REINFORCEMENT LEARNING APPROACHES FOR AUTONOMOUS GUIDANCE AND CONTROL IN A LOW-THRUST, MULTI-BODY DYNAMICAL ENVIRONMENT

by

Nicholas B. LaFarge

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Kathleen C. Howell, Chair
School of Aeronautics and Astronautics

Dr. Carolin Frueh
School of Aeronautics and Astronautics

Dr. Dengfeng Sun
School of Aeronautics and Astronautics

Dr. James M. Longuski
School of Aeronautics and Astronautics

Approved by:
Dr. Gregory A. Blaisdell

The Road goes ever on and on
   Out from the door where it began.
Now far ahead the Road has gone,
   Let others follow it who can!
Let them a journey new begin,
   But I at last with weary feet
Will turn towards the lighted inn,
   My evening-rest and sleep to meet.

J.R.R. Tolkein, *The Return of the King*

ACKNOWLEDGMENTS

I am endlessly grateful to Liene and Matiss for the constant joy you bring to my life and for your unwavering support over these many years. While I am thankful to the many supervisors, colleagues, and friends who have inspired many of my academic and career goals, I am even more grateful to you two for giving me a reason to make those aspirations come second. To my parents and sister, thank you for your constant love and encouragement throughout my academic career, I am immeasurably fortunate to call you family. Finally, but no less sincerely, thank you to my cat, Roxy, for always reminding me to appreciate sunny windows.

An immense debt of gratitude is owed to my advisor: Professor Kathleen Howell. I am still surprised and humbled that you accepted me into your research group as a first-year master's student with no engineering background (or clue about what I was doing in graduate school). Thank you for seeing my potential, for supporting and encouraging my professional development, and for teaching me most of what I know about research.

Thank you to my committee members, Professor Carolin Frueh, Professor Dengfeng Sun, and Professor James M. Longuski. I appreciate your helpful feedback during my preliminary examination and your generous use of time in reviewing this document.

Thank you to my research group: while graduate school can often be tumultuous, your company has always felt like an oasis. I appreciate your helpful suggestions in research meetings, your sound advice, and your willingness to always stop what you are doing to help me. Above all, I am thankful for your friendship.

I am immensely grateful to have collaborated on this research with some brilliant engineers. To my NASA research collaborator Dave Folta, thank you for the real-world expertise and insight you brought to this research and for hosting me at NASA Goddard. To Daniel Miller and Professor Richard Linares, thank you for the tremendous machine learning insight you both brought to this project.

Thank you to those at NASA who supported me thus far. At JSC, thank you to the Pathways program and, in particular, thank you to Chris D'Souza, Jack Brazzel, and Shane Robinson. I learned so much from you all, and I look forward to our paths crossing again in the future. At JPL, thank you to Mar Vaquero and Juan Senent for enabling my first experience at NASA; working with you both on Poincare was a highlight of graduate school.

I have many outstanding friends to thank for all their support along the way. Hank, Erik, Connor, and Fergus: thank you for decades of enduring friendship that has been unshaken by moves and time zones. Dane, TJ, Mitch, and Ken, thank you for the “rad” distractions.

I would finally like to thank and acknowledge all those who financially supported my academic studies. Thank you to my parents, the Purdue School of Aeronautics and Astronautics, the Truitt grant, Dr. Waterloo Tsutsui, and the NSTRF program.

I acknowledge MIT Supercloud and MIT Lincoln Laboratory Supercomputing Centers for providing HPC resources that have contributed to the research results reported within this document. In addition, this work was supported by a NASA Space Technology Research Fellowship, NASA Grant 80NSSC19K1175.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

To standardize mathematical nomenclature, several conventions are established for this document for typographic techniques and letter cases.

Quantity Type	**Style**	**Format**	**Example**
Scalar	normal	$◌$	$a = 3$
Nondimensional	normal	$◌$	$b = 0.1\,\mathrm{nondim}$
Dimensional	tilde	$\tilde{◌}$	$\tilde{b} = 10\,\mathrm{km}$
Dimensional time derivative	apostrophe	$◌'$	$f' = {}^{df}/_{d\tilde{t}}$
Nondimensional time derivative	overdot	$\dot{◌}$	$\dot{f} = {}^{df}/_{dt}$
Vector	bold	$\boldsymbol{◌}$	$\boldsymbol{c} = [c_x \;\; c_y \;\; c_z] \in \mathbb{R}^3$
Unit vector	circumflex	$\hat{◌}$	$\hat{d} = [d_x \;\; d_y \;\; d_z] \in \mathbb{R}^3, \lvert d \rvert = 1$
Rotating frame	lowercase	$\boldsymbol{v}$	$\boldsymbol{v} = [x_{\mathrm{rot}} \;\; y_{\mathrm{rot}} \;\; z_{\mathrm{rot}}]$
Inertial frame	UPPERCASE	$\boldsymbol{V}$	$\boldsymbol{V} = [X_{\mathrm{inertial}} \;\; Y_{\mathrm{inertial}} \;\; Z_{\mathrm{inertial}}]$
Neural network output	asterisk	$◌^*$	$p^* \in [-1, 1]$
Encoded variable	breve	$\breve{◌}$	$\breve{t} = \sqrt{t}$

LIST OF SYMBOLS

$\boldsymbol{A}$	Partial derivative matrix of the evolution functions with respect to state variables
$\boldsymbol{a}$	Reinforcement learning action vector
$\boldsymbol{a}^{\mathrm{lt}}$	Low-thrust acceleration vector
$\mathcal{A}$	Set of possible actions in a Markov decision process
B	CR3BP barycenter
b	Bonus applied in the reinforcement learning reward function
C	Jacobi constant
c	Constant value applied in reinforcement learning reward function
$\mathbb{E}$	Expected value operator
$\boldsymbol{F}$	Constraint vector
f	Nondimensional low-thrust magnitude
G	Nondimensional gravitational parameter
$\tilde{I}_{\mathrm{sp}}$	Specific impulse
$\mathcal{I}$	Inertial reference frame
$\mathcal{I}_{\mathrm{J2000}}$	J2000 reference frame
$\boldsymbol{J}$	Jacobian matrix
k	Distance to reference trajectory (norm of relative position and velocity vector)
$L_{◌}$	CR3BP Lagrange point (libration point): L_1, L_2, L_3, L_4, L_5
$\mathfrak{L}$	A sequence of multiple low-thrust segments
m	Nondimensional mass
$\mathfrak{m}$	Number of periodic orbit revolutions included in the targeting formulation
$\mathbb{m}$	Number of constraints in a targeting problem
$\mathbb{N}$	Set of natural numbers
$\mathcal{N}$	Normal (Gaussian) probability distribution
$\mathscr{N}$	Number of celestial bodies represented in an $\mathscr{N}$-body problem
n_x	Number of state variables in the evolution parameter vector $\boldsymbol{x}$
n_λ	Number of model parameters in the model parameter vector $\boldsymbol{\lambda}$
$\mathfrak{n}$	Number of steps taken in reinforcement learning environment

n	Nondimensional mean motion
$\mathbb{m}$	Number of free variables in a targeting problem
O	Origin of a frame of reference
$\mathcal{O}$	Set of possible observations in a partially observable Markov decision process
$\boldsymbol{o}$	Reinforcement learning observation vector
P	Particle (point mass)
$\mathbb{P}$	Orbital period
p	Penalty applied in reinforcement learning reward function
$\boldsymbol{q}$	Spatial rotating state vector (Nondimensional position, velocity, mass)
$\mathfrak{q}$	Planar rotating state vector (Nondimensional position, velocity, mass)
$\mathcal{Q}$	Action-value function (or state action-value function) in reinforcement learning
$\boldsymbol{R}$	Nondimensional position vector (inertial frame)
$\mathbb{R}$	Set of real numbers
$\mathcal{R}$	Rotating reference frame
$\mathcal{R}^{\text{ref}}$	Set of states that comprise a reference trajectory
$\mathfrak{R}$	Expected return in reinforcement learning
$\boldsymbol{r}$	Nondimensional position vector (rotating frame)
r	Reinforcement learning reward function
$\mathcal{S}$	Set of possible states in a Markov decision process
$\boldsymbol{s}$	Reinforcement learning state vector
$\mathfrak{s}$	Low-thrust segment
t	Nondimensional time
$\mathfrak{t}$	Thrust time (nondimensional time along a low-thrust arc)
$\mathbf{t}$	Time step in a reinforcement learning Markov decision process
$\mathcal{T}$	Ephemeris time
$\mathcal{T}_{\text{model}}$	Model epoch ephemeris time
$\mathfrak{T}$	Low-thrust throttle
U	Potential energy
$\mathcal{U}$	Uniform probability distribution

$\hat{u}$	Low-thrust direction vector
v_{e}	Nondimensional low-thrust exhaust velocity
$\boldsymbol{V}$	Nondimensional velocity vector (inertial frame)
$\mathcal{V}$	State-value function in reinforcement learning
$\boldsymbol{v}$	Nondimensional velocity vector (rotating frame)
$X,\ Y,\ Z$	Nondimensional position coordinates (inertial frame)
$\boldsymbol{X}$	Free variable vector
$\boldsymbol{x}$	Evolution parameter vector
$x,\ y,\ z$	Nondimensional position coordinates (rotating frame)
$\dot{x},\ \dot{y},\ \dot{z}$	Nondimensional velocity coordinates (rotating frame)
α	Activation function
β	Weighting coefficient in reinforcement learning reward function
γ	Reinforcement learning discount factor
Γ	Components of the reward function in reinforcement learning
$\Delta\tilde{V}$	Instantaneous change in velocity
$\Delta\tilde{V}_{\text{equiv.}}$	Equivalent low-thrust change in velocity (from the Tsiolkovsky rocket equation)
ζ	Rate at which the reward associated with relative state minimization increases along a reference path
η	Slack variable
θ	Azimuth angle from the rotating $\hat{x}$ axis
$\boldsymbol{\theta}$	Neural network trainable parameter vector
κ	Inclination angle measured from the $\hat{z}$ axis
λ	Scaling factor that adjusts the gradient of the reinforcement learning reward associated with relative state minimization increases along the reference path
$\boldsymbol{\lambda}$	Vector of model parameters
μ	Mass ratio in the CR3BP
ν	Stability index in the CR3BP
ξ	Encoded angle that locates states along multi-body periodic orbits
$\boldsymbol{\pi}$	Reinforcement learning policy

Π	Parameterization function
$\boldsymbol{\rho}$	Spatial rotating state vector (Nondimensional position, velocity)
$\boldsymbol{\varrho}$	Planar rotating state vector (Nondimensional position, velocity)
τ	Elapsed time since the model epoch ephemeris time ($\mathcal{T}_{\text{model}}$)
ϕ_{rot}	Angle between rotating and inertial reference frame
$\boldsymbol{\phi}$	Evolution function
$\boldsymbol{\Phi}$	State transition matrix (STM)
ω	Rotation rate in the CR3BP
Ω	CR3BP pseudo-potential function

ABBREVIATIONS

A2C Advantage Actor Critic 96

AI Artificial Intelligence 88

BIT-3 Busek Ion Thruster 3-cm 122

CR3BP Circular Restricted Three-Body Problem 40

CSI Constant Specific Impulse 57

DDPG Deep Deterministic Policy Gradient 96, 98

DPG Deterministic Policy Gradient 98

DQN Deep Q-Network 97

EDL Entry, Decent, and Landing 30

ER3BP Elliptic Restricted Three-Body Problem 42

ET Ephemeris Time 84

G&C Guidance and Control 111

GN&C Guidance, Navigation, and Control 111

KD-Tree K-dimensional tree 109

MDP Markov Decision Process 93

ML Machine Learning 88

NAIF Navigation and Ancillary Information Facility 84

NN Neural Network 89

NN-G&C Neural Network-aided Guidance and Control 27, 112

NNIT Neural Network-Initialized Targeting 113, 114, 116

NRHO Near Rectilinear Halo Orbit 120, 123, 209

POMDP Partially Observable Markov Decision Processes 93

PPO Proximal Policy Optimization 96

RL Reinforcement Learning 88

SAC Soft Actor Critic 96

SEP Solar Electric Propulsion 55

SK Stationkeeping 196, 197

STM State Transition Matrix 63

TD3 Twin Delayed Deep Deterministic Policy Gradient 97, 98

TDB Barycentric Dynamical Time 84

TRPO Trust Region Policy Optimization 96

XAC x-axis control 130, 214

ZVC Zero Velocity Curve 138, 163

ABSTRACT

Many far-reaching objectives and aspirations in space exploration are predicated on achieving a high degree of autonomous functionality. Traditional Earth-based mission operations are prohibitively expensive and may hinder the realization of various deep space mission objectives that necessitate swift decision-making capabilities despite extended communication delays. The ability of future space missions to overcome this challenge hinges on the development of onboard algorithms that maintain low computational costs without compromising on the precision necessary to ensure mission success. In developing autonomous functionality in safety-critical environments, recent progress has demonstrated the considerable potential of artificial neural networks as key elements in automation across a wide range of domains and disciplines. In the context of astrodynamics applications, evaluating the efficacy of employing a neural network onboard requires the identification of areas of applicability, thereby recognizing that these promising computational models are not universally suitable for every operation. After determining the application scope of neural networks, two interconnected objectives must be subsequently addressed. The first task involves selecting and establishing an appropriate training method for constructing the neural network. After training, the most effective strategy to utilize the neural network must be devised and examined through a systems-oriented lens. The overarching goal of this investigation is to address these objectives by demonstrating the versatility and promise of reinforcement learning as a machine learning framework that enables automated decision-making for challenging tasks in complex dynamical regions of space. Moreover, this research seeks to establish a class of composite algorithms that employ neural networks to augment and improve conventional methods, ultimately yielding an efficient and resilient strategy for autonomous spaceflight.

A notable challenge in automating space missions is the accommodation of off-nominal occurrences onboard that are typically addressed ad hoc by a team of specialists. In particular, determining maneuver plans in real-time for low-thrust mission applications in cislunar space remains challenging, particularly when confronted with unanticipated events. Many current low-thrust guidance and control approaches rely on either simplifying assumptions in the dynamical model or on abundant computational resources. However, future missions

to complex multi-body regions of space, such as the Earth-Moon neighborhood, require autonomous technologies that take advantage of the nonlinear dynamics to generate low-thrust control profiles without imposing an onerous workload on the flight computer, all while still ensuring that fundamental mission requirements are satisfied. This investigation addresses this challenge by leveraging neural networks trained via reinforcement learning to enhance onboard computational maneuver planning capability. The proposed reinforcement learning techniques function without explicit knowledge of the dynamical model, thus creating flexible learning schemes that are not limited to a single force model, mission scenario, or spacecraft. Moreover, this investigation develops a hybrid framework for low-thrust maneuver planning that capitalizes on the computational efficiency of neural networks and the robustness of targeting methods, thereby establishing a novel blended approach to autonomously construct and update maneuver schedules that satisfy mission requirements.

1. INTRODUCTION

Long-duration missions to cislunar space, characterized by its complex multi-body dynamics, are identified as a pivotal step toward achieving the ambition of a sustained presence for exploration and habitation in the far reaches of space. In support of the growing demand for efficient and cost-effective mission concepts, there is a pressing need for higher degrees of spacecraft autonomy that reduce the reliance on ground-based resources. Several factors motivate the migration of Guidance, Navigation, and Control (GN&C) functions to onboard flight computers. Key drivers for this transition, beyond time delays and operations cost, include the risk associated with communications failure, limitations in data transmission, and sensor tasking complexity. In overcoming these obstacles, future autonomous missions require advanced Guidance and Control (G&C) systems capable of responding to unanticipated occurrences given complex orbital dynamics. Developing such flexible and adaptive systems requires novel approaches that can overcome the challenges of multi-body spaceflight, i.e., the sensitive nonlinear dynamics, partially known environment, and limited onboard computing resources [1]. Hence, future G&C systems must achieve a high degree of accuracy without overly relying on individual mission characteristics or overburdening the flight computer. This investigation's overarching goal is to demonstrate Reinforcement Learning (RL), a subset of Machine Learning (ML), to be a powerful and flexible tool in enabling efficient, closed-loop G&C via a Neural Network (NN) in challenging, nonlinear regions of space. In support of this objective, mission scenarios within the Circular Restricted Three-Body Problem (CR3BP) are simulated to demonstrate the efficacy of NNs-based G&C concepts within challenging domains that incorporate multi-body gravitational effects.

This investigation seeks to establish a general foundation for RL applications that are both flexible and adaptive in support of autonomous low-thrust spaceflight. By devising approaches that can generalize well across diverse mission requirements, this research aims to underscore the important role that RL and NNs may play in the development of versatile onboard maneuver planning capabilities that will allow future spacecraft to respond more effectively and efficiently to unexpected events in challenging regions of space. Reinforcement learning excels in sequential decision-making problems that lack a priori knowledge of an effective policy, with NNs employed ubiquitously in recent applications to parameterize the control function. While ML methods (and NNs in particular) demonstrate considerable promise in serving as transformative technologies for autonomous systems, the surrounding enthusiasm, at times, borders on deification. As such, separating fact from fiction can be challenging, and an important step toward the utilization of these sophisticated computational models in space is the identification of areas of applicability and understanding their potential in enhancing G&C systems. In particular, NN behavior is often considered unpredictable, and this opaqueness introduces error and risk into potential future NN-augmented G&C concepts. This understandable concern is addressed in this analysis through the assessment of both advantages and disadvantages of leveraging a NN to support G&C tasks. Moreover, conventional G&C methods are explored to decrease the chance of NN function approximation errors adversely affecting mission outcomes. Various mission scenarios involving NN-aided G&C architectures are leveraged in this investigation to demonstrate the tangible benefits offered by NNs in terms of efficiency, accuracy, and robustness. The general class of NN-based approaches is referred to in this study as Neural Network-aided Guidance and Control (NN-G&C). Furthermore, this research aims to identify limitations of NN-G&C maneuver planning concepts to better understand their application scope in support of autonomous G&C objectives.

A typical mission concept plans for regular stochastic maneuvers to respond to a variety of anticipated deviations from the mission plan that stem from errors associated with state estimation, imperfectly executed maneuvers, and differences between real and simulated dynamics, among others [2]. Responding to such disturbances typically involves assuming that both the error and response are small. However, unforeseen occurrences, such as missed thrust events, may lead to more significant deviations that fall outside the scope of the implemented stochastic maneuver computation strategy. Depending on the scale of the departure, alternative methods may consider either a recovery plan that returns the spacecraft to its original orbit path or the construction of a new baseline trajectory. In the absence of a pre-determined contingency plan, an inadvertent departure from a desired orbit path poses a serious risk to mission success. With limited computational resources available, this investigation evaluates the role that NNs may play in the recovery maneuver planning process at varying error levels. In this NN-G&C concept, the NN is tasked with constructing a control history that considers both the control and timing components of maneuver planning.

The ubiquity of high-performance computing resources enables many recent advancements in G&C tasks. Optimization methods, in particular, are widely employed to construct low-thrust trajectories that minimize propellant expenditure. While optimization is an umbrella term that encompasses numerous algorithms, relatively few are practical for autonomous onboard implementation given the limited computing power. Furthermore, traditional optimization-based G&C concepts depend heavily on the accuracy of the dynamical model, and neglected nonlinearities can significantly impact the utility of the corresponding solution [1]. Moreover, solutions to optimization problems frequently occur at constraint boundaries; thus, fuel optimality is often obtained at the expense of maximizing the risk of violating mission requirements.

Targeting methods have recently emerged as promising tools in onboard computational G&C to iteratively construct feasible [3] or optimal [4] maneuver schedules while ensuring the necessary mission constraints remain satisfied. For example, the autoNGC flight software system, developed by NASA Goddard Space Flight Center, implements a forward shooting scheme for onboard maneuver planning optimization [4]. Similarly, the onboard GN&C architecture for Orion includes a two-level targeting algorithm capable of automatically altering maneuver plans based on navigation states [3]. Furthermore, targeting methods are commonly employed in multi-body Stationkeeping (SK) approaches [5], [6], with demonstrated applicability in Near Rectilinear Halo Orbit (NRHO) applications for the upcoming Gateway program [2], [7]. While targeting methods are effective in converging to feasible trajectories, they rely on sufficiently accurate startup solutions to achieve convergence and maintain current maneuver schedules. In the presence of large deviations or low-thrust propulsion options, an initially planned baseline reference trajectory may prove insufficient for achieving targeting convergence, and an alternative initial guess-generation strategy may be required.

The high cost of failure associated with spaceflight poses practical barriers to onboard ML applications, where NN accuracy and explainability issues introduce risk to mission success. While research into NN control stability in aerospace applications is ongoing [8], several previously suggested NN-G&C approaches benefit from NN function approximation while ensuring that the associated approximation errors do not impact mission safety [9]–[11]. In these paradigms, rather than the NN directly controlling the spacecraft, safety assurances are accomplished by tasking the NN with estimating startup solution components for conventional iterative algorithms that incorporate mission constraints. These blended approaches mitigate the computational footprint of iterative methods by leveraging NNs to rapidly identify basins of convergence. This investigation similarly leverages this NN initialization concept by introducing the Neural Network-Initialized Targeting (NNIT) approach to NN-G&C. This novel approach is enabled by an RL-based training process that constructs a NN that rapidly and autonomously generates accurate initial conditions for a targeter despite unexpected, prolonged deviations from a pre-planned mission scenario. A versatile approach to NN-G&C results, and multiple options for constructing an NNIT algorithm are evaluated through sample mission applications. In particular, directly incorporating traditional iterative methods into the training process broadens the available solution space, reduces the impact of chaotic dynamics on the training process, and ensures all mission criteria are satisfied by the corresponding solution.

1.1 Summary of Previous Contributions

Machine learning, as applied to astrodynamics, is becoming an increasingly active area of research and offers exciting new opportunities for applications in various phases of flight. As a motivating example, consider the Autonomous Exploration for Gathering Increased Science (AEGIS) algorithm implemented onboard multiple Mars rovers to aid in autonomously identifying geologic features of interest [12]. The AEGIS algorithm reduces the pointing refinement process for Curiosity to between 94 and 96 seconds [13] – substantially less than the time necessary to send images to Earth and wait for scientists to select features manually. While computational resources limit AEGIS to only a few trainable parameters, AEGIS is a motivating example for ML serving as a transformative technology in enabling autonomous spaceflight. With onboard computational capability increasing, more complex ML algorithms and models offer the potential for significant contributions to future autonomous missions. While these numerical methods are promising, identifying suitable ML techniques for specific problems in astrodynamics remains challenging. Understanding and reviewing previous contributions that lie at the intersection of ML and spaceflight offers insight into problem applicability.

The majority of recent ML research efforts in spaceflight involve training an artificial NN to approximate various functions in support of G&C tasks. Neural networks are most frequently trained via supervised learning or RL algorithms. In supervised learning, a NN is trained to estimate pre-generated data, commonly taking the form of 'behavior cloning,' where the goal of the learning process is reproducing desirable behavior, typically constructed in advance via optimization methods. In contrast to supervised learning, RL requires no such a priori knowledge and 'learns' by directly interacting with an environment. Both paradigms are frequently applied to similar problems, and since both processes lead to incorporating a NN in the G&C architecture, results in the literature for both supervised and reinforcement learning inform this research effort. Previous ML investigations are primarily focused on path-finding, guidance, small body operations, low-thrust G&C, as well as Entry, Decent, and Landing (EDL), and a literature survey in each of this application areas is included in this investigation. Additional relevant applications include rendezvous guidance [14]–[19], asteroid proximity operations [20], [21], collision avoidance [22], path-planning for asteroid surface exploration [23], attitude control [24], multi-target missions [25], and detection avoidance [26], [27]. Furthermore, several recent survey papers aggregate and elucidate a selection of ML-driven G&C applications in spaceflight [28], [29], and reference [1] focuses specifically on RL-based approaches for spacecraft control.

Autonomy is an essential component in EDL scenarios, and examining a selection of ML-based EDL studies reveals the breadth of application domains for NN-driven G&C applications in spaceflight. In particular, several EDL investigations leverage supervised learning to train a NN to estimate pre-computed optimal behavior. Sánchez-Sánchez and Izzo employ supervised learning to train a NN to represent solutions to the Hamilton-Jacobi-Bellman policy equation for several pinpoint lunar landing scenarios [30], and Furfaro et al. explore supervised learning to model a fuel-optimal lunar landing control history [31]. Applied to asteroid landing problems, Cheng et al. train a NN to estimate costate initial guesses for an indirect optimal control shooting scheme for asteroid landing [10], and to estimate both the optimal control history and the irregular gravity field dynamics of the asteroid [32].

Reinforcement learning is recently applied in a variety of recent EDL applications. Specifically, Furfaro et al. employ a ZEM/ZEV feedback approach to lunar lander guidance [33], Gaudet and Linares explore pinpoint Mars landing [34], and Gaudet, Linares, and Furfaro further this research in a six-degree-of-freedom simulation [35]. Furthermore, Scorsoglio et al. apply image-based RL to lunar landing [36], Jiang, Li, and Furfaro employ RL with a pseudospectral method for powered Mars descent [37], and Cheng et al. leverage deep RL in conjunction with indirect optimal control methods to produce an "actor-indirect" learning scheme [38]. In these studies, the swift responsiveness of NN controllers in EDL scenarios underscores the values of ML-based G&C approaches. The present investigation, however, differs from EDL applications in several notable ways. Specifically, EDL tasks demand a considerably higher degree of precision and occur over shorter timeframes compared to G&C operations in cislunar space. Furthermore, these application areas exist in a distinct dynamical domain, where EDL challenges factor in nonconservative forces, such as atmospheric drag, but benefit from the inclusion of only a single gravitational body.

Optimal control tasks supporting interplanetary transfers are of particular recent interest, with several investigations successfully applying supervised learning techniques to estimate various parameters in optimal control problems. Dachwald applied a NN to low-thrust trajectory optimization as early as 2004 [39], [40]. More recently, in Earth-to-Mars problems, Cheng et al. generate real-time optimal control values for a solar sail [41], Li, Baoyin, and Topputo train a NN to estimate both control states and indirect optimization costates for low-thrust spacecraft [9], and Schiassi et al. employ a Physics-Informed NN (PINN) to construct optimal planar trajectories [42]. Furthermore, Izzo and Öztürk expand on previous contributions with a massively large database of optimal low-thrust trajectories to train a NN to, in real-time, generate mass-optimal Earth-to-Venus interplanetary transfers in the presence of remarkably large perturbations [43]. In RL applications, Zavoli and Federici construct fuel-optimal closed-loop control policies that are resilient to missed thrust events, observation uncertainty, and maneuver execution errors [44], and Miller, Englander, and Linares employ RL for Earth-to-Mars transfers [45].

For path-finding and trajectory design in multi-body dynamical regimes, Das-Stuart, Howell, and Folta leverage Q-learning in conjunction with accessible regions to compute initial guesses for low-thrust transfers in the CR3BP, leverage supervised learning to influence the resulting solution geometry [46], and apply their framework to contingency planning [47]. In continuous state-space problems, Miller and Linares employ RL for low-thrust, multi-body trajectory guidance [48], De Smet and Scheeres use supervised learning to train a NN to identify heteroclinic connections in the CR3BP [49], Parrish and Scheeres explore many nearby optimized trajectories to train a NN to approximate costates in an optimal control problem [50], and Yan, Yang, and Li leverage NNs to aid in rapidly computing gravity assist trajectories for Jovian moons [51]. Furthermore, Sullivan et al. introduce "multi-objective" RL to uncover transfer paths between L_2 orbits [52], and extend this research to the Sun-Earth system and a point-mass ephemeris force model via transfer learning [53]. Federici et al. similarly leverages RL to uncover optimal guidance laws and training schemes that do not depend on a known "reference" motion for transfers between libration point orbits [54].

Several authors expand on previous contributions to apply RL to G&C problems along known reference paths in multi-body dynamical regimes. LaFarge et al. employ RL for guidance along multiple heteroclinic transfers between Lyapunov orbits in the Earth-Moon system [55], [56], and LaFarge, Howell, and Folta expand on this analysis in an NRHO unintended departure scenario that tasks a NN with identifying basins of convergence for a low-thrust targeting algorithm [57]. Furthermore, Sullivan et al. apply "multi-objective" RL to guide a spacecraft into a Lyapunov orbit [58] and between Lyapunov and halo orbits in the Earth-Moon system [59]. In orbit maintenance applications, several investigations employ RL to compute SK maneuvers that maintain libration point orbits. Guzzetti first leverages Q-Learning with a discretized state and action space for orbit maintenance of an L_1-Lyapunov orbit in the Earth-Moon system [60]. Applied to Sun-Earth halo orbits, Molnar leverages RL to augment Floquet mode control [61], Bonasera et al. employ RL to compute corrective maneuvers that offset momentum unloads in both the Sun-Earth system and a higher-fidelity ephemeris force model [62], and LaFarge, Howell, and Folta leverage RL in the Earth-Moon system to compute low-thrust SK maneuvers, and to determine effective maneuver placement strategies along periodic structures [63].

Several recent investigations demonstrate the utility of leveraging a NN to recover from missed-thrust events. Unlike many related tasks, missed-thrust occurrences are characterized by higher degrees of deviation from a mission plan. Autonomously recovering to pre-planned motion given an unexpected departure is particularly challenging, with prior missions frequently approaching missed-thrust analysis by discretizing the reference trajectory and exhaustively simulating and optimizing recovery plans [64]. Prior work demonstrates promise in leveraging ML to aid in autonomously overcoming missed-thrust events along interplanetary transfers by leveraging behavior cloning [65] or neuroevolutionary algorithms in conjunction with supervised learning [66] to construct a NN controller. Several factors that motivate missed-thrust analyses also motivate the present investigation. Specifically, while this research does not explicitly model missed-thrust events, large deviations resulting from unintended departures are of key interest in the proposed RL-based NN-aided G&C approaches.

1.2 Research Objectives

The primary aim of this investigation is to develop a versatile framework for creating flexible RL training environments that facilitate the development of adaptive NN-G&C processes in practical multi-body spaceflight applications. In particular, this research seeks to evaluate the potential impact and improvements that NN-G&C approaches may provide to onboard maneuver planning capabilities for future low-thrust missions in challenging regions of space. This overarching goal is subdivided into the following objectives:

1. **Identify areas of applicability within multi-body astrodynamics**

   Neural networks are leveraged in support of low-thrust G&C tasks that enable autonomous trajectory recovery and maintenance of challenging multi-body orbits and transfers in cislunar space. Multiple approaches for leveraging a NN within G&C systems are evaluated, and the associated RL training procedures are detailed. Potential benefits and drawbacks are explored through sample mission application simulations.

2. **Develop flexible RL training methods for low-thrust spaceflight**

   Reinforcement learning training environments that support low-thrust G&C are generically defined, with specific implementation decisions illustrated through sample mission applications. Problems are formalized to avoid assumptions regarding the selected dynamical model, individual mission characteristics, or domain-specific knowledge. Learning environments are implemented and evaluated for their generality and applicability to related problems.

3. **Identify and demonstrate practical approaches for NN-G&C**

   Reinforcement learning is employed to produce lightweight closed-loop controllers that offer considerable potential benefits to support onboard low-thrust G&C tasks. Rapid maneuver planning capability is demonstrated in response to large, unexpected deviations in sample mission applications. Benefits are weighted against limitations in evaluating the future applicability of NN-augment G&C systems.

4. **Introduce NN-G&C methods that blend NN and targeting methods**

   A novel 'hybrid' approach to low-thrust NN-G&C is developed, where a neural network is tasked with initiating a conventional iterative G&C algorithm (referred to as Neural Network-Initialized Targeting (NNIT) in this investigation). Multiple options for training and constructing NNIT architectures are evaluated, demonstrating the potential safety and accuracy benefits of incorporating an NN-G&C process into existing G&C systems.

The advancement of these objectives provides a foundation for future applications of RL in multi-body spaceflight. While the present research focuses on low-thrust propulsion in the Earth-Moon system, the proposed learning frameworks and G&C architectures are designed to be extendable to other dynamical regimes, propulsion options, and mission objectives, thus providing exciting prospects for future developments of the proposed learning framework.

1.3 Document Overview

The development and evaluation of the proposed RL and NN-G&C frameworks are discussed in this document through six chapters that each address a distinct aspect of this investigation. The document is organized as follows:

Chapter 1 - Dynamical Models The dynamical models employed in this investigation are developed. The equations of motion for the CR3BP and the $\mathcal{N}$-body ephemeris force models are derived, and the framework for augmenting process equations with low-thrust acceleration terms is presented.

Chapter 2 - Numerical Methods A corrections framework for computing continuous low-thrust trajectories is detailed. The process for numerically propagating sensitivities is developed, and the constraint-free variable approach to targeting is derived. Considerations for implementing the corrections procedure in low-thrust problems are outlined for the CR3BP and $\mathcal{N}$-body ephemeris models.

Chapter 3 - Reinforcement Learning The foundational concepts for RL and NNs are stated, and the background for this investigation's employed RL algorithm is presented. The general RL implementation procedure employed throughout this analysis is outlined, including the development of a flexible framework for RL signal design in support of low-thrust, multi-body applications. Furthermore, efficiency characteristics of the proposed NN-G&C techniques are examined.

Chapter 4 - Guidance and Control Architectures The two approaches employed in this investigation for leveraging a NN to support low-thrust G&C tasks are discussed. The 'standalone' procedure for directly leveraging a NN to generate control states is presented, and the proposed NNIT framework is introduced. Furthermore, two training approaches for NNIT implementation are considered, including both standalone and 'integrated' methods.

Chapter 5 - Sample Mission Applications The RL and NN-G&C techniques are evaluated through three sample mission applications. Section 6.1 implements the proposed integrated NNIT training algorithm in an unintentional departure scenario that tasks an NNIT-based agent with returning a low-thrust spacecraft to its reference NRHO orbit. Section 6.2 examines both NN-G&C and NNIT approaches to low-thrust G&C in generating recovery maneuver plans that return a spacecraft to heteroclinic transfers between Lyapunov orbits. Section 6.3 analyzes NN-G&C applicability in multi-body orbit maintenance applications that consider both the timing and control components of low-thrust SK in cislunar space. Finally, Section 6.4 explores the applicability of the CR3BP recovery maneuver plans (generated via NN-G&C) in a higher-fidelity $\mathcal{N}$-body force model.

Chapter 6 - Conclusion A summary of this investigation's key findings and results are presented and recommendations for future work are offered.

2. DYNAMICAL MODELS

A dynamical model is a mathematical representation of the forces that govern motion within a system. In astrodynamics problems, a dynamical model is constructed by identifying the dominant forces for a given application and evaluating simplification options. While increasing the number of modeled forces often leads to higher accuracy in predicting the motion of a celestial object, the corresponding increase in complexity limits insights into the fundamental motion. Hence, in the earlier stages of trajectory design and maneuver planning endeavors, a simplified model is often leveraged to gain insight into the properties of the system. In 1979, George Box described this balance between simplifications and fidelity under the section heading “All models are wrong, but some are useful[1]”,

> Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations ... there is no need to ask the question “Is the model true?”. If “truth” is to be the “whole truth” the answer must be “No”. The only question of interest is “Is the model illuminating and useful?” [67]

When a simplified model is suitably leveraged in this context, trajectories from the simplified model serve as excellent initial conditions for higher-fidelity simulations. This research employs a simplified dynamical model during preliminary training and evaluation stages, and demonstrates the persistence of the corresponding solutions as additional forces are introduced.

This investigation seeks to evaluate the utility of leveraging a NN for low-thrust G&C applications in cislunar space. In this region, the motion of the spacecraft is primarily influenced by the gravity of the Earth and the Moon; hence, the Circular Restricted Three-Body Problem (CR3BP) is employed throughout this study to train NN controllers and evaluate performance metrics. Trajectories constructed in the CR3BP are then transformed into a higher-fidelity $\mathcal{N}$-body ephemeris force model to evaluate the accuracy (or, as George

[1]↑This phrase has become a well-known aphorism in statistical/scientific modeling and machine learning. While typically attributed to George Box, variations of the aphorism occur in many works before his.

Box might observe, the usefulness), of the maneuver plans. In both models, additional acceleration terms are included to account for the influence of the low-thrust engine.

The dynamical models in this investigation are derived from Newton's laws of motion. Defined generically, a dynamical model that consists of second-order equations of motion is expressed generally as a series of first-order differential equations defined by an evolution function $\boldsymbol{\phi}$,

$$\dot{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}, t; \boldsymbol{\lambda}\right) \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is a time-dependent evolution parameter (defined as a state vector in this investigation), n_x is the number of elements in the state vector of the dynamical system, and $\boldsymbol{\lambda}$ is a vector of n_λ model-specific parameters that do not evolve over time. This generic definition provides a useful mathematical framework to derive dynamical models and implement numerical methods.

2.1 The Classical $\mathcal{N}$-Body Problem

The $\mathcal{N}$-Body Problem is a classical problem in celestial mechanics that describes the motion of a set of $\mathcal{N}$ objects under their mutual gravitational influence. This problem is originally formulated by combining Newton's second law of motion with the universal law of gravitation, both introduced by Newton in *Philosophiæ Naturalis Principia Mathematica*. In this model, each particle P_i, with mass $\tilde{m}_i$, moves in an inertial references frame originating at O in a gravity field consisting of the remaining $\mathcal{N}$-1 bodies. A specific problem of interest involves modeling the motion of a single massless particle, in this case P_i, as governed by nearby centrobaric attracting bodies. This investigation assumes P_i to be a spacecraft that is located by the dimensional position vector $\tilde{\boldsymbol{R}}_i = [\tilde{X}_i \;\; \tilde{Y}_i \;\; \tilde{Z}_i]$, as depicted in Figure 2.1. The relative equations of motions for this $\mathcal{N}$-problem is mathematically modeled in terms of the second-order vector differential equation,

$$\tilde{m}_i \tilde{\boldsymbol{R}}_i'' = -\tilde{G} \sum_{\substack{j=1 \\ j \neq i}}^{\mathcal{N}} \frac{\tilde{m}_i \tilde{m}_j}{\tilde{R}_{ji}^3} \tilde{\boldsymbol{R}}_{ji} \tag{2.2}$$

where $\tilde{G} \approx 6.674\,08 \times 10^{-20}$ $[\mathrm{km^3/kg\,s^2}]$ is the universal gravitational constant, $\tilde{m}_i$ and $\tilde{m}_j$ are the mass in kilograms of P_i and P_j, respectively. The relative position vector from P_j to P_i is easily computed from the difference in their positions $\tilde{\boldsymbol{R}}_{ji} = \tilde{\boldsymbol{R}}_i - \tilde{\boldsymbol{R}}_j$, with the corresponding distance magnitude computed via the l^2 norm $\tilde{R}_{ji} = |\tilde{\boldsymbol{R}}_{ji}|$. An apostrophe signifies derivative of a quantity with respect to dimensional, and capital letters represent inertial quantities; hence, $\tilde{\boldsymbol{R}}_i''$ represents the acceleration of particle P_i in the inertial frame.

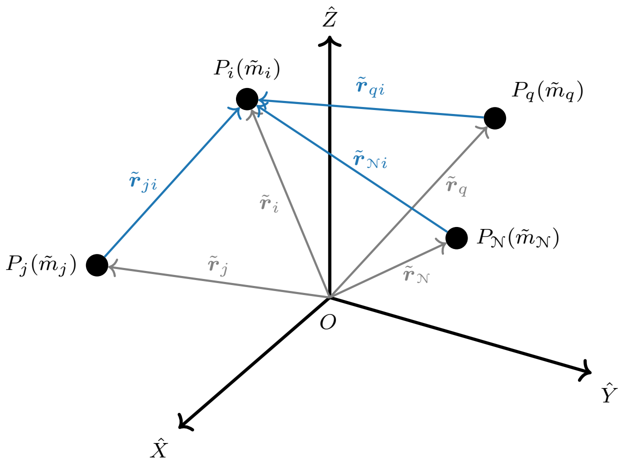

$\hat{Z}$
$P_i(\tilde{m}_i)$
$\tilde{\boldsymbol{r}}_{qi}$
$P_q(\tilde{m}_q)$
$\tilde{\boldsymbol{r}}_{ji}$
$\tilde{\boldsymbol{r}}_{\mathcal{N}i}$
$\tilde{\boldsymbol{r}}_i$
$\tilde{\boldsymbol{r}}_q$
$P_{\mathcal{N}}(\tilde{m}_{\mathcal{N}})$
$P_j(\tilde{m}_j)$
$\tilde{\boldsymbol{r}}_j$
$\tilde{\boldsymbol{r}}_{\mathcal{N}}$
O
$\hat{Y}$
$\hat{X}$

Figure 2.1. Vector definition in the $\mathcal{N}$-Body problem.

Expanding the number of celestial bodies ($\mathcal{N}$) included in the equations of motion increases both the fidelity of the model and the computational complexity of simulating trajectories. Examining motion in lower-fidelity models that include a subset of the relevant gravitational forces often provides insight into dynamical structures that govern motion in particular regimes. These simplified models are frequently employed in astrodynamics applications to enable preliminary analysis of the underlying dynamics and to construct initial guesses for higher-fidelity simulations.

The two-body problem ($\mathcal{N} = 2$) is the simplest special case of the general $\mathcal{N}$-body problem. Two-body motion has been extensively studied for centuries and admits the closed-form analytical solution discovered by Johannes Kepler. Assuming that the mass of the second body does not affect the motion of the first body ($\tilde{m}_1 >> \tilde{m}_2$), the equations of motion for $\tilde{m}_2$ are solved using conic sections. This geometric solution provides useful insight into the motion of objects with a single dominant gravitational force, e.g., an artificial or natural satellite's motion about a planet or star. While the two-body problem, and its analytical solution, is useful in a wide range of applications, its applicability is limited in dynamical regimes where two bodies significantly influence the motion of a spacecraft, such as in the vicinity of libration points. In these cases, additional gravitational bodies are frequently incorporated into the model to inform analysis of the multi-body dynamics. In mission design applications, the value for $\mathcal{N}$ is typically selected to be the smallest value that provides a reasonably accurate model for a given application. Once a preliminary analysis is complete, these lower-fidelity trajectories serve as initial guesses for the motion in increasingly higher-fidelity models. This investigation considers the motion of a spacecraft in cislunar space; hence, the CR3BP is employed to model the impact of both the Earth and Moon's gravity on the motion of the spacecraft. For subsequent analysis in more realistic scenarios, maneuver plans computed in this simplified three-body problem are transformed into a higher-fidelity $\mathcal{N}$-body ephemeris force model.

2.2 The Circular Restricted Three-Body Problem

The Circular Restricted Three-Body Problem (CR3BP) [68] is a model for the motion of an object with infinitesimal mass moving under the mutual gravitational influence of two celestial bodies. The CR3BP is a particular case of the more general 3-body problem, defined in Equation (2.2) for $\mathcal{N} = 3$, where the motion of the spacecraft, $P_3(\tilde{m}_3)$, is governed by,

$$\tilde{m}_3 \tilde{\boldsymbol{R}}_3'' = -\tilde{G}\frac{\tilde{m}_1\tilde{m}_3}{\tilde{R}_{13}^3}\tilde{\boldsymbol{R}}_{13} - \tilde{G}\frac{\tilde{m}_2\tilde{m}_3}{\tilde{R}_{23}^3}\tilde{\boldsymbol{R}}_{23} \tag{2.3}$$

In this dimensional second-order vector differential equation, $P_1(\tilde{m}_1)$ and $P_2(\tilde{m}_2)$ represent gravitational bodies that govern the motion of $P_3(\tilde{m}_3)$, as depicted in Figure 2.2. This Equation (2.3) is the most general form of the problem of three-bodies, and enables analysis of the mutual gravitational influence of all bodies. However, the case of the CR3BP, three additional simplifying assumptions are included to enable further analysis.

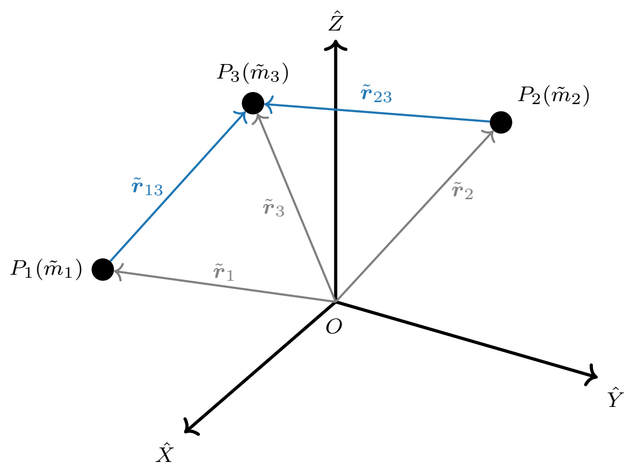


Figure 2.2. Vector definition in the 3-Body problem.

2.2.1 Simplifying Assumptions

The general three-body problem, represented in Equation (2.3), is focused on the analysis of the mutual gravitational influence of all three bodies. However, in many cases of interest, the mass of one body is much smaller than the other two and, thus, its relative gravitational impact is justifiably neglected. For example, it is reasonable to assume that the relatively small mass associated with a spacecraft results in a gravitational force that does not impact the orbits of nearby celestial bodies. In such a scenario, where three-bodies are considered and the mass of P_3 neglected, the motion of P_1 and P_2 forms the **primary system**: an isolated two-body system that is characterized by Keplerian motion about P_1 and P_2's common barycenter, B. The third particle, P_3, is propagated spatially with respect to B. Primary systems of interest include sun-planet systems (e.g., Sun-Earth) and planet-moon systems (e.g., Earth-Moon). For the CR3BP, the three simplifying assumptions are summarized as:

- **Assumption 1**: The mass of P_3 is infinitesimal relative to the masses of P_1, P_2 (i.e., $\tilde{m}_3 << \tilde{m}_1, \tilde{m}_2$).
- **Assumption 2**: P_1 and P_2 represent an isolated two-body conic system about their common barycenter.
- **Assumption 3**: P_1 and P_2 move on circular orbits around B.

By convention, the primary bodies are ordered such that $\tilde{m}_1 > \tilde{m}_2$. If the third assumption is omitted, the Elliptic Restricted Three-Body Problem (ER3BP) results. The ER3BP includes the pulsation of the primary two-body conic system in the model. While potentially useful in specific application, many primary systems of interest are nearly circular, the inclusion of the third assumption for the CR3BP produces a time-autonomous model that enables additional insight into the underlying dynamics that govern the motion. The Earth-Moon CR3BP is employed in this investigation as an example of a nonlinear dynamical environment that is relevant to upcoming missions to cislunar space.

2.2.2 Derivation from Newtonian Mechanics

The CR3BP, a model for motion of an infinitesimal mass moving under the influence of two massive bodies, is employed in this investigation to evaluate the proposed G&C schemes within the context of a complex dynamical regime. In this model, as depicted in Figure 2.3, the two spherically symmetric gravitational bodies, assumed in this investigation to be the Earth (P_1) and the Moon (P_2), form the primary system as they move in circular orbits about their common barycenter, B. The vector components that locate P_3 are propagated with respect to the system barycenter in a reference frame $\mathcal{R}$ that rotates with the orbit of P_2. This reference frame, here referred to as simply "the rotating frame", is spanned by dextral, orthonormal unit vectors $\mathcal{R} = [\hat{x} \;\; \hat{y} \;\; \hat{z}]$, denoted by red lines in Figure 2.3, such that $\hat{x}$-$\hat{y}$ are oriented by an angle ϕ_{rot} from the inertial unit vectors $\hat{X}$-$\hat{Y}$, and the two frames are aligned at the beginning of any propagation. The spatial component of both frames, $\hat{Z} = \hat{z}$, are defined as parallel to the orbital angular momentum vector, $\hat{x}$ is oriented from P_1 to P_2, and $\hat{y}$ completes the right-handed triad. Bodies P_1 and P_2 are located along the rotating $\hat{x}$ axis by the vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, respectively. The position of P_3 relative to P_1 is written as $\boldsymbol{r}_{13}$, and similarly $\boldsymbol{r}_{23}$ locates P_3 relative to P_2.

The CR3BP equations of motion are numerically integrated, and floating point and truncation errors result if state quantities span multiple orders of magnitude. To overcome this well-known numerical limitation, and to yield a more generalizable formulation of the problem, quantities in the CR3BP are nondimensionalized. Formulating the problem in terms of nondimensional quantities yields additional insight when extrapolating results between multiple CR3BP systems with similar mass ratios. In the CR3BP, the mass ratio between the first and second primary body is defined as,

$$\mu = \frac{\tilde{m}_2}{\tilde{m}_1 + \tilde{m}_2} \in [0,\, 0.5] \tag{2.4}$$

Introducing μ as a model parameter allows any primary systems with the same mass ratio to be represented by the same equations of motion. Thus, insight from a particular system is more easily applicable to another similar system. For example, analysis of the Saturn-Titan system ($\mu = 2.3664 \times 10^{-4}$) easily transitions into the Neptune-Triton system ($\mu = 2.0882 \times 10^{-4}$) due to their similar mass ratio. To nondimensionalize, characteristic quantities for position, mass, and time are heuristically selected to yield nondimensional values of the same order of magnitude. For this investigation, the characteristic length $\tilde{l}^*$ is defined as the distance between the primary bodies, i.e.,

$$\tilde{l}^* = \tilde{r}_1 + \tilde{r}_2 \tag{2.5}$$

where, as visualized in Figure 2.3, $\tilde{r}_1$ and $\tilde{r}_2$ are the dimensional distances from B to P_1 and P_2, respectively. In reality, the distance between two bodies varies due to the eccentricity in

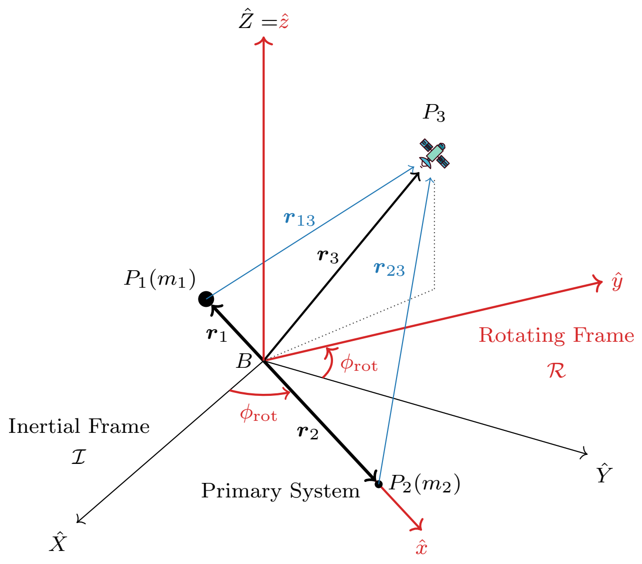


Figure 2.3. Vector definitions in the CR3BP where $\mu = 0.2$.

their orbits and other perturbing forces; this investigation selects $\tilde{l}^*$ to represent an average distance between the Earth and Moon based on ephemeris data retrieved from SPICE[2] [69]. The characteristic mass is defined as the sum of the masses of P_1 and P_2,

$$\tilde{m}^* = \tilde{m}_1 + \tilde{m}_2 \tag{2.6}$$

This quantity is related the mass ratio parameter μ, Equation (2.4) and, thus, the nondimensional masses m_1 and m_2, of P_1 and P_2, respectively, are expressed in terms of μ,

$$m_1 = \frac{\tilde{m}_1}{\tilde{m}^*} = 1 - \mu \qquad m_2 = \frac{\tilde{m}_2}{\tilde{m}^*} = \mu \tag{2.7}$$

Furthermore, by defining the origin of the system as the barycenter, the definition of center of mass (located solely in the $\hat{x}$ direction) delivers the relationship $(-\tilde{m}_1\tilde{r}_1 + \tilde{m}_2\tilde{r}_2)/\tilde{m}^* = 0$. Using the definition of $\tilde{l}^*$ in Equation (2.5) and the nondimensional mass values in Equation (2.7), it follows that the nondimensional distances from the barycenter to each primary body are also characterized by the mass ratio of the system,

$$\boldsymbol{r}_1 = (-\mu)\hat{x} \qquad\qquad \boldsymbol{r}_2 = (1-\mu)\hat{x} \tag{2.8}$$

The nondimensional distances in Equation (2.8) highlight the relationship between the mass of the primary bodies and their distances to the common barycenter.

Unlike the values of $\tilde{l}^*$ and $\tilde{m}^*$, which are intuitively selected to correspond to physical quantities, the characteristic time, $\tilde{t}^*$, is derived such that the nondimensional gravitational constant, G, is equal to one. Recall that the units for $\tilde{G}$ are $\mathrm{km^3/kg\,s^2}$. Hence, $\tilde{t}^*$ is derived by nondimensionalizing $\tilde{G}$ and solving for $\tilde{t}^*$,

$$G = \tilde{G}\left(\frac{\tilde{m}^*(\tilde{t}^*)^2}{(\tilde{l}^*)^3}\right) = 1 \quad \rightarrow \quad \tilde{t}^* = \sqrt{\frac{(\tilde{l}^*)^3}{\tilde{G}\tilde{m}^*}} \tag{2.9}$$

This investigation leverages the Earth-Moon system with the characteristic quantities listed in Table 2.1. Furthermore, noting that P_1 and P_2 are assumed to be in circular orbits around a common barycenter, the dimensional mean motion associated with these circular

[2]↑Thanks to Andrew Cox for insight into characteristic quantity selection!

orbits is evaluated as $\tilde{n} = \sqrt{\tilde{G}\tilde{m}^*/(\tilde{l}^*)^3}$. From Equation (2.9), it follows that $\tilde{n} = 1/\tilde{t}^*$. The nondimensional mean motion is straightforwardly computed as,

$$n = \tilde{n}\tilde{t}^* = 1 \tag{2.10}$$

Mean motion represents the angular speed necessary to complete one period of an orbit. Noting that $n = 2\pi/\mathbb{P}$, where $\mathbb{P}$ is the orbital period, it follows that the nondimensional period of the primary system is 2π.

Table 2.1. Characteristic quantities employed in this investigation

Earth-Moon System Characteristic Quantities		
μ	nondim	0.012 004 715 741 012
$\tilde{l}^*$	km	384 747.962 856 037
$\tilde{t}^*$	s	375 727.551 633 535
$\tilde{m}^*$	kg	$6.045\,825\,574\,495\,057 \times 10^{24}$

Leveraging the characteristic quantities in Table 2.1, the CR3BP nondimensional equations of motion that govern P_3 are derived from Newton's Second Law, as detailed in *Principia.* Recall that, for convenience, P_3 is located with respect to B in the rotating frame, as depicted in Figure 2.3. The nondimensional position and velocity vectors of P_3 are defined as,

$$\boldsymbol{r}_3 = x\,\hat{x} + y\,\hat{y} + z\,\hat{z}$$
$$\boldsymbol{v}_3 = \dot{x}\,\hat{x} + \dot{y}\,\hat{y} + \dot{z}\,\hat{z}$$

and together comprise the six-dimensional vector,

$$\boldsymbol{\rho} = \begin{bmatrix} \boldsymbol{r}_3 & \boldsymbol{v}_3 \end{bmatrix}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \in \mathbb{R}^6 \tag{2.11}$$

The $\boldsymbol{\rho}$ vector is the evolution parameter vector $\boldsymbol{x}$ for the CR3BP, as defined in Equation (2.1). To apply Newton's Second Law, the general form of the three-body problem from Equation (2.3) is first nondimensionalized, resulting in the nondimensional three-body problem,

$$\ddot{\boldsymbol{r}}_3 = -\frac{1-\mu}{r_{13}^3}\boldsymbol{r}_{13} - \frac{\mu}{r_{23}^3}\boldsymbol{r}_{23} \tag{2.12}$$

where $\ddot{\boldsymbol{r}}_3$ is the nondimensional acceleration vector. Throughout this investigation, an overdot signify a derivative with respect to nondimensional time. The position vectors that locate P_3 with respect to P_1 and P_2 are simply deduced as,

$$\boldsymbol{r}_{13} = (x+\mu)\hat{x} + y\,\hat{y} + z\,\hat{z} \tag{2.13}$$

$$\boldsymbol{r}_{23} = (x-1+\mu)\hat{x} + y\,\hat{y} + z\,\hat{z} \tag{2.14}$$

Using Equations (2.13) and (2.14), and the relationships in Equation (2.8), the scalar components of $\ddot{\boldsymbol{r}}_3$ are

$$\begin{cases} \ddot{\boldsymbol{r}}_3 \cdot \hat{x} = -\dfrac{(1-\mu)(x+\mu)}{r_{13}^3} - \dfrac{\mu(x-1+\mu)}{r_{23}^3} \\ \ddot{\boldsymbol{r}}_3 \cdot \hat{y} = -\dfrac{(1-\mu)y}{r_{13}^3} - \dfrac{\mu\,y}{r_{23}^3} \\ \ddot{\boldsymbol{r}}_3 \cdot \hat{z} = -\dfrac{(1-\mu)z}{r_{13}^3} - \dfrac{\mu\,z}{r_{23}^3} \end{cases} \tag{2.15}$$

where the magnitudes of $\boldsymbol{r}_{13}$ and $\boldsymbol{r}_{23}$ are

$$r_{13} = \sqrt{(x+\mu)^2 + y^2 + z^2} \tag{2.16}$$

$$r_{23} = \sqrt{(x-1+\mu)^2 + y^2 + z^2} \tag{2.17}$$

The acceleration vector $\ddot{\boldsymbol{r}}_3$ is computed by differentiating the position vector $\boldsymbol{r}_3$ in the rotating frame. Using the basic kinematic equation, also known as the transport theorem, the kinematic expression for velocity is first evaluated,

$$\dot{\boldsymbol{r}}_3 = \frac{{}^{\mathcal{I}}d}{dt}\boldsymbol{r}_3 = \frac{{}^{\mathcal{R}}d}{dt}\boldsymbol{r}_3 + {}^{\mathcal{I}}\boldsymbol{\omega}^{\mathcal{R}} \times \boldsymbol{r}_3 \tag{2.18}$$

where $\mathcal{I}$ represents the inertial frame, $\mathcal{R}$ represents the rotating frame, and ${}^{\mathcal{I}}\boldsymbol{\omega}^{\mathcal{R}}$ is the angular velocity of the rotating frame with respect to the inertial frame. For a circular orbit, this angular rate is simply the mean motion of the orbit, as noted in Equation (2.10), whose nondimensional magnitude is one (i.e. ${}^{\mathcal{I}}\boldsymbol{\omega}^{\mathcal{R}} = n\hat{z} = \hat{z}$). Substituting values and differentiating yields the kinematic expansion for the velocity of P_3 in the rotating frame,

$$\dot{\boldsymbol{r}}_3 = (\dot{x} - y)\hat{x} + (\dot{y} + x)\hat{y} + (\dot{z})\hat{z} \tag{2.19}$$

The BKE is again applied for the kinematic expression for the acceleration of P_3,

$$\ddot{\boldsymbol{r}}_3 = (\ddot{x} - 2\dot{y} - x)\hat{x} + (\ddot{y} + 2\dot{x} - y)\hat{y} + (\ddot{z})\hat{z} \tag{2.20}$$

Finally, the vector components of $\ddot{\boldsymbol{r}}_3$ in Equations (2.15) and (2.20) are combined to form the three nonlinear nondimensional second-order differential equations of motion for P_3,

$$\begin{cases} \ddot{x} - 2\dot{y} - x = -\dfrac{(1-\mu)(x+\mu)}{r_{13}^3} - \dfrac{\mu\,(x-1+\mu)}{r_{23}^3} \\ \ddot{y} + 2\dot{x} - y = -\dfrac{(1-\mu)\,y}{r_{13}^3} - \dfrac{\mu\,y}{r_{23}^3} \\ \ddot{z} = -\dfrac{(1-\mu)\,z}{r_{13}^3} - \dfrac{\mu\,z}{r_{23}^3} \end{cases} \tag{2.21}$$

To analytically integrate the CR3BP equations of motion, 12 integration constants are required. With only 10 known constants, no closed-form analytical solution exists. Thus, numerical integration methods are typically leveraged to produce the time history for a particle originating from an initial state, and are discussed further in Section 3.1.

The three-body problem is a conservative system and, hence, the force acting on P_3 is the gradient of a potential function. Therefore, in the inertial frame, Newton's second law for P_3 is expressed as the gradient the gravitational potential function,

$$m_3\ddot{\boldsymbol{r}}_3 = \boldsymbol{\nabla} U \tag{2.22}$$

where U is the specific gravitational potential energy expressed in nondimensional coordinates,

$$U = \frac{1-\mu}{r_{13}} + \frac{\mu}{r_{23}} \tag{2.23}$$

The CR3BP equations of motion are derived in a rotating reference frame, so additional terms are added to Equation (2.23) to accommodate the centrifugal potential. Combining yields the scalar *pseudo-potential* function,

$$\Omega = \underbrace{\frac{1-\mu}{r_{13}} + \frac{\mu}{r_{23}}}_{\text{Gravitational}} + \underbrace{\frac{1}{2}(x^2+y^2)}_{\text{Centrifugal}} \tag{2.24}$$

For convenience, the first partial derivative of Ω is denoted $\Omega_i = {}^{\partial\Omega}/_{\partial i}$. The gradient of the pseudo potential function results in the vector

$$\boldsymbol{\nabla}\Omega = [\Omega_x \;\; \Omega_y \;\; \Omega_z] \tag{2.25}$$

These partials are easily evaluated for each position vector measure number x, y, z,

$$\begin{cases} \Omega_x = -\dfrac{(1-\mu)(x+\mu)}{r_{13}^3} - \dfrac{\mu(x-1+\mu)}{r_{23}^3} + x \\ \Omega_y = -\dfrac{(1-\mu)y}{r_{13}^3} - \dfrac{\mu y}{r_{23}^3} + y \\ \Omega_z = -\dfrac{(1-\mu)z}{r_{13}^3} - \dfrac{\mu z}{r_{23}^3} \end{cases} \tag{2.26}$$

The pseudo-potential Ω allows the CR3BP equations of motion, Equation (2.21), to be concisely expressed in terms of Ω_i,

$$\begin{cases} \ddot{x} - 2\dot{y} = \Omega_x \\ \ddot{y} + 2\dot{x} = \Omega_y \\ \ddot{z} = \Omega_z \end{cases} \tag{2.27}$$

This succinct form of the CR3BP equations of motion offers insight into the motion as influenced by the gravitational potential.

Orbits in the CR3BP are commonly characterized by stability properties. In particular, one commonly used metric computes a linear estimate of stability based on eigenvalues of the monodromy matrix, i.e., the State Transition Matrix (STM), propagated for one orbital period (derived in Section 3.2). This 'stability index' is evaluated as,

$$\nu = \frac{1}{2}\left(|\lambda_{\max}| + \frac{1}{|\lambda_{\max}|}\right) \tag{2.28}$$

where $\lambda_{\max}$ represents the maximum eigenvalue of the monodromy matrix. Orbits with $\nu <= 1$ are considered linearly stable. Nearly-stable orbits, such as NRHOs, are of particular interest to upcoming missions. This quantity is simple useful way to evaluate linear stability; however, it does not account for the period of the underlying orbit. Thus, when it is necessary to compare the stability of orbits with dramatically different periods, an alternative stability index may provide more insight into the underlying motion.

2.2.3 Integral of Motion

While no general closed-form solution of the CR3BP currently exists, one known integral of motion emerges in the CR3BP formulation. This scalar term, denoted the **Jacobi constant**, C, offers useful insight into the orbital energy as formulated in the CR3BP rotating frame. To derive the Jacobi constant, first note that the pseudo-potential Ω, defined in Equation (2.24), is solely a function of rotating position coordinates and, therefore, is not explicitly a function of time. Therefore, its time-derivative is straightforwardly expressed as,

$$\frac{d\,\Omega}{dt} = \left(\frac{d\Omega}{dx}\right)\left(\frac{dx}{dt}\right) + \left(\frac{d\Omega}{dy}\right)\left(\frac{dy}{dt}\right) + \left(\frac{d\Omega}{dz}\right)\left(\frac{dz}{dt}\right) \tag{2.29}$$

$$= \Omega_x \dot{x} + \Omega_y \dot{y} + \Omega_z \dot{z} \tag{2.30}$$

Leveraging the CR3BP equations of motion, Equation (2.27), this expression may be written in terms of rotating velocity and acceleration terms,

$$\begin{aligned}\frac{d\,\Omega}{dt} &= \dot{x}(\ddot{x} - 2\dot{y}) + \dot{y}(\ddot{y} + 2\dot{x}) + \dot{z}\ddot{z} \\ &= \dot{x}\ddot{x} + \dot{y}\ddot{y} + \dot{z}\ddot{z}\end{aligned}$$

Note that this quantity is simply the dot product of P_3's velocity and acceleration in the rotating frame. The equation is integrated directly, yielding,

$$\Omega + \tilde{C} = \frac{1}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) \tag{2.31}$$

where $\tilde{C}$ is the constant of integration. By convention, the Jacobi constant, C, is defined as positive such that $C = -2\tilde{C}$,

$$C = 2\Omega - \left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) \tag{2.32}$$

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

2.2.4 Equilibrium Solutions

In the CR3BP, equilibrium solutions are frequently labelled the *libration points* or *Lagrange points*, and are determined when the energy gradient is equal to zero, $\boldsymbol{\nabla}\Omega = \mathbf{0}$, with each scalar component listen in Equation (2.26). It is evident from the Ω_z equation that the second term is necessarily positive, i.e., ${}^{(1-\mu)}/_{r_{13}^3} + {}^{\mu}/_{r_{23}^3} > 0$. Therefore, the equation is satisfied only if $z = 0$. Hence, all equilibrium points are located within the $\hat{x}$-$\hat{y}$ plane. Next, the two conditions that satisfy $\Omega_y = 0$ are,

$$y = 0 \qquad \text{or} \qquad 1 - \frac{1-\mu}{r_{13}^3} - \frac{\mu}{r_{23}^3} = 0 \tag{2.33}$$

These cases are solved separately, with $y = 0$ corresponding to the *collinear* libration points, and $1 - {}^{(1-\mu)}/_{r_{13}^3} - {}^{\mu}/_{r_{23}^3} = 0$ producing the *triangular* libration points.

Collinear Libration Points

In the CR3BP, three collinear equilibrium solutions exist along the $\hat{x}$ axis; these arise from $\Omega_y = 0$, Equation (2.26), for the case $y = 0$. The collinear libration points are determined by rearranging $\Omega_x = 0$, and solving the transcendental equation,

$$x = \frac{(1-\mu)(x+\mu)}{r_{13}^3} + \frac{\mu(x-1+\mu)}{r_{23}^3} \tag{2.34}$$

From among various options, Equation (2.34) is solved here numerically using a Newton-Raphson method. To distinguish between points, it is useful to iteratively solve for a distance with respect to a primary body, that is, rather than solving for x explicitly, each libration point is located by iteratively solving for the value of γ_i, as illustrated in Figure 2.4, where

L_i denotes the libration point with index i. This substitution allows the direct evaluation of $x_{L_1} = (1-\mu-\gamma_1)$, $x_{L_2} = 1-\mu+\gamma_2$, and $x_{L_3} = -\mu-\gamma_3$. Using the definition of γ_i, Equation (2.34) is reformulated in terms of γ_i. For each collinear libration point, these are,

$$1-\mu-\gamma_1 = \frac{(1-\mu)}{(1-\gamma_1)^2} - \frac{\mu}{\gamma_1^2} \tag{2.35}$$

$$1-\mu+\gamma_2 = \frac{1-\mu}{(1+\gamma_2)^2} + \frac{\mu}{\gamma_2^2} \tag{2.36}$$

$$\mu+\gamma_3 = \frac{(1-\mu)}{\gamma_3^2} + \frac{\mu}{(1+\gamma_3)^2} \tag{2.37}$$

These equations are solved using a basic Newton-Raphson method to produce three collinear libration points. By convention, L_1 is located between the primaries, L_2 is on the $+\hat{x}$ side of P_2, and L_3 is the remaining point in the $-\hat{x}$ direction from P_1, as depicted in Figure 2.4.

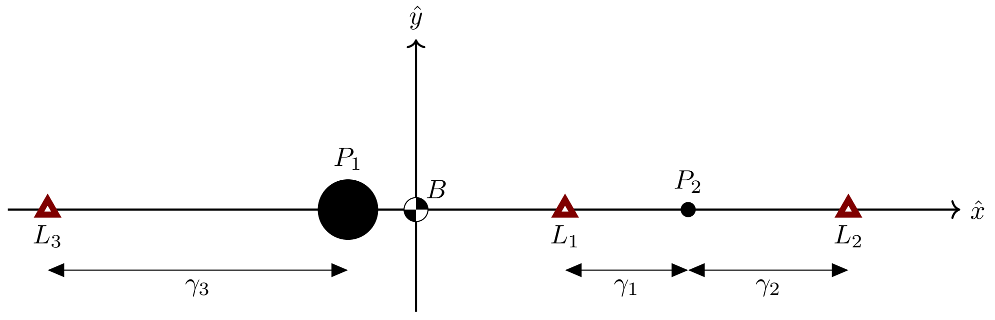


Figure 2.4. Collinear libration points for a CR3BP system with $\mu = 0.2$.

Triangular Libration Points

In the CR3BP, two equilibrium solutions are computed analytically. These solutions are termed the triangular libration points due to their geometry, and are labelled L_4 and L_5, as illustrated in Figure 2.5. The triangular libration points are derived from setting $\Omega_y = 0$ in Equation (2.26), for the case where,

$$1 - \frac{1-\mu}{r_{13}^3} - \frac{\mu}{r_{23}^3} = 0 \tag{2.38}$$

Since r_{13} and r_{23} are defined as physical quantities, this equation is only satisfied when the values for r_{13} and r_{23} possess no imaginary component; such a result only occurs for

$r_{13} = r_{23} = 1$. Substituting $r_{13} = r_{23} = 1$ into Equations (2.16) and (2.17) (given $z = 0$) produces,

$$1 = (x_{L_t} + \mu)^2 + y_{L_t}^2 \tag{2.39}$$

$$1 = (x_{L_t} - 1 + \mu)^2 + y_{L_t}^2 \tag{2.40}$$

Subtracting Equation (2.40) from Equation (2.39), and simplifying, yields the relationship,

$$x_{L_t} + \mu = \pm(x_{L_t} - 1 + \mu) \tag{2.41}$$

The positive sign implies a contradiction ($0 = -1$), clearly an invalid solution. Rather, consider the negative sign such that $x_{L_t} + \mu = -x_{L_t} + 1 - \mu$. This result is then simplified, with the final relationship,

$$x_{L_t} = \frac{1}{2} - \mu \tag{2.42}$$

The value of $x_{L_t}$is then substituted back into Equation (2.39) and, once simplified, yields the $\hat{y}$-component for the two triangular libration points, $y_{L_t} = \pm\sqrt{3}/2 \approx 0.866\,025$ [nondim]. Together, the locations of the triangular libration points are,

$$\boldsymbol{r}_{L_4} = \begin{bmatrix} \frac{1}{2} - \mu & \frac{\sqrt{3}}{2} & 0 \end{bmatrix} \tag{2.43}$$

$$\boldsymbol{r}_{L_5} = \begin{bmatrix} \frac{1}{2} - \mu & -\frac{\sqrt{3}}{2} & 0 \end{bmatrix} \tag{2.44}$$

The collinear and triangular libration points are together illustrated in Figure 2.5.

2.3 Higher-Fidelity Modeling

Astrodynamics problems are often approached by beginning analysis in a simplified environment and subsequently verifying the results in a higher-fidelity dynamical model. While the present investigation focuses primarily on RL applications within the CR3BP, a higher-fidelity $\mathcal{N}$-body ephemeris simulation is implemented to explore the validity of the proposed methods in a more realistic simulation. The $\mathcal{N}$-body problem, introduced in Section 2.1, models the motion of an object under the influence of an arbitrary number of celestial bodies. The equations of motion are derived in an inertial reference frame, but do not assume a

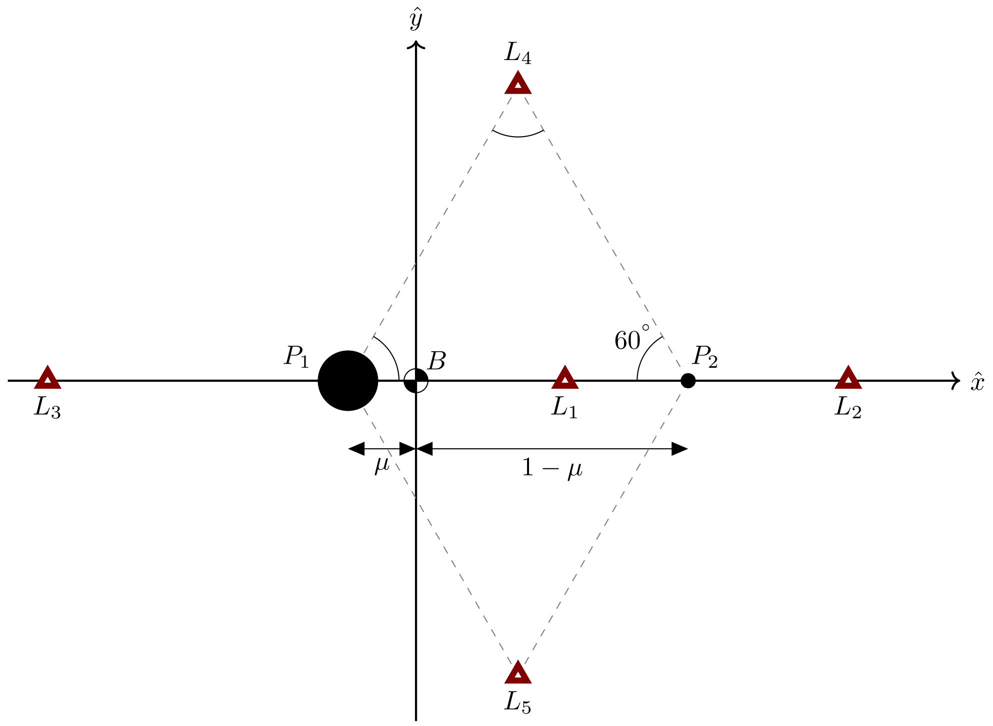


Figure 2.5. Libration points for a CR3BP system where $\mu = 0.2$.

specific orientation of the inertial axes. This investigation propagates motion in the J2000 equatorial frame as a standard inertial orientation and, optionally, transforms solutions to a rotating reference frame for comparison with the CR3BP.

The $\mathcal{N}$-body ephemeris model describes the motion of a massless particle P_i with respect to a primary body $P_q(m_q)$, with the remaining $\mathcal{N}-2$ bodies included as perturbing gravitational forces, as visualized in Figure 2.6. The selection of a central body influences the accuracy of numerical propagations and is, therefore, and important consideration. The nearby celestial body that generally imparts the largest gravitational influence on P_3 is frequently employed. The equations of motion are derived by employing Equation (2.2) to locate the position of P_i with respect to the primary body (P_q). Noting that $\tilde{\boldsymbol{R}}_{qi} = \tilde{\boldsymbol{R}}_i - \tilde{\boldsymbol{R}}_q$

(and therefore $\tilde{\boldsymbol{R}}''_{qi} = \tilde{\boldsymbol{R}}''_{i} - \tilde{\boldsymbol{R}}''_{q}$ in this inertial reference frame), the relative equations of motion is expanded as,

$$\tilde{\boldsymbol{R}}''_{qi} = \underbrace{-\tilde{G}\sum_{\substack{j=1\\j\neq i}}^{\mathcal{N}} \frac{\tilde{m}_j}{\tilde{R}^3_{ji}}\tilde{\boldsymbol{R}}_{ji}}_{\ddot{\tilde{\boldsymbol{R}}}_i} + \underbrace{\tilde{G}\sum_{\substack{j=1\\j\neq q}}^{\mathcal{N}} \frac{\tilde{m}_j}{\tilde{R}^3_{jq}}\tilde{\boldsymbol{R}}_{jq}}_{-\ddot{\tilde{\boldsymbol{R}}}_q} \tag{2.45}$$

where $\boldsymbol{R}_{[a][b]}$ represents the relative position of body $[b]$ with respect to body $[a]$. Equation (2.45) is simplified by extracting the $j = q$ term from the sum in the $\tilde{\boldsymbol{R}}''_i$ term, and neglecting the $j = i$ term from the sum within $-\tilde{\boldsymbol{R}}''_q$ (P_i is assumed massless, therefore $\tilde{m}_i = 0$). Equation (2.45) becomes,

$$\tilde{\boldsymbol{R}}''_{qi} = -\tilde{G}\frac{\tilde{m}_q}{\tilde{R}^3_{qi}}\tilde{\boldsymbol{R}}_{qi} - \tilde{G}\sum_{\substack{j=1\\j\neq i,q}}^{\mathcal{N}} \frac{\tilde{m}_j}{\tilde{R}^3_{ji}}\tilde{\boldsymbol{R}}_{ji} + \tilde{G}\sum_{\substack{j=1\\j\neq i,q}}^{\mathcal{N}} \frac{\tilde{m}_j}{\tilde{R}^3_{jq}}\tilde{\boldsymbol{R}}_{jq} \tag{2.46}$$

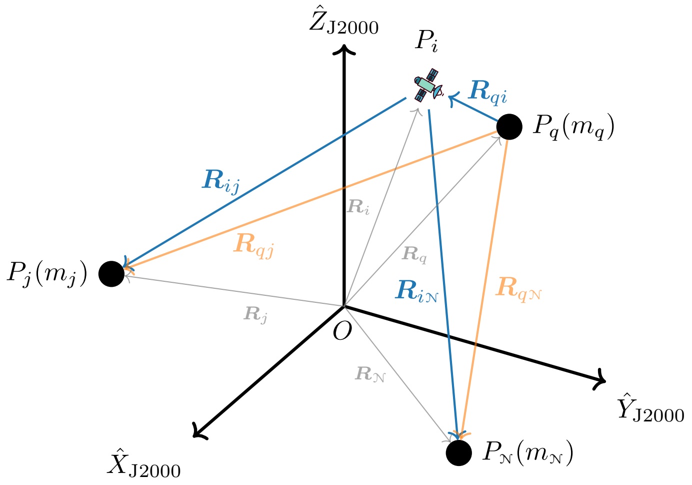
Next, the sums in Equation (2.46) are combined, and the direction and signs associated with $\tilde{\boldsymbol{R}}_{ji}$ and $\tilde{\boldsymbol{R}}_{jq}$ are switched for consistency with existing literature. These adjustments result in the dimensional equations of motion,

$$\tilde{\boldsymbol{R}}''_{qi} = -\tilde{G}\frac{\tilde{m}_q}{\tilde{R}^3_{qi}}\tilde{\boldsymbol{R}}_{qi} + \tilde{G}\sum_{\substack{j=1\\j\neq i,q}}^{\mathcal{N}} \tilde{m}_j\left(\frac{\tilde{\boldsymbol{R}}_{ij}}{\tilde{R}^3_{ij}} - \frac{\tilde{\boldsymbol{R}}_{qj}}{\tilde{R}^3_{qj}}\right) \tag{2.47}$$

As with the CR3BP, certain numerical errors associated with integration are mitigated by simulating nondimensional equations of motion with variables on the same order of magnitude. While the choice of dimensional constants is problem-dependent, this investigation focuses on cislunar space and, therefore, the $\mathcal{N}$-body ephemeris and CR3BP share the same characteristic quantities, listed in Table 2.1. Leveraging the same nondimensionalization process as the previous model, the dimensional equations of motion in Equation (2.47) becomes,

$$\ddot{\boldsymbol{R}}_{qi} = -G\frac{m_q}{R^3_{qi}}\boldsymbol{R}_{qi} + G\sum_{\substack{j=1\\j\neq i,q}}^{\mathcal{N}} m_j\left(\frac{\boldsymbol{R}_{ij}}{R^3_{ij}} - \frac{\boldsymbol{R}_{qj}}{R^3_{qj}}\right) \tag{2.48}$$

In this investigation, the Moon serves as the central body, with the Sun, Earth, and Jupiter comprising the perturbing bodies. The Moon-centered J2000 inertial reference frame (denoted ${}^{☾}\mathcal{I}_{\text{J2000}}$) is employed to represent and propagate state variables; rotating and J2000 reference frame transformations are detailed in [70]. Planetary ephemerides at specific points in time are obtained using the DE440 SPICE kernel, retrieved from JPL's Navigation and Ancillary Information Facility (NAIF) [69].



$\hat{Z}_{\mathrm{J2000}}$
P_i
$\boldsymbol{R}_{qi}$
$P_q(m_q)$
$\boldsymbol{R}_{ij}$
$\boldsymbol{R}_i$
$\boldsymbol{R}_{qj}$
$\boldsymbol{R}_q$
$P_j(m_j)$
$\boldsymbol{R}_{i\mathcal{N}}$
$\boldsymbol{R}_{q\mathcal{N}}$
$\boldsymbol{R}_j$
O
$\boldsymbol{R}_{\mathcal{N}}$
$\hat{Y}_{\mathrm{J2000}}$
$\hat{X}_{\mathrm{J2000}}$
$P_{\mathcal{N}}(m_{\mathcal{N}})$

Figure 2.6. Vector diagram for the $\mathcal{N}$-body ephemeris force model in a J2000 reference frame.

2.4 Low-Thrust Solar Electric Propulsion

Many mission architectures benefit from low-thrust Solar Electric Propulsion (SEP). In contrast to traditional chemical engines, electric propulsive engines are much more efficient but deliver energy changes over longer time intervals. Low-thrust spacecraft using SEP include ion thrusters that are powered through solar panels on the spacecraft. The inclusion of low-thrust propulsion in the dynamical model produces additional acceleration terms that are added to the ballistic equations of motion. Currently, ion thrusters are successfully employed on various missions [71]; several specific low-thrust spacecraft are listed in Table 2.2.

The impact of low-thrust propulsion on the underlying dynamics may be modeled at varying levels of fidelity. This investigation implements a straightforward low-thrust model that assumes a linear mass flow rate and a fixed thrust vector over any integration segment. An illustrative derivation and discussion of this model and other "pseudo-separable" low-thrust parameterization options are detailed by Cox in [72]. In this investigation, the dimensional low-thrust acceleration vector is defined as

$$\tilde{\boldsymbol{a}}^{\mathrm{lt}} = \frac{\tilde{\boldsymbol{f}}}{\tilde{m}_{3,0}}, \qquad \tilde{f} = |\tilde{\boldsymbol{f}}| \in [0, \tilde{f}_{\max}] \tag{2.49}$$

where $\tilde{f}_{\max}$ is the maximum thrust that the low-thrust engine is capable of producing (in mN), and $\tilde{m}_{3,0}$ is the initial mass of the spacecraft (in kg). Furthermore, A nondimensional mass quantity is defined as the ratio between the spacecraft's current and initial mass,

$$m = \frac{\tilde{m}}{\tilde{m}_{3,0}} \in [0, 1] \tag{2.50}$$

Unfortunately for low-thrust trajectory designers, spacecraft cannot gain mass and, therefore, the nondimensional mass is always less than one.

This investigation simulates nondimensional dynamical models and, thus, nondimensional thrust quantities are leveraged. In particular, f denotes the nondimensional thrust magnitude, and is derived by multiplying $\tilde{f}$ by characteristic quantities,

$$f = \tilde{f}\left(\frac{\tilde{t}^{*2}}{\tilde{l}^{*}\,\tilde{m}_{3,0}}\right) \in [0, f_{\max}] \tag{2.51}$$

where $f_{\max}$ is the nondimensionalized maximum thrust value. Thrust magnitude is also equivalently represented by a throttle value,

$$\mathcal{T} = \frac{f}{f_{\max}} \in [0, 1] \tag{2.52}$$

The throttle value is useful for communicating the relative magnitude of a low-thrust segment. To ensure that the simulations in this investigation are realistic, a collection of previous and planned low-thrust engine capabilities is summarized in Table 2.2, and visualized in Figure 2.7. This investigation implements propulsion models where f is in the range of values 0.0299–0.04; thus, the sample mission applications are at the middle-to-lower-end of current low-thrust capability. Given the nondimensional thrust magnitude f, the low-thrust acceleration vector is readily obtained by dividing f by the nondimensional mass,

$$\boldsymbol{a}^{\mathrm{lt}} = \frac{f}{m}\hat{a}^{\mathrm{lt}} \quad \rightarrow \quad a^{\mathrm{lt}} = \frac{f}{m} \tag{2.53}$$

where $\hat{a}^{\text{lt}}$ is a unit vector representing the thrust orientation, and a^{lt} quantifies the acceleration magnitude imparted on the system by the low-thrust engine. As detailed by Cox [72], the modeled low-thrust acceleration values employed in this investigation are on the on same order of magnitude to the perturbing acceleration of the Sun, with low-thrust possessing slightly higher acceleration values in the vicinity of the Moon. Thus, transitioning solutions to a force model that includes the gravitational effect of the Sun is an important step for implementing the proposed G&C methods in a practical scenario. While preliminary results demonstrate promise in the current transition process; however, a robust closed-loop transition process that is suitable for onboard computation remains future work.

Various options exist for defining the low-thrust direction coordinates that orient the $\hat{a}^{\text{lt}}$ acceleration vector in Equation (2.53). This investigation assumes the low-thrust orientation unit vector $\hat{a}^{\text{lt}}$ is fixed in the reference frame leveraged to propagate the model dynamics. In $\mathcal{N}$-body ephemeris applications, $\hat{a}^{\text{lt}} = \hat{u}_{\text{J2000}}$ signifies an inertially-fixed thrust direction in the J2000 reference frame. For CR3BP simulations, $\hat{a}^{\text{lt}} = \hat{u}$ is defined in the rotating frame visualized in Figure 2.3 and remains fixed over any integration segment. While a continuously changing inertial thrust direction may not be practical, precession of the rotating frame during thrusting time intervals is addressed when transferring CR3BP solutions into a higher-fidelity model. Furthermore, spherical coordinate angles $\{\theta, \kappa\}$ are employed in CR3BP targeting applications and, as visualized in Figure 2.8, are related to $\hat{u}$ via the transformation,

$$\hat{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T = \begin{bmatrix} \cos\theta \sin\kappa & \sin\theta \sin\kappa & \cos\kappa \end{bmatrix}^T \tag{2.54}$$

where, θ represents the azimuth angle from the rotating $\hat{x}$ axis, and κ denotes the inclination angle measured from $\hat{z}$.

When the low-thrust engine is operating, the mass of the spacecraft decreases over time as propellant is expended. This linear mass flow rate is computed as

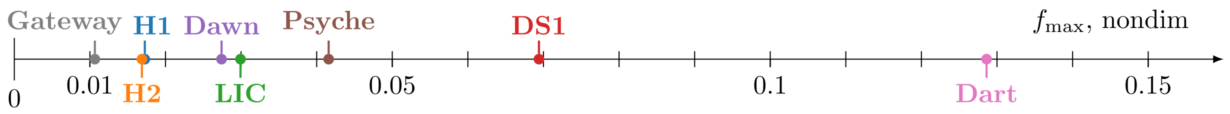
$$\dot{\tilde{m}} = -\frac{\tilde{f}}{\tilde{v}_{\text{e}}}\ , \quad \text{where} \quad \underbrace{\tilde{v}_{\text{e}} = \tilde{I}_{\text{sp}}\tilde{g}_0}_{\text{Exhaust velocity}} \tag{2.55}$$

where $\tilde{g}_0 = 9.806\,65 \times 10^{-3}\,{}^{\text{km}}/_{\text{s}}$ is the gravitation constant (a mean value for gravitational acceleration on Earth). The investigation assumes a Constant Specific Impulse (CSI) engine,

Table 2.2. Low-thrust capability of various spacecraft, nondimensionalized with the Earth-Moon characteristic quantities in Table 2.1.

Abbrv.	Spacecraft	$f_{\max}$, nondim	$\tilde{m}_{3,0}$, kg	$\tilde{f}_{\max}$, mN	Source
H1	Hayabusa 1	1.726×10^{-2}	510	24.0	[73]
H2	Hayabusa 2	1.688×10^{-2}	608.6	28.0	[74]
LIC	Lunar IceCube	2.992×10^{-2}	13.487	1.1	[75]–[77]
Dawn	Dawn	2.741×10^{-2}	1217.8	91.0	[78]
DS1	Deep Space 1	6.940×10^{-2}	486.3	92.0	[79]
Dart	Dart	1.286×10^{-1}	676	237	[80], [81]
Psyche*	Psyche	4.158×10^{-2}	2464	279.3	[82], [83]
Gateway*	Gateway	1.065×10^{-2}	39 000	1132.6	[84]

* Upcoming (planned) mission



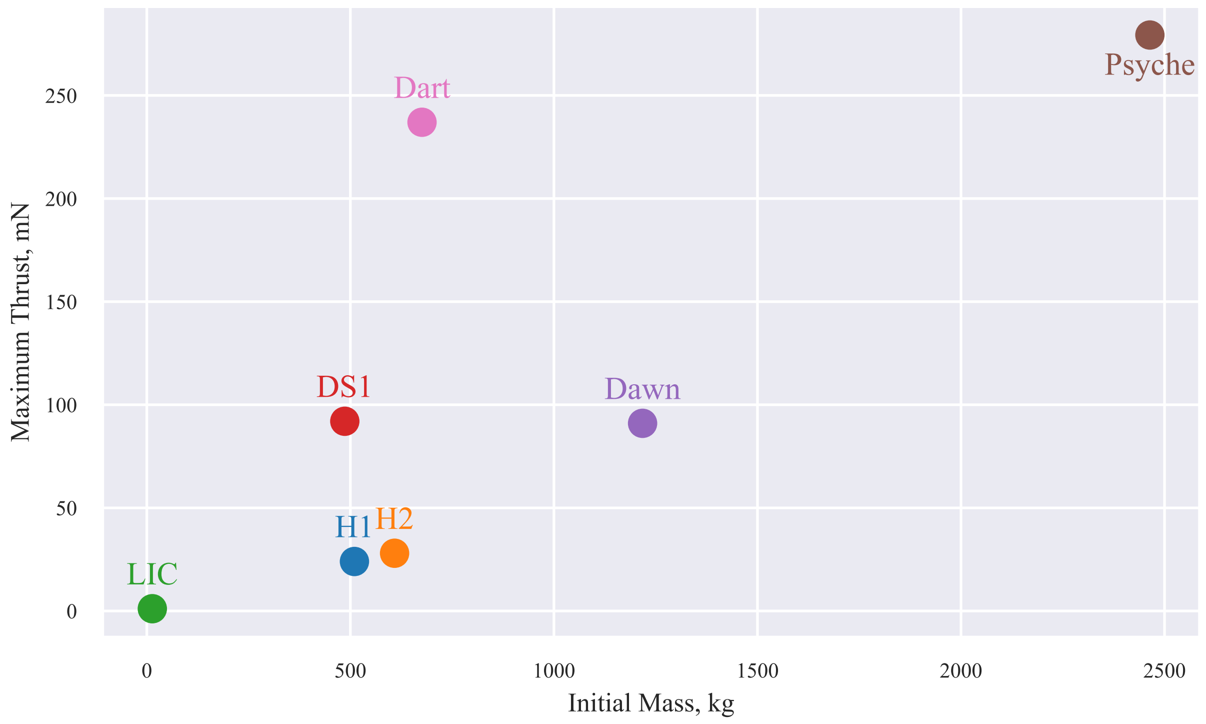


Figure 2.7. Launch mass and maximum thrust for the low-thrust spacecraft listed in Table 2.2. Gateway omitted from visualization due to the order-of-magnitude difference in size.

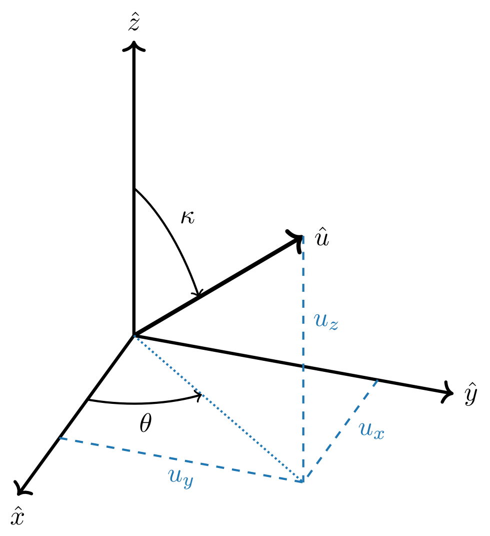


Figure 2.8. Low-thrust CR3BP direction coordinates employed in this investigation

i.e., $\tilde{I}_{\text{sp}}$ is constant value measured in seconds. A larger value for specific impulse implies less mass is expended to apply the same force; thus, $\tilde{I}_{\text{sp}}$ is intuitively understood as a measure of engine efficiency and is an important metric in motivating low-thrust mission concepts and comparing low-thrust engines. Furthermore, it is clear from Equation (2.53) that propulsive capability is inversely related to spacecraft mass; hence, as propellant is expended, the spacecraft is capable of higher acceleration values. This investigation employs nondimensional quantities and, thus, the mass flow rate is nondimensionally represented as

$$\dot{m}^{\text{lt}} = -\frac{f}{v_{\text{e}}}\,, \qquad \text{where} \qquad v_{\text{e}} = \tilde{v}_{\text{e}}\left(\frac{\tilde{t}^*}{\tilde{l}^*}\right) \tag{2.56}$$

The spacecraft's nondimensional exhaust velocity is represented as v_{e} and is obtained by nondimensionalizing $\tilde{v}_{\text{e}}$ in Equation (2.55), where $\tilde{l}^*$ and $\tilde{t}^*$ are the characteristic length and time of the dynamical system.

It is frequently useful to compare low-thrust solutions to their chemical counterparts. The possibility of long-duration thrusting segments allows for new path geometries compared to ballistic motion and enables new mission concepts. In comparing the resulting maneuver plans, propellant expenditure is a useful figure of merit and is frequently measured in terms of the instantaneous change in velocity in applications involving chemical engines. For low-thrust, an equivalent $\Delta\tilde{V}$ is available from the Tsiolkovsky rocket equation,

$$\Delta\tilde{V}_{\text{equiv.}} = \tilde{v}_{\text{e}} \log\left(\frac{m_0}{m_f}\right) \tag{2.57}$$

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

Low-thrust dynamics are implemented by augmenting a natural dynamical model, defined in Equation (2.1). In a general approach, a low-thrust evolution parameter $\boldsymbol{x}^{\text{lt}}$ is constructed by appending mass to a n_x-dimenstional natural state vector $\boldsymbol{x}$,

$$\boldsymbol{x}^{\text{lt}} = \begin{bmatrix} \boldsymbol{x} & m \end{bmatrix}^T \in \mathbb{R}^{n_x+1} \tag{2.58}$$

Given the linear mass flow rate defined in Equation (2.56), the low-thrust evolution function becomes,

$$\dot{\boldsymbol{x}}^{\text{lt}} = \boldsymbol{\phi}^{\text{lt}}\left(\boldsymbol{x}^{\text{lt}}, t, \boldsymbol{\lambda}^{lt}\right) = \begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{\phi}\left(\boldsymbol{x}, t, \boldsymbol{\lambda}\right) + \boldsymbol{a}^{\text{lt}} \\ \dot{m} = \dot{m}^{\text{lt}} \end{cases} \tag{2.59}$$

where $\boldsymbol{a}^{\text{lt}}$ is defined in Equation (2.53) such that $\hat{a}^{\text{lt}}$ is expressed in the same reference frame as $\boldsymbol{x}$, and $\dot{m}^{\text{lt}}$ is given in Equation (2.56). The general formulation of the low-thrust evolution function, Equation (2.59), is useful for adding low-thrust propulsion to an arbitrary dynamical model. This investigation is specifically concerned with the CR3BP and $\mathcal{N}$-body ephemeris models. The low-thrust-augmented CR3BP is given by combining Equations (2.27) and (2.59),

$$\begin{array}{c}\text{CR3BP w/}\\ \text{low-thrust}\end{array} \begin{cases} \ddot{x} - 2\dot{y} &= \Omega_x + a^{\text{lt}} u_x \\ \ddot{y} + 2\dot{x} &= \Omega_y + a^{\text{lt}} u_y \\ \ddot{z} &= \Omega_z + a^{\text{lt}} u_z \\ \dot{m} &= \dot{m}^{\text{lt}} \end{cases} \tag{2.60}$$

The low-thrust $\mathcal{N}$-body ephemeris model is similarly derived by adding low-thrust acceleration to the ballistic equations of motion in Equation (2.48),

$$\begin{array}{r}\mathcal{N}\text{-body w/}\\ \text{low-thrust}\end{array}\begin{cases}\ddot{\boldsymbol{R}}_{qi} = -G\dfrac{m_q}{R_{qi}^3}\boldsymbol{R}_{qi} + G\displaystyle\sum_{\substack{j=1\\ j\neq i,q}}^{\mathcal{N}} m_j\left(\dfrac{\boldsymbol{R}_{ij}}{R_{ij}^3} - \dfrac{\boldsymbol{R}_{qj}}{R_{qj}^3}\right) + a^{\text{lt}}\hat{u}_{\text{J2000}}\\ \\ \dot{m} \quad = \dot{m}^{\text{lt}}\end{cases} \tag{2.61}$$

The two low-thrust-augmented systems in Equations (2.60) and (2.61) form the basis of low-thrust simulations in this investigation.

3. NUMERICAL METHODS

This investigation focuses on dynamical models that lack a known analytical solution, i.e. the CR3BP and $\mathcal{N}$-body ephemeris force models. In these environments, numerical methods are frequently leveraged to simulate the dynamic and to construct trajectories. Numerical integration techniques are employed throughout this investigation to propagate paths through space that stem from an initial value problem, and to numerically represent the linear variational equations associated with the dynamical model. Furthermore, predictor-corrector targeting schemes are leveraged to construct and adjust continuous trajectories and low-thrust maneuver plans.

3.1 Numerical Integration

Numerical integration methods are leveraged in this investigation to produce a time history stemming from an initial value problem. The evolution parameters and functions for the dynamical models employed in this investigation are summarized in Section 7.3. In ballistic propagations, the evolution function $\boldsymbol{\phi}$, Equation (2.1), is defined by the equations of motions in either the CR3BP (Equation (2.27)) or $\mathcal{N}$-body ephemeris models (Equation (2.48)). For simulations that involve low-thrust propulsion, m is appended to either ${}^{\mathcal{R}}\boldsymbol{x}$ or ${}^{\mathcal{I}}\boldsymbol{x}$, depending on the simulated dynamical model, and the low-thrust evolution function is employed (Equation (2.59)). This investigation[1] leverages the Boost library in C++ to perform numerical integration (via `Boost.Numeric.Odeint`), and employs `pybind11`[2] to expose to C++ propagation methods to Python. In particular, a Runge-Kutta Fehlberg 78 integration method[3] is employed with absolute and relative error tolerances of 1×10^{-13}.

[1]↑Thank you to RJ Power and Brian McCarthy for your contributions to the numerical integration framework that enabled this research!
[2]↑https://pybind11.readthedocs.io/en/stable/
[3]↑https://live.boost.org/doc/libs/1_81_0/libs/numeric/odeint/doc/html/boost/numeric/odeint/runge_kutta_fehlberg78.html

3.2 Partial Derivative Propagation

Modeling the sensitivity of variables is an important component in many astrodynamics applications that leverage numerical methods, including trajectory design, state estimation, and other GN&C problems. In particular, representing the variation of the evolution function $\dot{\boldsymbol{\phi}} = \boldsymbol{\phi}\left(\boldsymbol{x}, t; \boldsymbol{\lambda}\right)$, Equation (2.1), provides an estimate for change in the dynamical model, and may be propagated to produce a time-history of derivative information. The evolution function depends on three variable that each possess sensitivity information. First, the linear variation of a downstream state with respect to changes in the evolution parameter (i.e. state) vector $\boldsymbol{x}_0$ is referred to as the State Transition Matrix (STM) and is a fundamental building block for many numerical methods in spaceflight. Next, variation with respect to time is directly available from the definition of the dynamical model, $\dot{\boldsymbol{\phi}} = {}^{d\phi}/_{dt}$. Finally, the first-order partial derivatives with respect to model parameters $\boldsymbol{\lambda}$ are represented similar to the State Transition Matrix (STM), and are useful in applications that seek change a characteristic of the dynamical model to satisfy an objective. Specifically, this investigation simulates the model parameter variational equations for epoch time quantities in the $\mathcal{N}$-body ephemeris force model and for low-thrust control variables. The evolution functions and parameters employed in this investigation are listed in Table A.1.

3.2.1 The State Transition Matrix

The STM is a matrix that describes how a state linearly evolves over time, and is an essential component of many astrodynamics and GN&C applications. Many numerical methods, including targeting schemes, exploit this linear approximation to the nonlinear dynamics. The STM quantifies how sensitivities evolve over time; that is, how small changes to an initial state affect downstream propagation. The STM is derived from the first-order variational equations. Recall that a dynamical system is defined in Equation (2.1), with the specific evolution parameters and functions employed in this investigation listed in Table A.1. As visualized in Figure 3.1, the isochronous variation $\delta\boldsymbol{x}$ is measured from nearby reference state $\boldsymbol{x}_{\text{ref}}$ as,

$$\boldsymbol{x}_{\text{ref}} + \delta\boldsymbol{x} = \boldsymbol{x} \tag{3.1}$$

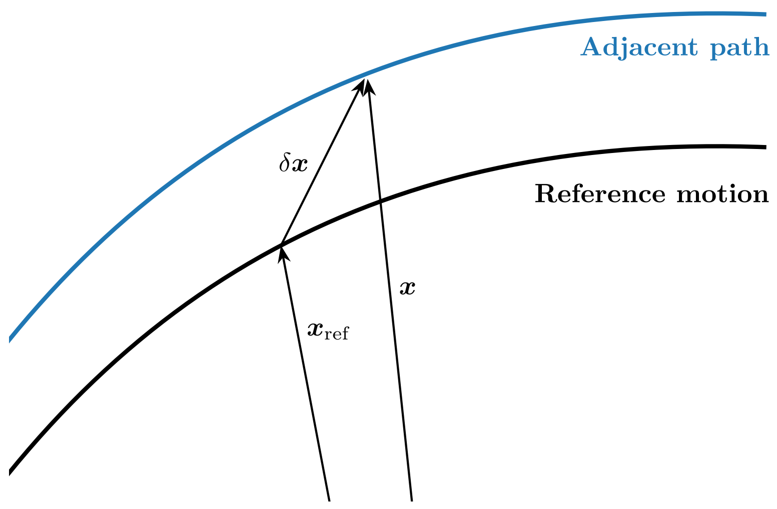


Figure 3.1. Vector diagram of variation in evolution parameter $\boldsymbol{x}$ with respect to a reference trajectory

This relationship is substituted into Equation (2.1) to yield the flow mapping,

$$\dot{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}_{\text{ref}} + \delta\boldsymbol{x}, t; \boldsymbol{\lambda}\right) \tag{3.2}$$

Where $\boldsymbol{\lambda}$ is the vector of model parameters that do not depend on time. A Taylor series expansion is applied to Equation (3.2) to derive the first-order effects relative to the reference trajectory. Noting that the time derivative of Equation (3.1) yields $\dot{\boldsymbol{x}}_{\text{ref}} + \delta\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}$, it follows that

$$\dot{\boldsymbol{x}}_{\text{ref}} + \delta\dot{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}_{\text{ref}}, t; \boldsymbol{\lambda}\right) + \left.\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}_{\text{ref}}} \underbrace{\left(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}\right)}_{=\delta\boldsymbol{x}} \quad + \text{H.O.T.s} \tag{3.3}$$

The STM is a linear mapping and, thus, Higher Order Terms (H.O.T.s) are truncated (including these effects yields a higher-order STM, though significant complexity and numerical computation is introduced by doing so [85]). By definition in Equation (2.1), $\dot{\boldsymbol{x}}_{\text{ref}} = \boldsymbol{\phi}\left(\boldsymbol{x}_{\text{ref}}, t\right)$; thus, terms cancel in Equation (3.3), leaving,

$$\delta\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\delta\boldsymbol{x} \ , \quad \text{where} \qquad \boldsymbol{A}(t) = \left.\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}_{\text{ref}}} \tag{3.4}$$

The Jacobian matrix $\boldsymbol{A}(t)$ consists of partial derivatives of the evolution functions with respect to the evolution parameters (i.e., the variation of the flow with respect to state variables).

The STM, $\boldsymbol{\Phi}(t, t_0)$, maps a linear variation over time from an initial state. This matrix quantity is defined as,

$$\boldsymbol{\Phi}(t, t_0) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} \in \mathbb{R}^{n_x \times n_x} \tag{3.5}$$

where n_x is the number of state variables in the evolution parameter vector $\boldsymbol{x}$. The STM provides a linear mapping of a variation over time,

$$\delta \boldsymbol{x}(t) = \boldsymbol{\Phi}(t, t_0) \delta \boldsymbol{x}(t_0) = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} \right) \delta \boldsymbol{x}(t_0) \tag{3.6}$$

Noting that $\boldsymbol{x}_0$ and t are independent, the first-order differential equation for ${}^{\partial \boldsymbol{x}}/_{\partial \boldsymbol{x}_0}$ is simply expressed as,

$$\dot{\boldsymbol{\Phi}}(t, t_0) = \frac{d}{dt} \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} \right) = \frac{d}{d\boldsymbol{x}_0} \left(\frac{d\boldsymbol{x}}{dt} \right) = \frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{x}_0} \tag{3.7}$$

From Equation (2.1), the value of $\dot{\boldsymbol{x}}$ is substituted, and a simple application of the chain rule results,

$$\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{x}_0} = \frac{d\boldsymbol{\phi}\left(\boldsymbol{x}, t; \boldsymbol{\lambda}\right)}{d\boldsymbol{x}_0} = \underbrace{\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} \right)}_{\boldsymbol{A}(t)} \underbrace{\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} \right)}_{\boldsymbol{\Phi}(t, t_0)} \tag{3.8}$$

Equations (3.7) and (3.8) are combined, yielding the expression,

$$\dot{\boldsymbol{\Phi}}(t, t_0) = \boldsymbol{A}(t) \boldsymbol{\Phi}(t, t_0) \tag{3.9}$$

This differential equation governs the evolution of the STM over time and may be numerically integrated along with the equations of motion by leveraging $\boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}^{n_x \times n_x}$ as the initial condition, where $\boldsymbol{I}^{n_x \times n_x}$ signifies the n_x-by-n_x-dimensional identity matrix. As an illustrative example, the STM for the CR3BP is written out in its entirety as,

$$\boldsymbol{\Phi}(t, t_0) = \begin{pmatrix} {}^{\partial x}/_{\partial x_0} & {}^{\partial x}/_{\partial y_0} & {}^{\partial x}/_{\partial z_0} & {}^{\partial x}/_{\partial \dot{x}_0} & {}^{\partial x}/_{\partial \dot{y}_0} & {}^{\partial x}/_{\partial \dot{z}_0} \\ {}^{\partial y}/_{\partial x_0} & {}^{\partial y}/_{\partial y_0} & {}^{\partial y}/_{\partial z_0} & {}^{\partial y}/_{\partial \dot{x}_0} & {}^{\partial y}/_{\partial \dot{y}_0} & {}^{\partial y}/_{\partial \dot{z}_0} \\ {}^{\partial z}/_{\partial x_0} & {}^{\partial z}/_{\partial y_0} & {}^{\partial z}/_{\partial z_0} & {}^{\partial z}/_{\partial \dot{x}_0} & {}^{\partial z}/_{\partial \dot{y}_0} & {}^{\partial z}/_{\partial \dot{z}_0} \\ {}^{\partial \dot{x}}/_{\partial x_0} & {}^{\partial \dot{x}}/_{\partial y_0} & {}^{\partial \dot{x}}/_{\partial z_0} & {}^{\partial \dot{x}}/_{\partial \dot{x}_0} & {}^{\partial \dot{x}}/_{\partial \dot{y}_0} & {}^{\partial \dot{x}}/_{\partial \dot{z}_0} \\ {}^{\partial \dot{y}}/_{\partial x_0} & {}^{\partial \dot{y}}/_{\partial y_0} & {}^{\partial \dot{y}}/_{\partial z_0} & {}^{\partial \dot{y}}/_{\partial \dot{x}_0} & {}^{\partial \dot{y}}/_{\partial \dot{y}_0} & {}^{\partial \dot{y}}/_{\partial \dot{z}_0} \\ {}^{\partial \dot{z}}/_{\partial x_0} & {}^{\partial \dot{z}}/_{\partial y_0} & {}^{\partial \dot{z}}/_{\partial z_0} & {}^{\partial \dot{z}}/_{\partial \dot{x}_0} & {}^{\partial \dot{z}}/_{\partial \dot{y}_0} & {}^{\partial \dot{z}}/_{\partial \dot{z}_0} \end{pmatrix} \in \mathbb{R}^{6 \times 6} \tag{3.10}$$

The partial derivative values that construct the $\boldsymbol{A}(t)$ matrix in the CR3BP and $\mathcal{N}$-body ephemeris models are derived by differentiating the evolution functions with respect to the evolution parameters; these partial derivative values are helpfully provided by Zimovan [86], Sections 3.1.1 and 3.1.2.

3.2.2 Model Parameter Sensitivity

It is frequently useful to incorporate model-specific parameters in targeting problems. As defined in Equation (2.1), the first-order vector equation of motion that governs a dynamical model is $\dot{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}, t; \boldsymbol{\lambda}\right)$, where $\boldsymbol{\lambda}$ is a vector that consists of n_λ model parameters. The STM provides a linear sensitivity mapping for changes in $\boldsymbol{x}_0$ and is propagated using the differential equation in Equation (3.9). While the STM measures downstream variation with respect to initial state variables, is similarly useful to understand the sensitivity due to changes in a model parameter; a collection of parameters associated with the dynamical models employed in this investigation are listed in Table 3.1. Throughout this investigation, low-thrust control variables are fixed over any integration segment, as detailed in Section 2.4. Since these values do not vary with time, the low-thrust orientation and magnitude values are implemented as model parameters in this research.

Table 3.1. Model parameters available in this investigation

CR3BP	**$\mathcal{N}$-body ephemeris**	**Low-thrust**
μ: mass ratio	τ: epoch time	f: thrust magnitude
		f_{max}: maximum thrust magnitude
		$\hat{a}^{\text{lt}}$: thrust direction
		$\tilde{I}_{\text{sp}}$: specific impulse

The derivation for the differential equation that governs the evolution of model parameter sensitivity over time is derived similarly to the STM, Equation (3.7). This first-order differential equation is a mathematical representation of the sensitivity of downstream state variables with respect to model parameters, i.e., $\partial\boldsymbol{x}/\partial\boldsymbol{\lambda}$. Observing that $\boldsymbol{\lambda}$ does not depend on time, the derivative is evaluated as,

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\lambda}}\right) = \frac{d}{d\boldsymbol{\lambda}}\left(\frac{d\boldsymbol{x}}{dt}\right) = \frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{\lambda}} \tag{3.11}$$

The time-invariance assumption in for $\boldsymbol{\lambda}$ in this representation is valid for the low-thrust control variables in this investigation, listed in Table 3.1, but would be violated for time-varying control laws. By combining Equations (2.1) and (3.11), and applying the chain rule, the vector equation becomes,

$$\frac{d\dot{\boldsymbol{x}}}{d\boldsymbol{\lambda}} = \frac{d\boldsymbol{\phi}\left(\boldsymbol{x}, t; \boldsymbol{\lambda}\right)}{d\boldsymbol{\lambda}} = \underbrace{\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}\right)}_{\boldsymbol{A}(t)}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\lambda}}\right) + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\lambda}} \tag{3.12}$$

where $\partial\boldsymbol{\phi}/\partial\boldsymbol{x}$ is Jacobian matrix defined in Equation (3.4). Hence, the first-order differential equation that describes the evolution of $\partial\boldsymbol{x}/\partial\boldsymbol{\lambda}$ over time is,

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\lambda}}\right) = \boldsymbol{A}(t)\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\lambda}}\right) + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\lambda}} \tag{3.13}$$

A variation in model parameter does not affect initial state; therefore,

$$\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{\lambda}} = \boldsymbol{0}^{n_x \times n_\lambda} \tag{3.14}$$

serves as the initial condition for numerically integrating Equation (3.13), where n_x is the number of variables contained in the evolution parameter vector $\boldsymbol{x}$. An excellent derivation and discussion for this process of obtaining general derivative information is provided by Pavlak [87].

3.3 The Targeting Method

Differential corrections (also referred to as *targeting* or *shooting*) is a common numerical technique in astrodynamics for constructing feasible trajectories. Targeting is formulated as a multidimensional generalization of the Newton-Raphson root finding method [88]. At a high-level, this algorithm seeks to leverage partial derivative information to adjust a set of design variables to satisfy a set of constraints. Various options exist for implementing a corrections process; this investigation employs a direct multiple shooting strategy that varies a column vector of $\mathbb{n}$ design variables,

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{\mathbb{n}} \end{bmatrix} \in \mathbb{R}^{\mathbb{n}} \tag{3.15}$$

where each X_i is a scalar value that is free to vary. The targeting process seeks to iteratively identify a vector $\boldsymbol{X}^*$ that is a solution to the set of $\mathbb{m}$ scalar constraints,

$$\boldsymbol{F}(\boldsymbol{X}^*) = \begin{bmatrix} F_1(\boldsymbol{X}^*) \\ F_2(\boldsymbol{X}^*) \\ \vdots \\ F_{\mathbb{m}}(\boldsymbol{X}^*) \end{bmatrix} = \boldsymbol{0}^{\mathbb{m}} \tag{3.16}$$

This "Free-Variable and Constraint" approach to targeting is a powerful way to generically define a corrections process in trajectory design applications. The selection of free-variables is problem-dependent, and common options include state, time, and control values.

A targeting algorithm requires a sufficiently accurate startup solution $\boldsymbol{X}_0$ to iteratively solve for $\boldsymbol{X}^*$. Stemming from the initial guess, a Taylor series for Equation (3.16) is expanded about $\boldsymbol{X}_j$ at each stage of the targeting process to achieve a first-order approximation of the constraint function at the next iteration, $\boldsymbol{F}(\boldsymbol{X}_{j+1})$,

$$\boldsymbol{F}(\boldsymbol{X}_{j+1}) = \boldsymbol{F}(\boldsymbol{X}_j) + D\boldsymbol{F}(\boldsymbol{X}_j)\,(\boldsymbol{X}_{j+1} - \boldsymbol{X}_j) + \underbrace{\cdots}_{\text{H.O.T.s}} = \boldsymbol{0} \tag{3.17}$$

the higher order terms are truncated, resulting in an iterative process that approaches $\boldsymbol{X}^*$. Furthermore, in Equation (3.17), $D\boldsymbol{F}(\boldsymbol{X})$ (evaluated at $\boldsymbol{X} = \boldsymbol{X}_j$) represents a matrix of partial derivatives of the constraint vector with respect to free variables,

$$D\boldsymbol{F}(\boldsymbol{X}) = \begin{bmatrix} {}^{\partial F_1}\!/_{\partial X_1} & {}^{\partial F_1}\!/_{\partial X_2} & \cdots & {}^{\partial F_1}\!/_{\partial X_{\mathbb{n}}} \\ {}^{\partial F_2}\!/_{\partial X_1} & {}^{\partial F_2}\!/_{\partial X_2} & \cdots & {}^{\partial F_2}\!/_{\partial X_{\mathbb{n}}} \\ \vdots & \vdots & \ddots & \vdots \\ {}^{\partial F_{\mathbb{m}}}\!/_{\partial X_1} & {}^{\partial F_{\mathbb{m}}}\!/_{\partial X_2} & \cdots & {}^{\partial F_{\mathbb{m}}}\!/_{\partial X_{\mathbb{n}}} \end{bmatrix} \in \mathbb{R}^{\mathbb{m}\times\mathbb{n}} \tag{3.18}$$

These partial derivative values are typically computed analytically, though numerical differentiation methods are available as well including central differencing, complex step differentiation, and automatic differentiation methods. In trajectory design applications, STM elements (e.g. Equation (3.10)) frequently arise as components of $D\boldsymbol{F}(\boldsymbol{X})$ to model first-order state variations.

Multiple strategies exist to compute the update from $\boldsymbol{X}_j \rightarrow \boldsymbol{X}_{j+1}$ that satisfies Equation (3.17), i.e.,

$$\boldsymbol{F}(\boldsymbol{X}_j) + D\boldsymbol{F}(\boldsymbol{X}_j)\left(\boldsymbol{X}_{j+1} - \boldsymbol{X}_j\right) = \boldsymbol{0} \tag{3.19}$$

In the simplest case, when a targeting problem involves the same number of free variables and constraints ($\mathbb{n} = \mathbb{m}$), a multidimensional Newton-Raphson scheme is directly implemented by solving Equation (3.19), i.e.,

$$\boldsymbol{X}_{j+1} = \boldsymbol{X}_j - D\boldsymbol{F}(\boldsymbol{X}_j)^{-1}\boldsymbol{F}(\boldsymbol{X}_j) \tag{3.20}$$

where $D\boldsymbol{F}(\boldsymbol{X}_j)$ is a full-rank (i.e. invertible) matrix. Beginning with $\boldsymbol{X}_0$, Equation (3.20) is iteratively applied until the $\boldsymbol{X}^*$ is found (such that$\boldsymbol{F}(\boldsymbol{X}^*) = \boldsymbol{0}$). As with any numerical process, machine precision and other numerical errors prevent Equation (3.16) from being perfectly satisfied. Hence, a common approach is to iterate until a pre-selected tolerance is reached, i.e.,

$$\boldsymbol{X}_0 \rightarrow \boldsymbol{X}_1 \rightarrow \cdots \rightarrow \boldsymbol{X}^* \;, \qquad \text{where} \qquad |\boldsymbol{F}(\boldsymbol{X}^*)| < \epsilon \tag{3.21}$$

This investigation leverages an L2 norm and typical employs $\epsilon = 1 \times 10^{-12}$ for the error threshold.

Targeting problems that posses more free variables than constraints, $\mathtt{n} > \mathtt{m}$ frequently arise in trajectory design application, in particular when low-thrust control variables are included. In these problems, an $(\mathtt{n} - \mathtt{m})$-dimensional null space implies that $\boldsymbol{X}^*$ is not unique; therefore, infinite solutions exist to the update equation Equation (3.19). A common approach when $\mathtt{n} > \mathtt{m}$ is to seek the solution that is closest to the initial guess. The 'minimum norm update' is a well-known solution to this optimization problem, and is obtained by computing the *psuedo-inverse* $D\boldsymbol{F}(\boldsymbol{X}_j)^+$,

$$D\boldsymbol{F}(\boldsymbol{X}_j)^+ = D\boldsymbol{F}(\boldsymbol{X}_j)^T \left(D\boldsymbol{F}(\boldsymbol{X}_j) D\boldsymbol{F}(\boldsymbol{X}_j)^T\right)^{-1} \tag{3.22}$$

This generalization of the matrix inverse is also referred to as the *MoorePenrose inverse*, and assumes that the rows of $D\boldsymbol{F}(\boldsymbol{X}_j)$ are linearly independent. The pseudo-inverse similarly arises in over-constrained targeting problems, $\mathtt{n} < \mathtt{m}$, resulting in a least-squares regression problem. A Singular Value Decomposition (SVD) is often employed when numerically implementing the pseudo-inverse to prevent small singular values from destabilizing the inversion process. When employed in targeting, the minimum-norm update is an orthogonal projection of the design variable vector onto the solution space,

$$\boldsymbol{X}_{j+1} = \boldsymbol{X}_j - D\boldsymbol{F}(\boldsymbol{X}_j)^+ \boldsymbol{F}(\boldsymbol{X}_j) \tag{3.23}$$

which minimizes the difference between $\boldsymbol{X}_j$ and $\boldsymbol{X}_{j+1}$ at each iteration Targeting applications in this investigation typically possess many more free variables than constraints; hence, Equation (3.23) is iterated on until $|\boldsymbol{F}(\boldsymbol{X}^*)| < \epsilon$ is satisfied.

Two common differential corrections algorithms in astrodynamics are termed 'single' and 'multiple' shooting methods. As the names imply, single shooting involves iterating on one trajectory to satisfy a set of constraints, while multiple shooting simultaneously accommodates numerous sub-problems consisting of various segments that comprise an overall trajectory or mission plan. While multiple shooting methods are more conceptually and computationally complex, they allow for applications that include different types of arcs (e.g., ballistic and low-thrust), are often more numerically stable, and expose a broader range of applications to the targeting formulation; an illustrative discussion of the merits of multiple shooting is detailed by Spreen [85]. This investigation leverages a multiple shooting strategy to simultaneously accommodate low-thrust and ballistic arc across a maneuver plan, correct discontinuities, and ensure mission criteria are satisfied.

3.3.1 Inequality Constraints

It is often useful to incorporate inequality conditions in a targeting process. However, the constraint vector formulation presented in Equation (3.16) is constructed as a series of scalar equality constraints. Consider a free variable X_i such that,

$$X_i \in [X_{i,\min}, X_{i,\max}] \tag{3.24}$$

In this case, the constraint/free-variable targeting method is adapted to incorporate the two inequality conditions as separate constraints. In both cases a *slack* variable η_i is introduced (here 'slack' is in reference to η 'picking up the slack' between inequality and equality conditions). The first inequality in Equation (3.24), $X_i \geq X_{i,\min}$, necessarily implies that,

$$X_i - X_{i,\min} \geq 0 \tag{3.25}$$

Furthermore, for any positive function, it must be true that there exists $\eta_i \in \mathbb{R}$ such that

$$X_i - X_{i,\min} = \eta_i^2 \tag{3.26}$$

is satisfied regardless of the sign of η_i. This relationship is readily transformed into an equality constraint,

$$F(\boldsymbol{X}) = X_i - X_{i,\min} - \eta_i^2 = 0 \tag{3.27}$$

The slack variable $\eta_i \in \boldsymbol{X}$ is included as a design variable in a targeting problem with the corresponding equality constraint subsequently appended to the constraint vector in Equation (3.16), $F(\boldsymbol{X}) \in \boldsymbol{F}(\boldsymbol{X})$. The initial value of the slack variable is determined by solving Equation (3.26) at the onset of targeting given $\boldsymbol{X}_0$. The upper-bound inequality constraint in Equation (3.24) is similarly derived, resulting in the equality condition,

$$F(\boldsymbol{X}) = X_i - X_{i,\max} + \eta_i^2 = 0 \tag{3.28}$$

The partial derivatives for X_i that construct the $D\boldsymbol{F}(\boldsymbol{X})$ matrix are trivially computed from taking the derivative of Equation (3.27) or Equation (3.28) with respect to X_i. The slack variable formulation for inequality constraints is summarized in Table 3.2. While slack variables are a useful tool in trajectory design problems, occasions arise where a poor initial guess prohibits the targeting process from finding a basin of convergence. In such instances, an alternative parameterization method that implicitly enforces inequality conditions, detailed in Section 3.3.2, may be helpful in overcoming numerical challenges in the targeting process.

Table 3.2. Slack variable equality constraint formulation

Inequality	Equality Constraint	Initial Condition	Partial
$X_i^* \geq X_{i,\text{min}}$	$X_i^* - X_{i,\text{min}} - (\eta_i^*)^2 = 0$	$\eta_{i,0} = \sqrt{X_{i,0} - X_{i,\text{min}}}$	${}^{\partial F_i(\boldsymbol{X})}/_{\partial \eta_i} = -2\eta_i$
$X_i^* \leq X_{i,\text{max}}$	$X_i^* - X_{i,\text{max}} + (\eta_i^*)^2 = 0$	$\eta_{i,0} = \sqrt{X_{i,\text{max}} - X_{i,0}}$	${}^{\partial F_i(\boldsymbol{X})}/_{\partial \eta_i} = 2\eta_i$

3.3.2 Variable Parameterization

It is frequently useful to leverage variable transformations in targeting problems to implicitly enforce bounds and adjust scaling. Scaling is useful in corrections processes that involve free variables on differing orders of magnitude. Similar to the motivation for nondimensionalization in the dynamical model and numerical integration, rounding and truncation errors may be addressed by ensuring all variables are on the order of 1. Furthermore, several quantities relevant to this research possess lower or upper bounds that reflect a physical limit. Specifically, time quantities must often remain positive, and thrust magnitude is bounded by zero and the maximum thrust of the modeled engine, Equation (2.51). Note that negative thrust magnitude values occasionally appear in the literature with the assumption of constant mass $\dot{m}^{\text{lt}} = 0$; a negative value for f in this investigation falsely implies the spacecraft gains mass over time.

Given a variable of interest w that possesses scaling or bounding challenges in a targeting problem, the parameterization process is implemented by defining a variable transformation

$$w = \Pi(\breve{w}) \tag{3.29}$$

where $\breve{w}$ is the parameterized value and Π is a function that transforms $\breve{w}$ to the original variable w. Given a selected parameterization function, $\breve{w}$ replaces w as a free variable in the targeting process. Several options employed in this investigation for $\Pi(\breve{w})$[4] are visualized in Figure 3.2. In each case, the x-axis corresponds to the unbounded parameterized values ($\breve{w}$), with the y-axis reflecting the original variable of interest (w). In Figure 3.2(a), the w variable possesses a large order of magnitude that may introduce numerical error to a targeting algorithm. A simple scaling function is applied in this case to ensure that $\breve{w}$ is on an order of magnitude that is consistent with other variables in the problem (the selection of the scaling factor's value is, of course, problem dependent). The square function, Figure 3.2(b), is useful for variables (such as time) that do not admit negative values, but are otherwise unbounded. Finally, Figure 3.2(c) depicts the function,

$$\Pi(\breve{w}) = \frac{1}{2}\left(\sin(\breve{w}) + 1\right) \in [0, 1] \tag{3.30}$$

Equation (3.30) is useful for implicitly enforcing lower and upper bounds, and is employed in this investigation for thrust magnitude values [89][5]. Scaling is often necessary as an additional step for Equation (3.30) to transform the function into the appropriate bounds for the given problem.

Implementing parameterized values in a differential corrections process requires a continuously differentiable function Π. The gradient information of the parameterized value is leveraged in the targeting process, and is easily computed as

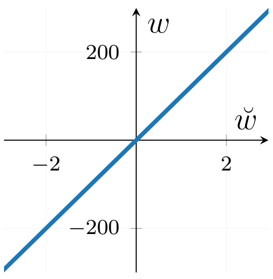
$$\frac{\partial \boldsymbol{f}(w)}{\partial \breve{w}} = \left(\frac{\partial \boldsymbol{f}(w)}{\partial w}\right)\left(\frac{\partial \Pi(\breve{w})}{\partial \breve{w}}\right) \tag{3.31}$$

where $\boldsymbol{f}$ is an arbitrary function of w. The derivatives of the suggested parameterization functions are plotted in Figures 3.2(d) to 3.2(f). In considering parameterization options, the gradient information is an important consideration in identifying critical values that lead to vanishing partial derivative values that may prevent updates in a targeting method (this phenomenon is related to the vanishing gradient problem in neural networks). Similarly,

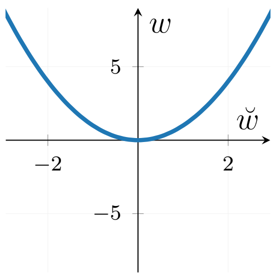
[4]↑Thank you to Andrew Cox and Emily Zimovan-Spreen for your helpful investigations and insight into low-thrust parameterization options!

[5]↑This parameterization option was introduced by Andrew Cox through his low-thrust research [72]

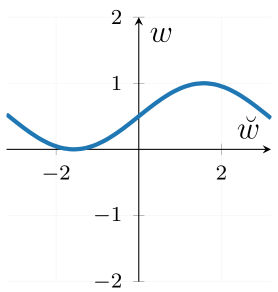
parameterization options that contain large derivative values may destabilize a gradient-based iterative method.



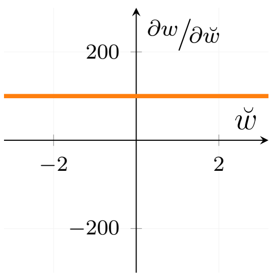
(a) $\Pi(\breve{w}) = 100\breve{w}$



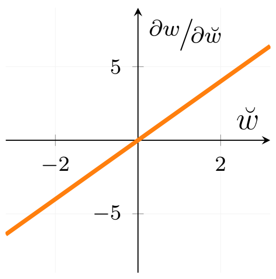
(b) $\Pi(\breve{w}) = \breve{w}^2$



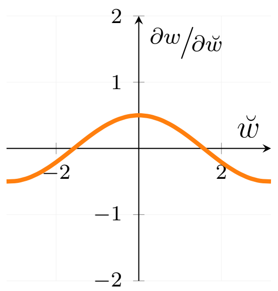
(c) $\Pi(\breve{w}) = \frac{1}{2}\left(\sin(\breve{w}) + 1\right)$



(d) ${}^{\partial \Pi(\breve{w})}/_{\partial \breve{w}} = 100$



(e) ${}^{\partial \Pi(\breve{w})}/_{\partial \breve{w}} = 2\breve{w}$



(f) ${}^{\partial \Pi(\breve{w})}/_{\partial \breve{w}} = \frac{1}{2}\cos(\breve{w})$

Figure 3.2. Parameterization options $w = \Pi(\breve{w})$ (blue) employed throughout this investigation, plotted alongside the associated partial derivatives ${}^{\partial \Pi(\breve{w})}/_{\partial \breve{w}}$ (orange).

This investigation parameterizes time and thrust magnitude variables in differential equations applications. The two parameterization methods are defined as,

$$t = \left(\breve{t}\right)^2 \qquad \leftrightarrow \qquad \breve{t} = \sqrt{t} \tag{3.32}$$

$$f = \frac{f_{\max}}{2}\left(\sin(\breve{f}) + 1\right) \qquad \leftrightarrow \qquad \breve{f} = \arcsin\left(\frac{2f}{f_{\max}} - 1\right) \tag{3.33}$$

where $t \in \mathbb{R}_{\geq 0}$ is nondimensional time quantity, and $f \in [0, f_{\max}]$ is the nondimensional thrust magnitude of the SEP engine, defined in Equation (2.51). A potential benefit, or drawback, of the thrust parameterization approach is that when f lies precisely on either bound, $f = \{0, f_{\max}\}$, the gradient of $\breve{f}$ vanishes. As visualized in Figure 3.2(f), the zero-gradient condition ${}^{\Pi(\breve{w})}/_{\partial \breve{w}} = 0$ occurs when w is a minimum or maximum value. While this may be useful in situations that seek to maintain a thrust magnitude at these critical values, it may cause unexpected results if the user is expecting the value to be updated during the targeting process.

3.4 Constructing Continuous Low-Thrust Trajectories

The first step in implementing a targeting method is to identify: 1) the constraints that the corresponding solution must adhere to, and 2) the variables that are free to change in the targeting process. A common approach for ballistic motion is to divide a trajectory into segments that are each allowed to vary, and employ a multiple shooting strategy to enforce continuity between successive arcs. Additional flexibility is often introduced by allowing time along the trajectory to vary, though variable time implementations may be disadvantageous in problems that require pre-specified timings.

As a simple illustrative example of targeting ballistic motion, visualized in Figure 3.3, consider segment (i) of a multiple shooting problem, where $\boldsymbol{x}_i$ represents the initial state (referred to as a *patch point*) that is propagated for t_i nondimensional time units, resulting in the final state $\boldsymbol{x}_{i,f}$. To enforce state continuity, the corresponding constraint from segment $(i) \rightarrow (i+1)$ is simply,

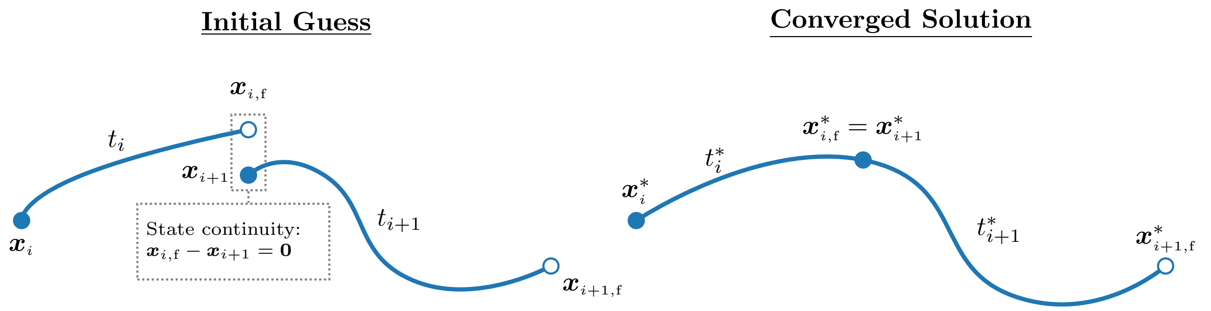
$$F_{i\rightarrow i+1}(\boldsymbol{X}) = \boldsymbol{x}_{i,f} - \boldsymbol{x}_{i+1} = \boldsymbol{0} \ , \quad \text{where} \qquad \{\boldsymbol{x}_i, t_i, \boldsymbol{x}_{i+1}\} \in \boldsymbol{X} \tag{3.34}$$

Of course, additional constraints may be introduced on patch point variables, specifying the Jacobi constant value, Equation (2.32), or a distance from a celestial body. Recall that the partial derivatives of constraints with respect to free variables constitute the $D\boldsymbol{F}(\boldsymbol{X})$ matrix, defined for the CR3BP in Equation (3.18). In this simple, the partial derivative values for the continuity constraint in Equation (3.34) are,

$$\frac{\partial F_{i\rightarrow i+1}}{\partial \boldsymbol{x}_i} = \boldsymbol{\Phi}(t_i, 0) \qquad \frac{\partial F_{i\rightarrow i+1}}{\partial t_i} = \dot{\boldsymbol{x}}_{i,f} \qquad \frac{\partial F_{i\rightarrow i+1}}{\partial \boldsymbol{x}_{i+1}} = -\boldsymbol{I}^{n_x \times n_x} \tag{3.35}$$

where $\boldsymbol{\Phi}(t_i, 0)$ is the STM evaluated at the end of segment (i). As discussed in Section 3.3.2, it is frequently necessary to ensure time variables t_i remain positive. Two techniques are available to enforce this inequality condition: 1) replace t_i with a parameterized time variable

$\breve{t}_i$ in the design variable vector, as defined in Equation (3.32), or 2) introduce a slack variable inequality constraint, as detailed in Equation (3.27).



Initial Guess
$\boldsymbol{x}_{i,\mathrm{f}}$
t_i
$\boldsymbol{x}_{i+1}$
$\boldsymbol{x}_i$
State continuity:
$\boldsymbol{x}_{i,\mathrm{f}} - \boldsymbol{x}_{i+1} = \boldsymbol{0}$
t_{i+1}
$\boldsymbol{x}_{i+1,\mathrm{f}}$
Converged Solution
$\boldsymbol{x}^*_{i,\mathrm{f}} = \boldsymbol{x}^*_{i+1}$
t^*_i
$\boldsymbol{x}^*_i$
t^*_{i+1}
$\boldsymbol{x}^*_{i+1,\mathrm{f}}$

Figure 3.3. State continuity targeting schematic

Including an engine model in the targeting process provides control authority for a spacecraft to alter its trajectory. These maneuvers are modeled differently in targeting problems for chemical and low-thrust propulsion options. For chemical engines, simply removing the velocity components of Equation (3.34) enables an instantaneous ΔV maneuver at patch point $\boldsymbol{x}_{i+1}$. However, low-thrust targeting applications require continuity in position, velocity, and mass components, thus creating a more challenging problem. Hence, the low-thrust targeting process alters thrust command variables to achieve state continuity. Throughout this investigation, each low-thrust propagation segment is defined by magnitude and direction coordinates that are typically included as a free variable, and problem-specific numerical challenges are addressed via parameter selection, parameterization, and slack variable inequality constraints. The targeting process for each thrust coordinate is summarized as:

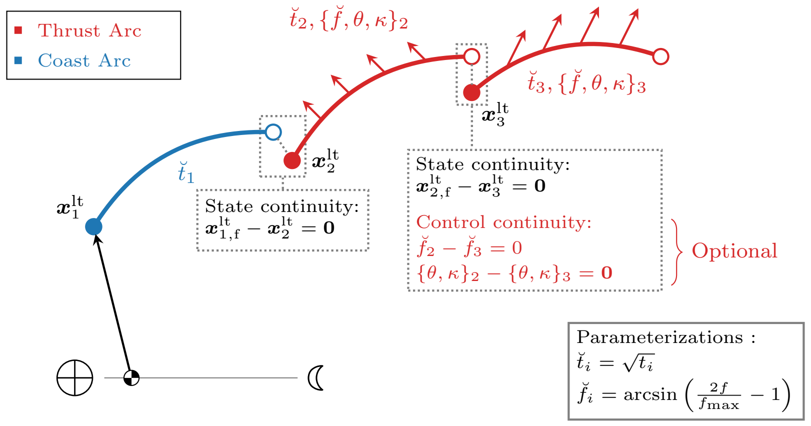
- Thrust Magnitude The parameterized nondimensional thrust magnitude $\breve{f}$, specified in Equation (3.33), is employed as a design variable in this investigation's targeting problems. Targeting f directly requires two additional inequality constraints, defined generically in Table 3.2, that may narrow the corrector's basin of convergence. Preliminary experiments indicate that $\breve{f} \in \boldsymbol{X}$ produces a more numerically stable process than $f \in \boldsymbol{X}$; however future work is required to precisely identify which types of targeting problems benefit from low-thrust magnitude parameterization options.

- Thrust Direction This investigation's targeting applications in the CR3BP typically employ spherical coordinates (Equation (2.54)) to represent thrust direction. It is convenient to implement angle variables (in radians) in targeting problems as they are unbounded; however scaling issues arise if the angle values change dramatically through the targeting process. Furthermore, when thrust direction continuity is enforced, spherical coordinates may prohibit convergences in cases where an angle differs by $\pm 2n\pi, n \in \mathbb{N}$. Alternatively, the three components of an orientation vector may be included as design variables $\hat{u} \in \boldsymbol{X}$ with an additional constraint to ensure $\hat{u}$ remains a unit vector, $|\hat{u}| = 1$.
- Thrust Time When thrust time is allowed to vary in this application, positivity conditions are enforced by either slack variable inequality constraints or by leveraging a variable parameterization. Time along low-thrust arcs is generally treated the same as time along coasting segments.

It is often necessary to intermix thrust and coast arcs depending on the objectives of a particular targeting application. To illustrate a general low-thrust targeting process, a high-level schematic of a low-thrust multiple shooting problem in the CR3BP is visualized in Figure 3.4. This illustrative problem consists of one coasting segment (blue) followed by two thrusting segments (red), where filled circles represent patch points and unfilled circles correspond to the final state of each arc. Continuity constraints are applied at the boundary between successive segments. In some targeting applications, control variable continuity is enforced; however, this condition is problem-dependent and is therefore deemed 'optional'. Solutions to targeting problems are highly-dependent on an accurate startup solution, and it is frequently challenging to predict an effective control profile in low-thrust applications. This research effort seeks to address this challenge by employing a neural network estimator to initialize low-thrust targeting processes.

3.4.1 Sensitivity propagation

As described in Section 3.3, applications of Newton's method requires derivative information to compute updates that iteratively converge on a solution. This investigation propagates thrust sensitivities by treating low-thrust control variables as model parameters



■ Thrust Arc
■ Coast Arc
$\breve{t}_2, \{\breve{f}, \theta, \kappa\}_2$
$\breve{t}_3, \{\breve{f}, \theta, \kappa\}_3$
$\boldsymbol{x}_3^{\mathrm{lt}}$
$\boldsymbol{x}_2^{\mathrm{lt}}$
$\boldsymbol{x}_1^{\mathrm{lt}}$
$\breve{t}_1$
State continuity:
$\boldsymbol{x}_{1,\mathrm{f}}^{\mathrm{lt}} - \boldsymbol{x}_2^{\mathrm{lt}} = \mathbf{0}$
State continuity:
$\boldsymbol{x}_{2,\mathrm{f}}^{\mathrm{lt}} - \boldsymbol{x}_3^{\mathrm{lt}} = \mathbf{0}$
Control continuity:
$\breve{f}_2 - \breve{f}_3 = 0$
$\{\theta, \kappa\}_2 - \{\theta, \kappa\}_3 = \mathbf{0}$
Optional
Parameterizations :
$\breve{t}_i = \sqrt{t_i}$
$\breve{f}_i = \arcsin\left(\frac{2f}{f_{\max}} - 1\right)$

Figure 3.4. Variable schematic for a low-thrust multiple shooting method in the CR3BP. Parameterization options detailed in Equations (3.32) and (3.33). Graphic adapted from Bosanac, Ref [90], p.181, Fig. 7.5.

(introduced in Section 3.2.2). The time-independence property of the employed control law in this investigation allows the state sensitivities with respect to control variables ($\partial \boldsymbol{x}^{\mathrm{lt}} / \partial \boldsymbol{\lambda}^{\mathrm{lt}}$) to be propagated using Equation (3.13), where $\boldsymbol{\lambda}^{\mathrm{lt}} = \{\breve{f},\ \hat{a}^{\mathrm{lt}}\}$ is the vector of model parameters and $\boldsymbol{x}^{\mathrm{lt}}$ is a seven-dimensional low-thrust state vector that includes position, velocity, and mass coordinates. Furthermore, the parameterized low-thrust magnitude $\breve{f}$, defined in Equation (3.33), is leveraged to bound the values of $f \in [0, f_{\max}]$ during the targeting process. Hence, as defined in Equation (3.31), the sensitivities are propagated using the partial derivatives of the flow equation, Equation (2.59),

$$\frac{\partial \boldsymbol{\phi}}{\partial f} = \begin{bmatrix} \mathbf{0}^{3\times1} \\ \frac{1}{m}\hat{a}^{\mathrm{lt}} \\ -\frac{1}{\tilde{v}_{\mathrm{e}}}\left(\frac{\tilde{l}^*}{\tilde{t}^*}\right) \end{bmatrix} \in \mathbb{R}^{7\times1} \quad \rightarrow \quad \frac{\partial \boldsymbol{\phi}}{\partial \breve{f}} = \left(\frac{\partial \boldsymbol{\phi}}{\partial f}\right)\left(\frac{\partial f}{\partial \breve{f}}\right) = \begin{bmatrix} \mathbf{0}^{3\times1} \\ \left(\frac{\cos(\breve{f})}{2m}\right)\hat{a}^{\mathrm{lt}} \\ -\frac{\cos(\breve{f})}{2\tilde{v}_{\mathrm{e}}}\left(\frac{\tilde{l}^*}{\tilde{t}^*}\right) \end{bmatrix} \in \mathbb{R}^{7\times1} \tag{3.36}$$

Similarly, the CR3BP control direction is equivalently represented in either Cartesian ($\hat{u}$) or spherical coordinates ($\{\theta, \kappa\}$). As a unit vector, the sensitivity of the evolution function is,

$$\frac{\partial \boldsymbol{\phi}}{\partial \hat{u}} = \begin{bmatrix} \mathbf{0}^{3\times3} \\ \left(\frac{f}{m}\right)\boldsymbol{I}^{3\times3} \\ \mathbf{0}^{1\times3} \end{bmatrix} \in \mathbb{R}^{7\times3} \tag{3.37}$$

Equation (3.37) describes the sensitivity of a downstream state with respect to a change in a unit vector thrust direction, which is readily transformed into spherical coordinates,

$$\frac{\partial \boldsymbol{\phi}}{\partial \theta} = \left(\frac{\partial \boldsymbol{\phi}}{\partial \hat{u}}\right)\left(\frac{\partial \hat{u}}{\partial \theta}\right) = \begin{bmatrix} \mathbf{0}^{3\times3} \\ -\left(\frac{f}{m}\right)\sin\theta\sin\kappa \\ \left(\frac{f}{m}\right)\cos\theta\sin\kappa \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{7\times1} \tag{3.38}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial \kappa} = \left(\frac{\partial \boldsymbol{\phi}}{\partial \hat{u}}\right)\left(\frac{\partial \hat{u}}{\partial \kappa}\right) = \begin{bmatrix} \mathbf{0}^{3\times3} \\ \left(\frac{f}{m}\right)\cos\theta\cos\kappa \\ \left(\frac{f}{m}\right)\sin\theta\cos\kappa \\ -\left(\frac{f}{m}\right)\sin\kappa \\ 0 \end{bmatrix} \in \mathbb{R}^{7\times1} \tag{3.39}$$

These low-thrust control sensitivity equations are integrated alongside the equations of motion of the underlying low-thrust-augmented dynamical model under the first-order vector differential equation defined in Equation (3.13).

3.4.2 Thrust Magnitude Adjustment

Many low-thrust engines are not physically able to produce thrust values below a certain magnitude, f_{engine}. Hence, a more realistic low-thrust model enforces the condition,

$$f_{\text{engine}} \leq f \leq f_{\max} \tag{3.40}$$

where $f_{\max}$ is the maximum possible thrust value defined in Equation (2.51). The precise value of the lower bound f_{engine} varies greatly from engine-to-engine. For example Hayabusa 1 and 2 are only capable of maneuvers above 58 % and 70 % of $f_{\max}$, respectively [74]. In scenarios where an initial guess for thrust magnitude falls below f_{engine} in a targeting problem, it can be useful to leverage the relationship between thrust time and magnitude to compute a new guess within feasible bounds. Several thrust-scaling heuristics that enable initial guess adjustment are employed in this investigation; the relationships derived in this section between time, mass, and equivalent $\Delta\tilde{V}$ are summarized in Table 3.3.

Table 3.3. Analytical expressing for transforming low-thrust segments

Given known...		Estimate...		Using equation...
Thrust time	$\mathbb{t}$	Change in mass	Δm	Equation (3.43)
Change in mass	Δm	Thrust time	$\mathbb{t}$	Equation (3.44)
Change in velocity	$\Delta\tilde{V}_{\text{equiv.}}$	Thrust time	$\mathbb{t}$	Equation (3.47)
Thrust time	$\mathbb{t}$	Change in velocity	$\Delta\tilde{V}_{\text{equiv.}}$	Equation (3.48)

To compute a low-thrust segment that satisfies Equation (3.40) given an existing arc (where $f < f_{\text{engine}}$), one approach is to select a desired thrust magnitude and solve for the thrust time that keeps propellant consumption constant. For a CSI engine, the mass flow rate, defined in Equation (2.56), is a function of a variable nondimensional thrust magnitude f and a fixed exhaust velocity v_{e}. Equation (2.56) is decoupled from the underlying dynamics and is, therefore, easily integrated analytically to express spacecraft mass as a function of time,

$$m_f = m(\mathbb{t}) = \dot{m}^{\text{lt}}\mathbb{t} + m_0 \ , \quad \text{where} \quad \dot{m}^{\text{lt}} = -\frac{f}{v_{\text{e}}} \tag{3.41}$$

where $\mathbb{t}$ is a nondimensional time variable that is equivalent to t; the additional superscript is leveraged in this section to distinguish time quantities that are assumed to be associated with a low-thrust maneuver. If the original spacecraft mass equals m_0 at time $\mathbb{t}_0 = 0$ and the thrust is active with a thrust magnitude of f for $\mathbb{t}$ nondimensional time units, then the final mass ratio ${}^{m_f}/_{m_0}$ is expressed as,

$$\frac{m_f}{m_0} = \frac{\dot{m}^{\text{lt}}}{m_0}\mathbb{t} + 1 \quad \in [0, 1] \tag{3.42}$$

Equation (3.42) provides a ratio for the remaining mass of the spacecraft after a thrusting from $\mathbb{t}_0$ to $\mathbb{t}$. Alternatively, the change in mass

$$\Delta m = m_f - m_0 = \dot{m}^{\text{lt}}\mathbb{t} \tag{3.43}$$

is a useful quantity to measure propellant consumption. Substituting Equation (3.43) into Equation (3.41) and solving for $\mathbb{t}$ yields a relationship for thrusting time as a function of mass expenditure,

$$\mathbb{t} = \frac{\Delta m}{\dot{m}^{\text{lt}}} = -\frac{v_{\text{e}} \Delta m}{f} \tag{3.44}$$

Equation (3.44) supplies a relationship to scale thrust magnitude and time given a fixed change in mass. In this approach, given a desired thrust magnitude $f_{\text{new}} \in [0, f_{\text{max}}]$, Equation (3.44) yields the necessary thrusting time that keeps propellant consumption constant. Given $\{f, \mathbb{t}\}$, this formulation results in the simple scaling function,

$$\mathbb{t}_{\text{new}} = \frac{f}{f_{\text{new}}} \mathbb{t} \tag{3.45}$$

where f_{new} is the desired thrust magnitude and $\mathbb{t}_{\text{new}}$ is the derived thrust time for f_{new}.

An alternative, but equivalent, formulation as Equation (3.44) involves measuring thrusting time as a function of $\Delta\tilde{V}_{\text{equiv.}}$, defined in Equation (2.57) using the ideal rocket equation. In this approach, the final mass ratio $({}^{m_f}/{}_{m_0})$ is derived by rearranging Equation (2.57) as,

$$\frac{m_f}{m_0} = \exp\left(-\frac{\Delta\tilde{V}_{\text{equiv.}}}{\tilde{v}_{\text{e}}}\right) \tag{3.46}$$

Equation (3.46) is equivalent to Equation (3.42). Equating the two expressions and solving for $\mathbb{t}$ yields,

$$\mathbb{t} = \frac{m_0}{\dot{m}^{\text{lt}}} \left(1 - \exp\left(-\frac{\Delta\tilde{V}_{\text{equiv.}}}{\tilde{v}_{\text{e}}}\right)\right) \tag{3.47}$$

The expression in Equation (3.47) is easily inverted to obtain an equation that relates thrusting time to the equivalent $\Delta\tilde{V}_{\text{equiv.}}$ value,

$$\Delta\tilde{V}_{\text{equiv.}} = \tilde{v}_{\text{e}} \log\left(1 - \frac{\dot{m}^{\text{lt}}}{m_0}\mathbb{t}\right) \tag{3.48}$$

This relationship between burn duration and change in velocity is also useful in applications that seek to estimate thrusting time given an impulsive $\Delta\tilde{V}$, as demonstrated by Scarritt et al. in an autonomous finite-burn targeting approach [91].

The heuristic estimation of thrust time $\mathbb{t}$ from propellant consumption (Equation (3.44)) or $\Delta\tilde{V}_{\text{equiv.}}$ (Equation (3.48)) offers a simple, computationally efficient method for improving initial guesses for low-thrust targeting algorithms by providing thrust segments that adhere to a practical limitation for low-thrust engines. The scaling functions in Equations (3.44) and (3.48) are visualized in Figure 3.5 on each of the dual y-axes. Each colored line signifies the propellant consumption over time for a low-thrust engine that is activated at a particular throttle level, defined in Equation (2.52). The maximum thrust f_{max} corresponds to 100 % throttle, and f_{engine} represents the minimum thrust value available for Hayabusa 1. In this graphic, adjusting thrust time to maintain constant $\Delta\tilde{V}_{\text{equiv.}}$ and final mass values is accomplished by placing a horizontal line across the plot at the desired level (visualized for $\Delta\tilde{V}_{\text{equiv.}} = 250\,{}^{\text{m}}/_{\text{s}}$) and observing where this value intersects other throttle percentage lines.

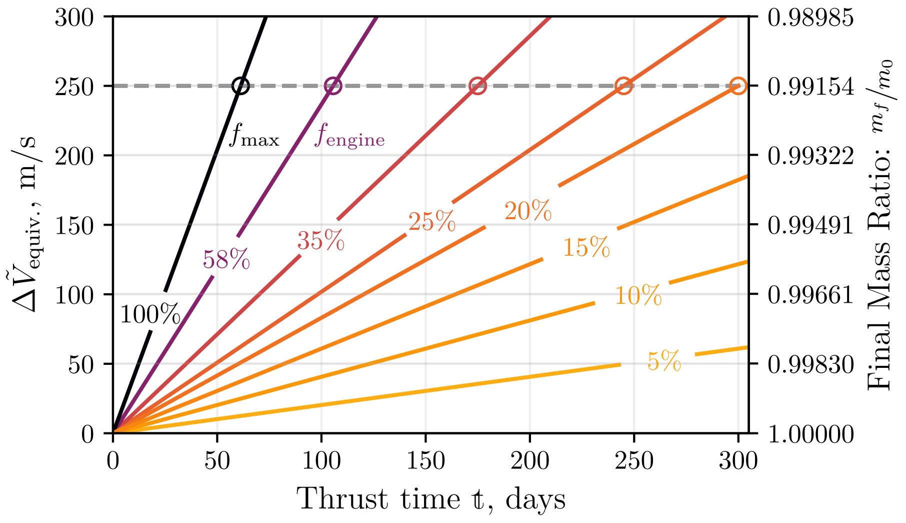


Figure 3.5. Relationship between thrust time and $\Delta\tilde{V}_{\text{equiv.}}$ for a spacecraft with thrust capability equal to Hayabusa 1. The listed percentages indicate throttle level, and the dashed line highlights estimated burn durations that all achieve $\Delta\tilde{V}_{\text{equiv.}} = 250\,{}^{\text{m}}/_{\text{s}}$. As throttle decreases, the estimated thrust time increases precipitously resulting in a less accurate estimation in targeting problems.

3.5 Low-Thrust Ephemeris Corrections

Low-thrust maneuver plans in this investigation are initially constructed in the CR3BP and later transformed into a higher-fidelity $\mathcal{N}$-body ephemeris force model. While the general low-thrust targeting process detailed in Section 3.4 remains applicable in an $\mathcal{N}$-body model, additional steps are necessary to account for the nonautonomous dynamics. In particular, an important consideration for targeting in an $\mathcal{N}$-body ephemeris model is ensuring that the resulting path is continuous in time. Epoch time is incorporated into a multiple shooting problem by defining a fixed initial model epoch $\mathcal{T}_{\text{model}}$. The epoch of each patch point $\tilde{\tau}_i$ is measured as the number of elapsed seconds since $\mathcal{T}_{\text{model}}$, and nondimensionalized using the characteristic time of the system $\tilde{t}^*$,

$$\tau_i = \frac{\tilde{\tau}_i}{\tilde{t}^*} \;, \qquad \text{where} \qquad \tilde{\tau}_i = \text{number of seconds since } \mathcal{T}_{\text{model}} \tag{3.49}$$

This investigation's characteristic time $\tilde{t}^*$ for the Earth-Moon system is given in Table 2.1. At each subsequent arc in an ephemeris multiple shooting problem, epoch continuity from arc (i) to $(i+1)$ is enforced by the simple constraint,

$$F_i(\boldsymbol{X}) = \tau_i + t_i - \tau_{i+1} = 0 \tag{3.50}$$

where $\{\tau_i, t_i, \tau_{i+1}\} \in \boldsymbol{X}$ are included as design variables.

A sample targeting schematic for a low-thrust ephemeris multiple shooting problem is visualized in Figure 3.6. In contrast to a similar CR3BP schematic, Figure 3.6, additional $\mathcal{N}$-body ephemeris considerations are included in the problem (visualized in brown). All ephemeris simulations in this investigation leverage the Moon as the central body, and include the Sun, the Earth, and Jupiter as perturbing bodies. Furthermore, a unit vector $\hat{u} = \hat{u}_{\text{J2000}}$ coordinate locates the direction of each thrust vector in a J2000 reference frame; therefore, additional constraints are necessary to ensure each $\hat{u}_i$ remains unit magnitude.

As depicted in Figure 3.6, epoch time free variables τ_i are allowed to vary at each patch point with additional constraints (defined in Equation (3.50)) included to ensure time remains continuous. The relationship between time variables in this simple problem is visualized in Figure 3.7. The model epoch time $\mathcal{T}_{\text{model}}$ is employed to reduce numerical error and, therefore, the selection of its value is arbitrary. To avoid large values in τ_i, a common approach is to select $\mathcal{T}_{\text{model}}$ such that the initial guess for $\tau_{i,0}$ is zero. Note that, in this problem t_i variables must remain positive; however, τ_i is relative measure and, therefore, admits negative values. Furthermore, recall from Section 2.3 that the $\mathcal{N}$-body ephemeris model employs planetary ephemerides from SPICE at specific points in time [69]. During the targeting process, the epoch values for each patch point, represented in Ephemeris Time (ET) (i.e. Barycentric Dynamical Time (TDB))[6], are computed as,

$$\mathrm{ET}(\mathcal{T}_i) = \mathrm{ET}(\mathcal{T}_{\mathrm{model}}) + \tilde{\tau}_i \tag{3.51}$$

where $\mathrm{ET}(\mathcal{T}_{\mathrm{model}})$ is the model epoch in ET, and $\tilde{\tau}_i$ is measured in seconds, as defined in Equation (3.49). Determining $\mathrm{ET}(\mathcal{T}_i)$ enables SPICE queries for the locations of planetary bodies during propagation segments.

3.5.1 Epoch Time Sensitivity

Along $\mathcal{N}$-body ephemeris integration segments in targeting applications, sensitivity information for the final states with respect to epoch times is sought to allow the timings of each patch point to vary. Epoch time, defined in Equation (3.49), is fixed over any integration segment and is, therefore, a parameter of the model; the general process for modeling the sensitivities associated with model parameters is detailed in Section 3.2.2. The variation of state variables with respect to epoch time is represented by $(\partial \boldsymbol{x}/\partial \tau)$ and, from Equation (3.13), evolves over time according to the first order differential equation,

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{x}}{\partial \tau}\right) = \boldsymbol{A}(t)\left(\frac{\partial \boldsymbol{x}}{\partial \tau}\right) + \frac{\partial \boldsymbol{\phi}}{\partial \tau}\ , \qquad \text{where} \qquad \boldsymbol{\phi} = \begin{bmatrix} \dot{\boldsymbol{R}}_{qi} \\ \ddot{\boldsymbol{R}}_{qi} \end{bmatrix} \tag{3.52}$$

In Equation (3.52), $\boldsymbol{R}_{qi}$ is a position vector that locations the massless particle P_i with respect to a central body P_q, and $\ddot{\boldsymbol{R}}_{qi}$ is the acceleration vector that comprise the relative equations of motion of the $\mathcal{N}$-body ephemeris model, defined in Equation (2.48). The $\boldsymbol{A}(t)$ matrix is the time-varying Jacobian of partial derivatives of the flow with respect to state variables, defined in Equation (3.5).

The only unknown term to derive in Equation (3.52) is the partial derivatives of the flow with respect to epoch time, i.e., $\partial \phi/\partial \tau$. The $\mathcal{N}$-body ephemeris model is represented with

[6]↑NAIF provides this helpful tutorial document to understand its time systems: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/15_time.pdf

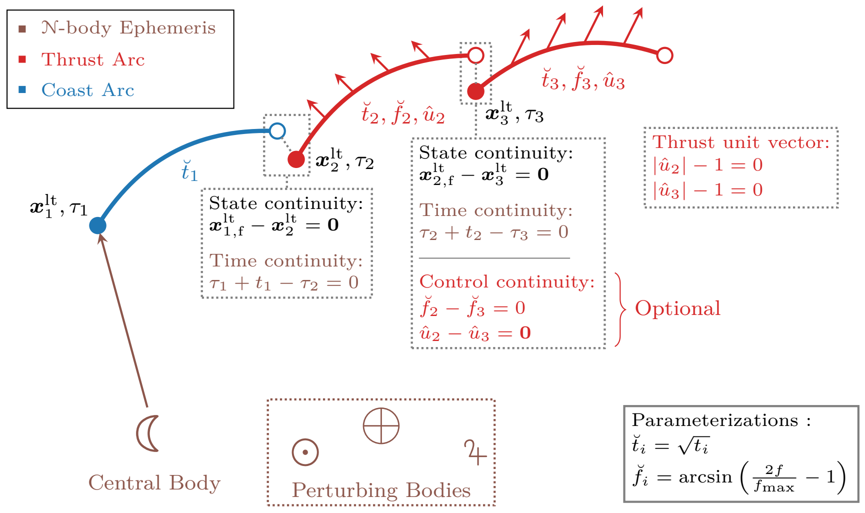

■ N-body Ephemeris
■ Thrust Arc
■ Coast Arc
$\boldsymbol{x}_1^{\text{lt}}, \tau_1$
$\breve{t}_1$
$\boldsymbol{x}_2^{\text{lt}}, \tau_2$
$\breve{t}_2, \breve{f}_2, \hat{u}_2$
$\boldsymbol{x}_3^{\text{lt}}, \tau_3$
$\breve{t}_3, \breve{f}_3, \hat{u}_3$
State continuity: $\boldsymbol{x}_{1,\text{f}}^{\text{lt}} - \boldsymbol{x}_2^{\text{lt}} = \mathbf{0}$
Time continuity: $\tau_1 + t_1 - \tau_2 = 0$
State continuity: $\boldsymbol{x}_{2,\text{f}}^{\text{lt}} - \boldsymbol{x}_3^{\text{lt}} = \mathbf{0}$
Time continuity: $\tau_2 + t_2 - \tau_3 = 0$
Control continuity: $\breve{f}_2 - \breve{f}_3 = 0$ $\hat{u}_2 - \hat{u}_3 = \mathbf{0}$
Optional
Thrust unit vector: $|\hat{u}_2| - 1 = 0$ $|\hat{u}_3| - 1 = 0$
Central Body
Perturbing Bodies
Parameterizations : $\breve{t}_i = \sqrt{t_i}$ $\breve{f}_i = \arcsin\left(\frac{2f}{f_{\max}} - 1\right)$

Figure 3.6. Free-variable/ schematic for a low-thrust multiple shooting method in the N-body ephemeris model. Parameterization options detailed in Equations (3.32) and (3.33). Graphic adapted from Bosanac, Ref [90], p.181, Fig. 7.5.

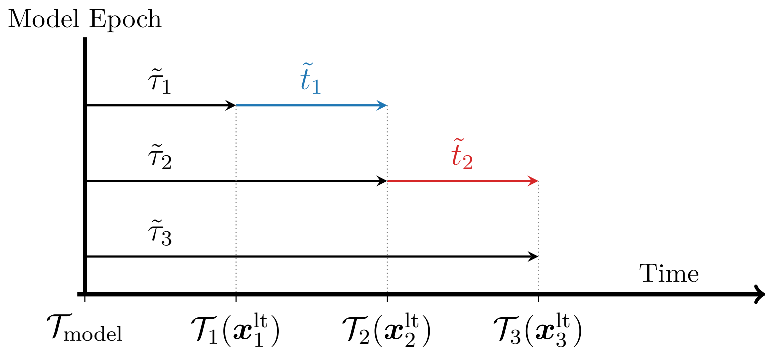

Model Epoch
$\tilde{\tau}_1$
$\tilde{t}_1$
$\tilde{\tau}_2$
$\tilde{t}_2$
$\tilde{\tau}_3$
Time
$\mathcal{T}_{\text{model}}$
$\mathcal{T}_1(\boldsymbol{x}_1^{\text{lt}})$
$\mathcal{T}_2(\boldsymbol{x}_2^{\text{lt}})$
$\mathcal{T}_3(\boldsymbol{x}_3^{\text{lt}})$

Figure 3.7. Epoch time variables introduced in N-body ephemeris multiple shooting problem, visualized for three patch points in the schematic in Figure 3.6. The relationship between dimensional and nondimensional time units is given by Equation (3.49).

respect to a central body, and its equations of motion do not explicitly contain τ. Hence, it is useful to perform a change of variables to perform this vector derivative,

$$\frac{\partial \boldsymbol{\phi}}{\partial \tau} = \left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{R}_{qj}}\right)\left(\frac{\partial \boldsymbol{R}_{qj}}{\partial \tau}\right) \tag{3.53}$$

where $\boldsymbol{R}_{qj}$ is the position of perturbing body P_j with respect to the central body P_q, as visualized in Figure 2.6. Note that the time rate of change term,

$$\frac{\partial \boldsymbol{R}_{qj}}{d\tau} = \frac{d\boldsymbol{R}_{qj}}{dt} = \dot{\boldsymbol{R}}_{qj} = \boldsymbol{V}_{qj} \in \mathbb{R}^{3\times 1} \tag{3.54}$$

is simply the velocity of P_j relative to P_q, and is determined from ephemeris data provided by SPICE.

The $\partial\boldsymbol{\phi}/\partial\boldsymbol{R}_{qj}$ matrix of partial derivatives in Equation (3.53) is analytically derived directly from the evolution function of the equations of motion,

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{R}_{qj}} = \begin{bmatrix} \partial\dot{\boldsymbol{R}}_{qi}/\partial\boldsymbol{R}_{qj} \\ \partial\ddot{\boldsymbol{R}}_{qi}/\partial\boldsymbol{R}_{qj} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}^{3\times 3} \\ \boldsymbol{J}_{\boldsymbol{R}_{qi}}(\boldsymbol{R}_{qj}) \end{bmatrix} \in \mathbb{R}^{6\times 3} \tag{3.55}$$

where $\boldsymbol{J}_{\boldsymbol{R}_{qi}}(\boldsymbol{R}_{qj})$ is the Jacobian matrix,

$$\boldsymbol{J}_{\boldsymbol{R}_{qi}}(\boldsymbol{R}_{qj}) = \begin{bmatrix} \partial\ddot{X}_{qi}/\partial X_{qj} & \partial\ddot{X}_{qi}/\partial Y_{qj} & \partial\ddot{X}_{qi}/\partial Z_{qj} \\ \partial\ddot{Y}_{qi}/\partial X_{qj} & \partial\ddot{Y}_{qi}/\partial Y_{qj} & \partial\ddot{Y}_{qi}/\partial Z_{qj} \\ \partial\ddot{Z}_{qi}/\partial X_{qj} & \partial\ddot{Z}_{qi}/\partial Y_{qj} & \partial\ddot{Z}_{qi}/\partial Z_{qj} \end{bmatrix} \tag{3.56}$$

Equations (3.54) and (3.55) are leveraged to rewrite the variational equations of motion in Equation (3.52) in the form,

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{x}}{\partial \tau}\right) = \boldsymbol{A}(t)\left(\frac{\partial \boldsymbol{x}}{\partial \tau}\right) + \begin{bmatrix} \boldsymbol{0}^{3\times 3} \\ \boldsymbol{J}_{\boldsymbol{R}_{qi}}(\boldsymbol{R}_{qj}) \end{bmatrix} \boldsymbol{V}_{qj} \tag{3.57}$$

The partial derivative values in Equation (3.56) are derived by noting that perturbing accelerations terms only appear within the summation component of the $\mathcal{N}$-body relative vector differential equation of motion, Equation (2.48). Leveraging the relationship $\boldsymbol{R}_{ij} = \boldsymbol{R}_{qj} - \boldsymbol{R}_{qi}$, the Jacobian is expressed as,

$$\boldsymbol{J}_{\boldsymbol{R}_{qi}}(\boldsymbol{R}_{qj}) = \frac{\partial \ddot{\boldsymbol{R}}_{qi}}{\partial \boldsymbol{R}_{qj}} \in \mathbb{R}^{3\times 3} \tag{3.58}$$

$$= \frac{\partial}{\partial \boldsymbol{R}_{qj}} \left(-G\frac{m_q}{R_{qi}^3}\boldsymbol{R}_{qi} + G\sum_{\substack{j=1\\ j\neq i,q}}^{\mathcal{N}} m_j \left(\frac{\boldsymbol{R}_{ij}}{R_{ij}^3} - \frac{\boldsymbol{R}_{qj}}{R_{qj}^3} \right) \right) \tag{3.59}$$

$$= G\sum_{\substack{j=1\\ j\neq i,q}}^{\mathcal{N}} m_j \frac{\partial}{\partial \boldsymbol{R}_{qj}} \left(\frac{\boldsymbol{R}_{qj} - \boldsymbol{R}_{qi}}{|\boldsymbol{R}_{qj} - \boldsymbol{R}_{qi}|^3} - \frac{\boldsymbol{R}_{qj}}{R_{qj}^3} \right) \tag{3.60}$$

$$= G\sum_{\substack{j=1\\ j\neq i,q}}^{\mathcal{N}} m_j \frac{\partial}{\partial \boldsymbol{R}_{qj}} \left(\frac{\boldsymbol{R}_{qj}}{|\boldsymbol{R}_{qj} - \boldsymbol{R}_{qi}|^3} \right) - \frac{\partial}{\partial \boldsymbol{R}_{qj}} \left(\frac{\boldsymbol{R}_{qj}}{R_{qj}^3} \right) \tag{3.61}$$

Each term of the Jacobian matrix represented by Equation (3.61) is evaluated as,

$$\frac{\partial \ddot{X}_{qi}}{\partial X_{qj}} = Gm_j \left(\frac{1}{R_{ij}^3} - \frac{3(X_{qj} - X_{qi})}{R_{ij}^5} - \frac{1}{R_{qj}^3} + \frac{3X_{qj}^2}{R_{qj}^5} \right) \tag{3.62}$$

$$\frac{\partial \ddot{Y}_{qi}}{\partial Y_{qj}} = Gm_j \left(\frac{1}{R_{ij}^3} - \frac{3(Y_{qj} - Y_{qi})}{R_{ij}^5} - \frac{1}{R_{qj}^3} + \frac{3Y_{qj}^2}{R_{qj}^5} \right) \tag{3.63}$$

$$\frac{\partial \ddot{Z}_{qi}}{\partial Z_{qj}} = Gm_j \left(\frac{1}{R_{ij}^3} - \frac{3(Z_{qj} - Z_{qi})}{R_{ij}^5} - \frac{1}{R_{qj}^3} + \frac{3Z_{qj}^2}{R_{qj}^5} \right) \tag{3.64}$$

$$\frac{\partial \ddot{X}_{qi}}{\partial Y_{qj}} = \frac{\partial \ddot{Y}_{qi}}{\partial X_{qj}} = 3Gm_j \left(-\frac{(X_{qj} - X_{qi})(Y_{qj} - Y_{qi})}{R_{ij}^5} + \frac{X_{qj}Y_{qj}}{R_{qj}^5} \right) \tag{3.65}$$

$$\frac{\partial \ddot{X}_{qi}}{\partial Z_{qj}} = \frac{\partial \ddot{Z}_{qi}}{\partial X_{qj}} = 3Gm_j \left(-\frac{(X_{qj} - X_{qi})(Z_{qj} - Z_{qi})}{R_{ij}^5} + \frac{X_{qj}Z_{qj}}{R_{qj}^5} \right) \tag{3.66}$$

$$\frac{\partial \ddot{Y}_{qi}}{\partial Z_{qj}} = \frac{\partial \ddot{Z}_{qi}}{\partial Y_{qj}} = 3Gm_j \left(-\frac{(Y_{qj} - Y_{qi})(Z_{qj} - Z_{qi})}{R_{ij}^5} + \frac{Y_{qj}Z_{qj}}{R_{qj}^5} \right) \tag{3.67}$$

The six additional variational equations of motion in Equation (3.61) are numerically integrated along with the equations of motion and provide sensitivity information regarding changes to the initial epoch. The variation in final state with respect to epoch introduces additional flexibility in $\mathcal{N}$-body ephemeris targeting problems.

4. REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a branch of ML that encompasses a broad range of goal-oriented algorithms in which an agent 'learns' to perform a task through trial-and-error. Agents explore their ability to manipulate an environment, and seek to associate particular action choices with positive outcomes. Reinforcement learning techniques are leveraged in many fields and applications and have demonstrated particular success in robotic control and playing games. Current state-of-the-art RL approaches employ NNs to approximate nonlinear functions, and policy gradient methods are of particular recent interest due to their demonstrated ability in continuous control tasks. One such algorithm, Twin Delayed Deep Deterministic Policy Gradient (TD3) [92], is employed in this investigation to train a NN controller.

4.1 Machine Learning Fundamentals

Machine Learning (ML) broadly refers to a class of data science techniques that involve computer systems that leverage predictive and task-performing abilities without explicit instructions. These algorithms leverage mathematical models that automatically improve their performance on a given task by identifying patterns and learning from experience. Such approaches are closely related to statistics and typically involve inferences based on large quantities of data. Research into machine learning techniques has experienced a surge in interest and advancements over the last decade. Noteworthy machine learning applications include Natural Language Processing (NLP), image and speech recognition, recommendation systems, predictive modeling, and autonomous robotics (including self-driving cars). This investigation seeks to understand and advance ML applications that will likely play an important role in the future of self-sufficient, autonomous spaceflight.

Machine learning is a subfield of Artificial Intelligence (AI) that focuses on replicating low-level decision processes in humans. Such approaches seek to emulate biological processes through experience-based learning and mathematical models that stem from cognitive neuroscience, particularly in the case of NNs. Artificial intelligence encompasses a significantly broader range of subfields than ML, including research into emulating or creating high-level reasoning processes on computer chips (a significantly). These loftier AI ambition range from science to science fiction, and are outside the scope of the present research; the ML algorithms and models in this investigation are best understood as numerical methods based on applied statistics and optimization techniques rather than emulations of high-level intelligent systems.

4.1.1 Artificial Neural Networks

An artificial Neural Network (NN) is a class of nonlinear computational models that are frequently employed in ML tasks [93]. While ubiquitous in supervised learning applications, NNs are also used extensively in modern RL algorithms due to their demonstrated ability to approximate nonlinear functions. Many traditional tabular RL approaches, such as Q-learning, rely on finely discretizing the state and action spaces, quickly becoming intractable as the number of dimensions in the problem increases. Thus, leveraging NNs allows modern algorithms to access continuous state and action spaces and easily incorporate additional dimensions.

Neural networks are potentially well-suited for a flight computer, with demonstrated capability on hardware suitable for spaceflight [94]. While training is computationally complex, the evaluation process is relatively simple. Furthermore, current flight software prototype investigations leverage iterative G&C algorithms onboard [3], [4]. These repetitive operations pose challenges in predicting the total number of CPU cycles and in excessive iterations resulting from an inaccurate initial guess. The NN employed in this investigation is evaluated in constant time and possesses a relatively small 0.5 MB memory footprint, thus offering a rapid means of identifying startup solutions. Augmenting iterative methods with a NN offers potential improvements in extending the range of convergence and reducing the total number of necessary iterations. Furthermore, specialized "neuromorphic" flight processors are currently being developed to enable low-power NN evaluations in space applications [95]. Adoption of neuromorphic hardware into flight systems will render ML approaches more accessible and productive, and may enable efficient autonomous control, decision-making, and onboard adaptive learning [96].

Neural Network Evaluation

To understand the relative simplicity of evaluating a NN, a simplified sample model is introduced. In this example, the small NN, depicted in Figure 4.1, receives two scalar inputs, processes them through a single two-node hidden layer, and outputs a scalar value. The depicted network is *feedforward*, that is, there are no cycles between nodes (values only pass from left-to-right as data is processed through the network). Understanding the evaluation process of NNs informs their suitability for on-board use.

Evaluating a feedforward neural network consists of straightforward linear algebra operations, with several nonlinear element-wise activation functions. After the input layer, each node is computed as a linear combination of weighted values and a bias, and processed through an activation function to incorporate nonlinearity into the model. Hence, in Figure 4.1, the lines connecting nodes each carry an associated weight and each node (except input variables) possesses an associated bias. The weights signify the impact that particular nodes exert on each node in the next layer. The bias allows the activation function to be shifted left or right for each node. Together, the weights and biases form the set of trainable parameters for the model. Deploying a pre-trained NN onboard a spacecraft would involve storing these weight and bias values in memory.

In general, the NN trainable parameters cannot be known a priori, and so no suitable initial guess of their value is possible. Hence, the weights are typically initialized with random values between 0 and 1 (depending on the chosen activation function), and the biases are initialized at zero. Some applications benefit from initializing weights and biases from a NN that was trained on a related problem. This initialization technique, termed *transfer learning*, can significantly reduce training times at the cost of limiting the potential solution space.

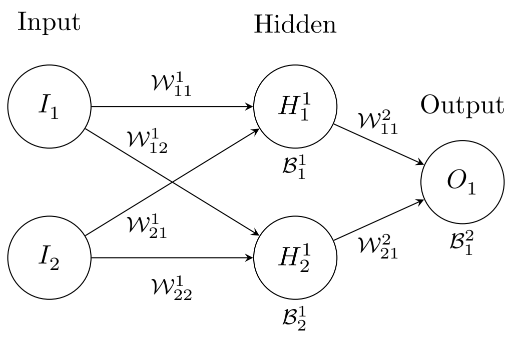


Figure 4.1. Basic feedforward neural network consisting of two inputs, two hidden nodes, and one output variable.

Consider the two nodes of the hidden layer in Figure 4.1, denoted H_1^1 and H_2^1, where the subscript denotes the index of the node and the superscript denotes the hidden layer to which the node belongs. The values in the hidden layer are computed as,

$$H_1^1 = \alpha^1(I_1\mathcal{W}_{11}^1 + I_2\mathcal{W}_{21}^1 + \mathcal{B}_1^1) \tag{4.1}$$

$$H_2^1 = \alpha^1(I_1\mathcal{W}_{12}^1 + I_2\mathcal{W}_{22}^1 + \mathcal{B}_2^1) \tag{4.2}$$

where α^1 is the activation function employed for the H^1 layer, $\mathcal{B}_j^1$ is the bias term for H_j^1, and $\mathcal{W}_{ij}$ is the weight connecting node I_i in the input layer to node H_j^1 in the hidden layer. The values of H_1^1 and H_2^1 are clearly simple linear combinations of weighted inputs, with the activation function applied to the total sum. Evaluating the two nodes together is expressed more concisely in matrix form,

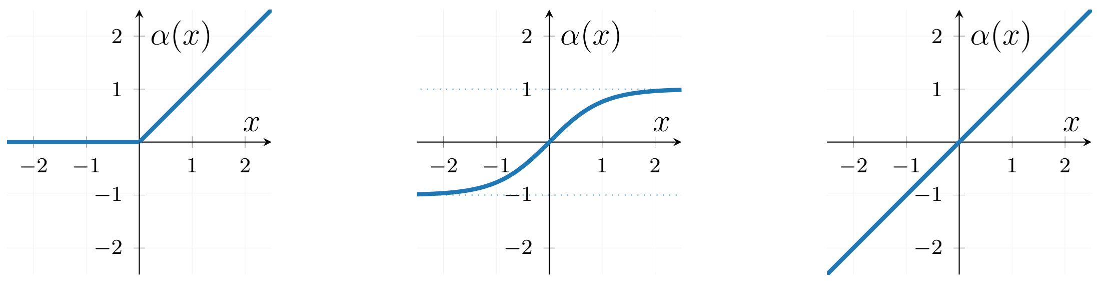
$$H_{2\times1}^1 = \alpha^1\left(\mathcal{W}_{2\times2}^1 I_{2\times1} + \mathcal{B}_{2\times1}^1\right) \tag{4.3}$$

The output is easily computed in the same manner, with inputs received from the hidden layer,

$$O_1 = \alpha^2\left(\mathcal{W}_{1\times2}^2 H_{2\times1}^1 + \mathcal{B}_1^2\right) \tag{4.4}$$

Equations (4.3) and (4.4) are the sole operations necessary to evaluate the depicted neural network.

In evaluating the NN depicted in Figure 4.1, the only opportunities to introduce nonlinearity are in the activation functions, α^1 and α^2. Without activation functions, the network is only able to model linear functions and, thus, the selection of the activation function is an important component in neural network performance. Furthermore, bounded functions are advantageous since they aid in normalizing the output of the network. For this reason, common activation functions include sigmoid, hyperbolic tangent (tanh) and Rectified Linear Unit (ReLU). This investigation employs ReLU to incorporate nonlinearity in hidden layers, and tanh to bound output functions for actor networks. Linear activation is also, at times, advantageous. Within RL, networks that produce a single scalar value output frequently employ a linear activation in the output layer. Together, ReLU, tanh, and linear are the only activation functions employed in this investigation and are plotted in Figure 4.2.



(a) ReLU (b) tanh (c) Linear

Figure 4.2. Neural network activation functions employed in this investigation.

4.2 Reinforcement Learning Fundamentals

Reinforcement Learning (RL) is a class of algorithms in which a goal-seeking *agent* seeks to complete a task by means of interaction with an *environment*. An agent is a controller that maps observed variables to actions. The learning process originates with the agent directly exploring an environment by means of trial-and-error. The environment communicates relevant information about its dynamics to the agent by means of a *state* or *observation* signal, which the agent then employs to perform some *action*. The environment updates its current state based on the action and computes a numerical reward that communicates the immediate benefit of the given action. This process repeats iteratively such that, over time, the agent improves its *policy* (the means by which decisions are accomplished) by seeking to maximize the reward signal. In many cases, terminal conditions exist that cease the learning process. In these cases, the environment typically resets to an initial configuration, and the process begins anew. These are identified as *episodic* tasks, where each episode signifies an attempt by the agent to solve a problem. A schematic for the high-level process of an RL agent is depicted in Figure 4.3. This diagram highlights the three signals that enable communication between the agent and environment at time steps $\mathtt{t}$ and $\mathtt{t}+1$: state, action, and reward. While the RL literature prefers the terms *agent*, *environment*, and *action*, these expressions are analogous to the more common engineering terms *controller*, *controlled system* (or *plant*), and *control signal* [97].

The training procedure is characterized as a Markov Decision Process (MDP), which involves both feedback and associative actions, and is a classical formalization of sequential decision-making problems [97]. The MDP is the idealized mathematical form of the problem and offers a straightforward implementation framework for RL algorithms. In the infinite-horizon discounted case, the MDP is formulated by a tuple $< \mathcal{S}, \mathcal{A}, p, r, p_0 >$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of all possible states and actions, respectively, $p(\boldsymbol{s}_{\mathtt{t}+1}|\boldsymbol{s}_{\mathtt{t}}, \boldsymbol{a}_{\mathtt{t}}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state-transition probability distribution, $r(\boldsymbol{s}_{\mathtt{t}}, \boldsymbol{a}_{\mathtt{t}}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $p_0(\boldsymbol{s}_0)$ is the density of the initial state distribution. For clarity, throughout this investigation, the time step associated with sequential states in an MDP is represented by the character '$\mathtt{t}$' to avoid confusion with the nondimensional time variable 't'.

Perfect state knowledge is frequently absent in realistic scenarios, thus a Partially Observable Markov Decision Processes (POMDP) characterization provides a more general framework for implementing environments that possess incomplete or uncertain state information.

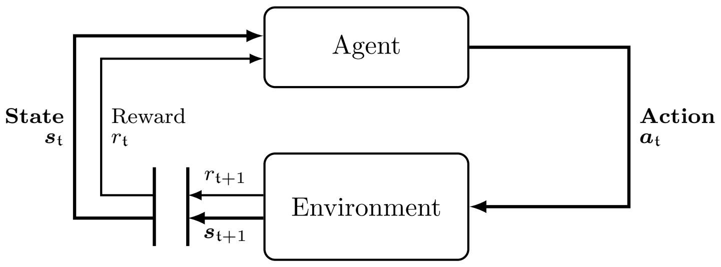

Agent
State
$\boldsymbol{s}_{\mathtt{t}}$
Reward
$r_{\mathtt{t}}$
Action
$\boldsymbol{a}_{\mathtt{t}}$
$r_{\mathtt{t}+1}$
Environment
$\boldsymbol{s}_{\mathtt{t}+1}$

Figure 4.3. The agent-environment process in an MDP. For POMDPs, the state variable $\boldsymbol{s}_{\mathtt{t}}$ is replaced with an observation $\boldsymbol{o}_{\mathtt{t}}$. Graphic reproduced from Sutton and Barto [97], p.48. Fig. 3.1).

In particular, a spacecraft's position and velocity are never precisely known (much to the disappointment of the navigation community), and thus RL applications within the space domain typically fall under the POMDP classification. For a POMDP, an "observation" signal is introduced that serves as an analog to the state signal in an MDP. The observation $\boldsymbol{o} \in \mathcal{O}$ is a function of the state $\boldsymbol{o} = \boldsymbol{o}(\boldsymbol{s})$, where $\mathcal{O}$ is the set of all possible observations and $\boldsymbol{s} \in \mathcal{S}$ is the hidden state. The observation is simply understood as a representation of the (potentially incomplete) state knowledge that is available to the agent. The delineation between state and observation (i.e., MDP and POMDP) is an important consideration when evaluating a learning environment in reinforcing the notion that the agent is acting on incomplete information; however, this distinction does not affect the implementation of the RL algorithm itself. Furthermore, the observation terminology avoids ambiguity when referring to state variables, as the word "state" in spaceflight applications typically refers to the evolution parameter of a dynamical system (Equation (2.1)), e.g., the vector of position, velocity, and mass variables. This investigation leverages the *state* $\boldsymbol{s}_\mathrm{t}$ terminology when discussing the theoretical MDP framework for the RL process, and *observation* $\boldsymbol{o}_\mathrm{t}$ when referring to this investigation's POMDP environmental configurations.

In the idealized MDP framework, each state variable must consider the *Markov property*, which requires that future states depend only upon the current state, and not on the series of events that preceded it [97], i.e., $p(\boldsymbol{s}_{\mathrm{t}+1}|\boldsymbol{s}_0, \boldsymbol{a}_0, \dots, \boldsymbol{s}_\mathrm{t}, \boldsymbol{a}_\mathrm{t}) = p(\boldsymbol{s}_{\mathrm{t}+1}|\boldsymbol{s}_\mathrm{t}, \boldsymbol{a}_\mathrm{t})$. The Markov property is useful in assessing the degree to which this investigation's POMDP learning environments approximate an MDP. In spaceflight G&C tasks, the position, velocity, and mass of the spacecraft are the together sufficient in applications where the RL state transition probability is solely governed by numerically integrating the equations of motion, since this dynamical state serves as the initial condition for numerically solving the initial value problem. While this property holds for simple implementation options, a variety of possible learning environment characterization decisions affects the Markov property. In particular, early termination options (defined by either 'failure' or 'success' criteria), dynamical state uncertainty, variations in propagation time between states, and modeled maneuver execution errors all impact the degree to which the POMDP observation signal approximates the idealized state.

The fundamental goal of an RL process is for the agent to develop a policy $\boldsymbol{\pi} : \mathcal{S} \rightarrow p(\mathcal{A})$ that maps given states to a probability distribution over the action space that maximizes future returns. The expected return $\mathfrak{R}_{\mathrm{t}}$ is a balance between immediate and future rewards, formalized as a finite-horizon discounted sum from time step t to the final time step $\mathrm{t_f}$,

$$\mathfrak{R}_{\mathrm{t}} = \sum_{i=\mathrm{t}}^{\mathrm{t_f}} \gamma^{(i-\mathrm{t})} r(\boldsymbol{s}_i, \boldsymbol{a}_i) \tag{4.5}$$

where $\gamma \in [0, 1]$ is the discount factor that determines the extent to which the agent prioritizes immediate versus future rewards (typically near 1). Reinforcement learning aims to construct a policy $\boldsymbol{\pi}$ to maximize the expected return of a task originating from the initial state distribution $p_0(\boldsymbol{s}_0)$, thus, maximizing the objective function $J = \mathbb{E}_{\boldsymbol{s}_i \sim p_{\boldsymbol{\pi}}, \boldsymbol{a}_i \sim \boldsymbol{\pi}}[\mathfrak{R}_0]$ where $p_{\boldsymbol{\pi}}$ is the discounted state visitation distribution following policy $\boldsymbol{\pi}$. The definition of expected return leads to the formalization of the *action-value* function. Value functions are estimated in nearly all RL algorithms and inform the agent of the quality of a particular state. The *state-value function* $\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}})$, defined as,

$$\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}_{i>\mathrm{t}} \sim p_{\boldsymbol{\pi}} \\ \boldsymbol{a}_{i\geq \mathrm{t}} \sim \boldsymbol{\pi}}} [\mathfrak{R}_{\mathrm{t}} |\, \boldsymbol{s}_{\mathrm{t}}] \tag{4.6}$$

outputs a scalar that represents the relative value of a particular state. It is similarly useful to estimate the value of a state-action pair. This *action-value function*, denoted $\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}})$, computes the value $\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}})$ of taking action $\boldsymbol{a}_{\mathrm{t}}$ at $\boldsymbol{s}_{\mathrm{t}}$, and subsequently following the policy $\boldsymbol{\pi}$. The action-value function is closely linked to the state-value function, Equation (4.6), and is computed as,

$$\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}}) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}_{i>\mathrm{t}} \sim p_{\boldsymbol{\pi}} \\ \boldsymbol{a}_{i> \mathrm{t}} \sim \boldsymbol{\pi}}} [\mathfrak{R}_{\mathrm{t}} \,|\, \boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}}] \tag{4.7}$$

The state-value and action-value functions are related by noting the distribution to which the action $\boldsymbol{a}_{\mathrm{t}}$ belongs,

$$\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}) = \mathop{\mathbb{E}}_{\boldsymbol{a}_{\mathrm{t}} \sim \boldsymbol{\pi}} [\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}})] \tag{4.8}$$

The value functions $\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}})$ and $\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}})$ differ in that the action $\boldsymbol{a}_{\mathrm{t}}$ in $\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}})$ is not necessarily sampled from policy $\boldsymbol{\pi}$. In practice, many modern algorithms estimate these value functions using neural networks. Alternatively, rather than employing $\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}}, \boldsymbol{a}_{\mathrm{t}})$ or $\mathcal{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_{\mathrm{t}})$

to represent value information directly, some RL methods are concerned with estimating the value of a state-action pair compared to average. In these approaches, a relative value, termed the *advantage* function $\mathcal{A}_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t)$, is computed as the difference between the state-action-value function and the state-value function [98],

$$\mathcal{A}_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathcal{Q}_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) - \mathcal{V}_\pi(\boldsymbol{s}_t) \tag{4.9}$$

The advantage function is positive when $\boldsymbol{a}_t$ produces a more desirable outcome than an action sampled from the current policy $\boldsymbol{\pi}$, and is negative when $\boldsymbol{a}_t$ produces a less desirable outcome.

4.2.1 Actor-Critic Reinforcement Learning

Policy optimization methods seek to directly 'learn' a parameterized policy, $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$, where $\boldsymbol{\theta}$ represents a policy parameter vector. In contrast to some value optimization methods, such as the classical $\mathcal{Q}$-Learning approach, policy optimization methods are well suited to problems with a continuous action space. The ability to incorporate continuous state and action spaces offers significant advantage in complex control tasks that suffer from discretization and are more extendable to higher-dimensional dynamical models. A particularly fruitful branch of RL research emerges from a class of hybrid algorithms, identified as actor-critic methods. These approaches seek both the policy (i.e., *actor*) and the value (i.e., *critic*) functions, where both actor and critic are typically represented as NNs. Common actor-critic schemes include Advantage Actor Critic (A2C) [99], Deep Deterministic Policy Gradient (DDPG) [100], Twin Delayed Deep Deterministic Policy Gradient (TD3) [92], Soft Actor Critic (SAC) [101], Trust Region Policy Optimization (TRPO) [102], and Proximal Policy Optimization (PPO) [103].

A key delineation between modern actor-critic methods surrounds an assumption concerning the policy distribution from which the actions are sampled. In "on-policy" algorithms, e.g., TRPO and PPO, update equations assume all actions are sampled from the most recent version of the parameterized policy, $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, causing the current data batch to be discarded following each optimization process. In contrast, "off-policy" approaches, actions may be sampled from *any* distribution, allowing for improved learning efficiency through replay buffers. The present investigation does not seek to identify the 'best' RL algorithm for low-thrust G&C applications as the state-of-the-art landscape is constantly evolving. Rather, this research seeks to establish environment implementation methods that are applicable to on-policy and off-policy methods alike. This investigation leverages TD3 as the core RL algorithm due to favorable data efficiency properties and demonstrates broader applicability through the usage of PPO for one sample mission application.

4.2.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (equivalently, Twin Delayed DDPG, or TD3) is a state-of-the-art off-policy actor-critic algorithm that concurrently learns a parameterized policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ and action-value function $\mathcal{Q}_{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ [92]. Various RL approaches over the past several decades leverage the action-value function to formulate an effective policy. An early breakthrough for problems with discrete state and action spaces occurred in 1989 with $\mathcal{Q}$-Learning: a temporal difference learning algorithm that allows powerful agents to be trained in an off-policy manner [104]. The core motivation for the $\mathcal{Q}$-Learning approach is simple: if the $\mathcal{Q}_{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ function is known, that is, the value of all state action pairs is accurately represented, then an optimal policy is available by simply selecting the available action that possesses the largest $\mathcal{Q}_{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ value. While $\mathcal{Q}$-Learning is very effective, it was limited, along with other RL algorithms at the time, by the assumption of discrete spaces due to the lack of effective nonlinear function approximators.

A major breakthrough in RL came in 2015 when the $\mathcal{Q}$-Learning algorithm was extended to continuous state space problems with the Deep Q-Network (DQN) algorithm [105]. The DQN approach involves the same update rule as $\mathcal{Q}$-Learning, but rather than direct computation, the $\mathcal{Q}$-function is effectively estimated using a NN: alleviating the need for tabulated results. However, similar to $\mathcal{Q}$-Learning, DQN still suffered from the discrete action space assumption. While acceptable for tasks with a small, finite number of controls (such as Atari games), the curse of dimensionality in the action space limits applicability for continuous control tasks, such as robotics.

To address the action-space dimensionality limitation with DQN, Lillicrap et al. introduce DDPG [100]: an extension of DQN and Deterministic Policy Gradient (DPG) [106], that uses a second "actor" NN to construct a parameterized action function, and performs the $\mathcal{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ update with the new actor network. A key portion of DDPG is the inclusion of "target" networks for both the actor and value (i.e., *critic*) networks (a similar target approach is also noted in the double-DQN extension of the DQN approach [107]). By including duplicate networks, the second "target" network may be held constant while the actual network is updated. A more stable process results because it doesn't involve continuously updating estimates to be used in the minibatch updates. While powerful, DDPG contains several notable limitations. In particular, the value function is often overestimated, which leads to a significant bias in learning.

Twin Delayed Deep Deterministic Policy Gradient addresses the instability of DDPG in several ways. First, two separate value networks are included. Incorporating the minimum value estimate of the two networks mitigates the effect of the value function overestimation bias present in DDPG (along with other actor-critic schemes). Next, DDPG involves *bootstrapping*, i.e., implementing updates based on estimates rather than true values. Noisy estimates cause noisy gradients that pose significant difficulty in network optimization. The TD3 approach seeks to mitigate this error by delaying updates to the actor network in hopes that additional updates to the critic network will provide more accurate estimates for the actor updates. Finally, to avoid peak overfitting in the policy network, policy smoothing is included, that introduces a small amount of clipped random noise to the output of the target policy. Together, these improvements form the TD3 variant of DDPG. While the inclusion of function approximations demands increased complexity, TD3 shares the same core motivation of $\mathcal{Q}$-Learning: optimize a policy network to compute actions that maximize the $\mathcal{Q}$-function. This investigation bases its implementation of TD3 on the open-source "Spinning Up" TensorFlow implementation provided by OpenAI [108]

4.3 Learning Environment Configurations

While both the selection and implementation of an appropriate RL algorithm is critical to the learning performance, so too is proper design of the RL environment. The environment represents the formulations for the observation, action, and reward signals that comprise the POMDP. The selection of a reward function is often both the most difficult and most important function for design and is, thus, a critical element of this learning framework. Proper signal design is vital because even the most robust learning algorithm consistently falls short in an ill-designed environment. Hence, a proper quantification of positive and negative behavior, given the goals of the guidance framework, is crucial in achieving desirable outcomes.

The selection of encoding (and decoding) techniques of relevant variables for NNs is a critical aspect in employing RL in spaceflight problems. Neural networks are initialized with small weights and, hence, improperly scaled input or output variables often lead to unstable learning processes. While this challenge may, at times, be addressed by a standardization process, an additional complicating factor is representing cyclic values, since NN inputs/outputs are typically understood as bounded and monotonic. This challenge manifests in this investigation in representing periodic orbits and thrust directions. Properly encoding and decoding cyclic values enables the learning processes, and allows the RL agent to make informed decisions.

4.3.1 Periodic Orbit Encoding

Periodic orbits are frequently leveraged in celestial mechanics applications and present a specific challenges for NNs. For two-body Keplerian applications, locations along an orbit are most frequently represented by an angle, e.g., true anomaly, eccentric anomaly, or mean anomaly. In multi-body dynamical regimes, angles are occasionally leveraged in specific applications, but the wide variety in periodic orbit geometry offers challenges in any general angle definition. Instead, elapsed time since a specified initial condition is most frequently employed to represent location along an orbit. Regardless of coordinate choice, directly including cyclic values, such as angle or time along a periodic orbit, in a NN architecture is ineffective due to the discontinuity at the beginning and end conditions. This challenge is further exacerbated by cyclic value extrema, typically occurring at apse conditions, that often arise as critically important locations along an orbit. Instead, this investigation leverages a trigonometric ('trig') encoding and decoding process [109] to represent locations along a periodic orbit.

To apply the trig encoding to periodic orbits in the CR3BP, time since a fixed state, $t_{\text{po}} \in [0, \mathbb{P}]$ represents an orbit location, where $\mathbb{P}$ is the period of the underlying structure. The selected fixed initial state may occur anywhere along the orbit. As a cyclic value, t_{po} is conveniently represented as a fraction of the orbit period. The encoding process may be interpreted as projecting t_{po} onto a unit circle, and applying the sine and cosine functions on the resulting angle, and decoding is accomplished via the two-argument arctangent function `atan2`, i.e.,

$$\begin{cases} \underline{\textbf{Encoding}}: \quad t_{\text{po}} \to [\sin\xi, \cos\xi], & \text{where } \xi = \dfrac{t_{\text{po}} \cdot 2\pi}{\mathbb{P}} \\ \underline{\textbf{Decoding}}: [\sin\xi, \cos\xi] \to t_{\text{po}}, & \text{where } t_{\text{po}} = \dfrac{\texttt{atan2}\,(\sin\xi, \cos\xi) \cdot \mathbb{P}}{2\pi} \end{cases} \tag{4.10}$$

The relationship between t_{po} and ξ is visualized in Figure 4.4, where colormaps emphasize the cyclic nature of the ξ coordinate (in contrast to t_{po}). The encoding and decoding spaces are visualized in Figure 4.5. When leveraged in an encoding process, Figure 4.5(a), ξ necessarily lies on unit circle. However, the $\sin\xi$ and $\cos\xi$ values are continuous in $\mathbb{R}_{[-1,1]}$. Thus, if the value $\sin\xi$ and $\cos\xi$ are computed separately, then duplicate values exist in the decoding space. The order of magnitude of the encoding and decoding spaces render them potentially useful for ML applications that involves NNs.

The evolution of ξ along four sample orbits is depicted in Figure 4.6. In each case, apolune is arbitrarily selected for $\xi = 0$ and, therefore, $\xi = \pi$ occurs at the half-period of the orbit. Then, $\xi = \pi$ coincides with perilune for the NRHO and halo orbit examples, and with the second apolune for the butterfly orbit. The angle ξ has no general physical significance. Several previous applications involving NRHOs label ξ, measured from perilune, as 'Mean Anomaly' [110], [111]. This interpretation provides physical insight for orbits with elliptic shapes in the rotating reference frame, but breaks down for more complex geometries

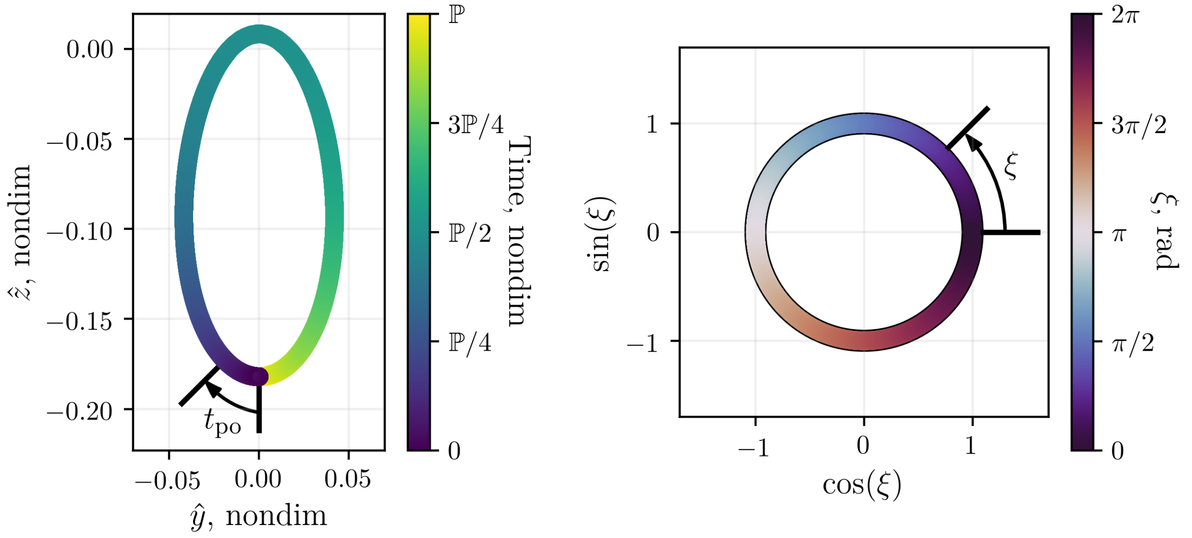


Figure 4.4. Relationship between t_{po} and ξ, where t_{po} is visualized on the $\hat{y}$-$\hat{z}$ projection of the 9:2 NRHO (left), and ξ evolves along a unit circle (right). The 9:2 NRHO, colored by ξ, is depicted in Figure 4.6(a).

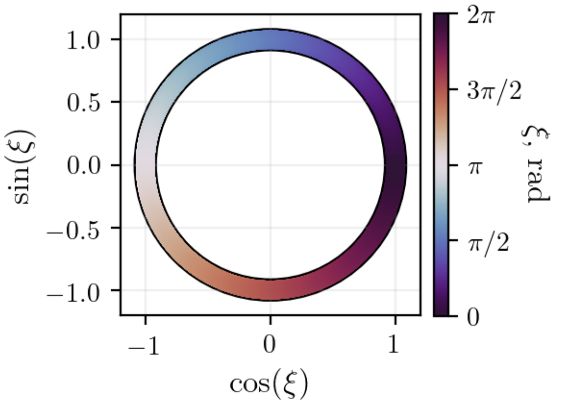
that involve multiple periapses in a single period. The different evolution of ξ across orbits from different families in Figure 4.6 reinforces ξ's lack of general physical significance in configuration space.

4.3.2 Observation Signal Design

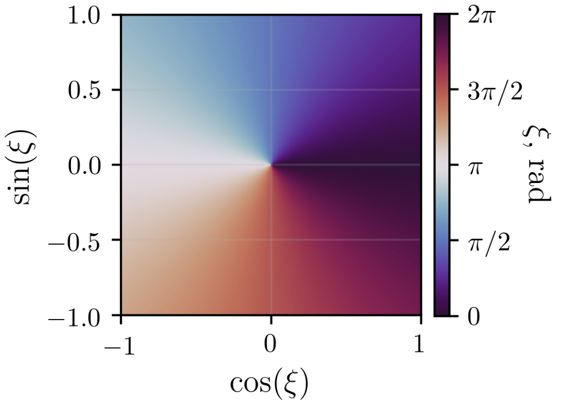
Under an MDP, the environmental state at time step $\mathfrak{t}$ ideally includes all necessary past environmental information that impacts the future [97]. However, recall that this investigation's environments are classified as POMDPs; thus, the observation signal $\boldsymbol{o}_{\mathfrak{t}}$ is employed to approximate the idealized state signal $\boldsymbol{s}_{\mathfrak{t}}$. For the low-thrust CR3BP, the Markov property is primarily influenced by the evolution parameter vector,

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\rho}^T & m \end{bmatrix}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & m \end{bmatrix} \in \mathbb{R}^7 \tag{4.11}$$

where $\boldsymbol{\rho}$ is the six-dimensional vector of position and velocity coordinates defined in Equation (2.11), and m is the spacecraft's mass. The $\boldsymbol{q}$ vector is sufficient to satisfy the Markov property in some theoretical spaceflight applications since future states are predicted by nu-



(a) Encoding Space



(b) Decoding Space

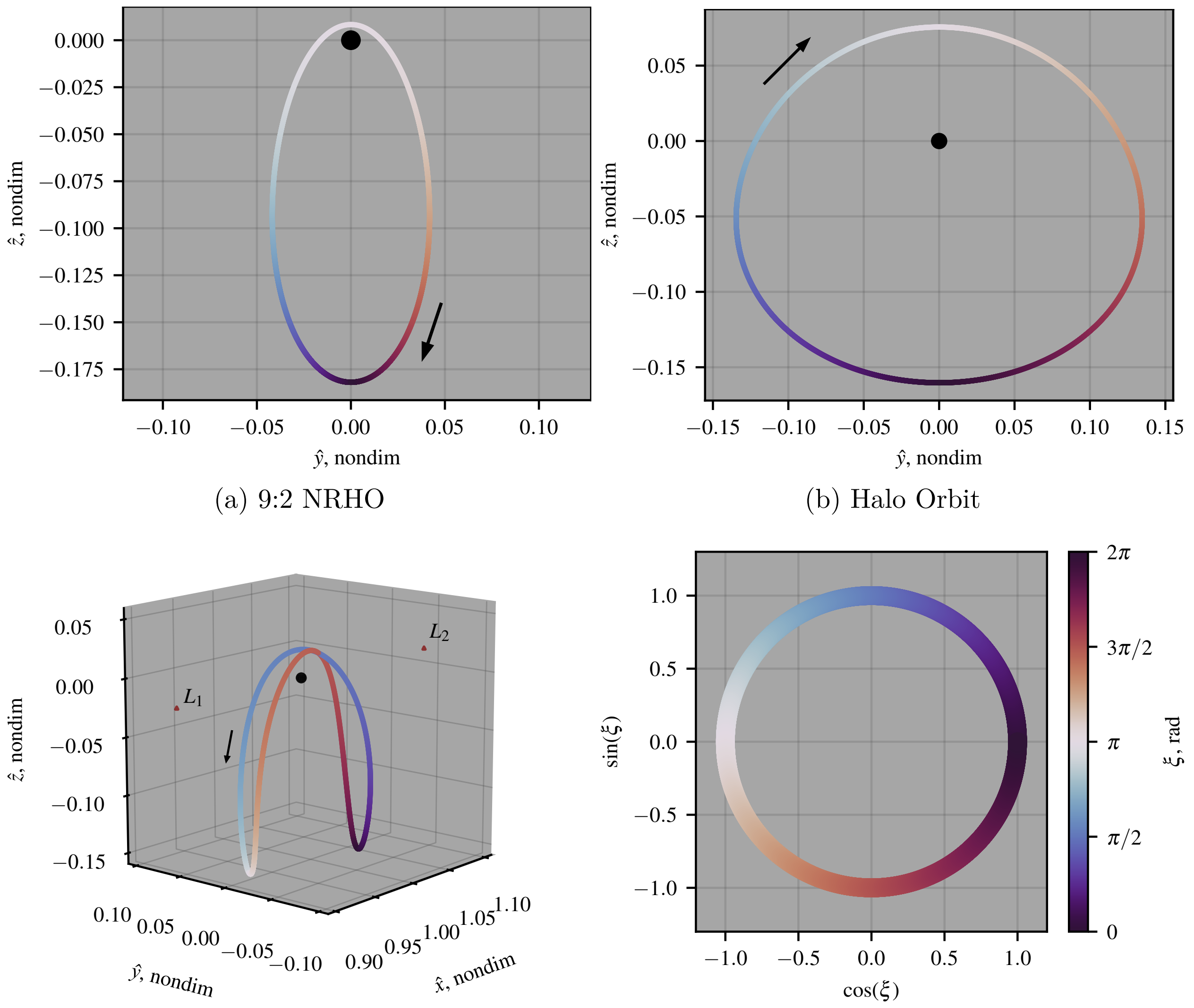
Figure 4.5. The observation and action spaces for ξ.

merically integrating the equations of motion specified in Equation (2.60). However, perfect state knowledge is generally absent in astrodynamics and, furthermore, this investigation employs termination criteria and other environment characterizations that are not included in the observation vector.

Agent performance throughout this investigation is greatly enhanced by augmenting the dynamical state, $\boldsymbol{q}$, with additional derived quantities to form the complete observation signal $\boldsymbol{o}$. In actor-critic RL formulations, both actor and critic networks receive the observation signal as inputs. Hence, both the policy and value functions are dependent on the selection of additional variables. Problems in this investigation involve various types of reference motion and, therefore, relative position and velocity vectors represent valuable data to the agent's performance. The relative information is computed simply as,

$$\boldsymbol{\delta\rho}\left(\boldsymbol{\rho}_{\mathrm{t}}\right) = \boldsymbol{\rho}_{\mathrm{t}} - \boldsymbol{\rho}_{\mathrm{ref}} = \begin{bmatrix} \boldsymbol{\delta r} & \boldsymbol{\delta v} \end{bmatrix} = \begin{bmatrix} \delta x & \delta y & \delta z & \delta \dot{x} & \delta \dot{y} & \delta \dot{z} \end{bmatrix} \in \mathbb{R}^6 \tag{4.12}$$

where $\boldsymbol{\rho}_{\mathrm{t}}$ is the position and velocity vector for the agent at time step t, and $\boldsymbol{\rho}_{\mathrm{ref}}$ is the position and velocity vector for the nearest neighbor along the given reference path. Here, "nearest" is defined as the state along the reference with the lowest L2 norm for the relative position and velocity. Note that this definition of "nearest" does not include time and, hence, is a



(a) 9:2 NRHO

(b) Halo Orbit

(c) Butterfly Orbit

(d) Angle encoding

Figure 4.6. Encoded angle ξ visualized across four sample CR3BP orbits in a rotating reference frame. The visualization of the NRHO (a) and halo orbit (b) are projections onto the rotating yz-plane.

normal rather than an *isochronous* correspondence. To quantify deviation from a reference, it is useful to compute relative dimensional deviations,

$$\boldsymbol{\delta}\tilde{\boldsymbol{r}} = (\tilde{l}^*)\boldsymbol{\delta r} \qquad\qquad \boldsymbol{\delta}\tilde{\boldsymbol{v}} = (\tilde{l}^*/\tilde{t}^*)\,\boldsymbol{\delta v} \tag{4.13}$$

If the reference orbit is represented as a set of discrete points, $\mathcal{R}^{\text{ref}}$, then the nearest state vector $\boldsymbol{\rho}_{\text{ref}}$ is defined as,

$$\boldsymbol{\rho}_{\text{ref}} \in \mathcal{R}^{\text{ref}} \quad \text{s.t.} \qquad k = |\boldsymbol{\delta\rho}|_2 = \sqrt{\boldsymbol{\delta\rho}\cdot\boldsymbol{\delta\rho}} \quad \text{is minimal} \tag{4.14}$$

The scalar value k is a function of both the position and velocity deviation. This relative state information vector $\boldsymbol{\delta\rho}$, along with the dynamical state $\boldsymbol{q}$ and other optional additional observations form the general observation signal employed in the applications in this investigation,

$$\boldsymbol{o}_{\text{t}} = \begin{bmatrix}\boldsymbol{q}_{\text{t}} & \boldsymbol{\delta\rho} & \text{additional observations}\end{bmatrix} \in \mathbb{R}^{13+j} \tag{4.15}$$

where j is the scalar number of optional additional observations that are incorporated. The elements of the observation signal must communicate sufficient information about the environment to enable the actor and critic networks to accurately characterize the system dynamics.

The additional observations are problem-dependent and are, thus, included here as optional parameters. Each application possesses different objectives and, hence, the additional observations are selected in support of those goals. The options evaluated in this investigation are summarized in Table 4.1. Preliminary results indicate that these optional parameters are helpful in some scenarios, and make converging on an effective policy more probable. While useful, these variables are termed "optional" due to the more significant impact of $\boldsymbol{q}$ and $\boldsymbol{\delta\rho}$. The selection of observations reflects a belief that a particular quantity is important for a given application, and is one way to indirectly incorporate expert knowledge into the learning process.

Table 4.1. Optional additional quantities included in the RL observation signal

$[\sin(\xi) \quad \cos(\xi)]$	In applications involving cyclic reference motion, such as station-keeping or arrival into a periodic orbit, it is useful to represent the location of $\boldsymbol{\rho}_{\text{ref}}$ using the trig encoding of the cyclic coordinate ξ, Equation (4.10).
$\{C_{\boldsymbol{o}}, C_{\text{ref}}, \delta C\}$	For problems in the CR3BP, the Jacobi constant, defined in Equation (2.32), communicates energy information, and is useful for problems that involve achieving motion at specific energy levels. This observation many include the Jacobi constant for the environmental observation ($C_{\boldsymbol{o}}$), reference state ($C_{\boldsymbol{o}}$), or the difference between them (δC).
r_{23}	In scenarios that involve close passes with the Moon, distance to the Moon (r_{23}) may be helpful to an RL agent.

4.3.3 Low-Thrust Action Signal Design

The action formalizes the means by which an agent alters its environment. During the training phase, the action is typically a stochastic process, where the NN outputs the mean value of each action parameter and uses these in conjunction with a derived variance to create a normal distribution for each value. The mean is essentially the agent's best guess for the action choice given a particular observation, and the variance is included to encourage exploration of the observation and action spaces. As in all policy optimization RL methods, over the course of training, the output of the network approaches an optimal policy. Once fully trained, exploration is no longer necessary, so the mean values are used directly to form a deterministic controller. For a NN controller, the bounds of the resulting action are governed by the selected activation function in the output layer of the network. The activation function employed for actor networks in this investigation is tanh and, therefore, the action values are bounded by $[-1, 1]$, and are decoded to reflect actual values. In actions defined in this investigation, an asterisk superscript denotes raw, bounded, outputs by the network.

Depending on the application, several options are available to decode NN output variables into low-thrust commands. In this investigation, low-thrust arcs are fundamentally comprised of three components: magnitude, direction, and time. Thrust magnitude and time variables may be either estimated by the NN or treated as constants. If both are constant, the NN is tasked with determining an effective thrust direction for a given time duration and thrust magnitude. Thrust magnitude and time are related by the ideal rocket question and, hence, it is effective in some applications for one to be held constant while the other is varied.

- Thrust direction The NN represents three components that comprise thrust direction $\boldsymbol{u}^* = [u_x^* \;\; u_y^* \;\; u_z^*]$, that are decoded to form a unit vector,

$$\hat{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T = \frac{\boldsymbol{u}^*}{|\boldsymbol{u}^*|} \tag{4.16}$$

  where $\hat{u}$ represents a constant thrust direction in the rotating reference frame, defined in Equation (2.54). While parameterizing thrust direction as a unit vector is straightforward, a potential drawback is an equation of constraint that is unknown to the controller, i.e., thrust direction is normalized after NN evaluation. While it seems appealing to reformulate the low-thrust parameterization to eliminate this constraint, note that including an angle as an output value for any bounded activation function results in a critical discontinuity in the available action space and, therefore, in the gradient of the action with respect to the observations. This discontinuity occurs because, once re-scaled to a range $[0, 2\boldsymbol{\pi}]$, the agent cannot perform an update step to push the output angle past either end bound.

  An interesting alternative formulation for a planar low-thrust control direction, detailed by Federici et al. [54], leverages a trig encoding similar to Equation (4.10), where duplication in the control space is removed by replacing the cosine estimation with a simple sign function, and may be extrapolated to the spatial case by including two additional parameters to model out-of-plane direction. Eliminating duplicate values from the action space may improve learning performance, however this possibility is not investigated.

- Thrust magnitude Neural network-approximated thrust magnitude, f^*, is simply rescaled by the maximum total allowable nondimensional thrust,

$$f = \left(\frac{f^* + 1}{2}\right) f_{\max} \in [0, f_{\max}] \tag{4.17}$$

Here, a minimal value for f^* results in nearly-ballistic coasting segments. An alternative option is to impose a nonzero lower-limit on thrust, $f = f_{\min} + \frac{1}{2}(f^* + 1)(f_{\max} - f_{\min})$. This formulation effectively removes nearly-coasting segments from simulation, which may or may not be beneficial depending on the specific application; however, this possible implementation remains future work.

- Thrust time Representing thrust time is challenging because the maximum allowable thrust time $\mathfrak{t}_{\max}$ must be pre-selected. The current selection process requires numerical experiments or imposing domain knowledge on the solution. The resulting thrust time $\mathfrak{t}$ is computed as,

$$\mathfrak{t} = \left(\frac{t^* + 1}{2}\right)\mathfrak{t}_{\max} \in [0, \mathfrak{t}_{\max}] \tag{4.18}$$

While zero provides an intuitive lower-bound for thrust time, the selection of an appropriate upper bound is challenging. If the selected value for $\mathfrak{t}_{\max}$ is too small, the spacecraft may not possess sufficient control authority to solve the given problem. If the value is too large, minor errors in control variable estimation become exacerbated, and the agent is more likely to converge on a propellant-inefficient policy.

A similar maximum-value scaling challenge arises in computing impulsive $\Delta\tilde{V}$ maneuvers with bounding the maximum allowable maneuver size. An alternative option is to employ a nonlinear decoding method, such as an exponential as implemented in an impulsive problem by Molnar [61]. While this approach may prove effective in some problems, it requires experimental tuning, and the introduced nonlinearity may pose challenges to the linear gradients computed during the backpropagation step of the neural network optimization process.

Timing Commands

For problems involving maneuver placement along periodic dynamical structures, the trig encoding/decoding process detailed in Section 4.3.1 is leveraged by an RL agent to estimate a suitable location for the next maneuver. In this trig-encoding approach, timing commands are implemented by tasking the NN with separately estimating the sine and cosine of the angle ξ^+; the '+' superscript here indicates that the time and angle are associated with the next maneuver (rather than the current state). This estimated angle ξ^+ is readily decoded to compute the time of the next maneuver, t^+_{po} using the decoding process detailed in Equation (4.10). The t^+_{po} time quantity is measured from an arbitrary fixed initial state along the periodic orbit; the relationship between ξ and t_{po} is detailed in Equation (4.10) and visualized in Figure 4.4. Hence, the timing action at time step $\mathtt{t}$, $\boldsymbol{a}_{\text{t}}$, is computed as,

$$\boldsymbol{a}_{\text{t}} = \begin{bmatrix} \sin(\xi^+) & \cos(\xi^+) \end{bmatrix} \in \mathbb{R}^2 \quad \rightarrow \quad t^+_{\text{po}} = \frac{\mathbb{P}}{2\pi}\texttt{atan2}(\sin(\xi^+), \cos(\xi^+)) \tag{4.19}$$

where $\mathbb{P}$ is the period of the underlying orbit. In this formulation, the coasting time between successive maneuvers, t, is a derived quantity, and is measured as the time between the current (t_{po}) and next (t^+_{po}) maneuver location, where $t_{\min}$ serves as a minimum allowable time between thrust segments. If the timing controller suggests a maneuver location such that $t < t_{\min}$, then one period of the orbit is added to the coast arc, $t = t + \mathbb{P}$. In a practical application, the selection of $t_{\min}$ would depend on many factors including operational constraints, science objectives, and navigation considerations.

4.3.4 Reward Signal Design

The reward function forms the feedback signal to communicate the effectiveness of a particular control choice to the agent. The general structure for reward functions within this investigation is given as,

$$r(\boldsymbol{o}_{\text{t}}, \boldsymbol{a}_{\text{t}}) = \begin{cases} c + \sum_i \beta_i \Gamma_i(\boldsymbol{o}_{\text{t}}, \boldsymbol{a}_{\text{t}}) & \text{Not deviated from reference} \\ r_{\text{f}} & \text{Final condition reached} \\ p & \text{Deviation criteria met} \end{cases} \tag{4.20}$$

where $c \in \mathbb{R}_{\geq 0}$ is a constant reward for not deviating at each time step, $\Gamma_i(\boldsymbol{o}_{\text{t}}, \boldsymbol{a}_{\text{t}}) \in \mathbb{R}_{\geq 0}$ are the individual components that comprise the overall reward function, and $\beta_i \in \mathbb{R}$ are tuning coefficients that adjust the sign and relative impact of each Γ_i term. The sign of β_i dictates whether its corresponding term serves as a bonus or penalty to the reward signal. Finally, $r_{\text{f}} \in \mathbb{R}_{\geq 0}$ is a final reward for applications that involve a terminal success condition, typically a bonus such that $r_{\text{f}} \in \mathbb{R}_{\geq 0}$, and $p \in \mathbb{R}_{\leq 0}$ is the penalty applied each time a deviation criterion is met. Options available to construct reward functions in this investigation are listed in Table 4.2, and specific implementations of these generic functions are detailed for each sample mission application.

Table 4.2. Components $\Gamma(\boldsymbol{o}_t, \boldsymbol{a}_t)$ for the Equation (4.20) reward function leveraged in this investigation

Symbol	Value	sign(β)	Explanation
$\Gamma_{\Delta\tilde{V}}$	$\Delta\tilde{V}_{\text{equiv.}}$	$-$	Penalty on propellant expenditure for a low-thrust maneuver, computed from the ideal rocket equation in Equation (2.57).
Γ_t	t	$+$	Coasting bonus $t \in \mathbb{R}_{\geq 0}$ for problems that involve maneuver scheduling. Including time in the reward encourages policies to maximize the time between maneuvers.
Γ_{ref}	$\mathrm{e}^{-\lambda k}$	$+$	Term to encourage relative state minimization to a reference trajectory, where $k(\boldsymbol{o}_t)$ is the distance to a reference state, Equation (4.14), and $\lambda \in \mathbb{R}_{\geq 0}$ is a tuning variable.

4.3.5 Nearest Neighbor Searching

The computation of the relative state for the reward and observation signals presents some practical challenges in implementation. First, the nearest reference state must be selected from a discrete set of states along the reference path, $\mathcal{R}^{\text{ref}}$. For an accurate assessment of nearness, the trajectory must include a large number of states. However, since the reward is computed at every time step, a brute force search through $\mathcal{R}^{\text{ref}}$ is computationally infeasible. To ease this computational burden, the nearest neighbor search is instead conducted by traversing through a K-dimensional tree (KD-Tree). A KD-Tree is a data structure frequently used for data clustering in unsupervised learning applications. This specialized type of binary search tree reduces the algorithmic complexity of the nearest neighbor problem from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$, a significant improvement when n is large. In this investigation, Scikit-learn's `neighbors.KDTree` implementation is employed to facilitate the nearest neighbor search process [112]. A similar approach is successful for autonomously locating neighbors in higher dimensional Poincaré maps [113], [114]. This application differs in that the neighbor search is conducted for only a single state, rather than intersections from two discrete sets.

In addition to time complexity improvements, approaching the nearness function from an unsupervised learning perspective allows for the inclusion of additional dimensions at no cost to algorithmic complexity. This functional extendability allows for a simpler transition of this guidance framework to higher-dimensional dynamical models. However, when including multiple variables in the nearness metric, the KD-Tree approach dictates that all variables are condensed into a single norm function. Thus, the process is more complex if the variables are scaled differently, since the norm is then biased toward the results with larger absolute values. This drawback is addressed by scaling all variables to, approximately, the same order of magnitude. This investigation demonstrates that nondimensional position and velocity in the CR3BP are close in magnitude and do not demand re-scaling, however, if units are dimensional, or if additional variables such as time or an angle are included, the individual variable scaling issue requires re-assessment.

5. GUIDANCE AND CONTROL ARCHITECTURES

GN&C comprise three main components of spaceflight that require specific considerations and novel approaches to enable autonomous flight systems. Within GN&C, guidance is tasked with planning a suitable trajectory that satisfies mission criteria, navigation is responsible for determining the position, velocity, and orientation of the spacecraft, and control involves determining steering commands in support of guidance objectives. In both pre-flight and ground-based analyses, where computational resources are abundant, G&C tasks are often formulated as optimization problems to minimize propellant expenditure or flight time. Many approaches exist for low-thrust optimal G&C, including nonlinear programming [115] and global optimization techniques [116]; an insightful discussion of various strategies for low-thrust trajectory optimization is provided by Pritchett [117].

Computational guidance methods are of recent interest for in-flight applications, and several recent flight software prototype investigations leverage iterative numerical methods to construct or update maneuver schedules while ensuring the necessary mission constraints remain satisfied. These methods seek to establish a balance between mission objectives and computing resource usage as the computational footprint of many state-of-the-art low-thrust maneuver planning approaches renders them impractical for onboard implementation. The autoNGC flight software system, under development at NASA Goddard Space Flight Center, provides several examples for proposed inflight GN&C methods, including a guidance scheme that implements a forward shooting algorithm for onboard maneuver planning optimization [4]. Similarly, the onboard GN&C architecture for Orion includes a targeting method capable of automatically altering maneuver plans based on navigation states [3]; Orion's targeting strategy seeks to reduce the onboard computational footprint by prioritizing feasibility over optimality [3]. Furthermore, targeting methods are commonly employed in multi-body SK approaches [5], [6], with demonstrated applicability in NRHO applications for the upcoming Gateway program [2], [7]. While targeting methods are effective in converging to feasible trajectories, they rely on sufficiently accurate startup solutions to achieve convergence and maintain current maneuver schedules. In the presence of large deviations or low-thrust propulsion options, an originally planned baseline reference trajectory may prove insufficient for achieving targeting convergence.

In contrast to more conventional iterative computational guidance methods, this investigation leverages RL techniques to train NN controllers in support of low-thrust G&C in cislunar space. A key challenge in employing a NN in this type of safety-critical problem is the abundance of mission constraints that a corresponding maneuver plan must adhere to. In RL applications, incorporating this type of mission criteria and constraints into the training environment presents several challenges. While adding penalties to the reward function encourages an agent to avoid particular behavior (such as the reference trajectory deviation criteria in the reward function in Equation (4.20)), it may also cause suboptimal convergence and unexpected behavior in policy optimization methods. For example, to encourage mass optimality, a thrust penalty may be included in the reward function to encourage propellant-efficient policies. However, this additional term may carry the unintended side effect of producing agents that avoid the penalty by simply never thrusting. Furthermore, leveraging reward penalties makes constraint violations less probable, but that probability is never provably zero; thus, there is no guarantee that the final NN-generated solution satisfies all the competing objectives. This investigation addresses this limitation through an alternative 'hybrid' G&C strategy that seeks to combine the efficiency of NN estimation with the safety assurances granted through conventional iterative approaches.

5.1 NN-G&C Options: Standalone vs. NNIT

Several recent approaches to low-thrust maneuver planning leverage a NN to estimate aspects of the control function, referred to in this investigation as **Neural Network-aided Guidance and Control (NN-G&C)**. To construct a NN-G&C algorithm, a NN is either trained via supervised learning or reinforcement learning. Once trained, the resulting NN may be employed in similar applications; thus, this investigation's NN-G&C architectures have potential additional applicability to supervised learning applications. In NN-G&C problems, NN controllers are demonstrated as potentially effective in overcoming challenging dynamical regions of space. However, the safety-critical nature of spaceflight poses practical barriers to implementing ML models onboard. In particular, concerns surrounding the complexity, reliability, and explainability of NNs must be addressed prior to deployment in onboard spaceflight applications. Future missions that seek to leverage the benefits of NN function approximation for G&C must contend with the corresponding challenges in introducing a 'black box' into a mission-critical system.

Two approaches to NN-G&C, visualized in the schematic in Figure 5.1, are evaluated in this investigation through sample low-thrust mission applications. The first approach, termed **Standalone NN-G&C**, implements a NN-G&C approach to directly compute a maneuver plan in the CR3BP stemming from a given navigation state. This maneuver plan is composed of a sequence of state variables and NN-generated thrust commands, originating from simulating a single episode of the RL training process. This standalone approach is by far the most common for continuous control applications in the astrodynamics G&C literature, and is favorable in applications that prioritize computational simplicity and establishing a direct connection to the underlying ML method. Alternatively, this investigation addresses the safety-criticality of incorporating a NN into the G&C architecture by blending ML with traditional methods. In this paradigm, rather than directly controlling the spacecraft, the NN-G&C simulated trajectory is transformed into an initial guess for a conventional iterative approach (in this case a multiple shooting targeting algorithm); this 'hybrid' architecture that is here termed **Neural Network-Initialized Targeting (NNIT)**. Directly incorporating traditional iterative methods into the training process broadens the available solution space, reduces the impact of chaotic dynamics on the training process, and ensures all mission criteria are satisfied by the corresponding solution. In both the standalone and NNIT approaches, the resulting CR3BP maneuver plan subsequently serves as an initial guess for a higher-fidelity $\mathcal{N}$-body ephemeris force model.

For both the standalone and NNIT approaches, while propellant efficiency is an important metric, RL is not an optimal control method and propellant minimization is not expected in the resulting agents. While NNs are extremely effective at function approximation, the control function is uncovered without any a-priori knowledge of desired behavior, and the estimation is never perfectly accurate. Thus, including domain-specific knowledge in the training process may improve propellant efficiency for specific applications, but it limits

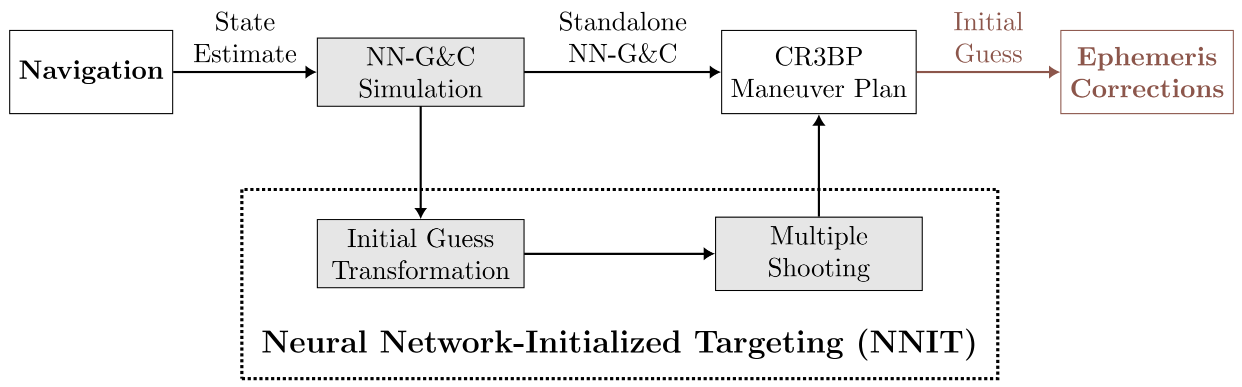


Figure 5.1. Information flow through G&C processes in this investigation.

applicability to other problems and is, therefore, not included in this investigation. In this application, propellant consumption is considered as one of many metrics when weighing the benefits and drawbacks of NN-G&C (for standalone or NNIT control), and is not the only factor in selecting a particular G&C strategy.

5.2 Neural Network-Initialized Targeting (NNIT) Overview

Neural Network-Initialized Targeting (NNIT) refers to the process of employing a NN to aid in the initialization process of a targeting algorithm. Evaluating the NNIT approach calls for a paradigm shift in discussing the NN-parameterized agent that results from an RL training process, where NNIT requires the NN to be understood as an estimation strategy for the control components of a conventional iterative algorithm's startup solution rather than viewing the NN as a standalone controller. Several benefits are observed in establishing a G&C architecture that blends NN and targeting approaches. First, the inclusion of differential corrections reduces the impact of NN estimation errors on the corresponding solution because, in an NNIT approach, the NN is tasked with identifying a basin of convergence rather than an exact result. Furthermore, the inclusion of targeting in NNIT offers significant benefit in ensuring that mission criteria are satisfied when compared with a 'standalone' NN-G&C approach, thus reducing any potential impact to mission safety.

Several related research efforts that utilize a NN in G&C tasks benefit from a similar function-approximation mindset to NNIT. In these paradigms, rather than the NN directly controlling the spacecraft, safety assurances are accomplished by tasking the NN with estimating components of a traditional G&C algorithm's startup process. The majority of previous work in this area involves leveraging behavior cloning (a type of supervised learning) to train a NN to estimate costate variables for indirect optimal control problems [9]–[11]. These types of blended approaches look at an existing G&C method and ask: "what component of this process would benefit from a NN approximation?" rather than "could this process be replaced with a NN approximation?" This prioritization of augmenting and improving over replacing is similarly reflected in this investigation.

As visualized in Figure 5.1, the addition of NNIT is implemented as a two-step process where, 1) the sequence of state-action signals from the NN-G&C simulation is transformed into an initial guess and 2) a multiple shooting targeting strategy is employed to identify a feasible solution to the problem. In combining NN-G&C and targeting paradigms for this framework, NNIT simultaneously benefits from the speed of NNs and the precision of shooting methods to ensure all mission constraints are satisfied. For example, prior work frequently assumes that a low-thrust engine possesses no lower bound on thrust, and is able to continuously and instantaneously alter both thrust direction and magnitude over the entire trajectory. The NN is trained prior to onboard deployment and, hence, cannot adjust a suggested control history in real time. One option for addressing this type of mission-specific constraint through leveraging the initial guess transformation and targeting components of NNIT. This investigation explores employing NNIT to alter and constrain low-thrust control variables, including the aforementioned lower bound on thrust magnitude, by applying transformation heuristics to construct an initial guess, and through explicit constraints within the targeting algorithm. In this investigation, control constraints are included to demonstrate the potential assurances granted though the NNIT process that may be absent in a standalone NN-G&C simulation; hence, these control considerations serve as stand-ins for other similar mission-specific criteria that motivate the NNIT approach.

5.3 RL Training Approaches for NNIT: Standalone vs. Integrated

Multiple options exist for implementing an RL training algorithm that constructs the NN-G&C component of an NNIT approach, as visualized in Figure 5.1. In the simplest approach, a 'standalone' NN-G&C training method tasks the RL agent with directly solving a given problem; standalone training results if the blue NNIT box is removed from Figure 5.1. The converged agent produced by standalone training is then either directly employed to construct a CR3BP maneuver plan (similarly referred to in this investigation as 'Standalone NN-G&C') or incorporated into a targeting initialization process for an NNIT framework. Alternatively, if the objective of training is to construct an NNIT algorithm, then an 'integrated' training approach (visualized in Figure 5.2) involves simulating NNIT at the end of each episode and leveraging the targeter's convergence characteristics to inform the terminal feedback signal provided to the agent.

The integrated RL training procedure, detailed in the blue box of Figure 5.1, involves simulating the NNIT algorithm upon each episode's termination (for $\mathfrak{t} = \mathfrak{n}$ where $\mathfrak{n}$ is the final time step). As visualized in Figure 5.1, NNIT involves two steps: transformation and targeting. In an integrated NNIT training algorithm, an additional step is introduced that allows NNIT to affect the terminal reward function. The terminal reward signal ($r_{\mathfrak{n}} = r(\boldsymbol{o}_{\mathfrak{n}}, \boldsymbol{a}_{\mathfrak{n}})$) is frequently leveraged in RL problems to communicate the degree of success of a particular episode. This terminal signal manifests in different RL problems; for example, $r_{\mathfrak{n}}$ could measure if an agent reaches a particular end result, if a game resulted in a win or loss, or if a poor decision caused an early termination event. In an integrated NNIT training approach, figures of merit of the targeting algorithm are leveraged to evaluate the degree of success, with select options summarized in Table 5.1. While intermediate rewards (for $\mathfrak{t} < \mathfrak{n}$) are useful to communicate the immediate impact of a thrust command (e.g., propellant consumption, distance from a reference trajectory), the inclusion of NNIT information in the terminal reward function provides direct feedback to the agent that quantifies its performance in initializing the differential corrections algorithm. This additional step in reward constructions provides a useful measure of 'success,' thus training the agent to compute actions that produce a higher probability of obtaining targeting convergence.

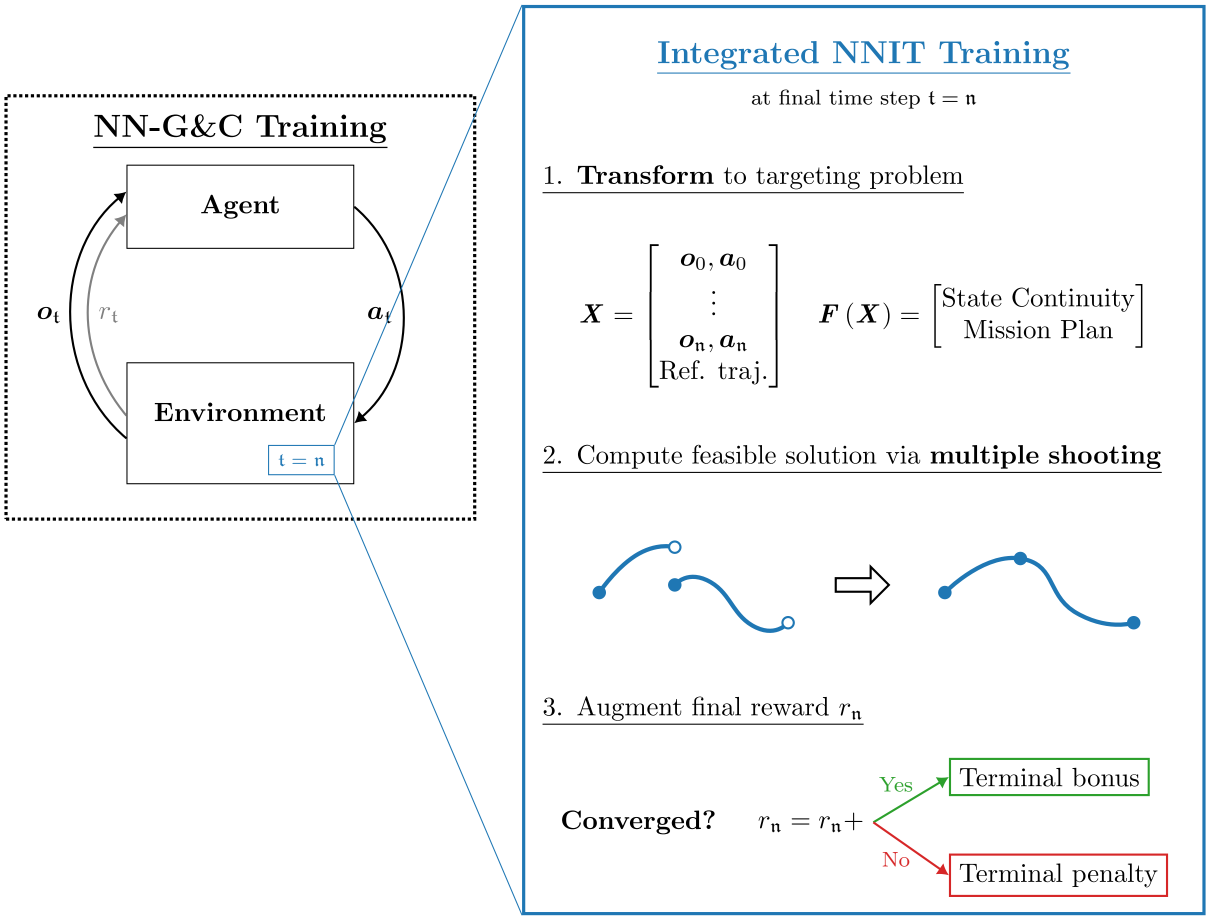

NN-G&C Training
Agent
Environment
$\boldsymbol{o}_t$
r_t
$\boldsymbol{a}_t$
$t = n$
Integrated NNIT Training
at final time step $t = n$
1. **Transform** to targeting problem
$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{o}_0, \boldsymbol{a}_0 \\ \vdots \\ \boldsymbol{o}_n, \boldsymbol{a}_n \\ \text{Ref. traj.} \end{bmatrix}$
$\boldsymbol{F}(\boldsymbol{X}) = \begin{bmatrix} \text{State Continuity} \\ \text{Mission Plan} \end{bmatrix}$
2. Compute feasible solution via **multiple shooting**
3. Augment final reward r_n
Converged?
$r_n = r_n +$
Yes
Terminal bonus
No
Terminal penalty

Figure 5.2. Reinforcement learning training options leveraged in this investigation. An 'integrated' training method simulates the NNIT process at the end of each episode, while a 'standalone' approach does not interact with the targeting algorithm until training is complete.

Several potential drawbacks are introduced by integrating NNIT into the RL training algorithm. Most notably, RL training is computationally expensive, occurring over hundreds of thousands of episodes. Integrated NNIT training requires the environment to simulate an iterative algorithm during each episode, causing the training time to markedly increase. This impact to the computational footprint is exacerbated by the fact that each iteration of the targeting algorithm employed in this investigation requires numeric integration across multiple trajectories that include additional terms for the variational equations of motion. However, one factor that mitigates the drawback of increased training time is that the targeting algorithm broadens the overall solution space, thus producing more opportunities for

Table 5.1. Options for applying NNIT bonus and penalty terms to the terminal reward function

Bonus	**Penalty**
Converges	Diverges
Converges in fewer iterations	Exceeds the pre-selected maximum number of iterations
Converges to a propellant-efficient maneuver plan	Converges to a maneuver plan with a high $\Delta\tilde{V}_{\text{tot.}}$

the NN to converge to an effective policy; expanding the solution space generally reduces the overall number of episodes to simulate. Furthermore, two techniques are leveraged in this investigation to overcome longer training times. First, parallel processing is employed to simulating many episodes at once. Second, the targeting error threshold ϵ, defined in Equation (3.21), is reduced during training to 1×10^{-8} under the assumption that the solutions that correspond to $\epsilon = 1 \times 10^{-8}$ and $\epsilon = 1 \times 10^{-12}$ lie within the same basin of convergence. Overall, evaluating a potential integrated NNIT training approach must weigh the benefits gained through NNIT informing the reward feedback with the increased computation time. In contrast, standalone training benefits from simplicity and computational efficiency, and may be advantageous in applications that observe fewer benefits from the inclusion of NNIT.

6. SAMPLE MISSION APPLICATIONS

This investigation explores several applications that focus on sample low-thrust CubeSat missions to cislunar space. These applications involve autonomously accessing and maintaining multi-body dynamical structures in the vicinity of the Moon despite large disturbances and limited onboard computational capabilities. Many factors motivate expanding the autonomous capability of future CubeSat missions that explore further from Earth. In particular, CubeSats operate on much smaller budgets than flagship missions and, therefore, autonomous capabilities dramatically reduce the operations cost associated with flying such missions. CubeSat mission applications are leveraged in this investigation to motivate the proposed Neural Network-aided Guidance and Control (NN-G&C) algorithms; however, as discussed in Section 2.4, the proposed methodology remains applicable to larger low-thrust spacecraft.

Within the cislunar vicinity, the motion of the sample CubeSat is primarily governed by its ion thruster and the gravity of Earth and the Moon. To model this motion for preliminary analysis, the low-thrust Earth-Moon CR3BP is employed to train NNs to support multiple G&C objectives in different phases of flight. To explore validity in a higher-fidelity model, CR3BP maneuver plans are, at times, subsequently evaluated in an $\mathcal{N}$-body ephemeris force model. Employing this higher-fidelity model informs the suitability of the proposed maneuver plans in practical scenarios that, for example, cannot neglect the gravitational influence of the Sun or the pulsation of the Moon in its noncircular orbit.

Three sample mission applications are considered in this investigation. First, Section 6.1 explores a CubeSat trajectory recovery scenario based on the conceptual mission plan for Lunar IceCube. In this low-thrust maneuver planning problem, the CubeSat has inadvertently departed from its desired NRHO orbit path and an NNIT approach is employed to autonomously determine an effective recovery plan that considers both thrust commands and timings. Next, Section 6.2 simulates a transfer phase between two periodic orbits where RL agents are tasked with returning to reference transfer paths despite large initial deviations. Heteroclinic transfers between L_1 and L_2 Lyapunov orbits serve as baseline trajectories to analyze the NN-G&C approach. Variations in control authority, transfer geometry, energy, and training algorithm are explored to understand the range of applicability, and potential drawbacks, of the proposed approach. Standalone NN-G&C is leveraged during training and preliminary analysis, with NNIT options evaluated after training is complete. In Section 6.3, RL agents are tasked with maintaining periodic orbits in the cislunar vicinity by determining strategies for placing maneuvers and for generating low-thrust commands. A highly flexible RL framework for SK is developed without dependencies on individual orbit characteristics or domain-specific knowledge. Furthermore, simulations demonstrate the RL maneuver placement strategy as highly effective in uncovering suitable maneuver locations and cadences with applicability to arbitrary control strategies. Finally, Section 6.4 presents higher-fidelity simulation and analysis of the NN-G&C-generated CR3BP maneuver plans, and evaluates their efficacy in an $\mathcal{N}$-body ephemeris force model.

6.1 Mission Recovery Scenario: NRHO Inadvertent Departure

A key challenge in enabling onboard autonomy is handling off-nominal situations that are typically addressed ad hoc by a team of specialists. For example, many prior investigations evaluate various SK methods along multi-body orbits, including techniques based on crossing control [7], [118] and dynamical systems theory [6], RL [63], [119], among others [120]. However, there is substantially less research on autonomous recovery given an inadvertent departure from a libration point orbit [121], with prior research efforts regarding NRHOs focusing primarily on intentional departure for both heliocentric disposal [122] and lunar impact [123]. In this chaotic region of space, large deviations over time rapidly render many traditional SK approaches ineffective. This sample mission application seeks to leverage RL to train a NN that determines accurate initial guesses and timings for low-thrust recovery maneuver sequences, both lengthening the possible recovery window for a potential onboard application and demonstrating the utility of blending RL with traditional G&C techniques.

In this simulated 'runaway' spacecraft scenario, an RL agent is tasked with identifying a control history that returns a CubeSat to an Near Rectilinear Halo Orbit (NRHO) orbit path given an inadvertent departure. Rapid low-thrust maneuver planning is deemed necessary in this autonomous scenario, where deviation over time from the planned trajectory renders stationkeeping ineffective and demands an alternative approach to determine an effective recovery plan. The low-thrust augmented CR3BP is employed during training in this analysis because it accurately represents the NRHO region while remaining sufficiently low fidelity for initial analysis of the NNIT scheme. To demonstrate applicability in a realistic mission setting, the modeled scenario bases its thrust characteristics and orbit selection on the Lunar IceCube mission plan.

This sample mission application is leveraged to demonstrate the utility of employing an NNIT framework and the associated 'integrated' RL training process, detailed in Sections 5.2 and 5.3, respectively. This NNIT approach involves tasking the RL-trained[1] NN with convergence basin identification for a low-thrust recovery multiple shooting algorithm that returns the spacecraft to the NRHO. Initialization performance is directly improved through the integrated NNIT training procure, depicted in Figure 5.2. In this integrated approach, the selected targeting method directly influences the agent's feedback signal, thus increasing the probability of accurate initialization decisions over time. In constructing recovery paths for the simulated spacecraft, the proposed NNIT algorithm significantly extends the recovery window compared to a crossing-control orbit maintenance approach and exhibits promising results in identifying both the timing and control components for low-thrust maneuver plans.

Inadvertently departing from a pre-conceived mission plan risks jeopardizing the continued success of the mission. For spacecraft moving along an NRHO, regular SK maneuvers are necessary to maintain the orbit [7]. However, in a missed-thrust scenario where multiple SK maneuvers are skipped, determining a recovery path is challenging, with significant additional complexity introduced for autonomous, low-thrust scenarios. While employing an NRHO as a nearly stable baseline orbit is advantageous due to a slow departure rate, it is correspondingly challenging to re-enter this region of near-stability. In addressing this challenge, Zimovan-Spreen et al. present a thorough analysis of NRHO departure characteristics and offer several potential recovery methods [121]. In their investigation, the suggested recovery response is determined based on two potential recovery 'regimes' categorized by departing flow. Regime 1 signifies that a standard SK approach readily converges

[1]↑TD3 [92] is employed as the training algorithm in this sample scenario, discussed in further detail in Section 4.2.2.

to a recovery plan, and Regime 2 encompasses situations where the departing flow remains in the lunar vicinity but is not recoverable by straightforward SK maneuvers (a third regime for cases that depart the lunar vicinity entirely is identified as future work). The authors leverage known structures and dynamical systems theory to formulate an impulsive recovery framework for trajectories in 'Regime 2'. While offering many potential recovery options, the computational footprint, human-in-the-loop interaction, and impulsive engine assumption present significant barriers to any potential onboard, autonomous implementation for low-thrust spacecraft. This investigation addresses a subset of recovery scenarios associated with a low-thrust analog to Zimovan-Spreen et al.'s 'Regime 2' by employing a deep neural network to aid low-thrust recovery planning for cases where SK no longer maintains the desired NRHO.

6.1.1 Spacecraft Model: Lunar IceCube

To more accurately reflect a realistic scenario, this investigation bases its spacecraft model and sample application on the originating conceptual mission plan for the Lunar IceCube mission. The Lunar IceCube spacecraft is a 6U CubeSat equipped with a Busek Ion Thruster 3-cm (BIT-3) low-thrust propulsion system [75], with specific spacecraft and engine characteristics listed in Table 6.1. A comparison between the propulsive capability of the Lunar IceCube spacecraft and other previous and planned low-thrust missions is listed in Table 2.2, and visualized in Figure 2.7.

Table 6.1. Lunar IceCube spacecraft data

Parameter	**Symbol**	**Value**
Initial mass	$\tilde{m}_{3,0}$	13.487 kg
Specific impulse	$\tilde{I}_{\text{sp}}$	2156 s
Maximum thrust	$\tilde{f}_{\text{max}}$	1.1 mN
	f_{max}	0.029 92 nondim

Launched as a secondary payload on the Artemis 1 vehicle, the Lunar IceCube spacecraft is designed to prospect, locate, and study ice, vapor, and liquid water features on the Moon from a highly-inclined 100 km perilune low-lunar science orbit [76], [124]. Lunar IceCube's mission plan incorporates a 9:2 lunar synodic resonant L_2 southern NRHO, depicted in Figure 6.1, as a staging orbit prior to the spiral-down leg into its science orbit [125], with at least one SK maneuver necessary to maintain the NRHO. The 9:2 NRHO is of particular recent interest as the planned baseline orbit for the Lunar Gateway [126]. This investigation simulates a scenario in which a disturbance causes a spacecraft modeled after Lunar IceCube to deviate significantly from the desired NRHO. A low-thrust maneuver plan is sought to return the spacecraft to the NRHO region. While Lunar IceCube is leveraged to model a realistic scenario, the proposed methodology is potentially applicable to other types of spacecraft, thrust capabilities, and reference trajectories.

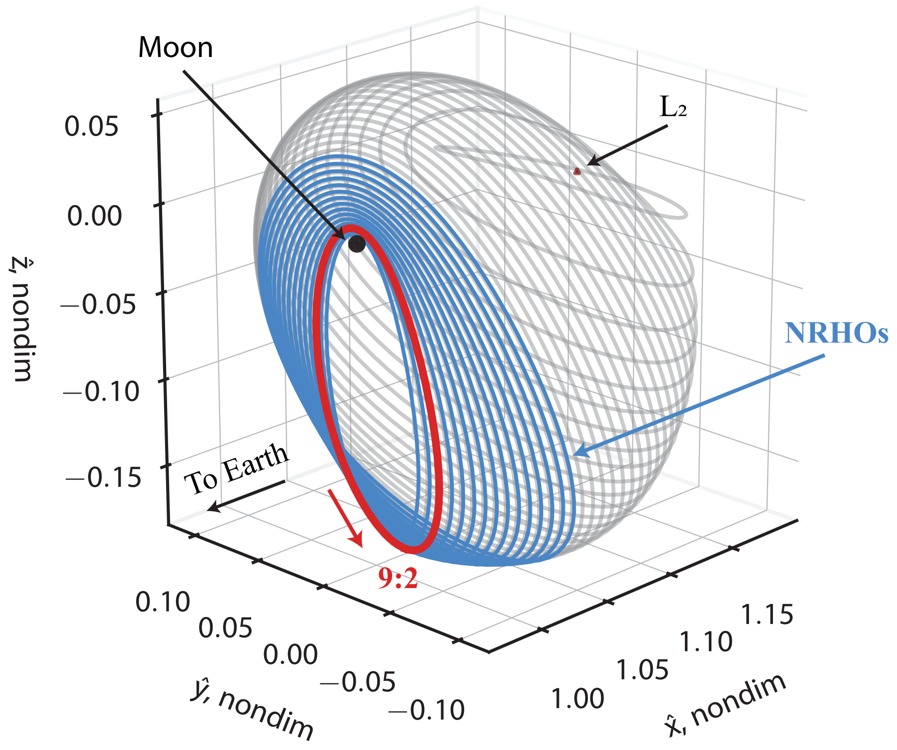


Figure 6.1. The Earth-Moon L_2 southern halo orbit family in the CR3BP, where blue denotes the NRHO region, and red signifies the NRHO at 9:2 lunar synodic resonance.

6.1.2 Learning Environment Implementation (Integrated NNIT)

In each episode of the RL training process, the agent is tasked with recovering in response to an induced departure from a reference NRHO. Prolonged departures are generated using a similar methodology to [121], where a randomly-selected initial state along the orbit is corrupted, with perturbation values for each position and velocity component sampled from independent and identically distributed Gaussian distributions (3σ: 10 km, 10 cm/s). These perturbed states are then propagated for $20-30$ days to simulate an unintended departure from the NRHO. This time range introduces significant deviations that still generally admit 'direct' returns to the NRHO. The end of the inadvertent departure trajectory serves as the initial condition for the RL episode.

Each episode consists of a fixed number of time steps ($\mathfrak{n} = 10$ in this investigation), after which the resulting low-thrust trajectory serves as an initial guess for the targeting algorithm defined by Equations (6.1) and (6.2). At each time step, the agent selects a thrust magnitude, direction, and time, which are then numerically integrated by the environment. The agent, therefore, seeks to determine a sequence of $\mathfrak{n}$ low-thrust arcs that terminate sufficiently close to the NRHO to achieve subsequent targeting convergence. This NNIT architecture is integrated directly into the training process to provide informed feedback to the agent regarding targeting outcomes. Furthermore, experiments demonstrate that this framework admits long-duration coasting segments through the selection of near-zero thrust magnitude values. Including coasting segments and allowing for significant variations in propagation time enables the proposed learning process to demonstrate remarkable adaptability in selecting a maneuver plan.

Previous research efforts in implementing RL to reach a desired orbit frequently employ arbitrary heuristics to formalize termination conditions, including the application of predefined position and velocity thresholds [127], [128]. A thresholding approach is particularly challenging in NRHO problems, where more significant velocity variations occur at the close perilune passes and, conversely, position deviations grow near apolune. If heuristic thresholds are employed in such a problem, the agent is unlikely to achieve satisfactory relative position and velocity values simultaneously due to their inverse relationship. Instead, this investigation leverages the convergence properties of the integrated NNIT targeter to produce a more informed terminal "success" metric for accessing a pre-selected destination orbit.

Targeting Algorithm

The integrated NNIT RL training process, presented in Sections 5.2 and 5.3, involves transforming an NN-G&C maneuver plan into an initial guess for a targeting algorithm[2]. Recall that, in this investigation, 'integrated' signifies that the targeting algorithm is simulated during each training episode and, thus, directly influences the RL reward function. This targeting method, like all variations of Newton's method, requires a suitably accurate initial guess to achieve convergence. For this recovery problem, the initial guess is comprised of: 1) $\mathfrak{n}$ low-thrust arcs with a control variable time history computed by a NN controller and 2) $\mathfrak{m}$ stacked ballistic revolutions of the desired periodic orbit (the 9:2 NRHO). By employing a minimum norm update, a solution is sought that maintains the stacked orbit states close to their original value, typically producing motion in the immediate vicinity of the desired orbit without precisely constraining the final state. Constraining the terminal condition to match a desired six-dimensional state vector presents additional challenges that are not always necessary or practical in the presence of navigation uncertainty and maneuver execution errors. An alternative targeting approach is to constrain the final state to be on the stable manifold, as detailed by Haapala and Howell [129]. While the stable manifold constraint approach preserves energy and periodicity, it poses challenges for a rapid closed-loop system due to the sensitivities in response to the initial time parameters. Stacking revolutions is a more flexible approach if consistent closed-loop convergence is required, and $\mathfrak{m} = 4$ revolutions is employed in this mission application.

[2]↑A detailed discussion of low-thrust multiple shooting in the CR3BP is presented in Section 3.4.

The multiple shooting scheme is implemented by defining free variable and constraint vectors, $\boldsymbol{X}$ and $\boldsymbol{F}(\boldsymbol{X})$, respectively, as detailed in Section 3.3. For the proposed NNIT approach in this NRHO recovery application, these vector quantities are defined as,

$$\boldsymbol{X} = \left(\underbrace{\left\{ \boldsymbol{\rho}_i \right\}_{i=1}^{\mathfrak{n}+\mathfrak{m}}}_{\text{States}} \quad \underbrace{\left\{ t_i \ \ \eta_i \right\}_{i=0}^{\mathfrak{n}+\mathfrak{m}}}_{\text{Propagation times}} \quad \underbrace{\left\{ \breve{f}_i \ \ \theta_i \ \ \kappa_i \right\}_{i=0}^{\mathfrak{n}}}_{\text{Controls}} \right)^T \tag{6.1}$$

$$\boldsymbol{F}(\boldsymbol{X}) = \left(\underbrace{\left\{ \boldsymbol{\rho}_{i,\mathrm{f}} - \boldsymbol{\rho}_{i+1} \right\}_{i=0}^{\mathfrak{n}+\mathfrak{m}-1}}_{\text{State Continuity}} \quad \underbrace{\left\{ t_i - \eta_i^2 \right\}_{i=0}^{\mathfrak{n}+\mathfrak{m}}}_{\text{Ensure } t_i>0} \right)^T \tag{6.2}$$

Throughout the differential corrections process, the state vector $\boldsymbol{\rho}_i$ at each intermediate patch point is allowed to vary; the initial state $\boldsymbol{\rho}_0$ is assumed fixed. Propagation times t_i are similarly included as design variables for both thrusting and coasting segments, with additional slack variables η_i incorporated to enforce the $t_i > 0$ condition[3]. Thrust magnitude and direction are design variables for the initial $\mathfrak{n}$ thrust segments, with direction represented in spherical coordinates (θ and κ are defined in Equation (2.54)). Furthermore, to ensure that the thrust magnitude remains bounded, i.e., $f_i \in [0, f_{\max}]$, the parameterized thrust magnitude $\breve{f}_i$, defined in Equation (3.33), is employed as a free variable. The design variables, together, comprise the $\boldsymbol{X}$ vector, which is updated iteratively to satisfy the desired $\boldsymbol{F}(\boldsymbol{X}^*) = \boldsymbol{0}$ constraints. In this investigation, the enforced constraints in Equation (6.2) are state continuity between successive patch points and positive value inequality on time variables.

Observation Signal

The observation signal informs the agent of its current dynamical state and its position and velocity relative to the desired orbit path. The method for defining observation signals leveraged throughout this investigation in specified in Equation (4.15) where, for this recovery application, the relative vector $\boldsymbol{\delta\rho}$ provides information about the spacecraft's behavior relative to the NRHO. Furthermore, the included observations from Table 4.1 are: 1) the trig-encoded location[4] of the nearest state along the NRHO (represented as $\sin(\xi)$ and $\cos(\xi)$),

[3]↑Inequality constraint options are discussed in Section 3.3.1
[4]↑The trig encoding method for representing location along a periodic orbit is detailed in Section 4.3.1.

and 2) the difference in the Jacobi constant value δC. These values, together, form the complete observation signal,

$$\boldsymbol{o}_{\mathfrak{t}} = \begin{bmatrix} \boldsymbol{q}_{\mathfrak{t}} & \boldsymbol{\delta\rho} & \sin(\xi) & \cos(\xi) & \delta C \end{bmatrix} \in \mathbb{R}^{16} \tag{6.3}$$

that serves as an input to both the actor and critic neural networks at time step $\mathfrak{t} \in [1, \cdots, \mathfrak{n}]$.

Action Signal

In constructing a maneuver plan in response to the induced NRHO departure, the agent is allowed to select low-thrust thrust direction, magnitude, and time values along each of the $\mathfrak{n}$ segments, together forming the action signal at time step $\mathfrak{t} \in [0, \mathfrak{n}]$,

$$\boldsymbol{a}_{\mathfrak{t}} = \begin{bmatrix} f^* & u_x^* & u_y^* & u_z^* & t^* \end{bmatrix} \in \mathbb{R}^5 \tag{6.4}$$

Each action value is bounded by $[-1, 1]$ due to the selection of tanh as the output activation function in the actor NN, specified in Table 6.2. The orientation vector for the maneuver $\hat{u}$ is fixed in the rotating reference frame, as defined in Equation (4.16). The thrust magnitude action, Equation (4.17), allows the agent to determine the throttle level at a given point in the recovery. This parameterization enables the selection of $f^* \approx -1$ (which decodes to $f \approx 0$) to produce coasting segments, thus allowing the agent to identify situations when thrusting is disadvantageous.

Table 6.2. Configuration of actor and critic neural networks for TD3 algorithm employed in the NRHO recovery sample mission application.

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

The action variable for thrusting time, Equation (4.18), enables the agent to determine the duration of both thrusting and near-coasting ($f \approx 0$) segments. The selection of thrust time and magnitude values in tandem results in a form of maneuver placement, where coasting segments are estimated such that future thrust arcs are placed in favorable locations. As defined in Equation (4.18), the decoding process for thrust time require a priori selection of a maximum allowable time for each segment $\mathfrak{t}_{\max}$. This mission scenario employs $\mathfrak{t}_{\max} = 24$ hours to allow for long-duration thrust and coast arcs necessary in the recovery phase. However, this choice limits the ability to accurately estimate the short burns often employed for SK, thus, increasing the NNIT cost at low error levels. A separate analysis for leveraging the proposed RL methods in support of SK is presented in Section 6.3. Furthermore, the total number of time steps in the recovery environment is limited to $\mathfrak{n} = 10$ in this sample application; thus each NN-G&C recovery maneuver plan cannot exceed 10 days. This overall upper limit restricts the possibility of recovery responses that involve longer times of flight.

Reward Signal

In this integrated NNIT approach, the reward response is determined based the fulfillment of the final condition criteria. For non-terminal states, the reward signal encourages the agent to minimize its relative distance to the NRHO in both position and velocity, with a small control penalty added to encourage propellant-efficient solutions and to allow near-ballistic segments to emerge. Trials are terminated when a large deviation threshold is exceeded, or the agent reaches $\mathfrak{n}$ time steps. Relative distance is measured from the nearest neighbor along the reference orbit, as defined in Equations (4.12) and (4.13). Deviation is defined as the agent impacting the Moon or exceeding a pre-defined relative position or velocity threshold, defined as,

$$\text{Deviation from NRHO} := \begin{cases} \tilde{r}_{23} < 1737.4\,\text{km} & \text{or} \\ |\boldsymbol{\delta}\tilde{\boldsymbol{r}}| > 8000\,\text{km} & \text{or} \\ |\boldsymbol{\delta}\tilde{\boldsymbol{v}}| > 250\,{}^{\text{m}}\!/\!{}_{\text{s}} & \end{cases} \tag{6.5}$$

Together, the reward function at time step $\mathfrak{t}$, as defined in Table 6.3, is formalized for this recovery application as,

$$r_{\mathrm{t}} = \begin{cases} \beta_{\mathrm{ref}}\Gamma_{\mathrm{ref}} + \beta_{\Delta\tilde{V}}\Gamma_{\Delta\tilde{V}} & \mathfrak{t} < \mathfrak{n}, \text{not deviated from reference NRHO} \\ r_{\mathrm{f}} & \mathfrak{t} = \mathfrak{n}, \text{final condition reached} \\ p_1 & \text{deviation criteria met} \end{cases} \tag{6.6}$$

where $\Gamma_{\mathrm{ref}} = \mathrm{e}^{-\lambda k}$ and $\Gamma_{\Delta\tilde{V}} = \Delta\tilde{V}_{\mathrm{equiv.}}$ are defined in Table 4.2. The relative weighting between the competing state minimization and propellant consumption objectives results from the selection of $\beta_{\mathrm{ref}} \in \mathbb{R}_{\geq 0}$ and $\beta_{\Delta\tilde{V}} \in \mathbb{R}_{\leq 0}$. The final integrated NNIT reward is defined here as,

$$r_{\mathrm{f}} = \begin{cases} b_{\mathrm{conv}} - \beta_3 \Delta\tilde{V}_{\mathrm{tot.}} & \text{targeter converges} \\ p_2 & \text{targeter does not converge} \end{cases} \tag{6.7}$$

Through this terminal reward, the RL process performs updates to NN trainable parameters such that targeting convergence becomes more probable. Together, each term that comprises the total reward function is defined in Table 6.3, along with the corresponding suggested values employed in this sample mission application.

Table 6.3. Components for reward functions leveraged in the NRHO sample mission application

Symbol	**Description**	**Value(s)**
β_{ref}	Constant coefficient for relative distance term	$\beta_{\mathrm{ref}} = 1$
k	Relative state norm, defined in Equation (4.12)	-
λ	Scaling factor that adjusts the gradient of the reward	$\lambda = 300$
$\beta_{\Delta\tilde{V}}$	Constant coefficient for $\Delta\tilde{V}_{\mathrm{equiv.}}$ term, where $\Delta\tilde{V}_{\mathrm{equiv.}}$ is measured $^{\mathrm{m}}/_{\mathrm{s}}$	$\beta_{\Delta\tilde{V}} = 0.03$
p_1	Penalty for violating deviation criteria	$p_1 = -25$
$\Delta\tilde{V}_{\mathrm{tot.}}$	Shorthand for the sum of the $\mathfrak{n}$ $\Delta\tilde{V}_{\mathrm{equiv.}}$ values (in $^{\mathrm{m}}/_{\mathrm{s}}$)	-
β_3	Constant relative weighting coefficient for $\Delta\tilde{V}_{\mathrm{tot.}}$ term in terminal bonus	$\beta_3 = 0.45$
b_{conv}	Bonus applied for achieving targeting convergence	$b_{\mathrm{conv}} = 28$
p_2	Penalty for targeter not converging in ≤ 10 iterations	$p_2 = -2$

6.1.3 Simulation Results

An NRHO recovery problem is simulated in this sample mission application to illustrate the proposed NNIT framework and integrated training method as applied to a spacecraft modeled after Lunar IceCube. During training, recovery situations are simulated by introducing deviations from the reference orbit and propagating those deviations forward in time. To verify the validity of this deviation modeling, a higher-fidelity simulation is implemented. This scenario assumes the spacecraft previously reached the 9:2 NRHO and subsequently executed several small SK maneuvers using the “Finite Burn” variant of x-axis control (XAC) detailed in [7]; the XAC method, in general, is discussed further in Section 6.3.2. In reality, the conceptual mission plan for Lunar IceCube involves accessing the NRHO over a shorter period of time; this scenario considers additional SK maneuvers and a longer recovery window to evaluate algorithmic performance. For this simulation, during the NRHO SK phase, 3σ errors of 10 km, 10 cm/s are modeled as an uncorrelated Gaussian process for each position and velocity component for both orbit determination and orbit insertion errors; a fixed 0.03 cm/s maneuver execution error is added in a random direction for each SK maneuver. Stationkeeping maneuvers are implemented at apolune at a once-per-orbit cadence. Leveraging XAC to initialize the departure simulation provides a more realistic scenario to evaluate NNIT performance.

Unintended Departure Simulation

A single representative scenario is simulated to illustrate the RL recovery process. In this sample problem, following its fifth orbit maintenance maneuver, an unanticipated thruster failure event is assumed for the simulated spacecraft that causes SK to cease entirely. An inadvertent departure from the NRHO results, causing the ‘runaway’ spacecraft to slowly drift from its desired orbit path, as depicted in Figure 6.2. Due to the quasi-stability of the 9:2 NRHO, departure slowly evolves over approximately 60 days until the spacecraft eventually leaves the vicinity of the orbit entirely. All times along the departure path are calculated from the time of the final SK maneuver.

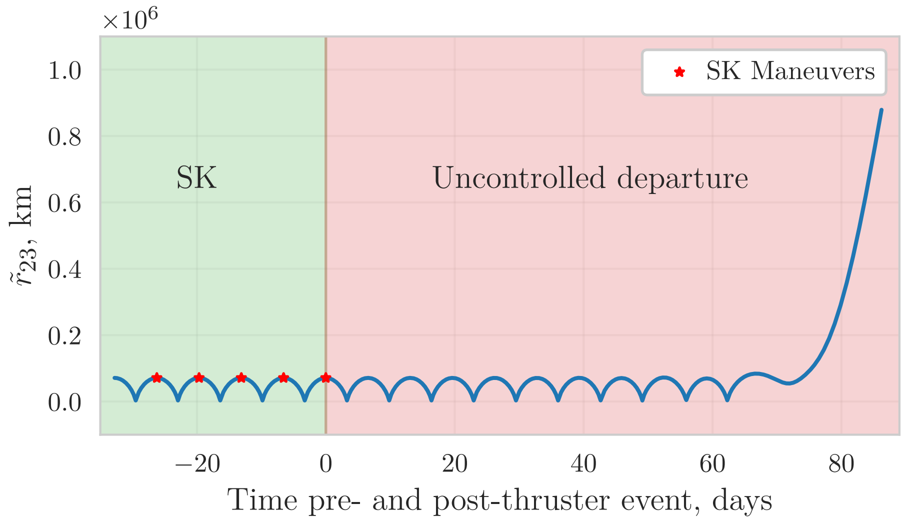


Figure 6.2. Distance between the spacecraft and the Moon during the sample recovery simulation's SK and uncontrolled departure phases.

At various points throughout the departure segment, XAC and NNIT are simulated to guide the spacecraft back to the 9:2 NRHO in the event that thruster functionality is suddenly restored. Leveraging XAC as a comparison metric for recovery is motivated by Zimovan-Spreen et al.'s investigation [121] and aids in designating scenarios when recovery, rather than SK, is necessary. Several XAC recovery trends identified by Zimovan-Spreen et al. emerge in this analysis. However, slight differences arise in XAC performance and the reported departure times due to differences in dynamics modeling, assumed spacecraft propulsion models, error sources, and XAC implementation. Zimovan-Spreen et al. employ a variant of XAC to compute impulsive maneuvers that accommodate phasing considerations by, at times, constraining future perilune passage times in an $\mathcal{N}$-body ephemeris force model. In contrast, the implemented XAC algorithm for low-thrust, suggested in [7], targets only the downstream $\dot{x}$ value in the CR3BP rotating frame, and leverages the ideal rocket equation to convert impulsive SK maneuvers to finite-burn, low-thrust arcs. This transformation remains accurate for multiple-hour burns but begins to introduce significant error as thrust times grow. Furthermore, as also observed by Zimovan-Spreen et al., a single converged XAC maneuver does not necessarily mean the orbit is 'stabilized'. Therefore, recovery costs for XAC are only reported if the algorithm continues to function and repeats the NRHO geometry for 20 subsequent periods of the orbit. As a SK algorithm, XAC is not expected to perform consistently in a recovery process. Instead, XAC is simulated here to provide insight into recovery and conditions when it may be required.

Performance of the XAC (blue) and NNIT (orange) processes over time for the sample scenario, Figure 6.2, are compared in Figure 6.3, with vertical yellow lines denoting apolune times. While propellant expenditure is an insightful metric in comparing the maneuver plans that result from XAC and NNIT, neither approach directly optimizes for fuel consumption. Rather, XAC identifies low-cost maneuvers by leveraging the symmetry of the underlying orbit, whereas NNIT seeks to identify a lower-cost basin of convergence for a targeting scheme. Occasionally, for XAC, a high recovery cost is incurred at the second or third SK maneuver and, therefore, the reported recovery cost sums the equivalent $\Delta\tilde{V}_{\text{equiv.}}$ of three sequential SK maneuvers. As expected, XAC initially offers lower-cost maneuvers than NNIT. After 27 days, XAC still maintains the orbit but is no longer dependably more propellant-efficient than NNIT. After 39 days, in this sample simulation, XAC no longer stabilizes the orbit. Conversely, during the initial departure segment, NNIT determines a high recovery cost, indicating that it is inefficient to employ the NNIT method directly for SK (though it does maintain the orbit). However, as deviations grow, NNIT begins to perform comparably to XAC. Once outside the region of applicability for short-horizon SK, NNIT extends the recoverable window for approximately three orbital periods (about 21 days). This extended recovery period results from an autonomous process and offers low-thrust maneuver plans without the additional complexities of Zimovan-Spreen et al.'s approach [121]. As expected, recovery costs generally increase over time during this extended window. As visualized in Figure 6.2, the spacecraft begins to significantly deviate from the NRHO path after 65 days, after which alternative recovery options must be considered.

Several key features of the NNIT recovery process are illustrated when examining maneuver plans at specific times. Recall that, in all presented cases, the NNIT approach consists of 10 NN-identified low-thrust arcs that serve as an initial guess for the targeting algorithm defined in Equations (6.1) and (6.2). Recovery paths originating at 29 days beyond the final stationkeeping maneuver, Figure 6.2, are depicted in Figure 6.4 to highlight the NNIT maneuver planning process that illustrates a representative example of the region where NNIT

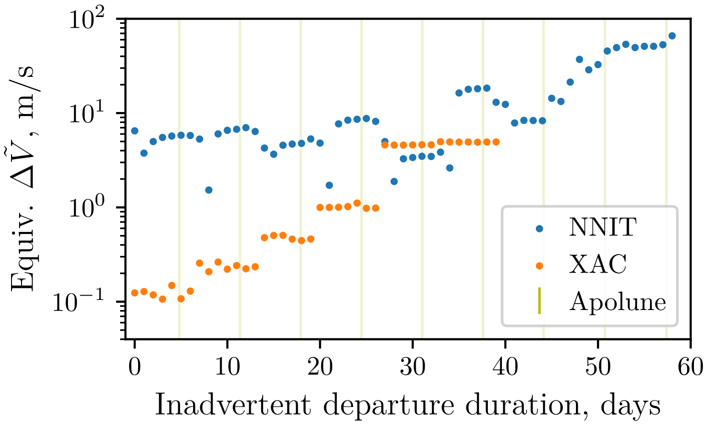
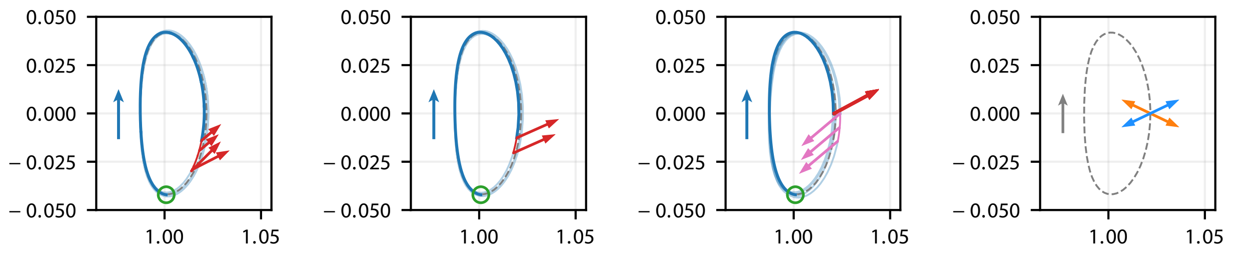


Figure 6.3. Comparison of the cost to recover to the 9:2 NRHO reference motion using SK (orange) and NNIT (blue), where vertical yellow lines denote apolune times. The NNIT method extends the recoverable window by 21 days.

and XAC perform comparably. In this case, at 29 days, NNIT suggests a thrust plan with $\Delta\tilde{V}_{\text{equiv.}} = 2.8\,{}^{\text{m}}\!/\!_{\text{s}}$ compared to $\Delta\tilde{V}_{\text{equiv.}} = 4.4\,{}^{\text{m}}\!/\!_{\text{s}}$ for XAC across two maneuvers. The NNIT and XAC maneuver plans (represented as x-y projections for clarity) are plotted in Figures 6.4(a) and 6.4(c), respectively, where green circles signify the departing trajectory state at 29 days and the gray dashed lines correspond to the underlying 9:2 NRHO. Notably, the NNIT maneuver plan independently identifies apolune as an effective location for its recovery maneuvers. While each of the 10 control choices for the NNIT process is implemented as a set of low-thrust arcs, the first 6 maneuvers introduced under this plan each possess a thrust magnitude less than 0.1 % of maximum thrust, thus rendering the low-thrust effect on the system insignificant for the first 4.66 days. A challenge in previous NN-based investigations for low-thrust G&C is the assumption of constant thrust. However, a notable feature of the NN trained in this investigation is its consistent suggestions of coasting segments when the spacecraft is near perilune. This knowledge that apolune maneuver placement is effective along NRHOs is consistent with previous studies [2], [118]. The ability of the RL training process to independently uncover an effective known solution suggests further applicability of RL in problems where such a priori knowledge is absent.

In the 29-day post-failure simulation for NNIT, Figure 6.4(a), the 4.66-day coasting segment suggested by the NN is followed by four successive thrust arcs (red) ranging from 1

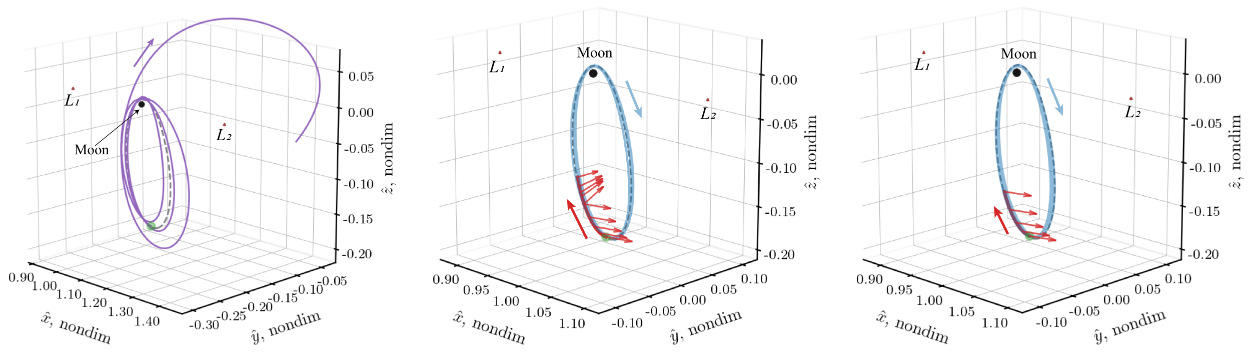


(a) Full NNIT recovery (b) NNIT single maneuver (c) Two XAC maneuvers (d) Eigenvector directions

Figure 6.4. Recovery simulated at 29 days post-failure event (x-y projections). Green signifies the initial state, dashed gray denotes the NRHO, and light blue represents the ballistic trajectory following the final low-thrust maneuver. The NNIT recovery path (a) coasts until several successive thrust arcs in similar directions are executed immediately following apolune. The NNIT arcs are combined into a single representative thrust segment (b). Two XAC SK maneuvers are implemented in (c), where the first maneuver (red) does not adequately stabilize the orbit, necessitating a much larger second SK maneuver (pink). Thrust directions occur near the position components of the stable (blue) eigenvector directions in (d).

to 8 hours. While thrust direction and magnitude instantaneously vary between the four arcs, their combined effect is well-approximated by a single 8.6-hour fixed-direction thrust profile, depicted in Figure 6.4(b). The thrust direction and time of this combined maneuver are determined via weighted averaging and scaling techniques detailed in Section 6.2.5, with the resulting discontinuity corrected using the same differential corrections technique employed in NNIT. Furthermore, tasked with recovering from the same deviation, XAC produces the bold red maneuver in Figure 6.4(c). This single 28-minute finite thrust arc is insufficient to recover to the NRHO fully, and upon propagating to the next apolune, a much larger 14.5-hour thrust segment (pink) is suggested. Furthermore, both the NNIT and XAC solutions produce thrust arcs that approximate the direction of the stable eigenvector associated with the 9:2 NRHO at apolune, plotted in Figure 6.4(d) (XAC is within 2 degrees of the stable direction, whereas the single NNIT arc differs by 30 degrees in the positive z direction). Implementing maneuvers in the direction of the stable eigenvector forms the basis of the powerful Floquet-mode SK strategy for unstable multi-body orbits, particularly in the Sun-Earth system [6]. This maneuver direction is known to be effective and is independently uncovered by the RL training process through this investigation.

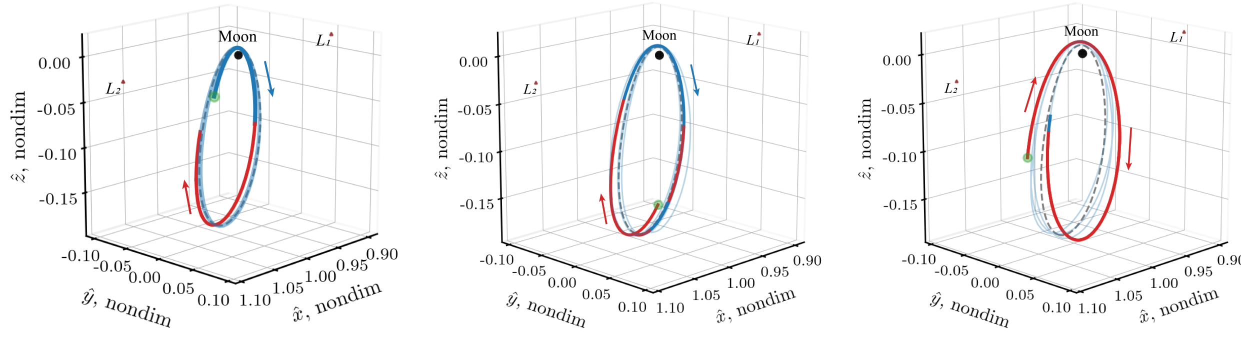
If spacecraft functionality is restored 46.4 days into the inadvertent departure period, XAC fails to converge on a solution that stabilizes the orbit, and recovery is deemed necessary. The state at 46.4 days, plotted green in Figure 6.5, rapidly departs the NRHO region over the subsequent month when left uncontrolled, as depicted in Figure 6.5(a). To return to the NRHO, NNIT suggests the maneuver sequence in Figure 6.5(b), consisting of 1) an immediate 31.3-hour thrust arc at maximum throttle, 2) five smaller thrust arcs ranging from 1-3 hours, and 3) four near-ballistic coasting segments totaling 32 hours. A feasible trajectory emerges from the targeting process resulting in a total cost equal to $\Delta\tilde{V}_{\text{equiv.}} = 12.36\,\text{m/s}$. All thrust arcs are in similar directions; therefore, their combined effect on the trajectory is well-approximated by the single fixed-attitude thrusting segment in Figure 6.5(c).



(a) Departing trajectory (b) NNIT-identified recovery (c) Single thrust-arc recovery

Figure 6.5. Recovery maneuver plan initiated 46.4 days after thruster failure event with initial state denoted by a green circle. The uncontrolled path rapidly deviates from NRHO over 33 days (a). The NNIT maneuver plan (b) stabilizes the orbit and is well-approximated by a single fixed-direction thrust segment (c).

Recovery paths originating at later points in time are depicted in Figure 6.6. As the spacecraft departs from the desired NRHO motion, longer thrust segments are employed to achieve recovery targeting convergence. At 49 days, Figure 6.6(a), the NNIT process still avoids thrusting near perilune with a 21-hour coast arc, followed by several successive thrust arcs that occur over a combined 5.19 days with a total cost equal to $\Delta\tilde{V}_{\text{equiv.}} = 35.2\,\text{m/s}$. By 51 days, Figure 6.6(b), the agent immediately thrusts for 4.6 days and subsequently alternates between thrusting and coasting segments, with total $\Delta\tilde{V}_{\text{equiv.}} = 45.4\,\text{m/s}$. Finally, by 55 days, Figure 6.6(c), the deviation is sufficiently large that the NNIT maneuver path no longer avoids perilune and instead immediately thrusts for 7.3 days continuously for a combined cost of $\Delta\tilde{V}_{\text{equiv.}} = 51.3\,{}^{\text{m}}\!/\!_{\text{s}}$. At this error level, while the recovery targeting algorithm still achieves convergence, the ballistic path initialized from the NRHO (plotted in light blue) begins to deviate from the desired NRHO motion. However, recall that the goal of recovery is not necessarily to return the spacecraft precisely to the NRHO, but instead to achieve a maneuver plan that guides the spacecraft to a region where short-horizon SK may be resumed. In this scenario, XAC and NNIT converge when simulated on the final state in Figure 6.6(c). As deviations grow further, convergence is eventually no longer achieved, and an alternative recovery approach or mission plan is deemed necessary. In these cases, longer-term low-thrust transfers that traverse further from the NRHO region must be considered, and alternate long-horizon baselines or intermediate orbit options may be necessary [121]; the feasibility of applying NNIT in such scenarios remains as future work.



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

Figure 6.6. Recovery trajectories determined by NNIT at additional times after the thruster failure event. By 55 days (c), the targeting algorithm achieves converges, but the resulting post-recovery ballistic trajectory begins to diverge from the NRHO.

6.1.4 Monte-Carlo Results

A Monte Carlo analysis approach is employed to evaluate NNIT efficacy across a broader range of simulations. As in the previous recovery scenario, trials originate with a simulated SK phase. After five low-thrust XAC maneuvers, a thruster failure is introduced at apolune, and an inadvertent departure commences. Each trial employs NNIT to return the spacecraft to the NRHO, and data is gathered at various times across each departure, where time is measured from the final SK maneuver. The convergence rates and propellant consumption statistics for the NNIT targeting algorithm across the 5000 simulations are depicted in Figures 6.7 and 6.8, respectively. The algorithm converges in more than 99 % of the trials for 46 days following the simulated failure and more than 98 % of trials up to 58 days. The convergence rate falls quickly over the proceeding 32 days, resulting in only 20.1 % after 80 days and 6.4 % after 90 days, indicating that, by this point, an alternative algorithm or mission objective is necessary.

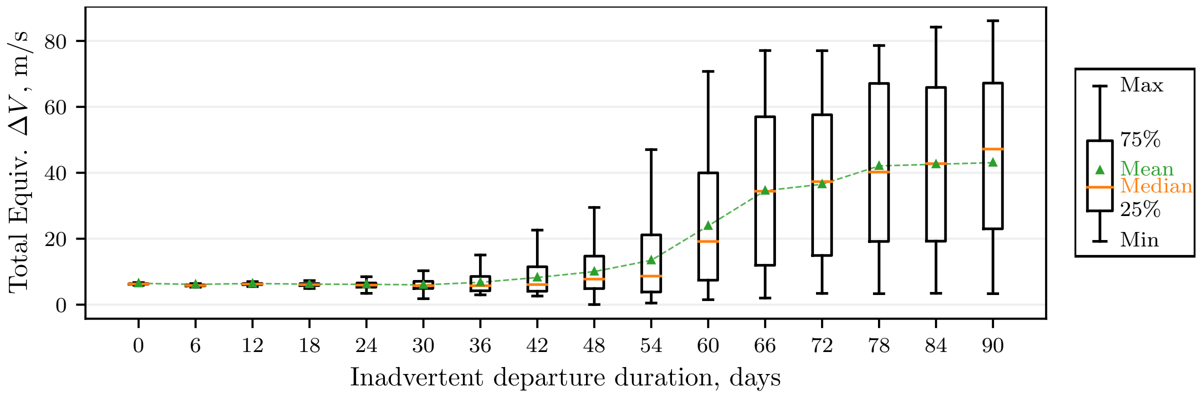


Figure 6.7. Propellant consumption statistics across 5000 Monte Carlo trials

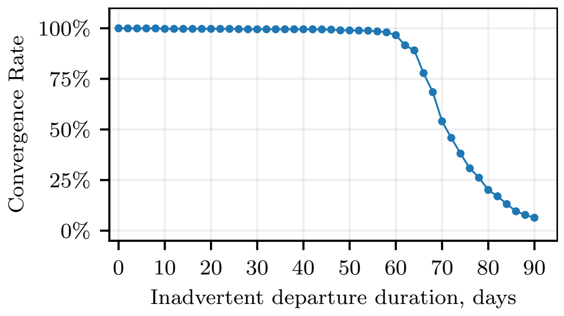


Figure 6.8. Convergence rates across Monte Carlo trials

Propellant consumption statistics across the Monte Carlo trials are visualized in Figure 6.7, where box plots are employed to represent interquartile ranges. As expected, propellant usage, measured by $\Delta\tilde{V}_{\text{equiv.}}$, increases substantially over the 90 days, beginning in the

$4\,\mathrm{m/s}$ to $7\,\mathrm{m/s}$ range for the initial four weeks, increasing to $24\,\mathrm{m/s}$ by 60 days, and eventually reaching an average greater than $40\,\mathrm{m/s}$ for each time duration after 74 days. Furthermore, Figure 6.7 demonstrates that the spread across trials increases precipitously toward the end of the inadvertent departure timeframe. This increase is primarily due to variations in the initial error and is similarly observed by Zimovan-Spreen et al. [121].

6.2 Mission Recovery Scenario: Heteroclinic Transfers

Reinforcement learning is applied in this sample mission application to support trajectory recovery along transfers for a fictitious low-thrust CubeSat mission near the Moon. In this scenario, large deviations are induced from a desired orbit transfer path and RL agents are tasked with returning the spacecraft to its original mission plan. Planar transfers between orbits in the vicinity of the L_1 and L_2 libration points, with various geometries, serve as illustrative test problems for the proposed NN-G&C framework. In particular, Lyapunov orbits at the same value of Jacobi constant are examined. With energy constrained, motion in the lunar vicinity is bounded in configuration space by a forbidden region, denoted the Zero Velocity Curve (ZVC) [130]. Furthermore, the shared Jacobi constant value between the orbits indicates that heteroclinic transfers may be available. Heteroclinic transfers occur when manifold intersections create purely-ballistic paths between two periodic orbits. These continuous trajectories are constructed by selecting an initial guess from manifold intersections on a Poincaré map and corrected using the methodology detailed by Haapala and Howell [129]. Heteroclinic transfers are one of the few cases in the CR3BP where globally optimal geometries may be identified and, thus, provide a useful test framework for the controller since no deterministic maneuvers are required.

While the agent is trained using only one reference trajectory, the controller's ability to generalize thrust histories to other geometries in the lunar region is also investigated. In reality, a variety of factors cause a planned path to shift in-flight. For example, onboard targeting yields trajectory corrections and a nearby solution is generated. In producing nearby transfers, it is often difficult to obtain any initial guess for the control history. As perturbations cause a spacecraft to deviate and maneuvers become necessary, a poor initial guess for control likely negatively impacts the performance of the targeter. To address this limitation, despite no training with other reference geometries, the ability for the proposed controller to generalize past experience is demonstrated. If a controller is only applicable to the particular reference it has seen, and the training process requires significant time and computational resources, then the practical uses of such a controller are limited in an onboard application.

The applicability of the proposed learning scheme in related problems is evaluated by simulating the training process for different thrust levels, reference trajectory geometries, and energy levels. While RL processes seek to be implemented as generally as possible, specific knowledge regarding the simulated domain or task are inescapably employed when constructing a learning environment. These decisions often carry unintended side effects in RL problems, and potentially limit the applicability of the proposed approach. Hence, varying the parameters of the given scenario helps inform the suitability of the training environment.

This investigation explores two options for incorporating a NN into a G&C framework: Standalone NN-G&C and NNIT (introduced in Chapter 5). As depicted in Figure 5.1, the standalone approach involves directly employing the NN-G&C simulation that results from an RL episode as a maneuver plan in the CR3BP. This straightforward NN-G&C method is utilized in this heteroclinic transfer mission application in evaluating and varying, the learning scheme. However, the maneuver plans that result from the standalone NN-G&C process possess several notable limitations that motivate the implementation of an NNIT algorithm that transforms the NN-G&C trajectories into a multiple shooting targeting process. Several heuristics are explored in the transformation process to enable rapid and accurate initial guess generation. Unlike the NRHO recovery application in Section 6.1, this sample mission application does not simulate the targeting algorithm during training (i.e. this is a ‘standalone‘ not ‘integrated’ NNIT training approach, as defined in Section 5.3). Leveraging standalone NN-G&C training with NNIT enables this mission application to demonstrate the utility of employing an NNIT approach given a pre-trained NN controller.

6.2.1 Low-Thrust CubeSat Model

The fictitious CubeSat simulated in this sample mission application is assumed to initially possess 11.46 kg wet mass and is equipped with a solar electric propulsion engine capable of producing 1.25 mN of thrust with a constant 3000 s Specific Impulse (Isp), resulting in a nondimensional thrust magnitude of $f = 0.04$ nondim. This thrust value approximately corresponds to a Busek Ion Thruster 3-cm (BIT-3) engine [77], but with a higher Isp. This nondimensional thrust value models any spacecraft with the same thrust-to-weight ratio and, hence, the sample applications in this investigation are equivalently applicable to larger spacecraft with correspondingly larger engines. The low-thrust engine parameters for this sample spacecraft are noted in Table 6.4. As observed in Table 2.2, this sample CubeSat possesses similar control authority to Lunar IceCube and the planned Psyche spacecraft. Furthermore, this investigation also considers the possibility that the sample engine is only capable of maneuvers above 58 % maximum thrust, or about 0.725 mN (which is above the lower limit of 0.67 mN for the BIT-3 [77]). The 58 % threshold is arbitrarily selected based on the low-thrust engine characteristics on the Hayabusa 1 mission [74].

Table 6.4. Sample CubeSat spacecraft data

Parameter	**Symbol**	**Value**
Initial mass	$\tilde{m}_{3,0}$	11.46 kg
Specific impulse	$\tilde{I}_{\text{sp}}$	3000 s
Maximum thrust	$\tilde{f}_{\max}$	1.25 mN
	$f_{\max}$	0.04 nondim
Lower thrust limit	$\tilde{f}_{\text{engine}}$	$0.58 f_{\max}$
		0.725 mN
	f_{engine}	0.0232 nondim

6.2.2 Learning Environment Implementation (Standalone NN-G&C)

The RL training environment is implemented to construct an agent able to recover, and follow, L_1-to-L_2 heteroclinic transfers. The observation, action, and reward signals, visualized in Figure 4.3, are defined to support this recovery objective. Numerous agents are trained in parallel, and a specific controller is selected based on consistency in solving the given task. This sample mission application leverages Proximal Policy Optimization (PPO) as an alternative actor-critic RL training algorithm to TD3, thus demonstrating the applicability of the proposed environment configurations and training processes to on-policy and off-policy RL methods alike[5]. The PPO training algorithm is originally proposed by Schulman et al. [103]; the specific implementation of PPO leveraged in this section is based on the open source work of Patrick Coady [131], and includes several other minor customizations to the core algorithm. Specific PPO implementation details for this sample mission application are presented in [127], Sections 3.1.3 and 3.1.4.

Observation Signal

This mission application involves multi-body orbit transfers in the CR3BP and, therefore, including some dynamical information in the observation signal is advantageous for learning performance. In particular, the Jacobi constant values of the current observation and reference states, detailed in Table 4.1, augment the general form of the observation vector in Equation (4.15). This additional information communicates energy deviations to the actor and critic networks. Hence, the observation vector in this application is given by,

$$\boldsymbol{o}_{\text{t}} = \begin{bmatrix} \mathbf{q}_{\text{t}} & \boldsymbol{\delta\varrho} & C_{\boldsymbol{o}} & C_{\text{ref}} \end{bmatrix} \in \mathbb{R}^{11} \tag{6.8}$$

This mission application assumes all motion occurs in the rotating x-y plane, visualized in Figure 2.3. Therefore, $\mathbf{q} \in \mathbb{R}^5$ and $\boldsymbol{\delta\varrho} \in \mathbb{R}^4$ in Equation (6.8) are planar analogs to $\boldsymbol{q}$ and $\boldsymbol{\delta\rho}$, respectively, where $z = \dot{z} = 0$. In this planar problem, the 'nearest' state along the reference, $\boldsymbol{\varrho}_{\text{ref}}$, is computed as,

$$\boldsymbol{\varrho}_{\text{ref}} \in \mathcal{R}^{\text{ref}} \quad \text{s.t.} \quad k = |\boldsymbol{\delta\rho}|_2 \quad \text{is minimal} \tag{6.9}$$

[5]↑The effectiveness of TD3 as the main RL training method in this investigation is demonstrated in the NRHO mission recovery scenario in Section 6.1 and in the stationkeeping problem presented in Section 6.3

where $\boldsymbol{\varrho}_{\text{ref}} \in \mathbb{R}^4$ is a planar analog to $\boldsymbol{\rho}_{\text{ref}}$ in Equation (4.14). The simulated scenario involves a transfer to a periodic destination orbit. Hence, the set of reference states leveraged for the nearest neighbor calculation in Equation (6.9) is composed of,

$$\mathcal{R}^{\text{ref}} = \mathcal{R}^{\text{transfer}} \cup \mathcal{R}^{\text{arrival}} \tag{6.10}$$

where $\mathcal{R}^{\text{transfer}}$ and $\mathcal{R}^{\text{arrival}}$ are the sets of states along the heteroclinic transfer and 'arrival' periodic orbit, respectively (assuming $\mathcal{R}^{\text{transfer}} \cap \mathcal{R}^{\text{arrival}} = \emptyset$).

Action Signal

The action signal is comprised of: 1) the planar thrust direction $\boldsymbol{u}^*$, specified in Equation (4.16) for $u_z = 0$, and 2) the thrust magnitude f^*, defined in Equation (4.17). This thrust command action is delivered by the vector quantity,

$$\boldsymbol{a}_{\text{t}} = \begin{bmatrix} f^* & u_x^* & u_y^* \end{bmatrix} \in \mathbb{R}^3 \tag{6.11}$$

The architecture for the feedforward actor NN that produces the planar thrust command, Equation (6.11), in this sample mission application is listed in Table 6.5, and visualized in Figure 6.9. For this application, thrust time $\mathfrak{t}$ is held constant for each time step and is, therefore, not estimated by the actor NN.

Reward Signal

The reward function in this transfer recovery application relies on relative state minimization, which reflects an assumption that it is advantageous for the spacecraft to maintain a close-proximity to its planned reference trajectory (as was the case in the Genesis mission [132]). Similar to the reward function for the NRHO recovery application, Equation (6.6), this relative state-based reward function is modeled as an exponential so that the reward grows rapidly as the agent's state nears the reference in both position and velocity. In this formulation, after the nearest neighbor along the reference is determined, the agent is rewarded for a thrusting plan such that the distance to the nearest state at the next time step is minimized. Deviation is defined as when the agent's relative state to the reference

Table 6.5. Configuration of actor and critic NNs for PPO algorithm employed in the heteroclinic transfer recovery mission application

	Actor Network		**Critic Network**	
Layer name	Size	Activation	Size	Activation
Input	11	-	14	-
Hidden 1	120	tanh	120	tanh
Hidden 2	60	tanh	24	tanh
Hidden 3	30	tanh	5	tanh
Output	3	tanh	1	linear
Learning Rate	0.000 11		0.002 04	

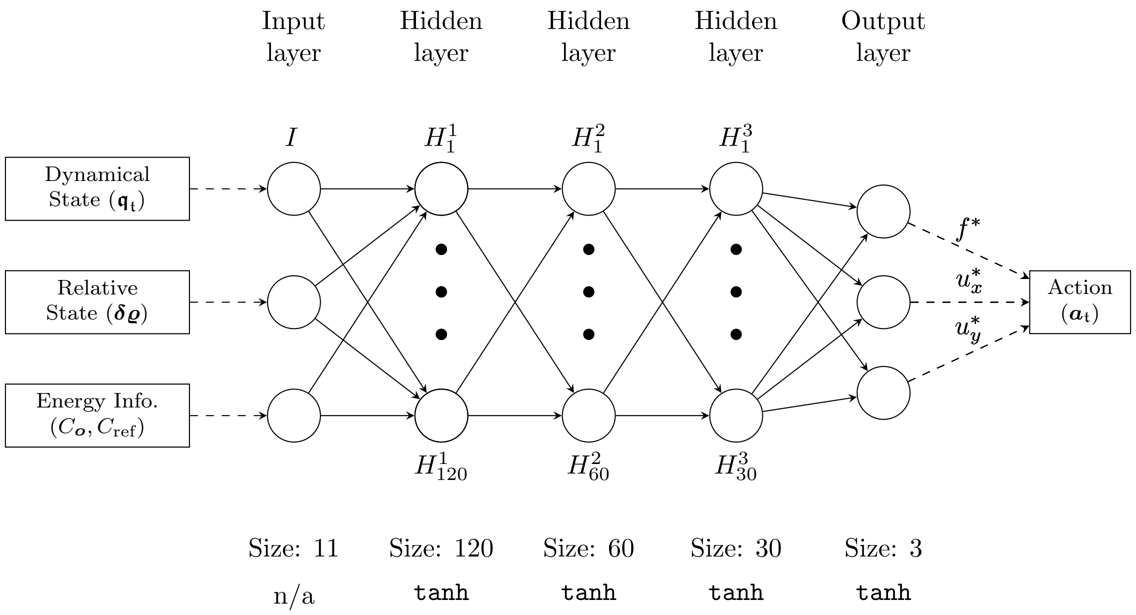


Figure 6.9. Actor NN employed in the heteroclinic transfer problem

trajectory exceeds maximum allowable deviation thresholds in either position or velocity, or when the agent impacts the Moon,

$$\text{Deviation from transfer} := \begin{cases} \tilde{r}_{23} < 1737.4\,\text{km} & \text{or} \\ |\boldsymbol{\delta\tilde{r}}| > 8000\,\text{km} & \text{or} \\ |\boldsymbol{\delta\tilde{v}}| > 35\,{}^{\text{m}}\!/\!_{\text{s}} \end{cases} \tag{6.12}$$

The general reward function formulation in Equation (4.20) is implemented for this mission application as,

$$r_t = \begin{cases} \beta_{\text{ref}}\Gamma_{\text{ref}} & \text{not deviated from transfer} \\ r_f & \text{final condition reached} \\ p & \text{deviation criteria met} \end{cases} \tag{6.13}$$

where $\Gamma_{\text{ref}} = \mathrm{e}^{-\lambda k}$ is defined in Table 4.2. In this formulation, unlike Equation (6.6), the scaling term β_{ref} increases over time for the reference transfer to encourage the agent to reach the target orbit, i.e.,

$$\beta_{\text{ref}} = \begin{cases} \left({}^{i}\!/\!_{n}\right)\zeta + 1 & \boldsymbol{\varrho}_{\text{ref}} \in \mathcal{R}^{\text{transfer}} \\ \zeta + 1 & \boldsymbol{\varrho}_{\text{ref}} \in \mathcal{R}^{\text{arrival}} \end{cases} \tag{6.14}$$

where i is the index for the reference trajectory state, $n = |\mathcal{R}^{\text{transfer}}|$ is the size of the set of states along the transfer trajectory, and ζ is a tuning variable to adjust the rate at which the reward increases along the reference path. If the nearest neighbor is along the arrival orbit and not the reference trajectory, then β_{ref} is assumed to be a maximum value $\beta_{\text{ref}}\big|_{\text{arrival}} = \max(\beta_{\text{ref}}) = \zeta + 1$. Formulating the reward signal to be at a maximum value when the agent reaches its target encourages the agent to fully complete the given transfer. The variables that comprise the reward function, Equations (6.13) and (6.14), are summarized in Table 6.6, with suggested values included for constant parameters.

Table 6.6. Components for reward functions leveraged in the heteroclinic transfer sample mission application

Symbol	Description	Value(s)
β_{ref}	Reward scaling coefficient, defined in Equation (6.14)	-
k	Relative state norm, defined in Equation (6.9) assuming $z = \dot{z} = 0$	-
λ	Scaling factor that adjusts the gradient of the reward	$\lambda = 340$
ζ	Rate of reward increase across reference transfer	$\zeta = 1$
r_{f}	Bonus applied for reaching terminal condition	$r_{\text{f}} = 15$
p	Penalty for exceeding deviation criteria	$p = -4$

The 'final' arrival condition in Equation (6.13) is triggered when the nearest neighbor to the spacecraft is along the destination orbit (instead of the transfer trajectory) and the relative position and velocity magnitudes are below pre-defined thresholds,

$$\text{Final condition} := \begin{cases} \boldsymbol{\varrho}_{\text{ref}} \in \mathcal{R}^{\text{arrival}} & \text{and} \\ |\boldsymbol{\delta\tilde{r}}| < 100\,\text{km} & \text{and} \\ |\boldsymbol{\delta\tilde{v}}| < 2\,{}^{\text{m}}\!/\!_{\text{s}} & \end{cases} \tag{6.15}$$

where $\mathcal{R}^{\text{arrival}} \subset \mathcal{R}^{\text{ref}}$ is the set of states along the arrival orbit (introduced in Equation (6.10)), and $\boldsymbol{\varrho}_{\text{ref}}$ is defined in Equation (6.9). This definition of arrival is not intended to indicate that the spacecraft has reached a particular state. Rather, the given tolerances simply detect that the spacecraft has reached the vicinity of the arrival orbit. If a tighter tolerance for arrival is employed, many false negatives exist that eventually reach the given tolerance, but not within the specified maximum number of time steps. For example, relative position and velocity errors over time for a sample episode are depicted in Figure 6.10, where annotations signify the time the arrival condition is satisfied given the noted tolerances. In the sample case, the arrival condition is triggered at 25.2 days when the relative position and velocity magnitudes along the arrival orbit drop below 100 km and 2 m/s, respectively. However, if 3 km and 0.07 m/s tolerances are instead selected, as depicted in Figure 6.10, with a maximum episode time of 100 days, the arrival condition would not be met, and the episode would be falsely designated a failure. Past 25 days, the errors remain bounded, and reaching tighter tolerances is, in general, a result of noise rather than controller accuracy. Hence, 100 km and 2 m/s are heuristically selected as relative position and velocity magnitudes to ensure early arrival criteria detection.

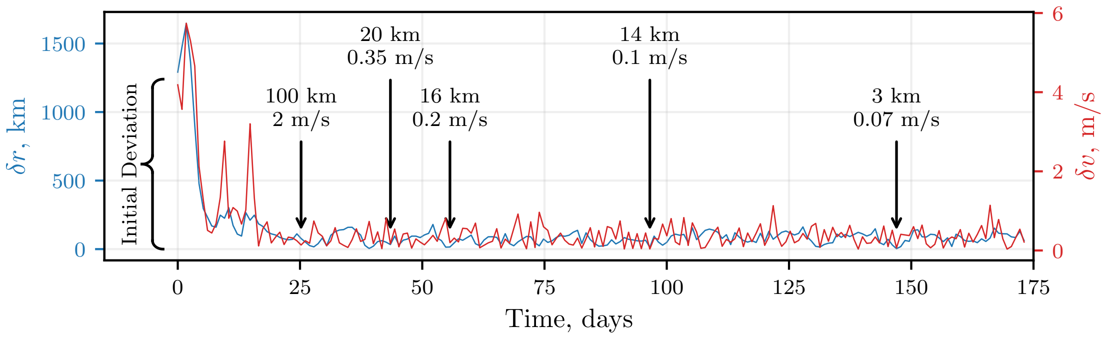


Figure 6.10. Position and velocity deviations over time for a sample episode. Each arrow signifies the time when relative errors drop below the noted tolerances. Errors remain bounded once the destination orbit is reached, and smaller arrival tolerances may eventually be met given a longer time horizon.

The measurement of nearness in the reward function, as detailed in Equation (6.13), is visualized in Figure 6.11. To illustrate the region of high reward surrounding the reference, perturbations are introduced in y_0 at the point where the trajectory crosses the $x = 1 - \mu$ plane. As each of these perturbed states is propagated forward in time, their deviation off the reference is visualized by the reward colormap. Once deviation beyond the threshold occurs, the trajectory is colored light gray to denote areas where a penalty is imposed. Due to the exponential term in $\Gamma_{\text{ref}} = \mathrm{e}^{-\lambda k}$, high reward exists solely in the region immediately surrounding the reference. Hence, to continue accruing reward, the agent is encouraged to maintain close proximity to the reference path.

A notable limitation of this time-autonomous approach is a reference trajectory that includes a departure periodic orbit. In this transfer recovery application, the reward function may cause an agent to learn to maximize the expected return by stationkeeping about the departure orbit rather than proceeding along the reference to the destination orbit. To combat this behavior, the variable β_{ref}, Equation (6.14), is leveraged to encourage the agent to continue along the reference, and discourage the initial periodic behavior. Like

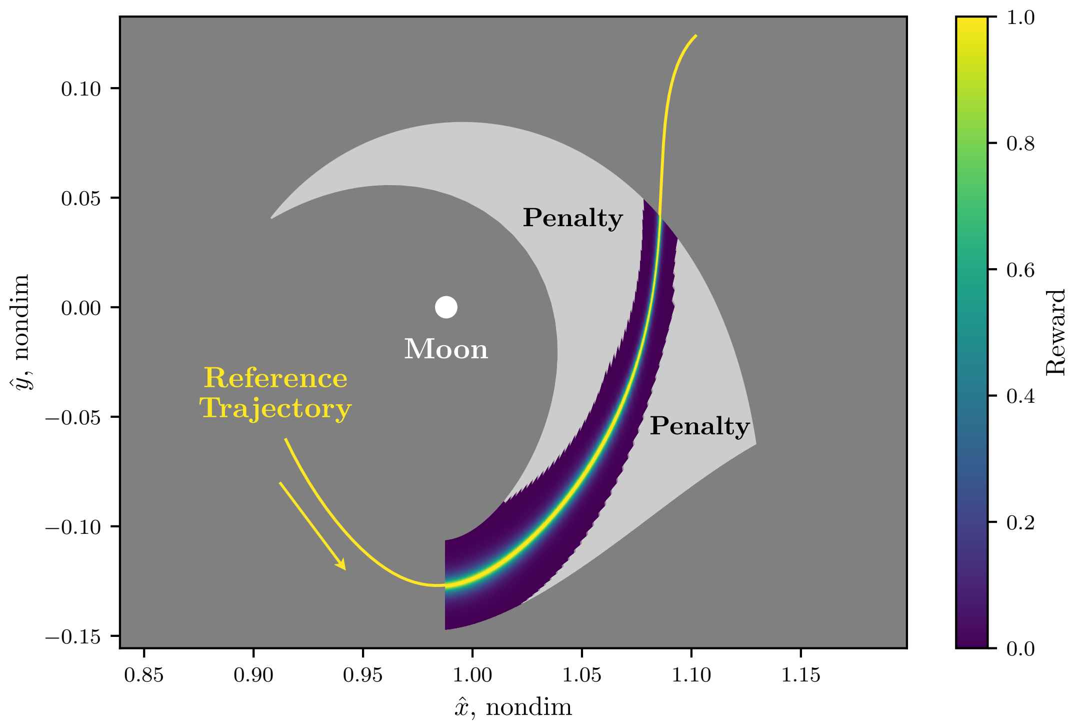


Figure 6.11. Motion nearby a reference trajectory originating at the plane defined by $x = 1 - \mu$. Perturbations are introduced in y_0, and then propagated without thrust. Each state is colored based on the reward function defined by Equation (4.20), where $\lambda = 100$, $\beta_{\text{ref}} = 1$, and the maximum deviations are given in Equation (6.12)

other optimization methods, abundant local minima encourage an agent to converge to suboptimal behavior. Including β_{ref} discourages this behavior, but does not entirely eliminate the possibility of suboptimal convergence and return to the stationkeeping local minimum.

Episode Details

The RL agent is trained over 150 000 finite-horizon episodes. During each episode, the agent attempts to return to the reference trajectory and complete the given transfer scenario despite a large initial deviation. Episodes are terminated when the agent diverges from the reference, Equation (6.12), or successfully reaches a specified arrival criteria, Equation (6.15). Over the course of training, the agent gradually improves at performing the task, until it consistently reaches the arrival condition. Each episode is allowed a total of $\mathfrak{n} = 100$ time steps to reach the arrival criteria, or 86.97 days. The overall trend in training over time is analyzed to understand performance, however, the specifics for the training protocols are most easily understood at the episode level.

Episodes begin by selecting a random initial state along the departure periodic orbit from a uniform distribution, and introducing a perturbation in position and velocity; initial mass is held constant. During training, $3\sigma_r = 1000\,\text{km}$ and $3\sigma_v = 10\,{}^{\text{m}}\!/\!_{\text{s}}$ model the initial deviations. The perturbation is sampled from two normal distributions (position and velocity), where each component of $\boldsymbol{\rho}_0$ is perturbed individually. The averages across the perturbation distributions are all zero, and σ is selected based on the desired disturbance for a particular simulation. As the L2 norm of the sum of two independent, normally distributed random variables, the scalar perturbation magnitude k follows a chi distribution rather than a normal distribution [133]. This lower-fidelity error model is intended to encompass a wide range of originating deviation sources. Navigation errors on the order of $3\sigma = 1\,\text{km}$ and $1\,{}^{\text{cm}}\!/\!_{\text{s}}$ [118] provide a useful comparison metric for the relative order of magnitude of the initial perturbation. During training, initial deviations are three orders of magnitude larger than expected orbit determination disturbances. In this mission scenario, orbit determination errors are discussed as a comparison metric for the initial perturbation distributions, and are not implemented during simulations.

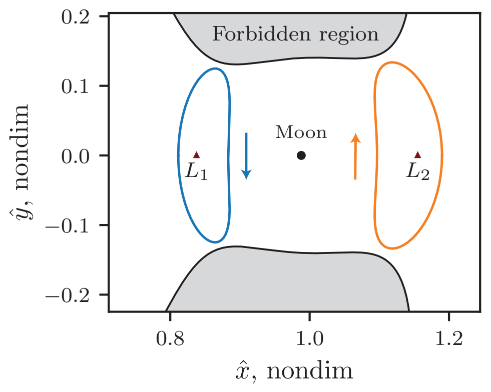
Once the initial state is generated, the departure orbit is no longer used in the simulation. Hence, to accrue reward over an episode, the agent must follow a particular reference trajectory as defined in the environment. After some disturbance is introduced into the environment, the agent computes an action, i.e., a thrust direction and magnitude. The equations of motion are then propagated within the environment for a particular time horizon ($\mathbb{t}$). The specified $\mathbb{t}$ between the actions is an important selection for the agent performance. If $\mathbb{t}$ is too large, then the nonlinearities become more pronounced, and the agent is offered fewer opportunities for sufficient actions over an episode. However, if $\mathbb{t}$ is too small, there is not sufficient time for the thrust direction to demonstrate a discernible impact on the system. For the examples included in this mission scenario, $\mathbb{t} = 0.2\,\text{nondim} \approx 20.87\,\text{h}$ strikes a balance between the extrema of being too large or too small. After propagating for $\mathbb{t}$, the agent again selects a new action. Actions are introduced sequentially, after each time interval, until the agent deviates from the reference, impacts a planetary body, arrives at the target orbit, or reaches a maximum number of time steps.

To accommodate variations in the controller training process, high-performance computing resources are leveraged to train many identically configured agents. Each agent is trained independently, and samples are not shared to ensure the on-policy assumption for PPO is satisfied. Training numerous agents increases the chances of producing an effective controller that generalizes well, and demonstrates the prevalence of local basins of attraction. Once trained, the arrival criteria is employed to select an agent with desirable characteristics. Upon completing training, 2000 deterministic episodes are simulated to estimate the frequency with which each agent reaches the arrival criteria. In the transfer scenario detailed in Section 6.2.3, of the 1275 trained agents, 40 % are at least 90 % successful and 25.7 % are at least 99 % successful. In practice, producing such a large number of agents is computationally challenging, and requires access to high performance computing resources. For this investigation, 1275 agents are produced to analyze training variance, however, only a small fraction of these are necessary to produce an agent that satisfies the mission criteria of this investigation.

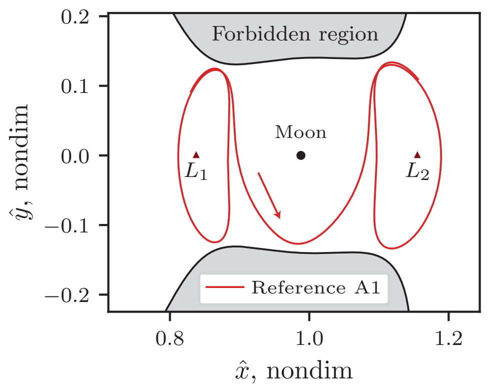
6.2.3 L1-to-L2 Transfer Simulation

A planar transfer between Lyapunov orbits in the vicinity of the L_1 and L_2 libration points at an energy value $C = 3.124102$ serves as an illustrative test problem for the proposed RL training method and NN-G&C framework. The transfer scenario is illustrated in Figure 6.12, where Figure 6.12(a) visualizes the periodic orbits, and Figure 6.12(b) depicts a continuous heteroclinic transfer from the L_1 Lyapunov orbit to the L_2 Lyapunov orbit. The RL agent is trained using this heteroclinic transfer, labeled reference ‘A1’, and is subsequently evaluated using other transfer geometries. All transfers at this energy level are labeled ‘A’-transfers.

At any given point during training, a deterministic controller is available by simply removing variance from the agent’s computed actions. Running an episode with the deterministic controller at various points in training, given a fixed initial condition, yields insight into the evolving improvement in the agent’s policy. For this simulation, the initial condi-



(a) L_1 and L_2 Lyapunov orbits



(b) Heteroclinic transfer employed in training

Figure 6.12. Periodic orbits and heteroclinic transfer in the Earth-Moon system applied to the sample scenario. All motion at $C = 3.124102$, with the corresponding forbidden regions in gray.

tions are generated by selecting an initial state along the L_1 Lyapunov departure orbit and introducing perturbations of 1106 km and 6.9 m/s. In reality, error is computed relative to the reference trajectory rather than the starting orbit, which adds a small amount of additional perturbation. For this sample case, the error relative to the reference is computed as 1108 km and 6.7 m/s. Without control, the resulting trajectory, as plotted in Figure 6.13, immediately deviates from the reference and impacts the Moon in less than a week. This specific perturbation is not included at any point during the training phase. In many types of machine learning, it is important to separate training and validation data. In RL, this data separation is not as explicit as in supervised learning approaches, but it is nevertheless an important consideration when selecting a test case.

The control history produced by a deterministic controller at various stages in the training is plotted in Figure 6.14. Arrows indicate the thrust magnitude and direction for a particular segment, where the length and color of the arrow corresponds to thrust magnitude. For visualization clarity, thrust values below a user-defined threshold cause the thrust direction magnitude indicator arrows to be omitted from visualization in Figure 6.14. In this case, 25 % is arbitrarily defined. When training begins, the agent's policy is determined by randomly initialized weights in the actor NN, depicted in Figure 6.9. As the number of

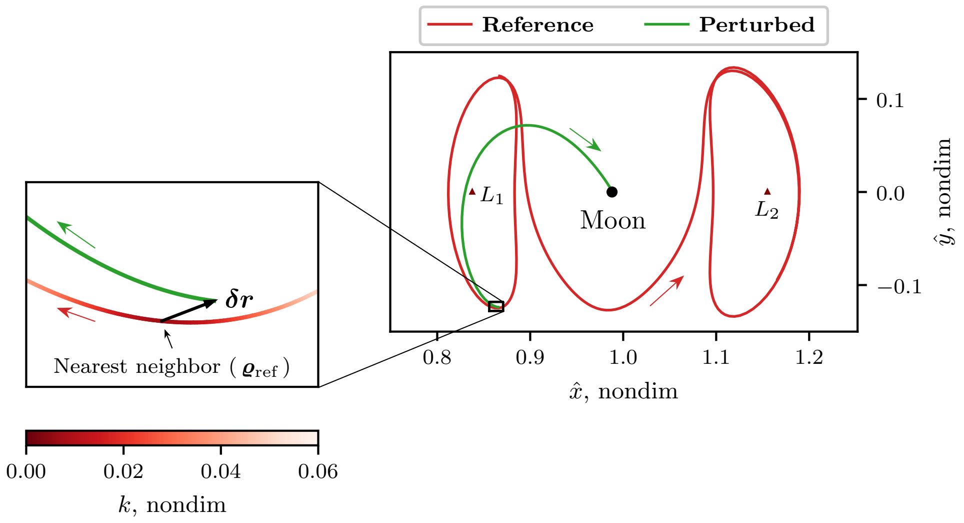
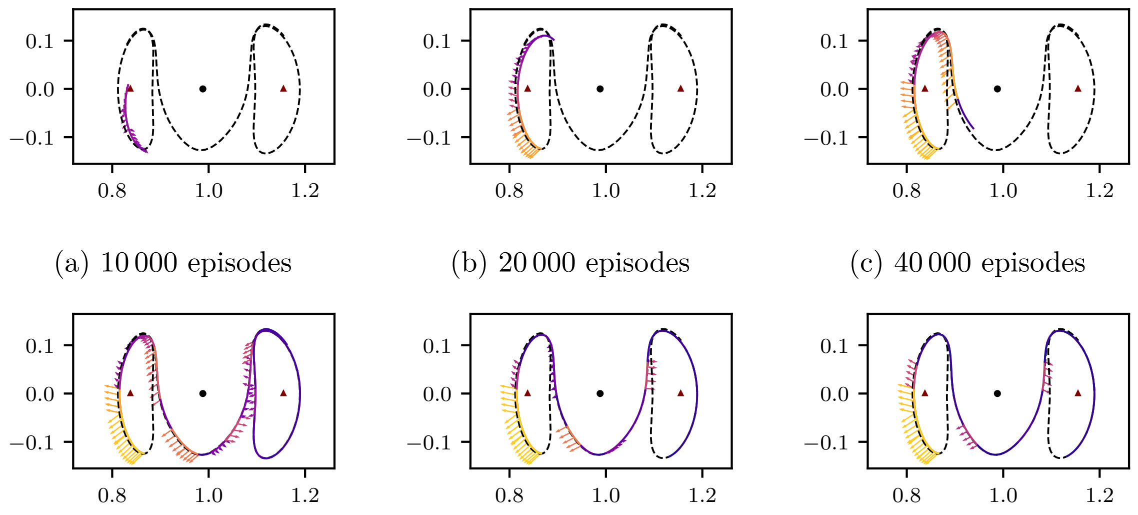


Figure 6.13. Sample perturbed initial state that impacts the Moon in 6.4 days. Position perturbation is plotted in configuration space ($\boldsymbol{\delta r}$), with magnitude 1106 km. States along the reference trajectory $\mathcal{R}^{\text{ref}}$ appear in the zoomed portion with the shade of red denoting the magnitude of the nearest neighbor distance to the perturbed state, k, defined in Equation (6.9).

episodes increases, the agent's ability to perform the given task gradually improves. After 1000 episodes, Figure 6.14(a), the agent is not able to discern the correct thrust direction. By episode 40 000, the agent departs the Lyapunov orbit at approximately the correct location. After only 10 000 additional episodes, the agent completes the given transfer, though with substantially more thrusting than is necessary. Over the next 100 000 episodes, the agent gradually improves by reducing the amount of thrust applied. To illustrate the thrust reduction over learning, at episode 50 000, the agent performs the given task, but its mass is reduced by 0.38 %. By episode 150 000, Figure 6.14(e), this mass-reduction figure is reduced to 0.20 %. The efficiency improvement trend is visually apparent in the departure segment from the L_1 Lyapunov orbit in Figure 6.14.

To analyze the performance of a particular controller after training, many initial conditions with various perturbations are generated, and the trained agent is evaluated based on its resulting output control history. A simulation is considered successful if the agent avoids deviating from the reference and reaches the arrival criteria, Equation (6.15). The



(a) 10 000 episodes (b) 20 000 episodes (c) 40 000 episodes

(d) 50 000 episodes (e) 100 000 episodes (f) 150 000 episodes

Figure 6.14. Deterministic agent over 150 000 episodes (x-y, nondim). Simulations (a – c) are penalized for diverging while (d – f) are terminated after successfully reaching the arrival criteria.

numerically computed arrival percentages for various levels of 3σ error in position and velocity are depicted in Figure 6.15. For 10 000 combinations of position and velocity 3σ values, 5000 deterministic episodes are simulated using the sample agent and the training reference trajectory (A1). The results of the Monte Carlo analysis are colored based on the arrival percentage across the 5000 episodes. Based on expected levels of position and velocity error, Figure 6.15 demonstrates the expectations for controller performance. For example, if all position and velocity errors are expected to be less than 1000 km and 10 m/s, respectively, then the bright yellow color at $[3\sigma_v = 10\,\mathrm{m/s}, 3\sigma_r = 1000\,\mathrm{km}]$ indicates that the controller is expected to reach the destination orbit in nearly 100 % of cases (actual value is 99.5 %). As expected, as the error increases, the controller becomes less successful. With large amounts of error, it is frequently unreasonable to return to a previous reference trajectory. Cases where the controller frequently fails to recover indicate error levels where a new reference trajectory is required.

The sample controller is more sensitive to perturbations in velocity than position, as depicted in Figure 6.15. With perturbations sampled from $3\sigma_v = 0\,\mathrm{m/s}$ and $3\sigma_r = 6000\,\mathrm{km}$,

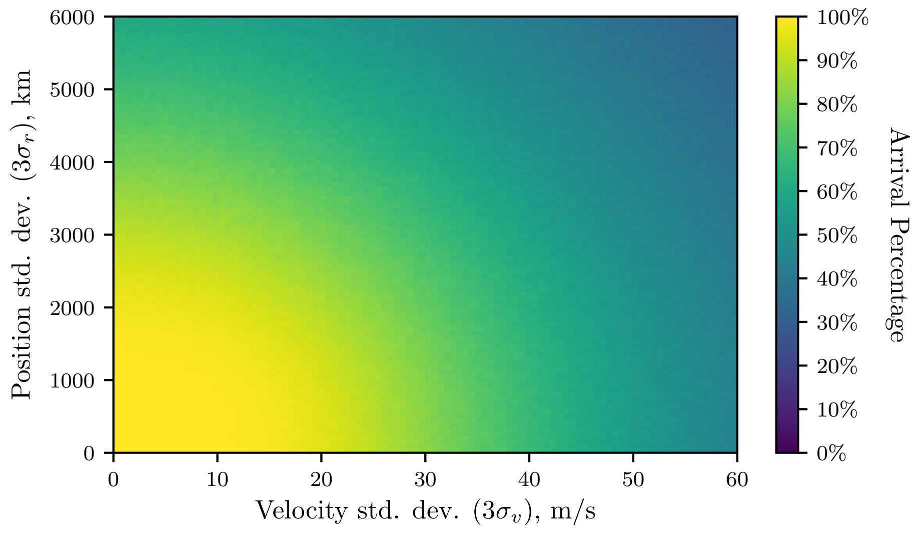
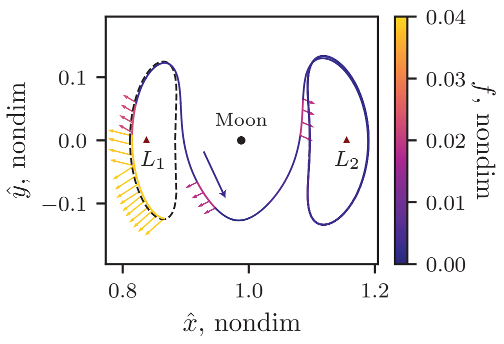


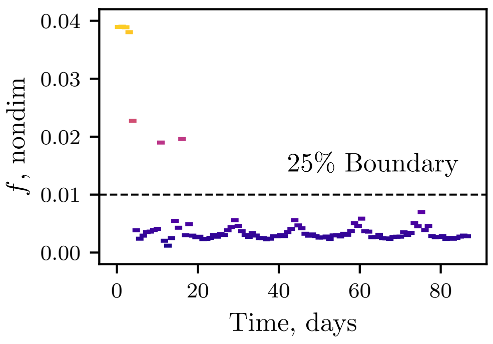
Figure 6.15. Arrival percentage for training reference (A1) given various 3σ levels of initial error.

the controller arrives successfully in 60.5 % of the cases. Conversely, if $3\sigma_v = 60\,\mathrm{m/s}$ and $3\sigma_r = 0\,\mathrm{km}$, the agent succeeds in only 46.7 % of the cases. This sensitivity to velocity errors is consistent with other applications in cislunar space. For example, Davis et al. demonstrate the significant impact of velocity navigation errors in NRHO orbit maintenance [2]. While this relative sensitivity to velocity appears consistent with prior work, the present analysis does not account for relative scaling differences between the mixed units (km and m/s). Therefore, this sensitivity trend in performance is mentioned here as an observation, but no definitive conclusion is available due to the difference in units between the axes.

For the perturbation in Figure 6.13, the sample controller produces the NN-G&C maneuver plan in Figure 6.16. With training completed, it is no longer necessary to terminate episodes upon reaching the arrival criteria. Without termination, upon completing the transfer, the agent performs orbit maintenance indefinitely about the L_2 Lyapunov orbit. The trajectory in Figure 6.16(a) is identical to Figure 6.14(f) since the resulting controller is simply the deterministic controller at the final episode, with the only difference being Figure 6.14(f) terminates at arrival, and Figure 6.16(a) does not.



(a) Trajectory and control history



(b) Thrust magnitude history

Figure 6.16. Sample case where controller successfully overcomes an initial perturbation and follows a given reference trajectory. Thrust direction and magnitude are plotted in configuration space and colored based on thrust magnitude. For thrust values below an arbitrary threshold (25 %), thrust direction indicators are omitted from the control history in (a).

Given the initial perturbation, as in Figure 6.13, with the goal of returning to the original reference trajectory, a delay in response time renders the original geometry inaccessible. However, the trained NN controller immediately outputs a maneuver plan that returns the spacecraft to its original path and successfully accesses the target L_2 Lyapunov orbit, as depicted in Figure 6.16(a). Again, thrust magnitudes below 25 % are omitted from visualization, but are still applied in the dynamical model. As expected, the majority of the spacecraft's thrusting occurs during the initial time intervals as the agent recovers from the introduced perturbation. Subsequent time steps require much less propellant. Upon arrival in the destination L_2 Lyapunov orbit, the controller maintains the arrival geometry. The orbit maintenance is apparent in configuration space, i.e., in Figure 6.16(a), as well as in the periodic sinusoidal thrust magnitude behavior, depicted in Figure 6.16(b). Hence, the trained controller performs three functions: 1) recovers from a large initial deviation, 2) guides the spacecraft along a reference path, and 3) maintains stationkeeping about the destination orbit.

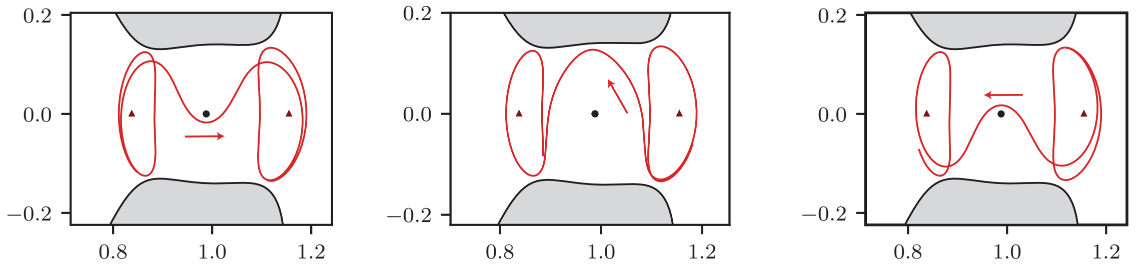
The 'standalone' NN-G&C architecture simulated in this sample mission application presents several practical challenges. While the sample scenario in Figure 6.16 is deemed successful, it requires the spacecraft to be continuously thrusting for the duration of the simulation, and offers no ability to include coasting segments. Furthermore, low-thrust engines typically possess a lower limit for thrust capability. The NN is not aware of such limitations and, therefore, assumes the spacecraft engine implements any suggested maneuver. There are opportunities to include this sort of constraint in the training process, but this possibility is not investigated here. Depending on the lower-bound for the thrust level, it is possible for the controller to generate a new solution by simply not applying control values below the given threshold. Depending on the lower limit, the agent may still reach the arrival criteria, but the corresponding thrust history requires frequent large maneuvers and suggests a clearly suboptimal path. Hence, this thrust limitation is better addressed by either modifying the environment or augmenting the NN guidance approach with an alternative guidance method. This investigation addresses thrust bound constraints by employing an NNIT architecture, detailed in Section 5.2, to alter the NN-generated maneuver plan.

Generalization to Other References

If a spacecraft deviates significantly from its reference path, it is not always reasonable to return to the original trajectory. The process of generating a new transfer arc is accomplished from a variety of options, including leveraging dynamical systems theory, applying a numerical strategy, and/or employing differential corrections. The methodology for generating new reference transfers is not considered in this investigation. However, assuming a new trajectory has been constructed, an important test of the proposed controller is its ability to perform despite training with only one original path. While NNs, in general, offer a demonstrated ability to generalize, their performance is always tied to the training data set. Hence, if the new geometry is vastly different, a NN-based approach's performance is limited by its training experience.

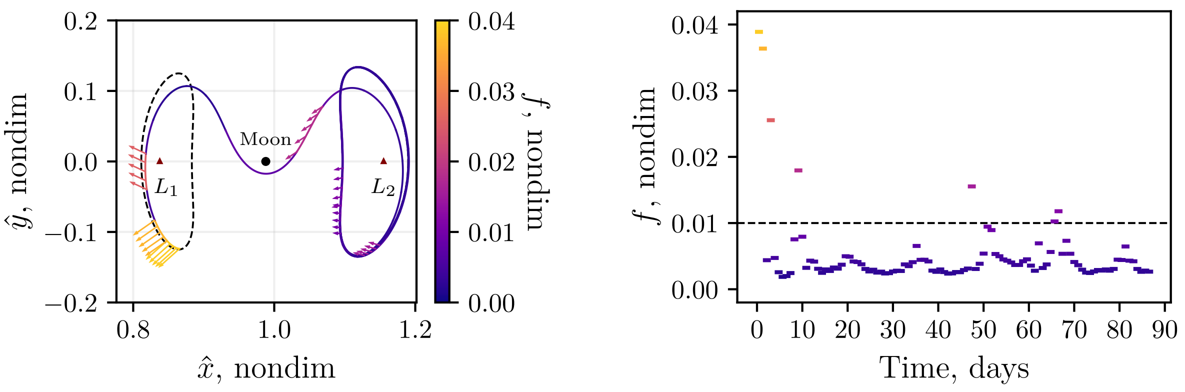
To test the expendability of the RL-trained controller, several new transfers are examined. First, a new path between the original L_1 and L_2 Lyapunov orbits is demonstrated via a second heteroclinic reference trajectory in Figure 6.17(a). Next, the reverse of the previous example is also included, in which the agent originates in the L_2 Lyapunov orbit and attempts to track heteroclinic transfers to the L_1 Lyapunov orbit. These two new heteroclinic paths are plotted in Figure 6.17(b) and Figure 6.17(c). To evaluate the scenario where a new reference is generated in-flight, reference A2 in Figure 6.17(a) is employed. Reference A2 is not included in the training process and, therefore, the controller must generalize its experience on the training reference to a different area of cislunar space, but at the same energy level. The new path is notably different from the original; the nearest distance to the Moon along references A1 and A2 are 34 546 km and 6725 km, respectively. Not only is this a large deviation in geometry, significant nonlinearity is added due to the close proximity with the Moon. Hence, this simulation adds difficulty for the controller, not only in generalizing to new locations in space, but also demanding that the controller overcome more nonlinearity than was present in the training. Returning to the previous example, the perturbation in Figure 6.13 is applied to the new reference geometry. The error from the perturbed state to its nearest neighbor along the new reference path is 860 km and 5.1 m/s. The agent generalizes its experience to the new reference trajectory, and the resulting control history and successful transfer are plotted in Figure 6.18.

For realistic scenarios, since the agent's performance is dependent on training data, it is generally inadvisable to reuse an old agent for a drastically different scenario without training on the new geometry. However, examples where no additional training is conducted are included to illustrate the agent's ability to perform well in regions of space that were not originally explored. In particular, the environment is reversed from the previous example, that is, the agent is required to transfer from the L_2 Lyapunov orbit to the L_1 Lyapunov orbit along heteroclinic paths A3 and A4, plotted in Figure 6.17. A single sample case is again examined to illustrate the NN-generated solution. For an arbitrary perturbation, the agent is tasked to return to one of the new references, without the benefit of previous training. The controller completes these new transfers, with the resulting control histories plotted in Figure 6.19. These examples demonstrate the RL controller's ability to perform well in challenging scenarios.



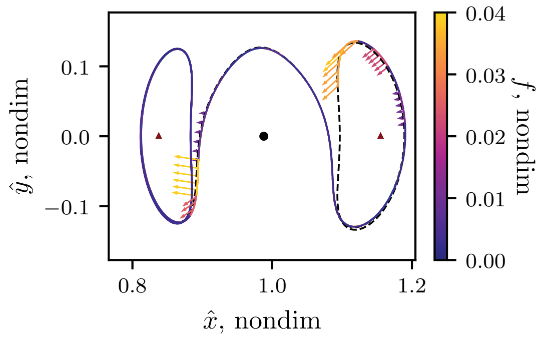
(a) Reference A2 (L_1 to L_2) (b) Reference A3 (L_2 to L_1) (c) Reference A4 (L_2 to L_1)

Figure 6.17. Additional heteroclinic transfers between Lyapunov orbits at $C = 3.124102$ ($\hat{x}$-$\hat{y}$, nondim). Reference A2 (a) connects L_1 to L_2, and passes closer to the Moon than A1. References A3 (b) and A4 (c) connect L_2 to L_1, and serve as mirrored analogs to references A1 and A2.

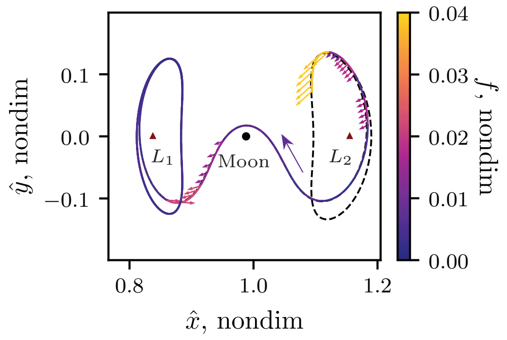


(a) Neural network control history (b) Thrust magnitude history

Figure 6.18. Resulting control history for previous sample deviation, plotted in Figure 6.13, but using reference A2 from Figure 6.17(a). Without additional training, the agent successfully tracks a new reference trajectory.



(a) Control history for Reference A3



(b) Control history for Reference A4

Figure 6.19. Sample computed control histories for the L_2 to L_1 reference trajectories plotted in Figure 6.17(b) and Figure 6.17(c)

Monte Carlo Results

A Monte Carlo analysis approach is applied to evaluate the sample controller's performance across different references. For this analysis, multiple levels of error are simulated for 50 000 trials each across the four sample reference trajectories. Initial position and velocity components are perturbed individually at varying orders of magnitude and larger than potential orbit determination navigation errors of $3\sigma = 1\,\text{km}$ and $1\,{}^{\text{cm}}/_{\text{s}}$, as detailed in Section 13. The proposed controller is tested with errors up to 2000 times the expected navigation error. The 1000× case corresponds to the error used in training, which represents the situation where a 1 km and $1\,{}^{\text{cm}}/_{\text{s}}$ perturbation is introduced, and the error is propagated for an orbital period, or about 12.9 days. Testing the controller beyond the error levels introduced during training provides insight into performance limitations.

The results from the Monte Carlo analysis are summarized in Table 6.7, with failure rates plotted in Figure 6.20. For the training reference A1, the controller is 100 % successful for all errors less than the 500× scenario. However, as the error is increased, trials emerge where recovery does not occur. In the 1000× case, 0.5 % of perturbations are not successfully recovered. For some of these examples, where the error bounds are $3\sigma = 1000\,\text{km}$ and $10\,{}^{\text{m}}/_{\text{s}}$, it is unreasonable to expect a return to the original motion, and these error levels

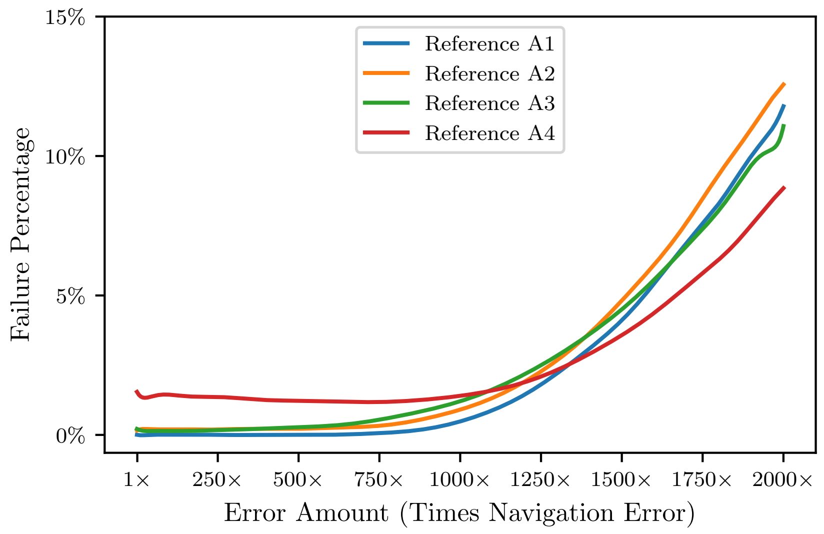


Figure 6.20. Failure rates for Monte Carlo simulations given the four reference trajectories with 50 000 deterministic episodes each, smoothed with polynomial fit. Performance remains mostly level until the error amount exceeds the 1000× threshold.

may correspond to conditions where alternate options should be considered. Further Monte Carlo simulations leveraging reference A1 are visualized in Figure 6.15.

To illustrate the failure characteristics that occur for the 1000× case, deviations over time for 100 sample episodes are represented in Figure 6.21. The red trajectories correspond to failures where the episode is terminated when either of the maximum deviation thresholds is violated, or if arrival conditions are not met within the maximum number of time steps. Episodes that reach the arrival criteria, in green, clearly maintain a small amount of error upon arrival to the target orbit. However, during the transfer phase, more variance is observed in the deviations. This variation is likely caused by the selection of a suboptimal action which forces the agent to recover from additional error.

The arrival percentages for references A2, A3, and A4 further demonstrate the agent's ability to generalize experience from the training reference (A1) to the other three geometries. For reference A2, failure rates are consistently $\approx 0.2\,\%$ less than the A1 test case. However, given the large amount of error, the agent still arrives $99.1\,\%$ of the time in the 1000× case. This high-performance level demonstrates the NN controller's ability to perform well despite

Table 6.7. Arrival percentages for Monte Carlo simulations given various reference trajectories with 50 000 deterministic episodes each. The agent is only trained using reference A1, and generalizes to references A2, A3, and A4 (Figure 6.17). The 1000× case corresponds to the amount of error the agent experiences during training.

	L_1 to L_2		**L_2 to L_1**	
Error Amount	Reference A1	Reference A2	Reference A3	Reference A4
1×	100 %	99.9 %	99.8 %	98.5 %
10×	100 %	99.8 %	99.8 %	98.6 %
100×	100 %	99.8 %	99.9 %	98.6 %
1000×	99.5 %	99.1 %	98.8 %	98.7 %
2000×	88.5 %	87.2 %	88.7 %	91.1 %

significant deviations. For reference A3, where the spacecraft does not approach the Moon, the agent performs exceedingly well, reaching the arrival criteria at least 98.8 % of the time for error levels up to 1000×. Furthermore, when the agent is required to fly nearby the Moon along reference A4, it is still successful in more than 98.5 % of the scenarios for error levels up to 1000×. In consistently satisfying the arrival criteria for this new scenario, the agent demonstrates a remarkable ability to generalize to new geometries. These results could potentially be improved by training the existing agent with the new transfers, i.e., employing transfer learning with the controller, but this possibility was not investigated.

Table 6.7 also demonstrates the controller's robustness to perturbation levels. Despite the three orders of magnitude difference between the 1× and 1000× error distributions, results remain within 1 % for each reference. During training, the agent experiences 1000× error levels and, hence, learns to overcome them. When error is further increased, performance degrades. However, for values up to those used in training, performance only slightly decreases for 3 out of the 4 references, and does not degrade for Reference A4.

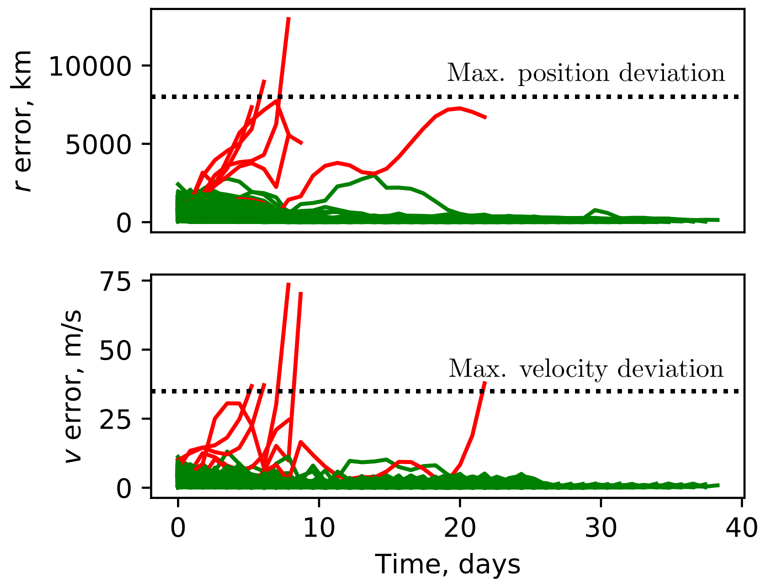


Figure 6.21. Deviation over time for 1000 sample episodes using Reference A1. The initial perturbation variance 3σ is 1000 times greater than the expected navigation error. Green denotes episodes that reach the success criteria, whereas red signifies trajectories that cross the maximum deviation threshold in position or velocity, or do not reach arrival conditions within the maximum number of time steps (87 days). In the trials, 994/1000 episodes are deemed successful for this sample batch of initial conditions.

6.2.4 Training Variation

Additional transfer scenarios are examined to test and understand the learning framework's flexibility. Section 6.2.3 details the training process for an agent on a single reference trajectory, with other transfer scenarios introduced to evaluate performance and generalization. By analyzing a more diverse set of agents and training references, insight is gained into the flexibility of the learning framework itself. In particular, all tuning parameters are heuristically determined to optimize the performance of the agent in Section 6.2.3. Applying the same set of parameters to different problems demonstrates the flexibility of the learning process.

Engine Capability

To analyze the training framework's applicability to more diverse low-thrust spacecraft, two additional thrust levels are examined. As depicted in Table 2.2, a spacecraft with lower thrust, $f_{\max} = 0.02$ nondim, possesses thrusting capability between Hayabusa and Dawn, and a spacecraft with higher thrust, $f_{\max} = 0.08$ nondim, corresponds to a spacecraft with slightly more propulsion capability than Deep Space 1. The algorithm is easily applied to each new scenario and produces a controller that effectively commands the given engine.

Given reference trajectory A1, Figure 6.12(b), training is conducted identically to the sample agent in Section 6.2.3, with new maximum thrust levels applied. Upon completing 150 000 episodes of training, two new controllers are produced for $f_{\max} = 0.02$ and $f_{\max} = 0.08$, nondim. When overcoming simulated deviations, the resulting solutions vary based on the specific spacecraft engine characteristics. To illustrate the impact an engine exerts on the corresponding solution, a single sample perturbation is considered. The three agents, one at each thrust level, are tasked with overcoming the sample perturbation, with corresponding solutions plotted in Figure 6.22. While all three episodes are deemed successful, the solutions clearly vary in the amount of initial thrusting required to return to the reference trajectory. The smallest engine, Figure 6.22(a), requires many segments at nearly maximum thrust. With more propulsive capability, the middle engine, Figure 6.22(b), recovers with only two large initial thrusting arcs. Finally, the largest engine employs a single significant thrusting segment with magnitude 65 % of $f_{\max} = 0.08$.

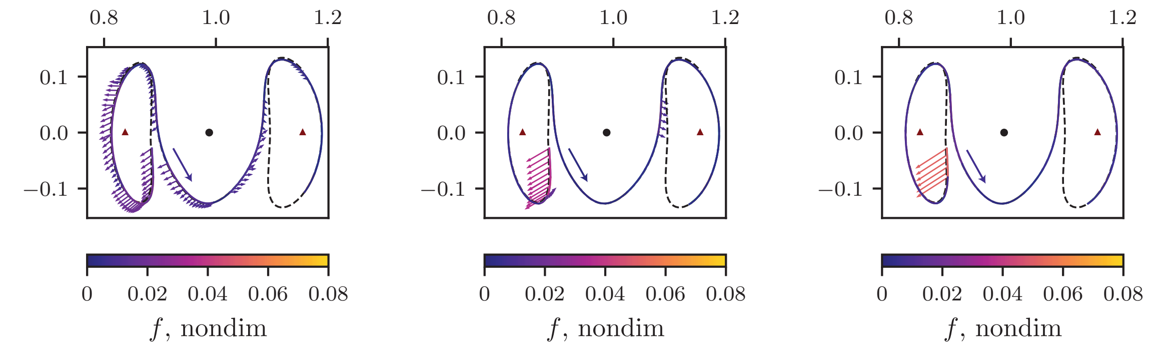
Newly trained agents with $f_{\max} = 0.02$ and $f_{\max} = 0.08$ are simulated in response to various degrees of initial error to analyze maximum thrust and the subsequent performance of the agent. A spacecraft with more propulsive capability is expected to overcome larger deviations and, conversely, a spacecraft with less thrust will overcome smaller perturbations. The result of 10 000 error combinations with 5000 episodes are each plotted in Figure 6.23. Compared to the previous sample spacecraft, Figure 6.15, the yellow region indicating nearly 100 % efficacy extends further for the engine with a higher thrust level Figure 6.23(b), and is more narrow for the less capable engine Figure 6.23(a). While success varies substantially for very large error magnitudes, all three agents are nearly 100 % successful for $3\sigma_r \leq 500$ km and $3\sigma_v \leq 6\,\mathrm{m/s}$. In producing effective agents for both new thrust levels, the resulting agents demonstrate the learning framework's flexibility and indicates applicability to various engine types.

Geometry Variation

To evaluate learning framework flexibility, additional agents are produced by employing different reference geometries in the training process. New reference trajectories are plotted in Figure 6.25. Each scenario, in addition to the four previous references (A1 – A4), is summarized in Table 6.8, where each row represents a separate agent trained on the corresponding transfer. The additional transfers cover a range of energy levels, and exhibit geometries across various regions of space. Transfers (A5 – A7) possess the same Jacobi constant as the four trajectories employed in Section 6.2.3, but differ in both Time of Flight (TOF) and geometry. Reference A5 departs the lunar vicinity through the L_2 gateway, while A6 and A7 venture toward the Earth before returning to the destination orbit.

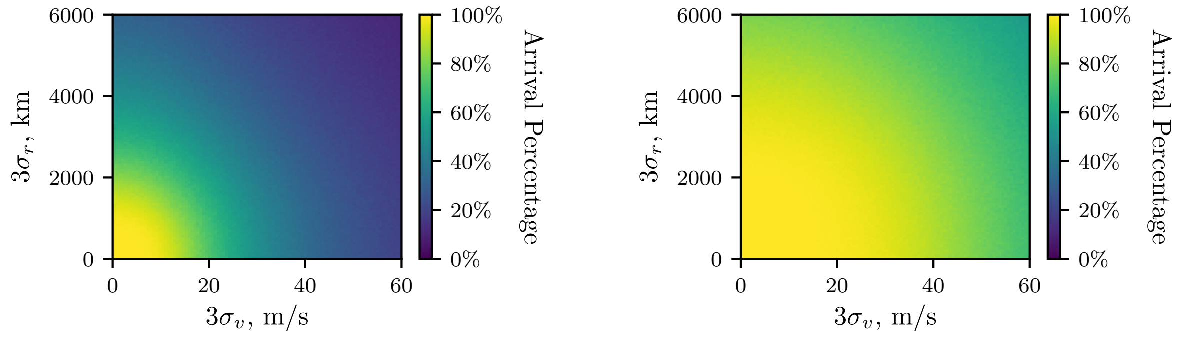
As depicted in Figure 6.24, transfers identified with the letter 'B' correspond to higher energy Lyapunov orbits (red) and, conversely, 'C' indicates lower energy transfers (blue); the 'A' energy level is visualized in purple. Table 6.8 categorizes each transfer by geometry relative to the ZVC. "Lunar Region" indicates that the entire transfer remains in the vicinity of the Moon, "Exterior" transfers pass through the L_2 portal and traverse the region of space outside the ZVC, and "Interior" signifies the transfer passes through the L_1 portal into the enclosed region of space near Earth. After each agent is trained along its corresponding reference, the 'arrival' metric is again leveraged to evaluate performance. Employing $3\sigma_r = 1000\,\mathrm{km}$ and $3\sigma_v = 10\,\mathrm{m/s}$ to model the initial deviation, each agent is simulated over 2000 episodes to compute an overall arrival percentage for each scenario.

High-performing agents are produced in a wide range of scenarios. References A5 and B3 demonstrate the learning framework's ability to overcome longer times of flight and exterior geometries, while A6, A7, and B4 exhibit high performance on interior references. Agents reach at least 99 % arrival on transfers covering all three levels of the Jacobi constant. With several notable exceptions in mind, i.e., C3 – C5, success across most scenarios indicates



(a) $f_{\max} = 0.02$ (b) $f_{\max} = 0.04$ (c) $f_{\max} = 0.08$

Figure 6.22. Control history generated by three separate agents for spacecraft with various propulsive capability. As thrust increases, less time is needed to recover from the initial perturbation.



(a) Lower thrust ($f_{\max} = 0.02$) (b) Higher thrust ($f_{\max} = 0.08$)

Figure 6.23. Arrival percentage along the training reference (A1) given different maximum thrust capability compared to the sample spacecraft in Figure 6.15. As expected, greater thrusting capability overcomes higher levels of initial error.

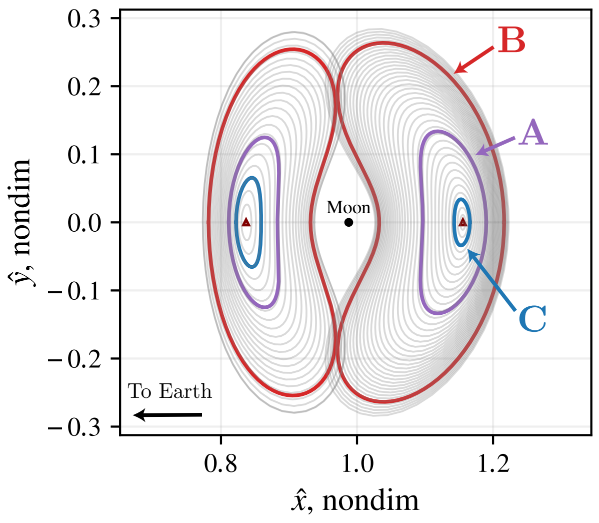


Figure 6.24. Lyapunov orbit energy levels leveraged in this transfer recovery application.

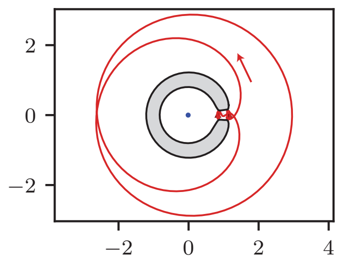
that the current parameters are effective in a wide range of scenarios, but are not globally applicable to every problem.

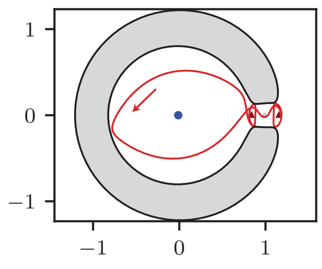
Notably suboptimal agents are produced on three lower energy transfer scenarios (C3 – C5), indicating that energy plays a key role in training parameters and agent performance. Performance degradation on these references indicates that, when changing energy levels, training framework parameters may need adjustment – in particular in response to lower energy situations. More productive agents for these scenarios are likely produced by adjusting certain parameters (e.g., time step length between successive actions, reward tuning values, RL hyperparameters).

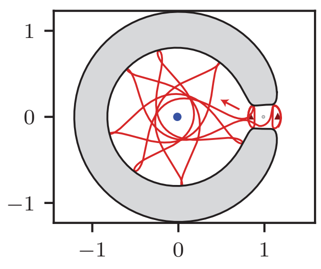
While the controller detailed in Section 6.2.3 demonstrates a remarkable ability in generalizing to new reference trajectories, there are several notable scenarios in which controllers are unable to generate an effective control strategy. Recall transfers are categorized by geometry in Table 6.8. In general, agents trained along lunar region references are unable to generalize to exterior or interior reference geometries. Conversely, since the orbits themselves are in the lunar vicinity, all agents demonstrate some ability to generalize to the lunar region, regardless of reference choice. Generalization between interior and exterior-trained agents does occur, but not always consistently. This inconsistency indicates that, if a drastic change in the reference geometry is possible, generalization performance may be increased by, during training, exposing the controller to all possible space regimes that it may encounter. One potential approach is to initialize each training episode with a randomly selected reference path, but this possibility is not investigated. The observed generalization trends do not hold for every scenario, and further research is necessary to determine precise predictions for NN generalization behavior.

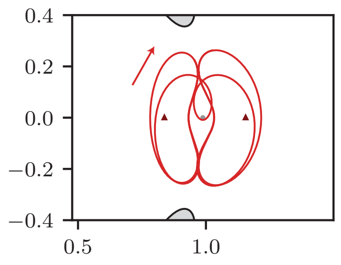
Table 6.8. Reference trajectories used in training. Arrival percentage computed using the training variance ($3\sigma_r = 1000\,\text{km}$, $3\sigma_v = 10\,\text{m/s}$).

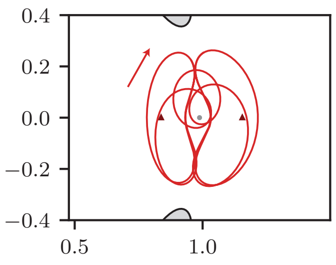
	Label	**Figure No.**	**TOF** (days)	**Lunar Region**	**Exterior**	**Interior**	**Arrival Pct.**
$C = 3.124102$ Section 6.2.3 Energy	A1	Figure 6.12(b)	43.49	✓	–	–	99.7 %
	A2	Figure 6.17(a)	43.49	✓	–	–	99.6 %
	A3	Figure 6.17(b)	39.14	✓	–	–	99.9 %
	A4	Figure 6.17(c)	39.14	✓	–	–	99.5 %
	A5	Figure 6.25(A5)	119.59	–	✓	–	99.6 %
	A6	Figure 6.25(A6)	69.58	–	–	✓	99.1 %
	A7	Figure 6.25(A7)	130.46	–	–	✓	97.2 %
$C = 3.02$ Higher Energy	B1	Figure 6.25(B1)	69.58	✓	–	–	98.5 %
	B2	Figure 6.25(B2)	73.93	✓	–	–	94.1 %
	B3	Figure 6.25(B3)	121.76	–	✓	–	98.3 %
	B4	Figure 6.25(B4)	95.67	–	–	✓	94.2 %
$C = 3.168$ Lower Energy	C1	Figure 6.25(C1)	52.18	✓	–	–	99.1 %
	C2	Figure 6.25(C2)	78.28	✓	–	–	91.9 %
	C3	Figure 6.25(C3)	73.93	✓	–	–	78.3 %
	C4	Figure 6.25(C4)	132.64	–	–	✓	57.8 %
	C5	Figure 6.25(C5)	158.73	–	–	✓	55.2 %

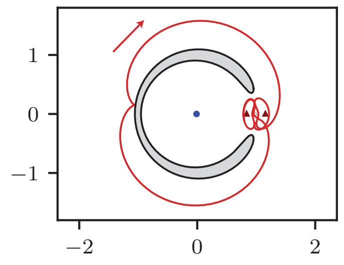

(A5) $C = 3.124102$, TOF $= 119.59$

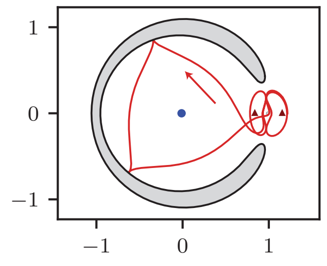

(A6) $C = 3.124102$, TOF $= 69.58$

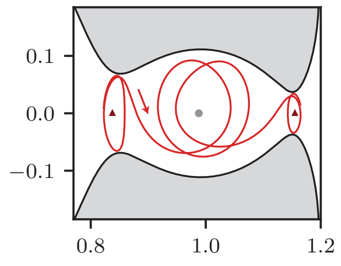

(A7) $C = 3.124102$, TOF $= 130.46$

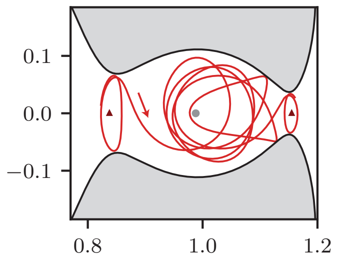

(B1) $C = 3.02$, TOF $= 69.58$

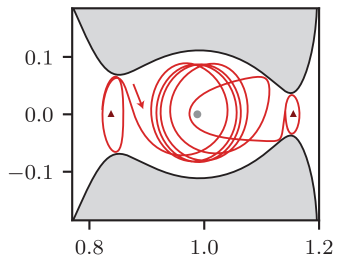

(B2) $C = 3.02$, TOF $= 73.93$

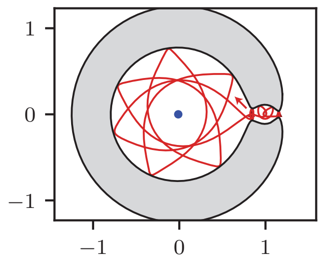

(B3) $C = 3.02$, TOF $= 121.76$


(B4) $C = 3.02$, TOF $= 95.67$


(C1) $C = 3.168$, TOF $= 52.18$


(C2) $C = 3.168$, TOF $= 78.28$


(C3) $C = 3.168$, TOF $= 73.93$


(C4) $C = 3.168$, TOF $= 132.64$

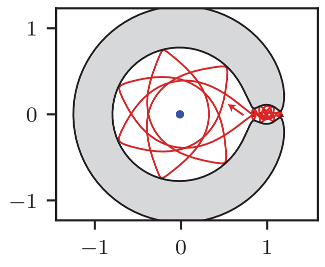

(C5) $C = 3.168$, TOF $= 158.73$

Figure 6.25. Additional transfers employed in training (x-y, nondim). TOF listed in days. Blue and gray circles represent the Earth and the Moon, respectively. Orbits are visualized in Figure 6.24.

6.2.5 NNIT Simulation

The NN-G&C recovery process simulated in Sections 6.2.3 and 6.2.4 is referred to in this investigation as “Standalone NN-G&C” (introduced in Section 6.2.4), where ‘standalone’ refers to fact that the NN directly produces the thrust commands that comprise the recovery response. However, several notable limitations emerge in analyzing these thrust plans that motivate the exploration of an NNIT algorithm, as detailed in Section 5.2. While the previous two sections frequently refer to the NN as a ‘controller’ that computes recovery responses, this alternative NNIT architecture instead understands the NN to be an ‘initializer’ for a targeting algorithm. This shift in terminology reflects the underlying motivation of approaching NN-aided spaceflight as a hybrid, rather than standalone, process. The NNIT approach is employed in this transfer recovery mission application to improve two facets of the standalone NN-G&C recovery maneuver plans: 1) the superfluous number of thrusting segments, and 2) the assumed feasibility of continuously thrusting at any throttle level.

Low-thrust G&C approaches that implement sequential decision-making algorithms (such as RL or dynamical programming) necessarily involve many consecutive control segments. However, in some trajectory recovery situations, a single fixed-orientation low-thrust arc is sufficient for recovering a spacecraft from its induced deviation. Furthermore, the thrust sequencing process employed in this investigation does not account for certain practical considerations, such as the feasibility of instantaneously adjusting attitude between successive segments. This sample mission application explores leveraging the NNIT process to simplify NN-G&C recovery maneuver plans to a single fixed-attitude thrust response. In support of this objective, thrust arc combination heuristics for initial guess construction are derived to enable rapid convergence that accomplishes the original heteroclinic transfer recovery goal.

The second NN-G&C limitation addressed through this mission scenario’s NNIT approach is the exclusion of segments that possess infeasibly low throttle values. For example, as apparent in sample simulation that is visualized in Figure 6.26, the NN employed in this NN-G&C response is unable to produce purely-coasting segments (where $f = 0$). Despite satisfying the stated goal of returning to the reference, the majority of the NN-generated transfer involves oscillations at very low levels of thrust; this oscillatory behavior is apparent in the visualized thrust history in Figure 6.26. Furthermore, the position and velocity relative to the reference trajectory for the scenario, as a normal correspondence, is plotted in Figure 6.27. While the preliminary deviations are large, the initial thrusting segments decrease the relative distances until both position and velocity errors remain bounded. These errors result from small inaccuracies in the NN estimation, causing a constant error introduction/reduction cycle. The NN-G&C maneuver plan is most effective for the initial segments with higher levels of thrust, and less accurate for later segments with lower thrust values. The employed targeting approach, combined with initialization heuristics, enable the construction of recovery plans that admit long-duration coasting segments that ensure all thrust segments are within feasible bounds of the assumed low-thrust propulsion system.

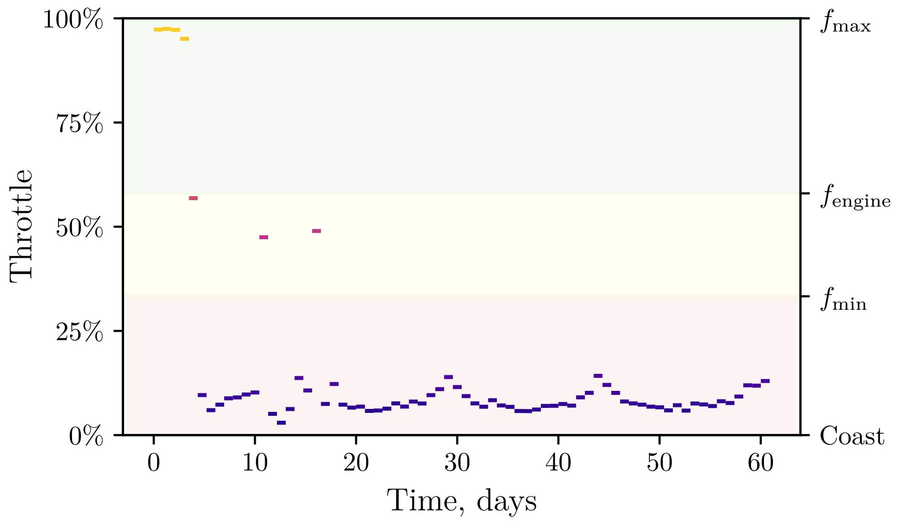


Figure 6.26. Throttle level history for simulation in Figure 6.16(a)

In leveraging an NNIT approach to address the noted limitations of the standalone NN-G&C simulations, it is useful to delineate NN thrust actions by their throttle level $\mathcal{T}$, defined in Equation (2.52). The NNIT algorithm employed in this section enforces a minimum allowable thrust constraint that corresponds to practical limits in low-thrust propulsion system. This boundary value, f_{engine}, forms the top region in Figure 6.26 where green reflects the fact that the given thrust magnitude is physically possible; the selected $\mathcal{T} = 0.58$ value is based on Hayabusa 1 data, as discussed in Table 6.9. Furthermore, it is useful to leverage

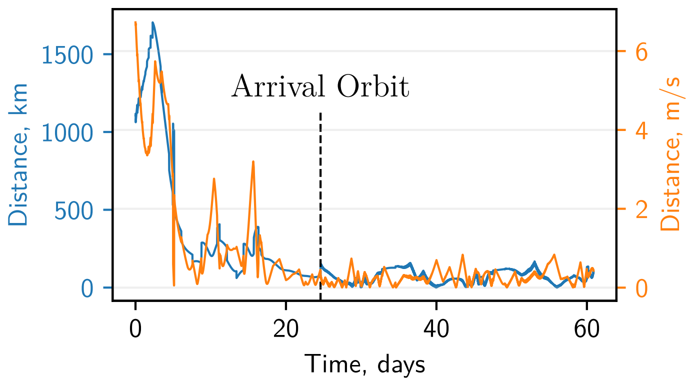


Figure 6.27. Variation from reference trajectory for the simulation visualized in Figure 6.16(a)

targeting techniques or low-thrust magnitude scaling heuristics (detailed in Section 3.4.2) to enforce this condition. However, as throttle decreases, so too does the impact of low-thrust on the spacecraft's dynamics, thus causing sensitivity to thrust direction to diminish. When this occurs, the orientation of a low-thrust arc serves as a poor initial guess for the targeting algorithm. This mission scenario's NNIT algorithm addresses this challenge by defining an $f_{\min}$ threshold that is heuristically selected to delineate segments that are better treated as ballistic, represented by the red region in Figure 6.26. The value of $f_{\min}$ is specific to the observed NN behavior and, for this transfer recovery problem, the throttle value $\mathcal{T} = 0.33$ provides a useful lower bound. The remaining yellow region in Figure 6.26 indicates that the given thrust value falls below the f_{engine} limit, but still serves as a sufficiently accurate initial guess for a targeting problem. The threshold values employed in this investigation are each summarized in Table 6.9.

Low-Thrust Initial Guess Transformation Heuristics

As depicted in the information flow schematic in Figure 5.1, the NNIT process involves a transformation step that converts an NN-G&C maneuver plan into an initial guess for a multiple shooting targeting algorithm. In the simplest form, this transformation process simply involves transcribing the sequence of observations and actions from the NN-G&C solution into the free variable vector $\boldsymbol{X}$ for a targeting process with state continuity enforced

Table 6.9. Low-thrust magnitude thresholds employed in this NNIT implementation, visualized in the boundary between colors in Figure 6.26

f threshold	$\mathcal{T}$ threshold	Description
$f = 0$	$\mathcal{T} = 0$	"Coast" arcs that possess purely ballistic motion, thus equating the low-thrust CR3BP equations of motion (Equation (2.60)) with the natural equations of motion (Equation (2.27)).
$f = f_{\text{min}}$	$\mathcal{T} = 0.33$	A lower-bound that is intended to identify thrust segments that supply useful thrust direction information (the 0.33 value is heuristically selected from experiments).
$f = f_{\text{engine}}$	$\mathcal{T} = 0.58$	The minimum thrust that a low-thrust engine is physically capable of producing. The $\mathcal{T} = 0.58$ value is selected for this sample mission application from Hayabusa 1 [74]
$f = f_{\text{max}}$	$\mathcal{T} = 1$	The maximum possible thrust that a low-thrust engine can produce.

by the $\boldsymbol{F}(\boldsymbol{X})$ constraint vector (this constraint-free variable targeting approach is detailed in Section 3.3). Some low-thrust NNIT applications benefit from additional transformation techniques to account for problem-specific considerations or known limitations of the employed NN-G&C approach. For example, in NNIT applications that employ a pre-trained NN, practical constraints (such as instantaneous attitude changes in or minimum thrust thresholds) may be absent from the NN-G&C training process or inadvertently violated by the resulting NN estimation.

For maneuver plans generated by the NN-G&C recovery response, control values are assumed constant over integration segments and, hence, subsequent thrust arcs include an assumed instantaneous attitude and power shift as the spacecraft transitions from one segment to another. The inclusion of multiple sequential thrust segments is, for many applications, an artifact of the employed numerical method and not always necessary for solving the given problem. In these cases, a strategy for combining multiple control arcs is useful to yield a single segment that is representative of the combined effect of each discrete component. This combination process is particularly productive for NN controllers that do not estimate time values and, instead, assume all control variables offered by the NN occur over a pre-defined time interval. For RL approaches, in particular, these time intervals are typically short and, hence, a NN-G&C solution may include many interactions with the NN controller. In these cases, computing a unified guess for thrust magnitude, direction, and time is challenging due to occasional large shifts in control values between successive thrust arcs produced by the NN. To address these discontinuities, variation in thrust direction and mass are incorporated into the combination process to yield accurate initial guesses that are representative of the NN-suggested motion. Furthermore, smoothing of the discontinuities may be incorporated into the NN training process by, for example, penalizing a change in thrust direction [58], but this possibility is not currently investigated.

An averaging approach is employed to compute a single thrust arc that estimates the combined effect of multiple sequential thrusting segments. Define $\mathfrak{L}$ to be a sequence of n low-thrust segments $\mathfrak{s} = \{\boldsymbol{\rho}, \hat{u}, f, \mathbb{t}\}$ that share the same thrust time $\mathbb{t}$, i.e.,

$$\mathfrak{L} = \left\{\mathfrak{s}_i\right\}_{i=1}^{n} = \left\{\boldsymbol{\rho}_i \quad \hat{u}_i \quad f_i \quad \mathbb{t}\right\}_{i=1}^{n} \tag{6.16}$$

The representative thrust vector across the multiple control segments in $\mathfrak{L}$ is termed the 'combined' thrust segment $\mathfrak{s}_\text{c}$. The low-thrust command represented by $\mathfrak{s}_\text{c}$ is oriented in the averaged direction of the n nondimensional low-thrust acceleration vectors $\boldsymbol{a}_i^\text{lt}$, defined in Equation (2.53), for the segments in $\mathfrak{L}$, i.e.,

$$\boldsymbol{a}_\text{avg}^\text{lt} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{a}_i^\text{lt} = \sum_{i=1}^{n} \frac{f_i}{m_i}\hat{u}_i \tag{6.17}$$

Hence, the thrust orientation unit vector $\hat{u}_\text{avg}$ for $\mathfrak{s}_\text{c}$ is obtained by simply normalizing $\boldsymbol{a}_\text{avg}^\text{lt}$,

$$\hat{u}_\text{avg} = \frac{\boldsymbol{a}_\text{avg}^\text{lt}}{|\boldsymbol{a}_\text{avg}^\text{lt}|} \tag{6.18}$$

An alternative approach directly averages the thrust directions of the individual segments, $\hat{u}_i$, which does not incorporate the thrust level along each path. With the goal of creating a single representative segment, determining the control direction via the acceleration vectors via Equation (6.18) reflects a belief that segments with higher thrust values impose larger impacts on the unified solution. Then, the combined nondimensional thrust magnitude of $\mathfrak{s}_\text{c}$, f_avg, is derived from multiplying the acceleration vector magnitude by the initial nondimensional mass value, i.e.,

$$f_\text{avg} = m_0|\boldsymbol{a}^\text{lt}_\text{avg}| \tag{6.19}$$

The expressions for $\hat{u}_\text{avg}$ and f_avg assume the feasibility of representing the discrete thrust segments by a single fixed-orientation thrust arc. As propagation time and control complexity increases, the set of n control arcs may be subdivided into multiple subsets to produce more accurate estimations.

A key challenge in combining thrust segments is estimating the final propagation time. The n segments that comprise $\mathfrak{L}$, Equation (6.16), each possess a thrust time $\mathfrak{t}$. Hence, the sum of the nondimensional times across $\mathfrak{L}$ is simply,

$$\mathfrak{t}_\text{total} = n\mathfrak{t} \tag{6.20}$$

Employing $\mathfrak{t}_\text{total}$ as the time estimate for the combined segment $\mathfrak{s}_\text{c}$ frequently results in overestimations that prohibit convergence. To combat this phenomenon, a weighted sum for total thrust time, $\mathfrak{t}_\text{avg}$, is computed as the L2 norm of the vector sum,

$$\mathfrak{t}_\text{avg} = \left|\sum_{i=1}^{n}\left(\frac{f_i}{f_\text{max}}\right)\left(\frac{m_0}{m_i}\right)\hat{u}_i\mathfrak{t}\right|_2 \tag{6.21}$$

where $({}^{f_i}/_{f_\text{max}})$ prioritizes segments with larger thrust magnitudes such that they carry greater influence in the propagation time summation; $({}^{m_0}/_{m_i})$ then compensates for mass variation across the segments. Including $\hat{u}_i$ in the summation decreases the thrust time estimate when large variations in control occur. Rearranging Equation (6.21), and substituting the expressions in Equations (6.18) and (6.19), yields,

$$\mathfrak{t}_\text{avg} = \frac{f_\text{avg}}{f_\text{max}}\mathfrak{t}_\text{total} \tag{6.22}$$

The derived values for $\hat{u}_\text{avg}$, f_avg, and $\mathfrak{t}_\text{avg}$ are leveraged to form combined thrust segment,

$$\mathfrak{s}_\text{c} = \left\{\boldsymbol{\rho}_0 \quad \hat{u}_\text{avg} \quad f_\text{avg} \quad \mathfrak{t}_\text{avg}\right\} \tag{6.23}$$

where $\boldsymbol{\rho}_0$ is the initial position, velocity, and mass of the first segment in $\mathfrak{L}$. The procedure for performing this thrust combination initialization process is detailed in Algorithm 6.1.

Algorithm 6.1: `CombineThrustSegments`
Combine successive low-thrust thrust segments into a unified thrust arc

```
Input: $\mathfrak{L}$: Low-thrust trajectory sequence with $n$ segments, $\mathfrak{L} = \{\mathfrak{s}_1, \cdots, \mathfrak{s}_n\}$. Each
       segment is defined as $\mathfrak{s}_i = \{\mathbf{q}_i, f_i, u_i, \mathfrak{t}\}$ where $\mathfrak{t}$ is the same for each arc.
Output: $\mathfrak{s}_\text{c}$: Low-thrust segment that represents the average impact of the input
        trajectory sequence

// Compute average acceleration vector, Equation (6.17)
$\boldsymbol{a}^\text{lt}_\text{avg} \leftarrow [0, 0, 0]$
foreach $\mathfrak{s}$ in $\mathfrak{L}$ do
    $\boldsymbol{a}^\text{lt}_\text{avg} \leftarrow \boldsymbol{a}^\text{lt}_\text{avg} + (\mathfrak{s}.f/\mathfrak{s}.m) * \mathfrak{s}.\hat{u}$

// Normalize $\boldsymbol{a}^\text{lt}_\text{avg}$ for average direction vector, Equation (6.18)
$\hat{u}_\text{avg} \leftarrow \boldsymbol{a}^\text{lt}_\text{avg}/\texttt{norm}(\boldsymbol{a}^\text{lt}_\text{avg})$

// Compute averaged thrust magnitude, Equation (6.19)
$f_\text{avg} \leftarrow m_0 * \texttt{norm}(\boldsymbol{a}^\text{lt}_\text{avg})$

// Estimate thrust time of combined segment, Equation (6.22)
$\mathfrak{t}_\text{avg} \leftarrow (f_\text{avg}/f_\text{max}) * (n\mathfrak{t})$

// Return result
$\mathfrak{s}_\text{c} \leftarrow \{\mathbf{q}_0, \hat{u}_\text{avg}, \hat{u}_\text{avg}, \mathfrak{t}_\text{avg}\}$
return $\mathfrak{s}_\text{c}$
```

The combined low-thrust magnitude value f_avg does not always fall withing the practical limits of the spacecraft. In this mission application, an additional heuristic is employed to transform the thrust segment that results from the above combination approach into practical bounds using the thrust magnitude adjustment relationships detailed in Section 3.4.2. Given a pre-selected desired thrust magnitude, the thrust scaling approach leverages the ideal rocket equation to compute an updated thrust time at the desired value of f that results in the same $\Delta\tilde{V}_\text{equiv.}$. In this approach, any thrust magnitude greater than the specified lower bound, f_engine, may be arbitrarily selected to transform the segment into practical bounds for a given engine. One option available in this selection process is to leverage prior knowledge from optimal control theory to bias the targeting solution. For CSI engines in a two-body model, Longuski et al. derive a bang-bang optimal control law that alternates between null-thrust (coast) and maximum-thrust ($f_{\max}$) arcs [134]. While this analysis does not involve optimization, $f = f_{\max}$ is leveraged as the desired magnitude value to bias the initial guess toward a propellant-efficient solution. Applying this transformation to the combined thrust segment in Equation (6.23) delivers a new thrust arc at $f_{\max}$,

$$\mathfrak{s}_{\text{ic}} = \left\{ \boldsymbol{\rho}_0 \quad \hat{u}_{\text{avg}} \quad f_{\max} \quad \mathfrak{t}_{\text{ic}} \right\} , \qquad \text{where} \qquad \mathfrak{t}_{\text{ic}} = \left(\frac{f_{\text{avg}}}{f_{\max}} \right) \mathfrak{t}_{\text{avg}} \tag{6.24}$$

An important point to consider in this selection is that if the parameterized thrust magnitude f in Equation (3.33) is employed as a free variable, it will not be updated during targeting due to the zero gradient at the $f = f_{\max}$ extremum.

Sample Explanatory Transformation Process

To illustrate the complete combination process, a sample sequence of three thrust arcs in the vicinity of L_1 is depicted in Figure 6.28(a), where the arrow directions and lengths signify thrust direction and magnitude, respectively, and thrust time $\mathfrak{t} \approx 20.87\,\text{h}$. Specific thrust command values are summarized in Table 6.10. For this sample scenario, Equations (6.18), (6.19) and (6.22) are leveraged to combine the three sample segments into the single representative thrust arc in Figure 6.28(b). In this case, including $\hat{u}_i$ in the computation of f_{avg} and $\mathfrak{t}_{\text{total}}$ lessens the impact of the third arc on the overall solution since f_3 is only 20 % of $f_{\max}$.

While the thrusting segment in Figure 6.28(b) resembles Figure 6.28(a), the simulated low-thrust engine is not physically able to produce a propulsive force at this magnitude, since $f_{\text{avg}} < f_{\text{engine}}$. Considering the potential implementation of the initial guess, Equation (3.47) is leveraged to compute the thrust time that achieves the same $\Delta\tilde{V}_{\text{equiv.}}$ (4.11 m/s) when the engine thrusts at maximum power, $f_{\max}$. This conversion is visualized in Figure 6.29, where the thrust level corresponding to Figure 6.28(b) is red, and maximum thrust is black. Thrust time is scaled between 41 % and 100 % by noting the intersection points along the dashed line, ensuring that $\Delta\tilde{V}_{\text{equiv.}}$ and mass consumption are preserved.

Table 6.10. Thrust commands for sample scenario depicted in Figure 6.28.

	$\mathcal{T}$	$\hat{u} = [u_x, u_y]$	$\mathfrak{t}$, nondim	$\mathfrak{t}$, hours	$\Delta\tilde{V}_{\text{equiv.}}$, m/s
Segment 1	0.5	$[-1, 0]$	0.2	20.87	4.10
Segment 2	0.9	$[-0.9578, 0.2873]$	0.2	20.87	7.38
Segment 3	0.2	$[0.7071, -0.7071]$	0.2	20.87	1.64
Combined	0.41	$[-0.9954, 0.0955]$	0.245	25.60	4.11
Adjusted	1	$[-0.9954, 0.0955]$	0.100	10.46	4.11

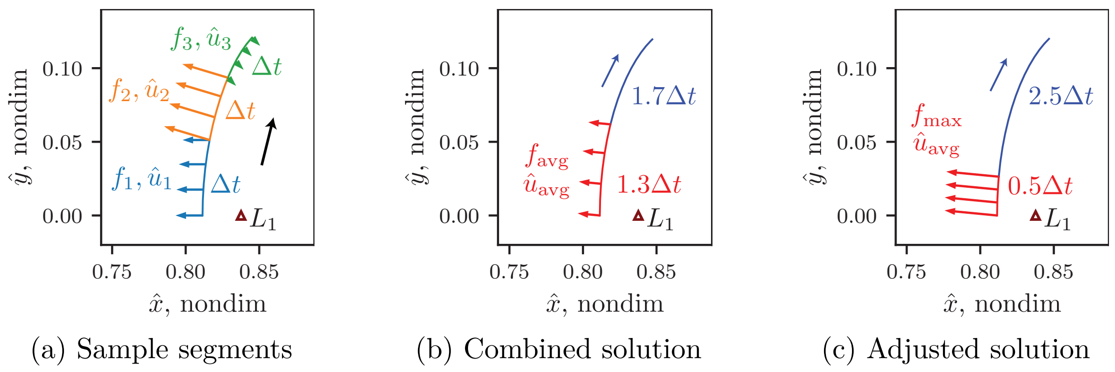


Figure 6.28. Sample thrust arc sequence combination where three discrete thrust segments (a) are combined into a single thrust arc where total thrust time is decreased by 56.7 % (b). Thrust command values are listed in Table 6.10.

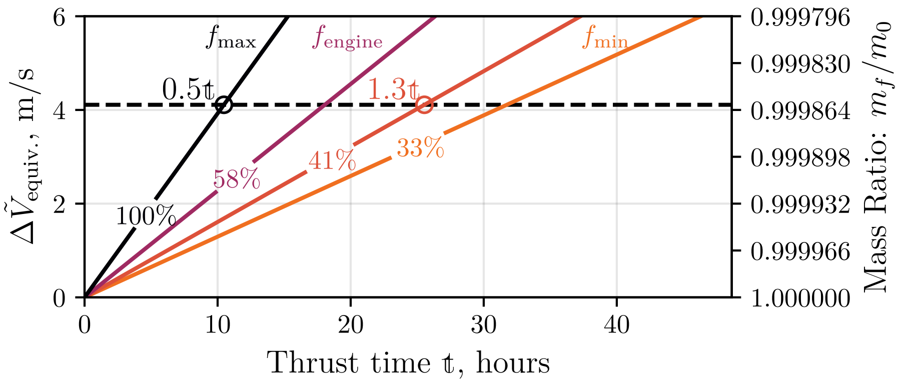


Figure 6.29. Relationship between $\Delta\tilde{V}_{\text{equiv.}}$ and thrust time for an ideal rocket with a CSI engine. Horizontal line denotes the thrust time adjustment for increasing f_{avg} (red) to f_{max} (black) for the segment depicted Figure 6.28(b), resulting in a new thrust arc in Figure 6.28(c).

NNIT Implementation

The NNIT approach is formalized by defining the initial guess transformation process and the multiple shooting targeting scheme, as visualized in Figure 5.1. For a standalone NN-G&C algorithm, let $\mathfrak{L}_{\text{NN}}$ represent the sequence of $\mathfrak{n}$ low-thrust segments with time $\mathfrak{t}$ that result from a NN-G&C simulation; a low-thrust trajectory $\mathfrak{L}$ is generally defined in Equation (6.16). The NNIT process commences by transforming $\mathfrak{L}_{\text{NN}}$ into an initial guess for a constraint-free variable targeting method that is subsequently iterated on to produce a feasible maneuver plan in the CR3BP. The procedure for simulating this transfer scenario's implementation of NNIT is provided in Algorithm 6.2. This initialization process requires multiple transformation steps, as visualized in the information flow schematic in Figure 6.30, where each step incrementally transforms the NN-G&C simulated maneuver plan into a practical startup solution for a differential corrections algorithm. A key motivation for this NNIT implementation is the observation that the overall NN-G&C response is primarily driven by the initial control choices. This behavior is expected, as large deviations are only introduced at the initial state; thus, as evident in Figure 6.26, the beginning thrust segments of $\mathfrak{L}_{\text{NN}}$ carry the greatest relative value for performing the given recovery task. To capitalize on this insight, an NNIT initial guess transformation process is derived to identify and employ these crucial early segments of the recovery plan.

As detailed in Figure 6.30 and Algorithm 6.2, the NNIT initial guess transformation process in this transfer recovery application begins by identifying the n thrust segments from $\mathfrak{L}_{\text{NN}}$, represented as $\mathfrak{L}_{\text{r}}$, that comprise the initial recovery response. The number n is determined by the first location in the $\mathfrak{L}_{\text{NN}}$ sequence where the thrust f or the combined effect of averaging the suggested control segments f_{avg}, defined in Equation (6.23), falls below the selected value for f_{min}, listed in Table 6.9. The procedure for averaging thrust arcs is provided in Algorithm 6.1. This thrust threshold criterion assumes that the NN tends to output low values for f_i when the distance to the reference trajectory is small and, hence, indicates that the recovery task is complete. To account for cases in which the NN delays in its response to a perturbation, i.e., initially produces a nearly-coasting segment followed by a large amount of thrust, at least two thrust arcs are always included along the trajectory.

The final state and the end of the series of initial recovery segments is assumed to terminate in the vicinity of the reference trajectory.

Once the initial recovery sequence $\mathfrak{L}_\mathrm{r}$ is identified, the transformation heuristics derived in Section 6.2.5 are employed. First, the segments in $\mathfrak{L}_\mathrm{r}$ are combined into a single thrust arc $\mathfrak{s}_\mathrm{c}$ (as defined in Equation (6.23)). Next, $\mathfrak{s}_\mathrm{c}$ is scaled to $\mathfrak{s}_\mathrm{ic}$ (i.e., $\{f_\mathrm{avg}, \mathfrak{t}_\mathrm{avg}\} \to \{f_\mathrm{max}, \mathfrak{t}_\mathrm{ic}\}$) to ensure the throttle value is within practical bounds, detailed in Equation (6.24). The terminal state along segment $\mathfrak{s}_\mathrm{ic}$ is assumed to be in the vicinity of the reference trajectory, hence a nearest neighbor search reveals the 'closest' transfer state $\boldsymbol{\varrho}_\mathrm{ref} \in \mathcal{R}^\mathrm{transfer}$, as defined in Equation (6.9), where the time between $\boldsymbol{\varrho}_\mathrm{ref}$ and the end of the transfer is represented by t_traj. The targeter is tasked with correcting the discontinuity between the recovery segment and reference trajectories. Finally, several revolutions of the arrival orbit are stacked to produce a transfer the terminates in the vicinity of the desired orbit. Each of these transformation steps is visualized in Figure 6.30 and detailed in Algorithm 6.2.

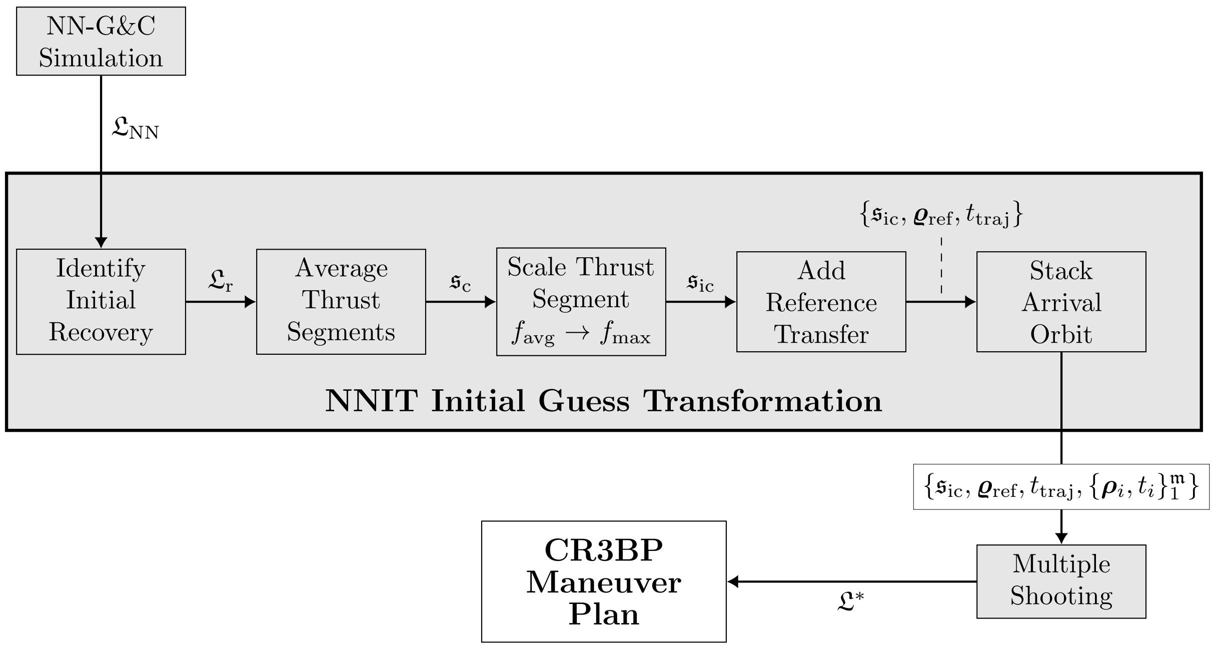

NN-G&C Simulation
$\mathfrak{L}_\mathrm{NN}$
Identify Initial Recovery
$\mathfrak{L}_\mathrm{r}$
Average Thrust Segments
$\mathfrak{s}_\mathrm{c}$
Scale Thrust Segment $f_\mathrm{avg} \to f_\mathrm{max}$
$\mathfrak{s}_\mathrm{ic}$
Add Reference Transfer
$\{\mathfrak{s}_\mathrm{ic}, \boldsymbol{\varrho}_\mathrm{ref}, t_\mathrm{traj}\}$
Stack Arrival Orbit
NNIT Initial Guess Transformation
$\{\mathfrak{s}_\mathrm{ic}, \boldsymbol{\varrho}_\mathrm{ref}, t_\mathrm{traj}, \{\boldsymbol{\rho}_i, t_i\}_1^\mathfrak{m}\}$
Multiple Shooting
$\mathfrak{L}^*$
CR3BP Maneuver Plan

Figure 6.30. Information flow schematic for the NNIT initial guess generation procedure employed in the heteroclinic transfer mission application

Algorithm 6.2: `SimulateNNIT`

NNIT process for heteroclinic transfer problem

```
Input: $\mathfrak{L}_{\text{NN}}$: NN-G&C trajectory of $\mathfrak{n}$ low-thrust arcs, $\mathfrak{L}_{\text{NN}} = \{\mathfrak{s}_1, \cdots, \mathfrak{s}_\mathfrak{n}\}$. Each
       segment is defined as $\mathfrak{s}_i = \{\mathbf{q}_i, f_i, u_i, \mathfrak{t}\}$ where $\mathfrak{t}$ is the same for each arc.
Output: $\mathfrak{L}^*$: Feasible maneuver plan that results from the targeting algorithm

// Initialize problem to always include the first segment
$i \leftarrow 1$
$\mathfrak{L}_\text{r} \leftarrow [\mathfrak{s}_1]$
$\mathfrak{s}_\text{c} \leftarrow \mathfrak{s}_1$

// Proceed through trajectory $\mathfrak{L}_{\text{NN}}$ until the thrust magnitude (or
   averaged thrust magnitude) falls below the minimum threshold $f_{\min}$
while $i \leq \mathfrak{n}$ and $\mathfrak{s}_i.f > f_{\min}$ do
    $\mathfrak{L}_\text{r}$.append($\mathfrak{s}_i$)
    $\mathfrak{s}_\text{c} \leftarrow$ CombineThrustSegments($\mathfrak{L}_\text{r}$)
    if $\mathfrak{s}_\text{c}.f > f_{\min}$ then
        $i \leftarrow i + 1$
    else
        break

// Scale the averaged thrust segment to $f_{\max}$, Equation (3.45)
$\mathfrak{t}_{\text{scaled}} = (\mathfrak{s}_\text{c}.f / f_{\max}) * (\mathfrak{s}_\text{c}.\mathfrak{t})$

// Define the combined and adjusted low-thrust segment
$\mathfrak{s}_{\text{ic}} \leftarrow \{\mathfrak{s}_0.\mathbf{q}, \hat{u}_{\text{avg}}, f_{\max}, \mathfrak{t}_{\text{scaled}}\}$

// Estimate intersection point along transfer
$\boldsymbol{\varrho}_{\text{ref}} \leftarrow$ NearestNeighbor($\mathfrak{s}_\text{c}.\mathbf{q}_f, \mathcal{R}^{\text{transfer}}$)

// Run multiple shooting algorithm to enforce continuity
$\mathfrak{L}^* \leftarrow$ TargetRecovery($\mathfrak{s}_{\text{ic}}$)
```

Targeting algorithm

The targeting method derived in Section 3.3 is constructed by selecting the free variables and constraints for a given problem. In this recovery application, the design variables updated through the targeting process are divided into three distinct categories, as depicted in Figure 6.31. First, the red arc represents the thrust segment produced from the heuristic transformation steps. For simplicity, the targeting algorithm is described in this section using numeric subscripts to indicate the ordering of segments. Hence, this initial thrust segment originates at a fixed initial state $\boldsymbol{\rho}_0$, and is parameterized by θ_0, f_0, and $\mathbb{t}_0$. Startup values for control variables are received from $\mathfrak{s}_{\text{ic}}$, defined in Equation (6.24), where, initially, θ_0 is the angle between $\hat{u}_{\text{avg}}$ and the rotating $\hat{x}$ axis, $f_0 = f_{\max}$, and $\mathbb{t}_0 = \mathbb{t}_{\text{ic}}$. The trained NN only affects this initial control segment, with the remainder of the initial guess for the targeting process constructed from the original reference trajectory and target orbit. The second part of the startup solution, plotted in blue, corresponds to the initial reference trajectory depicted in Figure 6.12(b). To minimize state discontinuity, the initial state along the reference $\boldsymbol{\rho}_1$ is selected by conducting a nearest-neighbor search in position and velocity between the final state along the thrust arc, $\boldsymbol{\rho}_{0,f}$, and the reference trajectory. The reference trajectory time, t_1, is simply the time between $\boldsymbol{\rho}_1$ and the end of the reference trajectory, and possesses and initial value of t_{traj}.

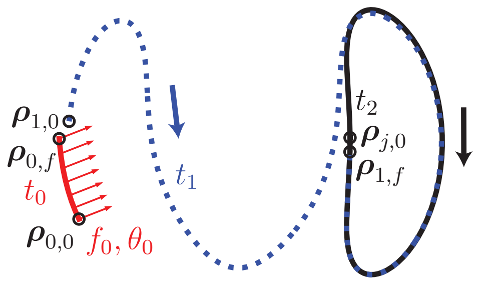

$\boldsymbol{\rho}_{1,0}$
$\boldsymbol{\rho}_{0,f}$
t_0
$\boldsymbol{\rho}_{0,0}$
f_0, θ_0
t_1
t_2
$\boldsymbol{\rho}_{j,0}$
$\boldsymbol{\rho}_{1,f}$

Figure 6.31. Targeting variables for NNIT multiple shooting algorithm

Revolutions of the arrival orbit comprise the final step of the startup solution, represented by the black orbit in Figure 6.31. In this approach, a nearest-neighbor search is again conducted between the end state along the reference trajectory, $\boldsymbol{\rho}_{1,f}$, and the orange arrival periodic orbit in Figure 6.12(a). Assuming this orbit possesses a period t_2, and the nearest-neighbor state is $\boldsymbol{\rho}_2$, $\mathfrak{m}$ revolutions are included in the initial guess, where $\mathfrak{m} = 4$ is employed in this investigation. Stacking revolutions biases the converged path to maintain Lyapunov orbit-like motion, however, does not ensure that the specific energy level is preserved. A different method for formalizing the terminal condition involves constraining $\boldsymbol{\rho}_{1,f}$ to be precisely on the arrival orbit's stable manifold, as introduced by Haapala and Howell [129]. This manifold constraint approach, while effective in many applications, presents challenges for an autonomous targeting algorithm due to its sensitivity to the selected initial time parameter that locates the manifold step-off state. Thus, utilizing stacked revolutions offers greater adaptability in achieving consistent closed-loop convergence for this NNIT algorithm.

The transfer recovery targeting process in this recovery application follows the general formulation for low-thrust targeting in the CR3BP that is detailed in Section 3.4. To ensure that the thrust magnitude remains bounded, i.e., $f \in [0, f_{\max}]$, the parameterized thrust magnitude variable $\breve{f}$, defined in Equation (3.33), is employed. Formulated as design variables, $\boldsymbol{X}$, and constraints, $\boldsymbol{F}(\boldsymbol{X})$, the targeting approach is summarized as,

$$\boldsymbol{X} = \left(\underbrace{\breve{f}_0 \quad \theta_0 \quad \boldsymbol{\rho}_0 \quad \mathfrak{t}_0}_{\text{Thrust Arc}} \quad \underbrace{\boldsymbol{\rho}_1 \quad t_1}_{\text{Ref. Traj.}} \quad \underbrace{\left\{ \boldsymbol{\rho}_j \quad t_2 \right\}_{j=2}^{\mathfrak{m}+1}}_{\text{Arrival Orbit}} \right)^T \qquad \boldsymbol{F}(\boldsymbol{X}) = \begin{pmatrix} \boldsymbol{\rho}_{0,f} - \boldsymbol{\rho}_{1,0} \\ \boldsymbol{\rho}_{1,f} - \boldsymbol{\rho}_{j,0} \\ \boldsymbol{\rho}_{j,f} - \boldsymbol{\rho}_{j+1,0} \\ \vdots \\ \boldsymbol{\rho}_{\mathfrak{m},f} - \boldsymbol{\rho}_{\mathfrak{m}+1,0} \end{pmatrix} \tag{6.25}$$

where Figure 6.31 depicts the "Thrust Arc" (red), the "Ref. Traj." (blue) and the "Arrival Orbit" (black). This targeter serves as a stand-in for any iterative approach that leverages a startup solution. The targeting scheme here is not intended to satisfy the strict requirements of flight software, but rather to illustrate an RL-generated controller and its incorporation into the initial guess generation process for existing corrections and optimization schemes. A NN-enabled targeting guidance framework is an appealing hybrid approach, offering the NN efficiency alongside targeting robustness to ensuring all mission criteria are satisfied.

Representative NNIT Scenarios

Three representative scenarios of the NNIT initial guess transformation process are depicted in Figures 6.32 to 6.34. For each scenario, plot (a) depicts the throttle control history of the NN-G&C initial recovery trajectory $\mathfrak{L}_{\mathrm{r}}$, where the colored horizontal lines represent NN-generated nondimensional thrust magnitude values (f_i) over time, and gray dashed lines signify the running value for the combined thrust magnitude value (f_{avg}), updated given each new control segment i by Equation (6.19). Next, the segments that comprise $\mathfrak{L}_{\mathrm{r}}$ are visualized in the first plot of (b) for each scenario, followed by the combined segment $\mathfrak{s}_{\mathrm{c}}$, and the scaled (i.e., "Adjusted"), initial guess $\mathfrak{s}_{\mathrm{ic}}$; recall, these transformation steps are visualized in Figure 6.30. The three scenarios are each summarized as follows:

- Scenario 1 (Figure 6.32) corresponds to the most common, expected outcome of the NN-G&C process. In this case, the NN selects three sequential thrust segments without large direction changes, and a fourth thrust arc that possesses a small throttle value. The NNIT transformation step recognizes that the spacecraft reaches the vicinity of the reference trajectory by observing $f_4 < f_{\mathrm{min}}$, plotted in Figure 6.32(a). Crossing the f_{min} threshold terminates thrust segment sequencing, and the $\mathfrak{s}_4$ is not included in the initial guess. Given $\mathfrak{L}_{\mathrm{r}} = [\mathfrak{s}_1 \;\; \mathfrak{s}_2 \;\; \mathfrak{s}_3]$, the direction of the combined and adjusted solutions in (b) and (c) closely resembles the NN-generated arcs, with a 12.9-hour reduction in thrust time for the combined case, and an additional 10.25-hour reduction for the final adjusted solution.

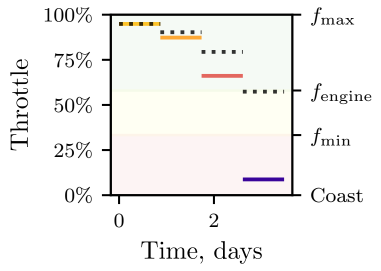
- Scenario 2 (Figure 6.33) illustrates a case where the NN-generated thrust sequence terminates when the average thrust is small, i.e., $f_{\mathrm{avg}} < f_{\mathrm{min}}$. Due to the large discrepancy between $\hat{u}_1$ and $\hat{u}_2$, visualized in Figure 6.33(b) under "NN Guidance", f_{avg} is characterized by a small throttle magnitude despite both f_1 and f_2 possessing large magnitudes, depicted in Figure 6.33(a). The 136° difference between $\hat{u}_1$ and $\hat{u}_2$ results in a combined solution that drastically reduces the thrust magnitude. With $f_{\mathrm{avg}} < f_{\mathrm{min}}$, the combined solution indicates that coasting is appropriate for the original state estimate, and the scaling step is not necessary.

- Scenario 3 (Figure 6.34) demonstrates the case where the NNIT initial guess transformation process proceeds past the first segment despite $f_1 < f_{\text{min}}$. In this case, the NN is delayed in its response to the perturbation, and a large second thrust magnitude, f_2, is computed, depicted in Figure 6.34(a). This second segment is added to the trajectory $\mathfrak{L}_{\text{r}}$, and the process terminates when $f_3 < f_{\text{min}}$ occurs. This scenario illustrates the transformation behavior when a large disparity in thrust magnitude exists, in this case, $f_1 << f_2$. The combined control variables are computed by weighting thrust magnitude and, hence, f_2 exerts a larger impact on the resulting solution than f_1. This disparity is observed in the combined thrust direction closely resembling $\hat{u}_2$.

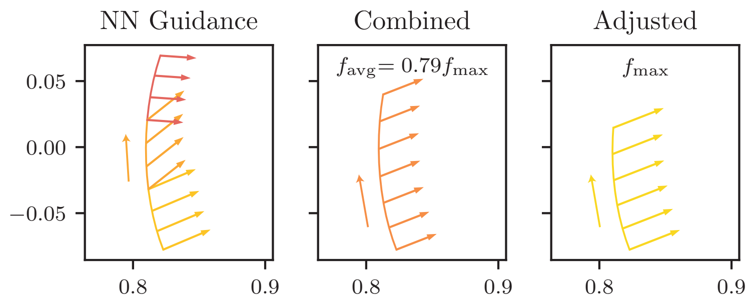
The converged results for each representative scenario are plotted in Figure 6.35. Scenario 1 and 3 remain close to the initial guess, depicted in Figures 6.35(a) and 6.35(c). In both cases, the change in thrust angle is less than 0.1°. Thrust time increases by 5.6 hours for Scenario 1, and by 2.7 minutes for Scenario 3. The utility of thrust time scaling is exemplified by the very small change in thrust time after Scenario 3 targeting. In contrast to Scenarios 1 and 3, the initial guess transformation step for Scenario 2 results in a 6-hour coasting segment due to the small combined thrust magnitude, as depicted in Figure 6.33(b). Each time a coast arc is suggested, the NNIT is simulated from the beginning with the potential to compute a future thrust or coast segment. For Scenario 2, eight coasting segments are sequentially introduced, resulting in 2 days of coasting followed by a 7-hour thrust arc, depicted in Figure 6.35(b). The parameterized thrust magnitude variable is not updated for any scenario, since $f = f_{\text{max}}$. Together, the three representative scenarios illustrate how NNIT improves the standalone NN-G&C recovery plans, and avoids control choices that violate hardware constraints.

Monte Carlo Experiments

To evaluate the proposed NNIT framework, 2000 randomly perturbed initial conditions are considered, where $3\sigma_r = 1000\,\text{km}$ and $3\sigma_v = 10\,{}^{\text{m}}\!/\!{}_{\text{s}}$ model the initial deviations. Each initial condition is simulated using NNIT to achieve solutions similar to those in Figure 6.35. In analyzing the algorithm performance, the first, and most important, factor to consider is

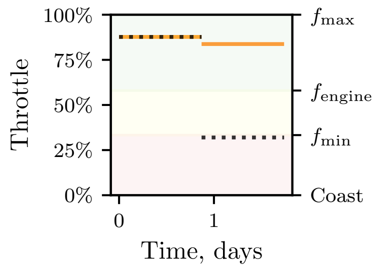


(a) Thrust magnitudes

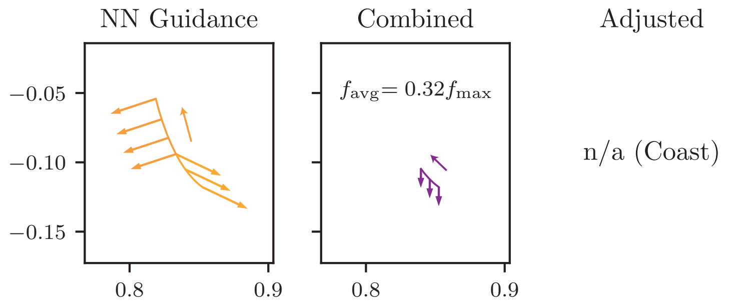


(b) NNIT Initial Guess Transformation Steps ($\hat{x}$-$\hat{y}$, nondim)

Figure 6.32. Scenario 1: Expected NN-G&C behavior. Controller recovers from deviation in three steps, ceasing iteration when control values are produced such that $f_4 < f_{\min}$. Total thrust time reduces from 62.62 $\rightarrow$ 49.71 $\rightarrow$ 39.71 hours. Solid and dashed lines indicate values for f_i and f_{avg}, respectively.

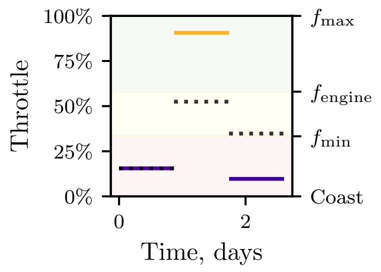


(a) Thrust magnitudes

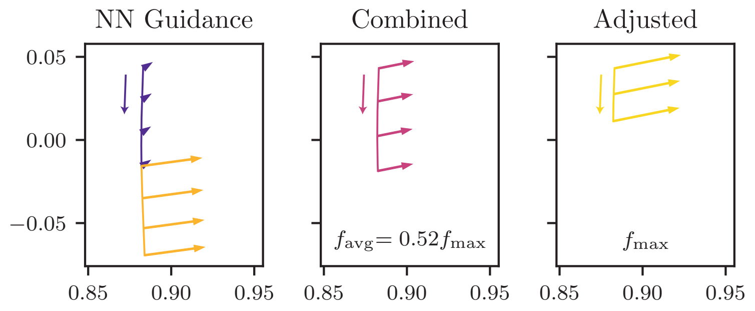


(b) NNIT Initial Guess Transformation Steps ($\hat{x}$-$\hat{y}$, nondim)

Figure 6.33. Scenario 2: Behavior when the average low thrust magnitude, f_{avg}, computed using Equation (6.19) and denoted by dashed lines, falls below the imposed lower limit, $f_{\text{avg}} < f_{\min}$, causing the NNIT initialization process to suggest a coasting segment (omitting the need for scaling to $f_{\max}$).

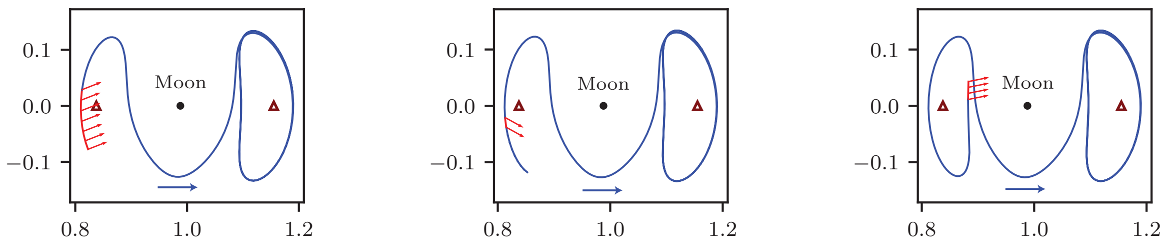


(a) Thrust magnitudes



(b) NNIT Initial Guess Transformation Steps ($\hat{x}$-$\hat{y}$, nondim)

Figure 6.34. Scenario 3: NN-G&C scenario that begins with $f_1 < f_{\min}$, but proceeds such that $f_1 > f_{\min}$. Through NNIT initial guess transformation, thrust time reduces from 41.75 $\rightarrow$ 21.9 $\rightarrow$ 11.50 hours.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 6.35. Converged solutions for the three representative cases (x-y, nondim)

convergence, visualized in Figure 6.36. While the algorithm seeks to improve a NN-generated path, cases arise where the startup solution is not sufficiently close to the converged solution, and the targeter diverges. For the simulated error level, the algorithm converges in the vast majority (98.15 %) of simulations, and divergences for the other 1.85 %. Next, the number of iterations is considered since rapid convergence is an important factor for potential onboard implementations. The targeter reaches a specified convergence threshold of 1×10^{-12} in 5 iterations or fewer in 42.3 % of cases, and in 6 or fewer in 81.1 % of trials, as depicted in Figure 6.36. The largest number of iterations reached across the 2000 trials is 15.

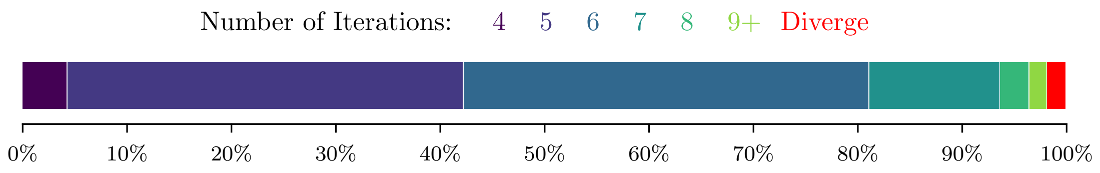


Figure 6.36. Number of iterations to convergence across 2000 simulations

Each simulation is additionally evaluated using the ‘standalone’ NN-G&C approach detailed in Section 6.2.2 and simulated in Section 6.2.3; a sample episode of the standalone NN-G&C approach is depicted in Figure 6.16. Of the 2000 simulated condition, the standalone NN-G&C solution reaches the vicinity of the target orbit in 99.4 % of case (1988/2000). The NNIT approach converges in only 2 of the 12 cases where the standalone approach fails, indicating that the implemented recovery NNIT method is not able to consistently overcome situations where the NN-G&C solution causes the spacecraft to diverge from the reference. Of the failure cases depicted in Figure 6.36, about 1 in 3 divergent cases correspond to simulations where the NN-G&C solution also fails.

Combined Solution Accuracy

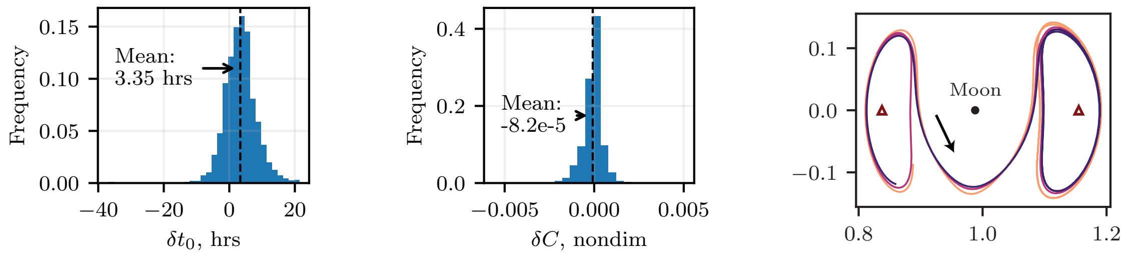
Each simulation proceeds through the proposed NNIT initial guess transformation step and, hence, the accuracy of the computed thrust time and direction is considered. Accuracy is defined as the distance to the basin of convergence. Therefore, the converged solution differing significantly from the initial guess indicates that, for the set of conditions, the thrust segment combination step may be providing a poor estimate of the resulting solution. Conversely, a small change in value indicates a suitable startup solution.

Thrust direction does not significantly change in the majority of the 2000 simulations, indicating that the relationship for combined thrust direction, Equation (6.18), provides an accurate estimation for the given scenarios. In 94.5 % of cases, the total change in thrust direction after targeting is less than two degrees, and less than four degrees in 98 % of simulations. Thrust magnitude remains unchanged at $f_{\max}$ for all targeting experiments.

Total thrusting time is the most challenging quantity to approximate because it is not directly estimated by the NN. The initial guess is computed as a weighted sum over the initial thrust segments, scaled for $f = f_{\max}$, as summarized in Equations (6.22) and (6.24). An empirical distribution for the overall change in thrust time, $\delta\mathbb{t}_0$, over the 2000 simulations is plotted in Figure 6.37(a), where the average update is 3.35 hours. Figure 6.37(a) demonstrates that targeting causes thrust time to undergo a larger transformation than thrust direction and, hence, illustrates the degree of inaccuracy in the time estimate. In this case, the positive mean update value signifies an underestimation bias in the computed approximation. This bias may be addressed via alternative estimation methods, or employing a NN architecture that directly evaluates time.

Energy Variation

The proposed NNIT transfer recovery targeting scheme involves stacking revolutions of the arrival orbit and enforcing position and velocity continuity throughout the trajectory. While stacking revolutions biases the converged solution to resemble the desired Lyapunov orbit, it does not ensure that the target energy level is achieved. To determine the suitability of this approach, the resulting error in Jacobi constant for the 2000 trials is depicted in Figure 6.37(b), where δC is the difference in the Jacobi constant value between the reference orbit and the converged coasting segment after targeting. This empirical distribution demonstrates that the targeting process does not significantly alter energy, where the average case of $\delta C = -8.16 \times 10^{-5} \approx 0$ indicates no significant introduced energy bias. Furthermore, the simulations that undergo the largest change in energy still closely resemble the reference motion, depicted in Figure 6.37(c). While there is a small discernible difference in the amplitude of the resulting Lyapunov orbit-like motion, this variation may be acceptable for a given application.

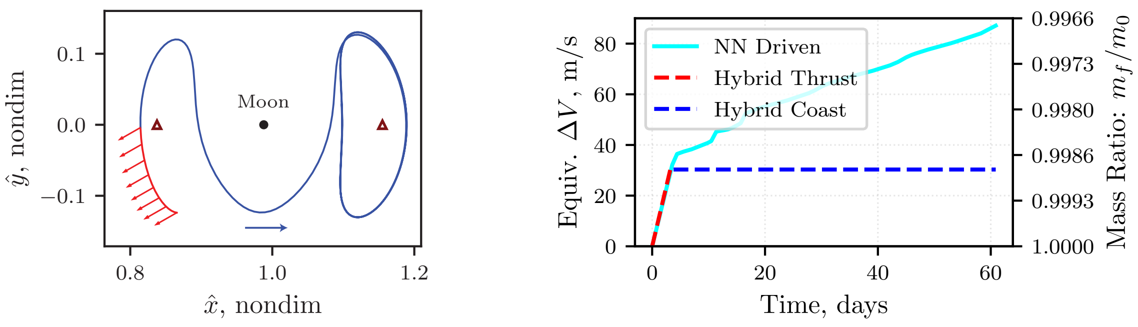


(a) Time updates (b) Energy errors (c) Min/Max δC ($\hat{x}$-$\hat{y}$)

Figure 6.37. Changes from reference motion over trials.

Mass Consumption

Recovery plans resulting from the NNIT process consume significantly less propellant than the standalone NN-G&C approach. The simulated NN is not able to determine coasting segments and, hence, produces numerous arcs at very low throttle levels. While fuel consumption across any one of these arcs remains small, the collected impact results in propellant-inefficient solutions. To illustrate the difference in mass consumption, consider the standalone NN-G&C solution depicted in Figure 6.16. This scenario serves as an initial guess for the NNIT process, producing the converged trajectory in Figure 6.38(a). While similar in configuration space, these solutions differ significantly in propellant consumption, plotted in Figure 6.38(b). The mass change in the NNIT solution closely resembles standalone NN-G&C maneuver plan for the first several days. However, these two approaches differ dramatically as the low-throttle arcs cause the standalone NN-G&C solution to continue increasing, resulting in more than double the propellant consumption of the NNIT algorithm during the 61-day simulation.



(a) Converged NNIT solution (b) Propellant consumption

Figure 6.38. Comparison of the standalone NN-G&C and NNIT approaches for the scenario depicted in Figure 6.16. Over 61 days, the NNIT recovery plan (labeled ‘Hybrid’ with colors red and blue) demands significantly less propellant than the standalone NN-G&C result (labeled ‘NN Driven’ with cyan coloring).

6.2.6 Spatial Transfer Analysis: Integrated NNIT

This sample mission application considers a trajectory recovery problem in the planar CR3BP. While planar heteroclinic transfers provide useful applications in analyzing the proposed learning framework, limitations in implementation and analysis emerge in evaluating the implemented standalone NN-G&C process. Specifically, several noted weaknesses in the maneuver plans that result from standalone NN-G&C motivate this investigation’s NNIT analysis; these drawbacks include practical constraints on the types of low-thrust control profiles that are physically possible, and inaccuracies and inefficiencies in the NN estimations. In addressing these challenges, Section 6.2.5 demonstrates encouraging promise in applying an NNIT architecture given the pre-trained NN; however, an alternate approach may yield more consistent outcomes. Thus, the integrated NNIT training algorithm detailed in the NRHO recovery application, Section 6.1, is employed to evaluate an alternative method to achieve this mission application's goal of transfer recovery. The RL framework for integrated NNIT is implemented identically to the NRHO recovery environment in Section 6.1.2, with the only differences being the intial state-generation process, and the utilization of a heteroclinic transfer for the reference trajectory in the reward function and targeting processes.

An important consideration in evaluating real-world applicability of this transfer recovery mission application is considering extendability to the spatial dynamics. While NN-G&C efficacy in spatial domains is demonstrated in the NRHO recovery application presented in Section 6.1, a natural next step is to evaluate recovery concepts that consider more challenging mission plans; this spatial heteroclinic recovery simulation extends and unifies the NRHO and planar transfer recovery tasks by evaluating an integrated NNIT training algorithm that returns the modeled spacecraft to a spatial transfer. Furthermore, the planar transfer recovery application's RL controllers are trained using PPO (an on-policy actor-critic RL algorithm). While the present investigation does not seek to identify the 'best' RL algorithm for spaceflight, a well-formulated environment is applicable to multiple RL approaches; thus, TD3 is employed for this spatial heteroclinic recovery application to demonstrate the applicability and flexibility of the proposed learning environment to alternative training schemes.

To complete the planar transfer recovery task using the standalone NN-G&C training environment, the NN must possess the capability to implement recovery thrust segments, following the reference trajectory, and indefinitely maintain the desired target orbit. In this alternative integrated NN-G&C approach, the NN is only tasked with estimating control states for the initial recovery segment; the targeting algorithm is responsible for constructing the remainder of the transfer and target orbit paths. Thus, the integrated NNIT environment allows the NN to focus its optimization process solely on the most valuable initial recovery segments which creates a more 'focused' initialization model that is not required to represent extraneous functionality.

Implementation Details

The spatial heteroclinic transfers employed in this integrated NNIT analysis is depicted in Figure 6.39. This purely ballistic transfer path, denoted 'HH1-7' by Haapala and Howell [129, Fig. 24], connects an L_1 northern halo orbit with an L_1 southern halo orbit at the same Jacobi constant value of $C = 3.07229$. In addition to evaluating NNIT suitability given a spatial transfer recovery problem, this sample application additionally seeks to evaluate the flexibility of the integrated NNIT learning environment defined in Section 6.1.2; thus, the addition of the heteroclinic transfer and an adjustment to the episode initialization process serve as the only alteration to Section 6.1.2's environment. In reality, RL applications require a certain degree of tuning for hyperparameter values, reward scaling terms, and application-specific considerations. Thus, in evaluating the applicability of the NRHO implementation without alteration, the present approach does not seek to develop the 'best' learning process for the given recovery task. Rather, the present analysis seeks to understand the extent to which an environment 'tuned' for a different application (NRHO recovery) remains immediately applicably to this spatial NNIT problem.

Given the spatial heteroclinic transfer in Figure 6.39, twelve agents are trained in parallel over 165 000 episodes. As detailed in the planar heteroclinic environment in Section 6.2.2, episodes are initialized by selecting a uniformly-distributed random state along the departure orbit and adding perturbations in position and velocity ($3\sigma_r = 1000\,\text{km}$ and $3\sigma_v = 10\,\text{m/s}$ for each coordinate individually). Following the induced deviation, the episode process and signal definitions are identical to the NRHO recovery framework detailed in Section 6.1.2. Once training is complete, the trained agents are evluated based on their convergence and propellant consumption characteristics. In simulating 2000 deterministic episodes for each of the candidate agents, eleven reach targeting convergence in at least 90 % of trials (seven are > 95 %, four are > 99 %). Average per-episode propellant consumption, measured in $\Delta\tilde{V}_{\text{equiv.}}$, ranges from 9.22 m/s to 18.22 m/s. For this mission application, the 9.22 m/s agent

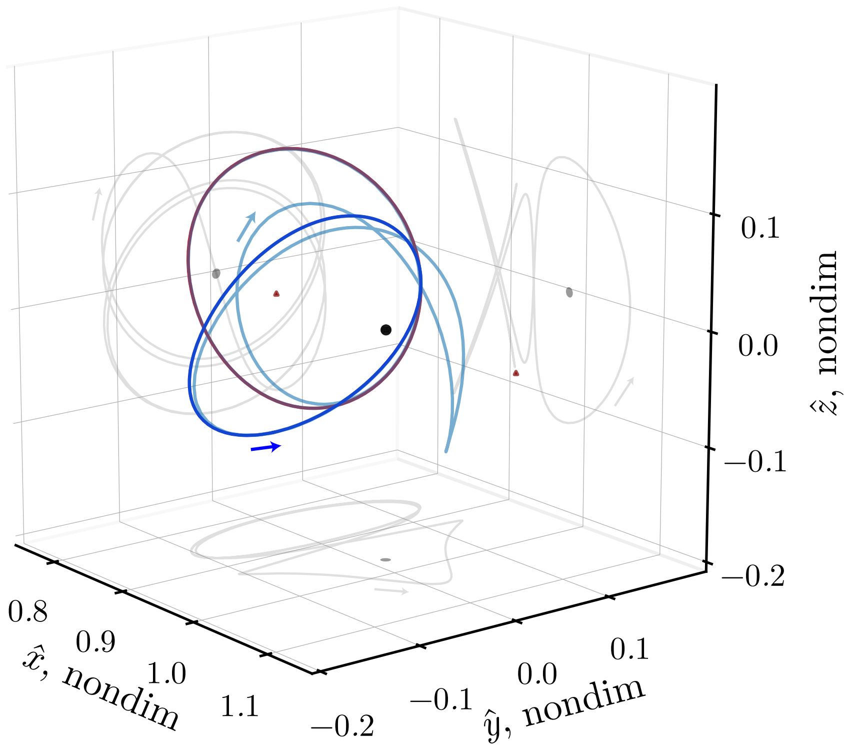
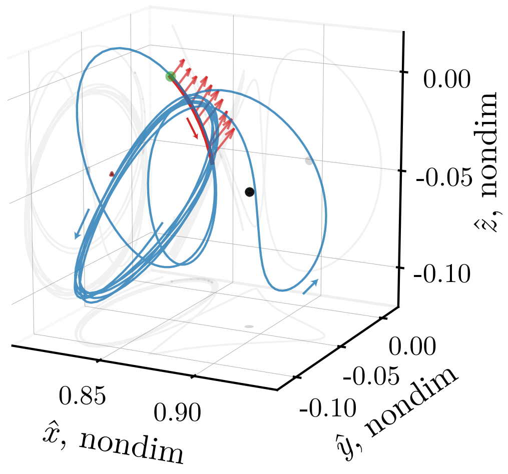


Figure 6.39. Spatial L_1 northern-to-southern heteroclinic transfer at $C = 3.07229$, identified as 'HH1-7' by Haapala and Howell [129, Fig. 24].

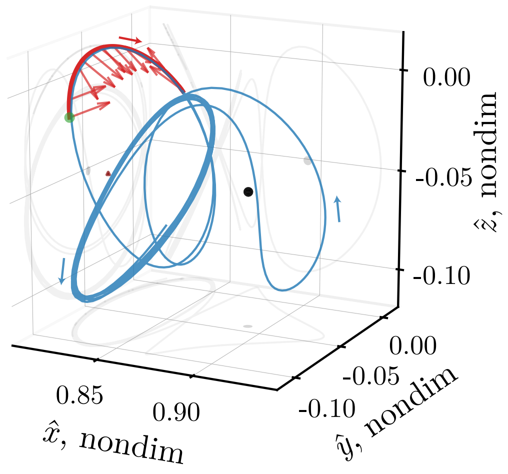
is selected due to its favorable propellant efficiency characteristics. However, this selected agent is among the least consistent in its targeting convergence benchmark, reaching the desired error threshold in only 92 % of simulations; thus, selecting this agent prioritizes fuel savings over consistency.

Simulation Results

To evaluate the performance of the selected agent, NNIT recovery plans are generated across a range of possible scenarios; two such NNIT simulations are depicted in Figure 6.40. First, Figure 6.40(a) illustrates a representative case in which a 1574 km and 11.65 m/s perturbation is applied along the L_1 northern halo orbit. With the initial state generated, the departure orbit is no longer employed in the training process, and the relative state error is computed with respect to the heteroclinic transfer (2635 km, 15.53 $^{\mathrm{m}}/_{\mathrm{s}}$). Despite this large error in both position and velocity, the NNIT agent successfully identifies a recovery path that involves 1.8 days of thrusting at maximum throttle ($\Delta\tilde{V}_{\mathrm{equiv.}} = 17.1\,^{\mathrm{m}}/_{\mathrm{s}}$). Next, Figure 6.40(b) depicts a simulation where, despite a similar relative error magnitude to Figure 6.40(a), a significant amount of additional thrusting is deemed necessary by the NNIT process, resulting in 4.6 days of continuous thrust ($\Delta\tilde{V}_{\mathrm{equiv.}} = 43.46\,^{\mathrm{m}}/_{\mathrm{s}}$). In both cases, the NNIT process rapidly and successfully identifies a control profile that enables the given spacecraft to return to its pre-planned transfer path.



(a)

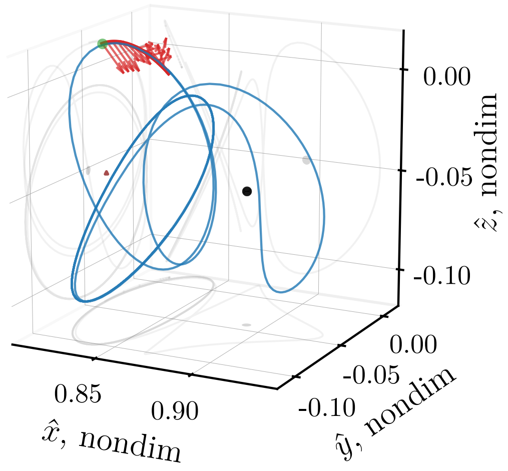


(b)

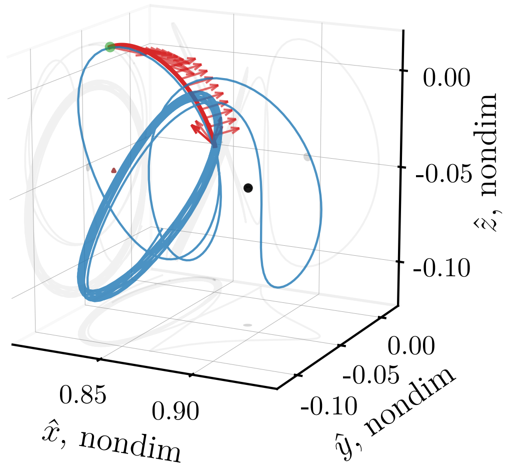
Figure 6.40. Sample successful recovery maneuver plans to the heteroclinic transfer visualized in Figure 6.39

Unlike the NRHO application, the selected agent tends to immediately initiate recovery maneuvers, as evident in comparing the sample episodes in Figures 6.6 and 6.40. While the dynamics in the NRHO region make thrusting near perilune disadvantageous, no close perilune passes exist along on the reference path in Figure 6.39 and, thus, the transfer recovery agent 'learns' that this application benefits from immediate recovery initiation. The difference between these maneuver timing approaches demonstrates the flexibility of the integrated NNIT approach in adapting its learned response based on the given problem. Moreover, the agent simulated in this heteroclinic mission application tends to operate as a 'bang-bang' controller, where throttle values are typically at maximum when thrusting, and the recovery response is then driven by the selected time and orientation values. This behavior is markedly different from the planar analysis that included no penalty on thrust magnitude and, therefore, tended to frequently suggest low-throttle values in performing its NN-G&C task, as depicted in Figure 6.16

From the twelve concurrently-trained agent evaluated in this sample mission application, the selected agent possesses a relatively low convergence percentage (at 92 % of trials). While 'failure' cases take a variety of forms, a representative scenario for a common source of failure is depicted in Figure 6.41. In this case, given an initial error from the reference transfer of 2153 km and 12.43 m/s, the NN selects the 2.1-day thrust sequence plotted in Figure 6.41(a). This NN-G&C Markov chain initializes the environment's targeting algorithm, which subsequently diverges after 5 iterations. However, if an attenuation factor is heuristically employed when applying the targeting update, the converged solution represented in Figure 6.41(b) is identified. This feasible recovery plan's 3.7 days of thrusting suggests that the favorable propellant efficiency properties that motivated the selection of the employed agent also occasionally lead to a thrust time underestimation tendency that prohibits consistent convergence.



(a) Initial Guess

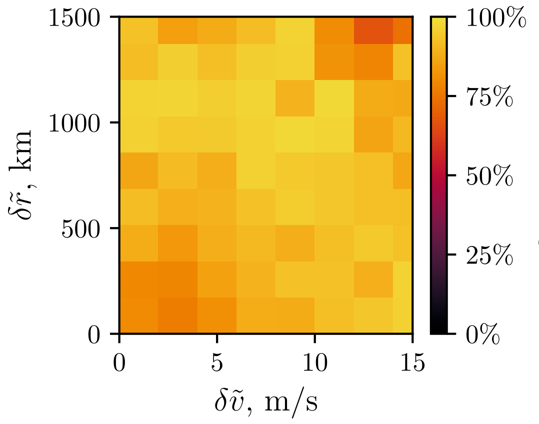


(b) Feasible Maneuver Plan

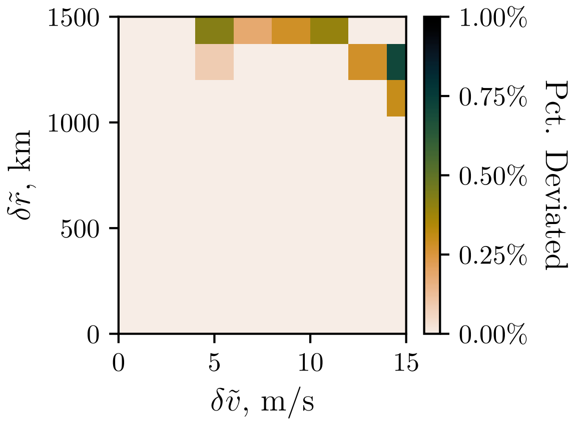
Figure 6.41. Sample unsuccessful initialized maneuver plan (a) that is successfully converged with an attenuation factor (b)

The agent's performance given a larger range of scenarios is evaluated via a Monte Carlo approach. In this analysis, the NNIT agent is tasked with recovering from 28 000 randomly-perturbed initial conditions that span various error levels; performance is recorded through convergence, deviation, and fuel consumption figures of merit. The results of this Monte Carlo analysis are depicted in Figure 6.42. As expected from the initial agent selection, the NNIT targeter does not converge in 100 % of scenarios; rather, Figure 6.42(a) demonstrates this particular controller consistently converges for initial error levels up to $\delta\tilde{r} = 1000\,\text{km}$ and $\delta\tilde{v} = 10\,{}^{\text{m}}\!/\!_{\text{s}}$ (measured from the departure orbit). Interestingly, this agent's converges worsens at the lowest errors level. This counterintuitive behavior stems from the fact that when errors are small, the low-thrust response is correspondingly minimal. When this occurs, the relative impact of the low-thrust control variables diminishes in the targeting posses given the insignificant thrust times, thus adding a challenge for the NNIT agent to overcome at smaller error levels, and potentially revealing a potential drawback of this NNIT implementation if consistent performance is required for minor deviations. However, as demonstrated in Section 6.1.3, a ballistic response is likely sufficient to initialize the targeting algorithm when errors are small. Therefore, the proposed NNIT implementation prioritizes the response characteristics at the larger error levels that render a NN initialization process more valuable to the onboard G&C system.

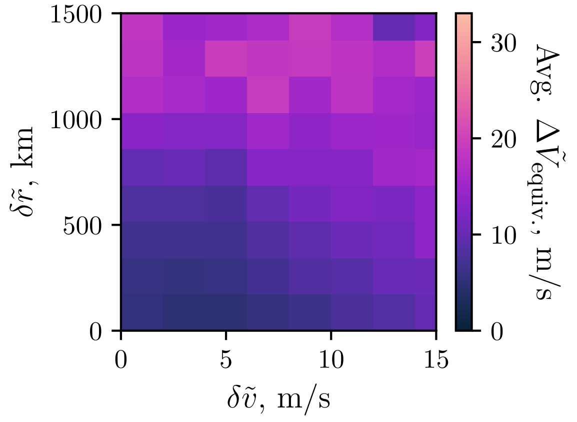
This sample recovery application's deviation threshold, identical to Equation (6.5), is seldom activated by the selected agent, as visualized in Figure 6.42(b). For errors are high as $\delta\tilde{r} = 1500\,\text{km}$ and $\delta\tilde{v} = 15\,{}^{\text{m}}\!/\!_{\text{s}}$, the deviation threshold is reached in fewer than 1 % of trials. While the convergence performance is more sporadic, as observed in Figure 6.42(a), the Monte Carlo analysis demonstrates that the selected agent is effective in 'learning' to avoid departing the vicinity of the reference motion. Furthermore, average propellant consumption across the simulated trials is represented in Figure 6.42(c). As expected, fuel usage tends to increase alongside the magnitude of the induced perturbations; the unexpected fuel consumption decrease as errors become largest is explained by the corresponding drop in convergence percentage in Figure 6.42(a). As demonstrated in Figure 6.41, this NNIT agent tends to underestimate thrust time variables to achieve a lower overall $\Delta\tilde{V}_{\text{equiv.}}$ value, which carries the unintended side effect of occasionally causing the targeting process to diverge.



(a) Convergence



(b) Deviation



(c) Propellant Consumption

Figure 6.42. Convergence, deviation, and propellant consumption metrics collected across 28 000 Monte Carlo simulations.

A high degree of algorithm and training flexibility is demonstrated through the simulated agent's performance in this spatial transfer recovery application. Despite the relatively small number of trained agents (12), and without any additional evaluation of hyperparameter options or problem-specific tuning variables from the NRHO recovery analysis in Section 6.1.2, multiple effective NNIT agents readily emerge. Given additional analysis and an evaluation of the tuning options, it is certainly possible to train agents that possess higher degrees of performance. The present simulation demonstrates the remarkable adaptability of the proposed integrated NNIT learning algorithm in its applicability to this new problem with its differing dynamical characteristics, thus suggesting future applicability of the proposed RL process.

6.3 Multi-body Orbit Stationkeeping Scenario

The advent of low-thrust propulsion systems offers exciting new prospects for space missions, allowing for increased maneuverability, reduced fuel consumption, and longer mission lifetimes. However, the complexity of these systems presents a challenge for traditional computational control methods, which often rely on intricate domain-specific knowledge and predetermined orbit-specific characteristics. The challenge of establishing efficient onboard G&C processes that will enable future autonomous low-thrust spaceflight is exacerbated by the limited computational resources available on flight computers. In particular, Stationkeeping (SK) presents unique challenges for algorithms that must satisfy the often competing objectives on accuracy, computational footprint, and propellant consumption within complex dynamical regimes. Furthermore, determining maneuver locations and frequencies is often accomplished heuristically, with relatively few analytical or numerical tools available to aid in the process. Through this sample SK mission application, RL is leveraged to address both the timing and the control components for low-thrust SK problems, producing NN controllers that successfully maintain cislunar periodic orbits without a priori knowledge of effective maneuver placement schemes. Practical examples demonstrate a NN's ability to 'learn' a suitable maneuver schedule for both NN and crossing control schemes alike, resulting in effective maneuver patterns that balance coasting time against propellant consumption, while nevertheless ensuring mission success.

The previous two sample mission scenarios, in Sections 6.1 and 6.2, demonstrate NN-G&C approaches that are capable of maintaining a given periodic orbit. However, simulations of these cases indicate that the learned G&C policy are significantly more useful at larger error levels due to their excessive propellant use in responding to scenarios where perturbations are small. Hence, this sample mission application seeks to evaluate the efficacy of the proposed RL-based NN-G&C process in seeking a more sparse thrusting strategy that prevents excessive fuel use and avoids the constant interruption to science objectives evident in, for example, the NN-G&C simulation depicted in Figure 6.16. Furthermore, both prior mission applications leverage relative state minimization terms in designing their reward function. However, many realistic SK problems do not require close adherence to a specific reference trajectory to satisfy mission objectives. In particular, SK along a libration point orbit without minimizing the relative state is proven successful on previous missions such as ARTEMIS [6], ACE, SOHO, and WIND [135]. Thus, this application seeks to develop an NN-G&C method, and RL training approach, capable of supporting these varying SK objectives.

Stationkeeping, at a high-level, is comprised of two distinct tasks: control and timing. Depending on mission objectives, previous research efforts suggest a wide variety of orbit maintenance control approaches, including recent investigations that leverage RL to train a NN controller [60], [61], [119]. Once a control approach is selected, an additional challenge is determining suitable locations along the orbit to implement the maneuvers. Too few maneuvers may result in deviation from the reference orbit and higher SK costs, while too many maneuvers adds operational complexity, and may interrupt science objectives. As applied in this SK mission application, RL is demonstrated as an effective tool for separately determining both the timing and the control states for low-thrust SK maneuvers along complex multi-body orbits. This approach seeks to improve NN-G&C performance in responding to small disturbances and to demonstrate a novel method for encoding the timing and the location of future maneuvers as learnable NN parameters, reducing the need for a priori knowledge of maneuver frequency and placement by allowing the NN to 'learn' a suitable maneuver schedule. Separating the planning and control tasks renders the maneuver placement approach applicable to a broad range of control methods (NN-G&C or otherwise) and enables the 'timing' NN to uncover dynamical regions in the space that lead to low-cost SK solutions.

For the control task, many current SK techniques in multi-body dynamical regimes lack global applicability, instead relying on specific characteristics of the underlying problem to construct propellant-efficient maneuvers. For example, Floquet mode control approaches leverage stability information to compute maneuvers that offset the orbit's unstable modes [6]. Alternatively, crossing control methods target certain conditions at later crossings of an orbit's plane of symmetry [118]. While powerful and effective, these approaches are often application-specific, and not easily generalized to a wider range of mission scenarios and spacecraft. Alternatively, modern RL techniques offer general approaches to train a NN controller without direct knowledge of the dynamical environment. Neural networks offer appealing benefits in enabling computationally efficient G&C with potentially suitability for onboard use. When trained via RL, the separation between the agent and the environment (visualized in Figure 4.3) provides significant flexibility for spacecraft G&C by removing the reliance on underlying assumptions and characteristics of the dynamical model and mission scenario. Furthermore, the separation between training and closed-loop evaluation produces a computationally efficient controller with attractive onboard characteristics, requiring neither iteration nor numerical integration to operate effectively.

Multiple previous investigations demonstrate the feasibility of RL as an encouraging novel approach for closed-loop guidance near multi-body libration points. In particular, several authors employ RL to compute SK maneuvers. Guzzetti first leverages $\mathcal{Q}$-Learning with a discretized state and action space for orbital maintenance of an L_1-Lyapunov orbit in the Earth-Moon system [60]. Alternatively, several authors employ recently developed RL techniques to access and manipulate continuous dynamical environments. Applied to Sun-Earth halo orbits, Molnar applies a Soft Actor Critic (SAC) [101] approach to augment Floquet mode control applied to SK maneuvers [61], and Bonasera et al. employ Proximal Policy Optimization (PPO) [103] to compute corrective maneuvers that offset momentum unloads in both a Sun-Earth gravitational system and a higher fidelity ephemeris force model [119].

While each previous RL-enabled SK strategy demonstrates promising results, the training processes often incorporate domain-specific knowledge, limiting their applicability in problems that lack specific characteristics. This application builds on prior research by removing domain-specific knowledge from the SK training process, creating a flexible approach that is more easily applied to new mission scenarios. In particular, previous RL training methods rely on application-specific plane crossing geometries [60], [119] or stability [61] information. The current work introduces a generalization of the training approach to avoid orbit-specific characteristics, producing an RL training algorithm that is applicable to a wider variety of problems, and successfully maintains challenging orbits without relying on relative state error minimization in the training process. For many mission applications, such error minimization is not necessary, and extra propellant is expended to correct what may be an acceptable amount of error.

6.3.1 Candidate Cislunar Orbits

Missions to cislunar space are becoming increasingly common. Many current and upcoming spacecraft plan to leverage periodic orbits in the vicinity of the L_1 and L_2 Earth-Moon CR3BP Lagrange points as baseline trajectories. These missions range from small CubeSats, such as Lunar IceCube [76] and CAPSTONE [136], to the flagship Artemis and Gateway programs [126]. In general, cislunar orbits are computed in the CR3BP during preliminary investigations, and subsequently transferred to a higher-fidelity ephemeris force model. To evaluate preliminary results, the CR3BP serves as the basis for this SK mission application.

Three candidate destination orbits in the lunar vicinity are employed in this mission application to evaluate the proposed NN-G&C SK framework. Orbits in the CR3BP are commonly characterized by stability properties; this sample mission application leverages a 'stability index' ν, defined Equation (2.28), to characterize relative differences in stability between the candidate orbits. The three orbits, visualized in Figure 6.43, are summarized as:

1. The 9:2 synodic resonant southern L_2 NRHO (planned baseline for the Lunar Gateway [126]), plotted in red. This orbit is also visualized within the L_2 southern halo orbit family in Figure 6.1.
2. A distinctly unstable member of the L_2 southern halo orbit family, plotted in blue.
3. An L_2 southern butterfly orbit at nearly 2:1 resonance, similar to the butterfly orbit in Davis et al. that possesses a larger perilune radius [137], plotted in green.

Specific values relevant to each orbit are listed in Table 6.11. The stability properties for each orbit are particularly important for SK applications. The stability index, defined in Equation (2.28), across the L_2 southern halo and butterfly families are plotted in Figures 6.44(a) and 6.44(b), respectively, where specific colors highlight the stability indices of three sample destination orbits in their respective families. However, it is important to exercise cuation when comparing stability index values across orbit families, as this metric neglects to incorporate the period associated with the underlying orbit. Consequently, larger orbital periods result in inflated stability index values that do not accurately represent the sensitivity inherent to the underlying dynamical system.

6.3.2 Stationkeeping Overview (Axis Control)

Numerous strategies exist for SK leveraging dynamical structures in the vicinity of libration points. In many cases, specific characteristics of the underlying periodic orbit are exploited to compute low-cost maneuvers, limiting the applicability of any one strategy to a wide theater of operations. For example, *Floquet mode* control exploits stability information for unstable orbits to compute corrective maneuvers that cancel unstable modes [138], [139]. While effective on distinctly unstable orbits, Floquet mode approaches assume accurate computation of the unstable directions: a condition that is frequently unavailable in cislunar space. Numerous alternative strategies are demonstrated for libration point orbits, including classical control theory, Hamiltonian structure preserving control, and continuation strategies [118]. Summaries of previous contributions, available SK techniques, and mission applications in multi-body regimes are compiled by Shirobokov et al. [120] and Muralidharan [140].

One particular SK approach, crossing control, has emerged as an especially powerful strategy for maintaining symmetric orbits in multi-body regimes [118], [141], with particular recent interest in SK for Earth-Moon NRHOs [2]. Many structures in the vicinity of the collinear libration points, including the L_1, L_2, and L_3 halo families, are symmetric across the rotating xz-plane in the CR3BP. The x-axis control (XAC) algorithm leverages this symmetry by employing a differential corrections technique to target a spacecraft's x-velocity at near-perpendicular crossings of the plane of symmetry at locations downstream. This approach yields remarkably low-cost maneuvers, is effective in an $\mathcal{N}$-body ephemeris force model, and is extendable to low-thrust propulsion options when maneuvers are small [7].

Table 6.11. Orbit characteristics

	Period, days	C, nondim	Stability ix., ν	Perilune radius, km
9:2 NRHO	6.56	3.046 767	1.32	3210
Halo Orbit	13.59	3.059 435	71.02	36 000
Butterfly Orbit	13.64	3.080 410	10.03	9000

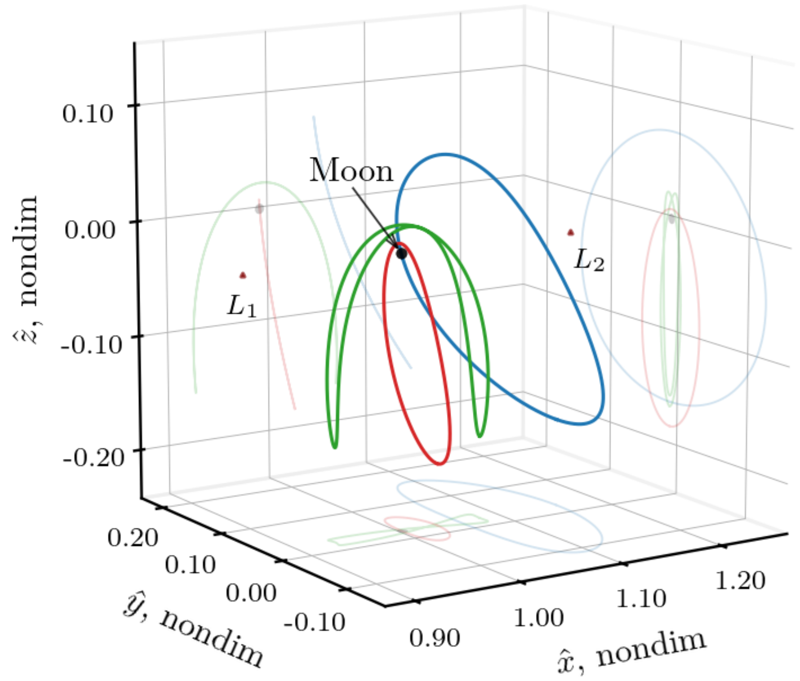
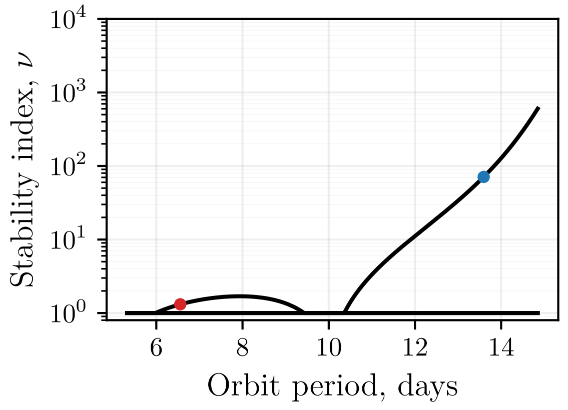
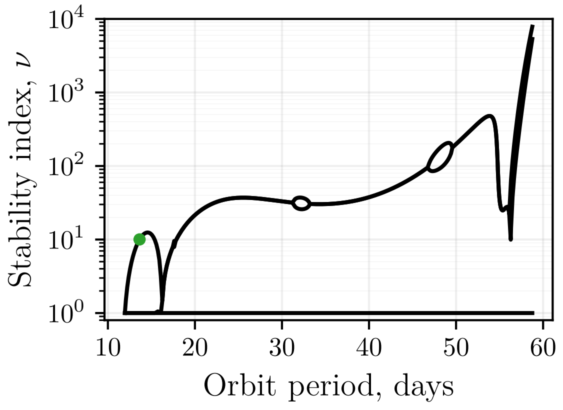


Figure 6.43. Sample destination orbits employed in this investigation.



(a) L_2 Southern Halo Orbit Family



(b) L_2 Southern Butterfly Orbit Family

Figure 6.44. Stability indices along the Earth-Moon L_2 southern orbit families employed in this investigation, plotted on a logarithmic scale.

To evaluate the proposed RL schemes, a simple XAC strategy is implemented. In a full $\mathcal{N}$-body ephemeris model, XAC is comprised of: 1) 'long horizon maneuvers' to compute a long-term virtual baseline solution and 2) 'short horizon maneuvers' to target near-term crossing conditions along the virtual baseline trajectory [118]. This research effort is a preliminary investigation into the proposed RL approaches and the numerical studies are, therefore, limited to the CR3BP. In this context, the periodic orbit itself serves as a long-term baseline, and only short-horizon maneuvers are necessary for the XAC algorithm to maintain the orbit. The algorithm itself is detailed by Guzzetti et al. [118], and the resulting impulsive maneuvers are transformed into low-thrust arcs using the ideal rocket equation, as suggested by Newman et al. [7].

Maneuver Placement Strategies

Maneuver locations and frequency depend on many factors including, but not limited to, operational constraints, spacecraft hardware, science objectives, navigation and tracking constraints, dynamical sensitivity, impacts to attitude control, and fuel availability [135]. Many applications rely on prior experience, heuristics, and trade studies to determine maneuver placements. Stationkeeping is necessary to maintain periodic orbits about collinear libration points and, therefore, past mission experience provides a useful background for determining maneuver schedules. In Sun-Earth multi-body applications, the ACE mission places maneuvers at one z-axis extremum every 6 months, while WIND and SOHO implement maneuvers at an approximate 90-day cadence [135]. The Genesis mission similarly distributed maneuvers evenly in time, with 2–4 maneuvers implemented per orbital period [142]. In these missions, maximizing the time between maneuvers is particularly important to avoid interruptions to science objectives. Within the Earth-Moon system, the ARTEMIS mission implemented maneuvers at xz-plane crossings and y-max amplitudes [143].

In applications involving the upcoming Gateway and Artemis programs, maneuvers along NRHOs are planned near apolune to avoid the dynamically sensitive region near the Moon [7]. For NRHOs, assuming one maneuver per revolution, ΔV placement is determined via Monte Carlo simulations across a range of orbits and maneuver locations [118]. While an exhaustive search of all ΔV locations is effective in this simple case, this approaches relies on discretization and expensive Monte Carlo simulations, becoming quickly intractable for complex mission scenarios and maneuver patterns. Furthermore, the circumstances for different missions and spacecraft influence maneuver planning and, hence, no one pattern of maneuvers is effective for all types of controllers.

An alternative newer methodology for maneuver placement, proposed by Muralidharan and Howell, is to determine SK locations by observing the angle between maneuvers and downstream stretching directions, computed via singular value decomposition of the STM [111]. This approach is demonstrated as effective for as many as three maneuver locations per orbit in the L_1 NRHO region, with potential applicability for additional maneuvers.

In RL efforts, previous applications place SK maneuvers either at axis crossings [60], or at regular intervals that are based on previous research efforts [61] or correspond to particular mission characteristics such as the frequency of momentum unload cycles [119]. In other NN-G&C applications, including the mission scenario detailed in Section 6.2, low-thrust spacecraft are assumed to be constantly thrusting, with control parameters adjusted at regular intervals [54]. Implementing maneuvers at a fixed cadence is well-suited to RL, but poses several challenges. First, this approach assumes the maneuver frequency is known a priori and, second, the resulting controller may be unable to accommodate shifts in the schedule due to one of many operational constraints.

6.3.3 Control Environment Implementation

The control environment seeks to train an agent capable of producing SK maneuvers in the presence of state uncertainty from anywhere along an orbit. This process is formalized through three signals that facilitate the agent-environment communications process (observation, action, and reward), and the transition dynamics that defines the process by which the environment steps from one state to the next. An overarching goal of the SK environment implementation is to create a flexible training process that avoids domain-specific knowledge and is potentially applicable to numerous SK problems.

Observation Signal

The observation signal communicates information about the agent's location relative to its desired orbit path. The general form of the observation signal in Equation (4.15) is augmented to include location information regarding the nearest reference state. Similar to the NRHO recovery mission application, Equation (6.3), location of $\boldsymbol{\rho}_{\text{ref}}$ along the underlying periodic orbit is represented by the trig encoding of the angle coordinate ξ, as defined in Section 4.3.1. The complete observation vector for the control task is computed as,

$$\boldsymbol{o}_{\text{t}} = \begin{bmatrix} \boldsymbol{q}_t & \boldsymbol{\delta\rho} & \sin(\xi) & \cos(\xi) \end{bmatrix} \in \mathbb{R}^{15} \tag{6.26}$$

Unlike the heteroclinic transfer mission application in Section 6.2, spatial orbits are leveraged in this simulation and, hence, the spatial vector quantities $\boldsymbol{q}_t$ and $\boldsymbol{\delta\rho}$ are employed.

Action Signal

The control agent commands low-thrust thrust direction, magnitude, and time values. Hence, the action vector is defined as,

$$\boldsymbol{a}_{\text{t}} = \begin{bmatrix} f^* & u_x^* & u_y^* & u_z^* & t^* \end{bmatrix} \in \mathbb{R}^{5} \tag{6.27}$$

This action signal is identical to the NRHO recovery mission application, Equation (6.4). However, unlike the recovery simulation, thrust arcs are separated by coasting segments in the environment; hence, a smaller maximum time value $\mathbb{t}_{\text{max}}$, defined in Equation (4.18), is required in this application. Furthermore, simulations of the NRHO recovery problem demonstrate poor propellant-efficiency for lower levels of error, as depicted in Figure 6.3, which further motivates lowering the value of $\mathbb{t}_{\text{max}}$ to increase performance in SK problems. This mission application leverages $\mathbb{t}_{\text{max}} \in [1, 3]$ hours (compared to 24 hours in the NRHO mission recovery problem).

Reward Signal

While the reward choice for the transfer phase, defined in Equation (6.13), encourages the agent to maintain a very close proximity to the reference, such strict adherence to the reference motion is often unnecessary in practical applications, and causes excessive propellant expenditure in correcting permissible levels of error [6]. Rather than minimizing the distance with respect to the underlying orbit, the RL process for SK formulates success as simply "not deviating" from the reference, with a small additional penalty added to encourage propellant-efficient maneuvers. Similar to the previous two mission applications, deviation is formalized by enforcing maximum bounds on position and velocity deviation, and detecting simulations that impact the Moon,

$$\text{Deviation from orbit} := \begin{cases} \tilde{r}_{23} < 1737.4\,\text{km} & \text{or} \\ |\boldsymbol{\delta\tilde{r}}| > 10\,000\,\text{km} & \text{or} \\ |\boldsymbol{\delta\tilde{v}}| > 250\,{}^{\text{m}}\!/\!_{\text{s}} \end{cases} \tag{6.28}$$

Since the expected return (defined in Equation (4.5)) is formulated as the sum of future discounted reward, a deviation penalty, applied at any point, impacts the entire trajectory of preceding actions. This formulation encourages actions that avoid future deviation, thus, encouraging SK without minimizing relative state errors. Stationkeeping is expected to proceed indefinitely and, hence, possesses no terminal condition. Therefore, the "Final condition reached" portion of the reward function, as defined in Equation (4.20), is not incorporated for the SK application. The SK control reward function at time step $\mathfrak{t}$ is defined in this mission application as,

$$r_{\mathfrak{t}} = \begin{cases} c + \beta_{\Delta\tilde{V}}\Gamma_{\Delta\tilde{V}} & \text{not deviated from reference orbit} \\ p & \text{deviation criteria met} \end{cases} \tag{6.29}$$

where $c \in \mathbb{R}_{\geq 0}$ is a constant reward for not deviating at each state, $\beta_{\Delta\tilde{V}} \in \mathbb{R}_{\leq 0}$ is a scaling factor to determine the extent to which propellant use is penalized, and $\Gamma_{\Delta\tilde{V}} = \Delta\tilde{V}_{\text{equiv.}}$ is defined in Table 4.2.

Error Models

For preliminary analysis, two sources of error are included in this SK mission application: orbit insertion and orbit determination. For both the control and timing tasks, episodes begin with a spacecraft inserting into a periodic orbit with some degree of uncertainty. This uncertainty is modeled by constructing $\boldsymbol{\delta\rho}$ measure values from zero-mean normal distributions, where 3σ error bounds are defined as 10 km and 10 $^{\mathrm{cm}}/_{\mathrm{s}}$ for position and velocity values, respectively. Following insertion, 3σ orbit determination errors of 1 km and 10 $^{\mathrm{cm}}/_{\mathrm{s}}$ are implemented each time the environment communicates an observation signal to the agent. While maneuver execution errors are not explicitly implemented during training of the control agent, training actions are computed stochastically and, therefore, resilience to maneuver execution errors occurs naturally via the RL training process. These error models evaluate the RL agent's ability to perform its task in the presence of uncertainty. However, the employed models are lower-fidelity and do not consider measurement correlations or potential filtering options.

Episode Details

The agent is trained over either 110 000 episodes or 1.5 million interactions with the environment, whichever occurs first. Stationkeeping control episodes proceeds as follows:

1. Initial location along the orbit is selected from a uniform distribution where $t_{\mathrm{po}} \sim \mathcal{U}(0, \mathbb{P})$, with an additional perturbation introduced to model an insertion error.
2. A stochastic control action, defined in Equation (6.27), is computed by the agent, and simulated in the environment.
3. A ballistic coasting arc is introduced, where propagation time is selected from a uniform distribution where $t \sim \mathcal{U}(t_{\min}, t_{\max})$. Varying propagation time exposes the agent to many locations along the orbit, and discourages policies that rely on one particular maneuver frequency. After propagation is complete, an orbit determination error is added to the final state estimate included in the observation signal, Equation (6.26).

4. Steps 2 and 3 alternate until either the agent reaches a pre-determined maximum number of steps, or the deviation criteria (Equation (6.28)) is reached.

Once training is complete, the actor NN is saved, and may then be leveraged as a deterministic controller. Due to the stochasticity in NN initialization, it may be necessary that the training process is run several times before a suitable agent is produced.

6.3.4 Timing Environment Implementation

Given any SK control strategy, the timing environment trains an agent to estimate the next maneuver location along the reference orbit. The given controller may be an RL-trained NN or a traditional control strategy. Sample mission applications in this investigation demonstrate timing networks for both NN-G&C and XAC approaches. The timing agent is rewarded for maximizing the time between maneuvers without escaping from the vicinity of the orbit. The observation, action, and reward signals again formulate the learning process:

Observation Signal

The observation signal for the timing environment is identical to the control environment, detailed in Section 6.3.3. The observation vector is provided in Equation (6.26).

Action Signal

The timing agent estimates the location of the next maneuver by leveraging the timing command process detailed in Section 4.3.3. The action vector is defined in Equation (4.19).

Reward Signal

The reward signal for the timing environment is similar to the reward in the control environment, detailed in Section 6.3.3 and defined in Equation (6.29). For the timing environment, an additional term $\beta_t \Gamma_t$ is added to Equation (6.29) to encourage policies that maximize the time between maneuvers t,

$$r_{\mathrm{t}} = \begin{cases} c + \beta_{\Delta\tilde{V}} \Gamma_{\Delta\tilde{V}} + \beta_t \Gamma_t & \text{not deviated from reference orbit} \\ p & \text{deviation criteria met} \end{cases} \tag{6.30}$$

where $\beta_t \in \mathbb{R}_{\geq 0}$ and $\Gamma_t = t$, as listed in Table 4.2.

Episode Details

The timing agent is again trained over either 110 000 episodes or 1.5 million interactions with the environment, whichever occurs first. Each timing environment episode proceeds as follows:

1. Initial location along the orbit is selected from a uniform distribution where $t_{\mathrm{po}} \sim \mathcal{U}(0, \mathbb{P})$, with an additional perturbation introduced to model an insertion error.
2. The specified controller computes control states given the current navigation state.
3. The timing agent selects the next maneuver location.
4. The time between maneuvers is calculated, such that $t \in [t_{\min}, t_{\min} + \mathbb{P}]$.
5. Propagate t and evaluate the deviation criteria in Equation (6.28). If the spacecraft is not deviated, and the agent has not yet reached the pre-determined maximum number of steps, orbit determination errors are added to the observation, and steps 2–5 are repeated.

Upon training completion, the actor NN is saved, and may then be directly leveraged as a deterministic function to compute maneuver locations. Alternatively, experiments demonstrate that agents tend to converge on repeating location sequences that may be represented as simple patterns. These patterns are identified from NN simulation, and may be directly employed to avoid NN evaluation altogether. As with the control environment, stochasticity in NN initialization often leads to suboptimal convergence and, hence, the algorithm may need several simulations before a suitable policy is uncovered.

6.3.5 Simulation Results

The proposed training processes for control and timing tasks are evaluated along the three candidate cislunar destination orbits introduced in Section 6.3.1 and plotted in Figure 6.43. These orbits include the 9:2 synodic resonant L_2 southern NRHO (the planned baseline orbit for the upcoming Lunar Gateway), a more unstable member of the L_2 southern halo family, and an L_2 southern butterfly orbit at nearly 2:1 resonance. Characteristics of each orbit are listed in Table 6.11, where the stability index is defined in Equation (2.28) and visualized in Figure 6.44. The variation in orbital period, energy, stability, and geometry between these three orbits provides a useful basis for evaluating the proposed RL approaches.

Controller Training

Neural network controllers are trained using TD3 along each of the sample destination orbits. In each case, to overcome the stochasticity in the NN initialization, and to explore different reward coefficients, many agents are trained in parallel. Once training is complete, a specific controller is selected based on simulation results and desired behavior. Recall that, during training, the next maneuver location is selected from a uniform random distribution between user-specified lower and upper bounds. During training, actions are stochastic, with small perturbations introduced to better explore the action space. After training is complete, performance statistics are gathered on the resulting deterministic controllers. In particular, deviation frequency and propellant consumption are key selection metrics. Then, rather than computing locations randomly, the resulting controller is employed in the timing training process to determine an effective maneuver pattern strategy for the particular controller. For this analysis, a single controller is selected for each mission scenario, with Monte Carlo results listed in Table 6.12 under the “Training” pattern for each RL controller. The results are summarized as follows,

- **9:2 NRHO**: An inconsistent controller is intentionally selected to evaluate the training process for the timing network. This controller produces relatively propellant-efficient maneuvers, but struggles to maintain the NRHO when maneuvers are located near the Moon. This behavior is consistent with previous research efforts that demonstrate maneuvers near perilune to be ineffective for NRHO SK [2], [118]. With 1–3 randomly placed maneuvers per revolution, this controller maintains the vicinity of the 9:2 NRHO for 25 revolutions in 79.5 % of scenarios.

- **Halo orbit**: A controller is selected that consistently maintains the given halo orbit. In the Monte Carlo simulation with 5–15 randomly-placed maneuvers per revolution, this agent successfully avoids escaping from the reference for 25 revolutions in 100 % of trials.
- **Butterfly orbit**: The selected agent consistently maintains the given butterfly orbit, although 1.5 % of Monte Carlo trials did escape when maneuvers were randomly placed. This controller produced lower-cost maneuvers than most other trained agents, and is selected for its balance between successfully maintaining the orbit and lower annual SK cost.

While propellant-efficiency is an important metric in the control task, it is important to note that RL is not an optimal control method and propellant minimization is not expected in the resulting agents. While NNs are extremely effective at function approximation, the control function is uncovered without any a priori knowledge of desired behavior, and the estimation is never perfectly accurate. Furthermore, controllers in this investigation are expected to compute SK maneuvers from any location along their respective orbit – a condition that further complicates a particular controller's ability to consistently minimize overall fuel costs. While including domain-specific knowledge in the training process often improves performance in specific applications, it limits applicability to other problems and is, therefore, not included in this investigation. In this application, propellant consumption is considered as one of many metrics when weighing the benefits of NN control, and is not the only factor in selecting a particular controller from a batch of trained agents.

Maneuver Placements and Timing

The timing environment is leveraged to train a controller to select the location for the next maneuver along an orbit, where the maneuver location decoding process is specified in

Equation (4.19). Again, many agents are trained in parallel, and desirable agents are selected based on several factors, including deviation frequency, number of maneuvers per revolution, and overall propellant consumption. Furthermore, empirical results demonstrate that timing agents tend to converge on specific, repeating, patterns for placing maneuvers. Therefore, the NN-suggested maneuver sequences may be extracted from simulations and independently analyzed. In this investigation, rather than evaluating the timing network directly in simulations, the computed patterns are employed directly. Initial analysis demonstrates that replacing the NN with its converged pattern does not degrade overall SK performance.

To illustrate the pattern identification process, a representative simulation of a converged timing agent along the 9:2 NRHO is plotted in Figure 6.45. The episode begins with insertion and, thus, a maneuver is automatically implemented at the initial state. As time progresses in this example, maneuvers fall into a repeating two-maneuver pattern, where colors corresponding to ξ may be correlated to locations along the 9:2 NRHO via Figure 4.6(a), and the resulting pattern is visualized in Figure 6.46(a). Once identified, the repeating two-maneuver sequence, located by ξ, is directly employed in simulations.

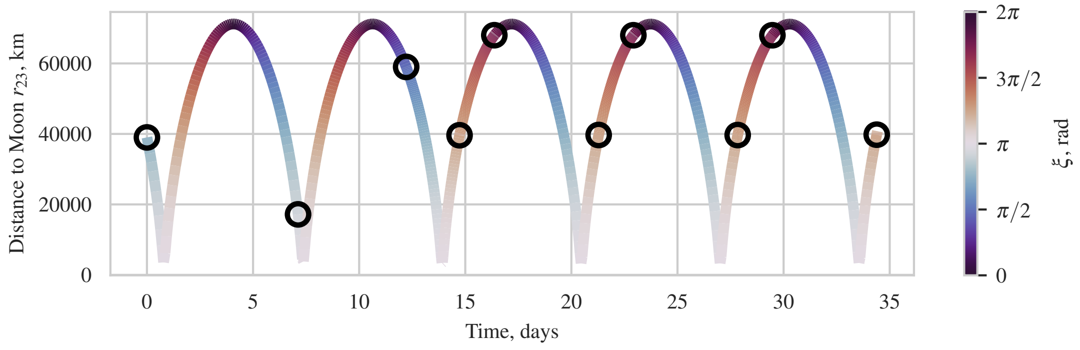


Figure 6.45. Distance to the moon over time for a simulation along the 9:2 NRHO, where black circles indicate locations of maneuvers, and color signifies locations along the NRHO, as depicted in Figure 4.6(a).

9:2 NRHO Patterns

The timing agent is tasked with determining effective maneuver locations along the 9:2 NRHO leveraging a frequently inaccurate NN controller. Given random locations, this controller escapes from the reference orbit in more than 20 % of simulations. Several timing agents are trained in parallel, and two patterns are extracted from two converged results, plotted in Figures 6.46(a) and 6.46(b). These RL-generated patterns are then compared with an apolune control strategy known to be effective for crossing control techniques [118], depicted in Figure 6.46(c). Without a priori knowledge of this behavior, pattern (b) independently uncovers a similar apolune-placement strategy, demonstrating the timing environment's effectiveness in identifying suitable regions of space for maneuver placement. Furthermore, given larger error levels, a similar preference for placing thrust segments near apolune is uncovered in this investigation's NRHO recovery mission application, detailed Section 6.1.

A Monte Carlo analysis is implemented to evaluate the efficacy of the control strategy given the three specified maneuver patterns, and results are summarized in the "9:2 NRHO" section of Table 6.12 for Figures 6.46(a) to 6.46(c). While the agent frequently diverges given random locations, as implemented in the control training process, the SK process is significantly stabilized with the identified patterns, successfully maintaining the orbit in more than 97.5 % and 95 % of simulations for patterns (a) and (b), respectively In both cases, the mean annual SK cost is approximately 2.3 m/s. This analysis is not intended to represent a practical mission scenario, as a 5 % deviation is likely an unacceptable level of error. Instead, this example illustrates the ability of the timing network training process to identify regions of space where a control strategy is particularly ineffective, and place maneuvers elsewhere. Furthermore, the performance difference between the RL-generated one-maneuver pattern (b), and the known pattern (c), illustrates the balance imposed during training between deviation and propellant consumption, where the RL-generated pattern leads to lower-cost solutions that deviate more frequently.

Butterfly Orbit Patterns

The butterfly orbit, depicted in Figure 4.6(c), involves two passes of the Moon during each orbital period and, hence, presents a challenge to the maneuver location selection process. As detailed in Equation (4.10), location along the orbit is represented by the angle

ξ and, in contrast to halo orbits, the two values $\xi = \{0, \pi\}$ represent separate apolunes. This scenario tests the timing agent's ability to 'learn' an effective pattern despite the more complex evolution of ξ across one orbital period.

A batch of agents is trained along the butterfly orbit, and three patterns are extracted from the resulting converged solutions, depicted in Figure 6.47. Patterns (a) and (b) involve four maneuvers per revolution, while (c) implements five. Consistent with the NRHO results, each configuration exhibits a strong preference toward placing maneuvers near apolune. Monte Carlo results leveraging these patterns are listed in the "Butterfly Orbit" section of Table 6.12. Notably, no escapes occur in the Monte Carlo trials along any of the RL-generated maneuver sequences. Furthermore, each pattern results in a significant annual cost reduction compared to the random maneuver locations employed during training. In particular, pattern (a) possesses a mean annual SK cost of 18.4 m/s with no failure cases, compared to 35.6 m/s in the training example. Adding the additional maneuver in pattern (c) increases the overall cost, indicating that a four-maneuver cadence is appropriate for this controller-orbit pair.

Halo Orbit Patterns

The halo orbit employed in this investigation, plotted in Figure 4.6(a), is significantly more unstable than the NRHOs and is, therefore, a basis for evaluating the maneuver selection process on a more numerically sensitive dynamical structure. Different agents converge on four representative maneuver patterns, depicted in Figure 6.48. Each scenario involves 5 maneuvers per revolution, roughly evenly spaced in time. While a two-maneuver cadence is often employed for halo orbits, the NN's performance indicates that additional maneuvers are necessary in this application. Mission constraints coupled with the selected control strategy affect the maneuver frequency, as observed in the Genesis mission where up to four maneuvers per orbital period of a Sun-Earth L_1 halo orbit were necessary to avoid deviation cases and satisfy mission requirements for the target point control scheme [132], [142]. As opposed to the NRHO and butterfly orbit scenarios, no specific region of space is outright avoided in the RL-converged patterns, though each pattern demonstrates a slight preference toward placing maneuvers closer to apolune than perilune. Each pattern spaces maneuvers approximately equally in time. This placement strategy is consistent with Genesis, where distributing maneuvers evenly in time was demonstrated as effective for the Sun-Earth L_1 halo orbit [132].

Monte Carlo results for each pattern are specified in the "Halo Orbit: RL control" section of Table 6.12. Due to overall similarity in the resulting patterns, annual cost remains relatively close between the four cases and, again, in contrast to the NRHO and butterfly orbits, no significant reduction in SK cost is observed when compared to random maneuver placements (though the total number of maneuvers is reduced from between 5 and 15, to only 5). Of the four included configurations, pattern (a) corresponds to the lowest annual cost of 17.01 m/s. No deviations occur in any of the simulated configurations for the halo orbit scenario.

Maneuver Placement: x-axis Control

The timing network training process employs an arbitrary control policy and is, hence, applicable to NNs and traditional methods alike. To demonstrate this flexibility, an XAC scheme for low-thrust is implemented following the process detailed by Newman et al. [7]. To evaluate the RL approach, the timing network is tasked with computing XAC maneuver sequences along the L_2 southern halo orbit plotted in Figure 4.6(a). Three representative patterns are produced from RL timing agents, and are illustrated in Figures 6.49(a) to 6.49(c). The fourth pattern, Figure 6.49(d), represents a two-maneuver cadence that alternates between perilune and apolune. This alternating sequence, with additional maneuvers performed at y-max amplitudes, was employed in the ARTEMIS mission [143], and is further demonstrated by Davis et al. in an impulsive XAC application applied along a similar halo orbit [2]. The inclusion of a known maneuver schedule provides a useful comparison for the RL-generated pattern results.

Each scenario implements two maneuvers per revolution, and Monte Carlo results are detailed in Table 6.12 in the section titled "Halo Orbit: XAC". Each RL-generated sequence yields lower annual SK costs, and higher a success percentage, compared to the simple perilune-apolune alternating scheme in Figure 6.49(d), with pattern (a) resulting in a mean annual cost of 20 cm/s. Costs are, as expected, notably lower than the RL controller for the same halo orbit, though the computational footprint is two orders of magnitude higher. Despite the large variation between these control schemes, the timing training process is, nevertheless, able to produce effective maneuver sequences for their respective controllers, demonstrating the flexibility of the proposed maneuver sequencing approach, and suggests future applicability to alternative control techniques.

Multi-Period Patterns

While the majority of the timing agents converge on patterns that repeat every revolution, several cases arise where an agent produces maneuver location sequences that repeat at period multiples. Two such patterns are depicted in Figure 6.50 and Figure 6.51, where the former repeats every two periods along the NRHO leveraging an RL controller, and the latter repeats every three periods of the halo orbit employing an XAC scheme. Monte Carlo simulation results for each pattern are listed in Table 6.12, demonstrating that both cases are equal to, or better than, other uncovered maneuver sequences in both success percentage and annual cost (though neither performs dramatically better). These non-homogeneous patterns demonstrate the NNs' ability to uncover and estimate more complex maneuver arrangements, and suggest this methodology is applicable in more challenging domains where such complexity is beneficial for the given problem.

6.4 Higher-fidelity Simulation

This investigation's three sample mission applications leverage the CR3BP as the simulation model in training and evaluating the proposed NN-G&C and NNIT frameworks. Leveraging the CR3BP enables the evaluation of the proposed methodologies in the context of a challenging dynamical model that is characterized by complex multi-body gravitational effects. While this modeling approach evaluates the efficacy the NN-G&C framework, the selected dynamical model possesses several simplifying assumptions that must be accounted for in a realistic mission scenario. In particular, the nonautonomous dynamics, the gravitational

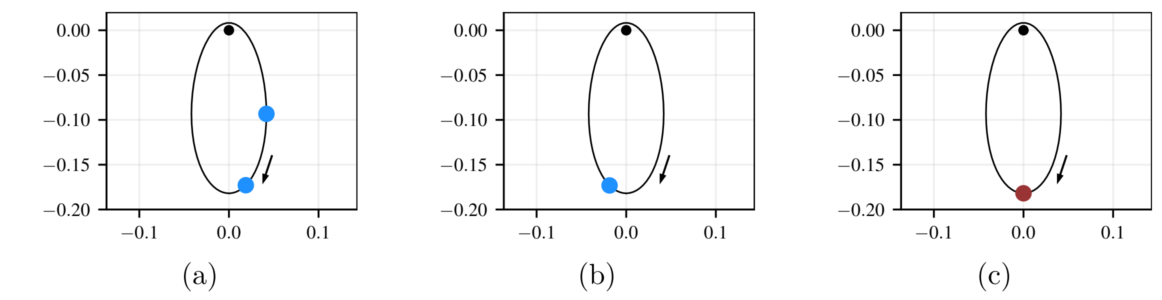


Figure 6.46. RL-generated patterns (blue) and the apolune pattern suggested in [118] (red), for a NN controller along the 9:2 NRHO ($\hat{y}$-$\hat{z}$, nondim)

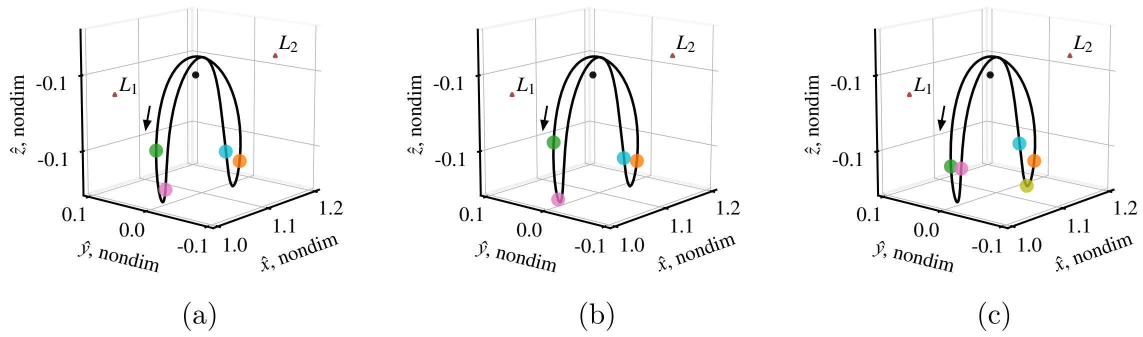


Figure 6.47. NN-generated control patterns along the sample butterfly orbit

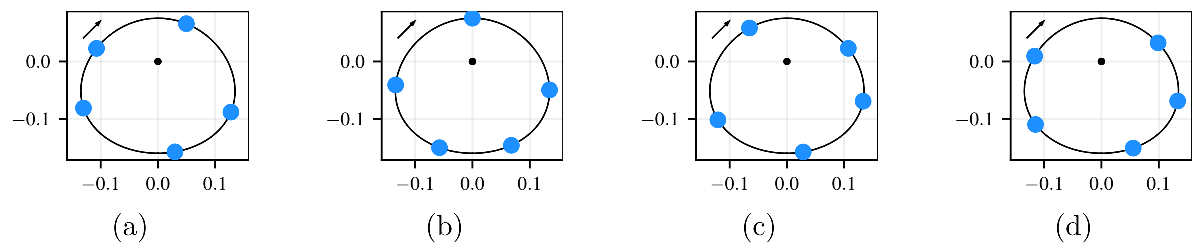


Figure 6.48. Sample patterns for a NN controller, halo orbit ($\hat{y}$-$\hat{z}$, nondim)

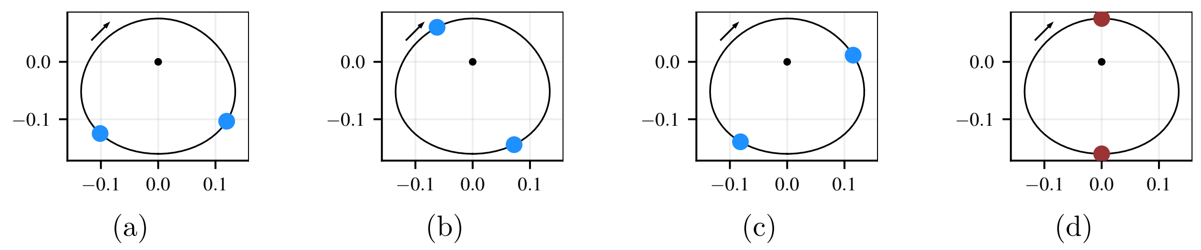


Figure 6.49. RL-generated two-maneuver patterns (blue) and an existing configuration from [2] (red) for an XAC scheme along the sample destination halo orbit ($\hat{y}$-$\hat{z}$, nondim)

Table 6.12. Monte Carlo results with 200 simulations and 25 orbit periods for each cislunar orbit plotted in Figure 6.43, where "success" indicates the controller did not deviate from the reference orbit. In cases where RL-control is employed, "Training" indicates the random maneuver location selection process that is employed during training of the RL controller.

	Pattern Fig. No.	**ΔV/ Rev.**	**Success Pct.**	**Annual Cost** μ, m/s	**Annual Cost** σ, m/s	**Pattern** ξ, rad
9:2 NRHO RL control	Training	1–3	79.5 %	5.66	1.37	Random
	Figure 6.46(a)	2	97.5 %	2.32	1.78	[3.9291, 5.4982]
	Figure 6.46(b)	1	95 %	2.35	0.76	[0.7741]
	Figure 6.46(c)	1	98 %	3.16	0.83	[0]
	Figure 6.50	3,1	98 %	2.32	0.76	[2.3545, 3.9756, 5.1958, 5.4999]
Butterfly RL control	Training	5–15	98.5 %	35.60	5.05	Random
	Figure 6.47(a)	4	100 %	18.40	0.62	[0.7833, 2.2087, 3.6375, 5.4107]
	Figure 6.47(b)	4	100 %	21.76	3.79	[0.7858, 2.1442, 2.9305, 5.4986]
	Figure 6.47(c)	5	100 %	28.33	1.78	[0.7854, 2.3570, 3.9278, 5.3160, 6.2392]
Halo Orbit RL control	Training	5–15	100 %	20.18	1.80	Random
	Figure 6.48(a)	5	100 %	17.01	0.91	[1.5419, 2.4496, 3.4051, 4.8138, 6.0126]
	Figure 6.48(b)	5	100 %	22.41	0.97	[0.5345, 1.9185, 3.1408, 4.4414, 5.6436]
	Figure 6.48(c)	5	100 %	19.33	1.39	[1.3195, 2.7830, 3.8332, 4.6227, 6.0247]
	Figure 6.48(d)	5	100 %	23.71	0.99	[1.2235, 2.3373, 3.7519, 4.6194, 5.7684]
Halo Orbit XAC	Figure 6.49(a)	2	100 %	0.20	0.030	[1.0233, 4.9817]
	Figure 6.49(b)	2	100 %	0.23	0.036	[2.8031, 5.5979]
	Figure 6.49(c)	2	99.5 %	0.21	0.042	[0.7854, 3.9270]
	Figure 6.49(d)	2	99.0 %	0.24	0.040	[0, π]
	Figure 6.51	2,2,2	100 %	0.20	0.025	[2.3562, 4.3136, 2.3562, 4.3136, 1.4497, 5.3210]

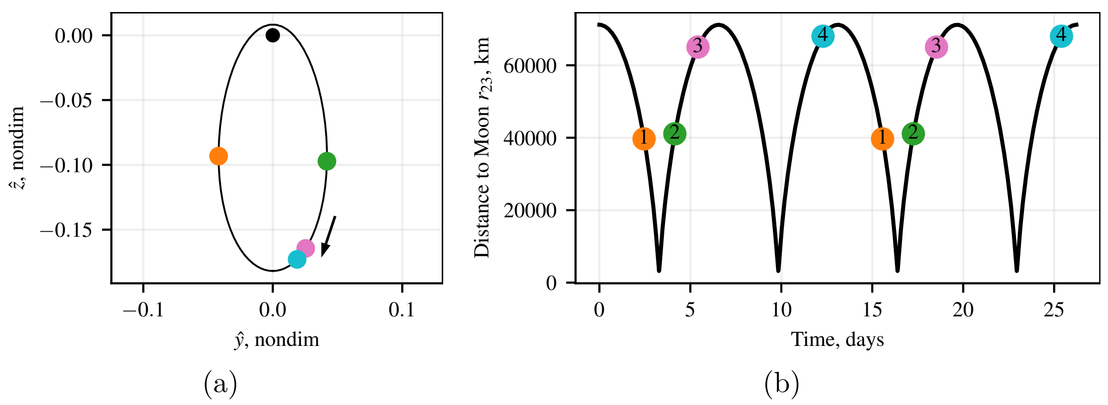


Figure 6.50. Sample non-homogeneous RL-generated maneuver pattern for an RL controller along the 9:2 NRHO

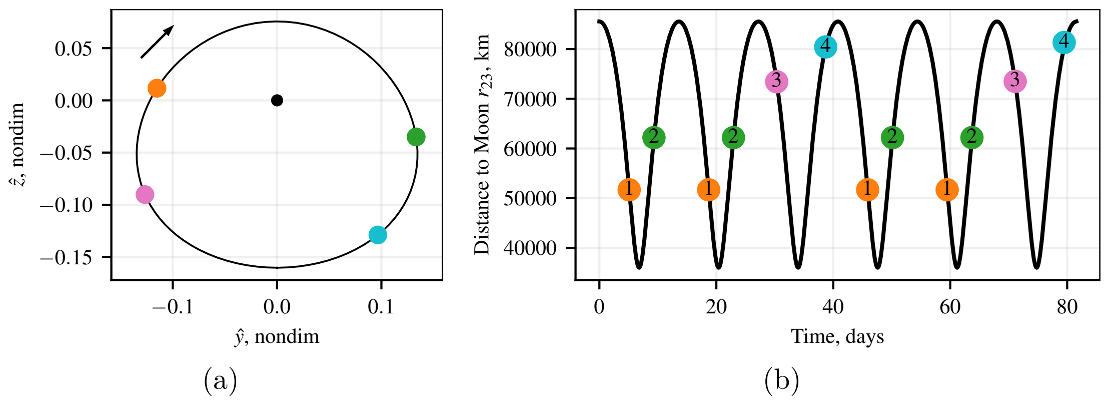


Figure 6.51. Maneuver sequence that repeats for every three periods of the sample halo orbit, employing an XAC scheme

influence of the sun, and the pulsation of the Moon within its orbit impart a non-negligible impact on a spacecraft's motion in cislunar space. Furthermore, while helpful for preliminary analysis, it may not be feasible to fix the spacecraft attitude in a rotating frame; therefore, an inertially-fixed direction is employed in this analysis to evaluate applicability in a more realistic simulation.

6.4.1 NRHO Recovery Ephemeris Targeting Process

An additional targeting step is leveraged in this investigation to transform CR3BP solutions into the $\mathcal{N}$-body ephemeris force model. As detailed in Section 3.5, differential corrections is a powerful tool to transition lower-fidelity simulations into higher-fidelity models.

To illustrate this capability, first consider a higher-fidelity simulation that does not employ a targeting procedure. Recall, Figure 6.5 illustrates an NNIT maneuver plan in the CR3BP that recovers the spacecraft to its desired NRHO at 46.4 days post-thruster failure, with Figure 6.5(b) depicting the entire 42.8-day NNIT recovery plan, and Figure 6.5(c) representing a simplified control strategy that fixes the maneuver direction in the rotating reference frame for a 31.4-hour burn. These simulations do not consider additional perturbing forces and, hence, a higher-fidelity simulation provides insight into the feasibility of implementing the NNIT-computed CR3BP maneuver plan directly in a more realistic scenario.

To demonstrate the ephemeris transition process, the transformation of a single representative scenario is considered. To convert the NNIT-generated CR3BP maneuver plan in Figure 6.5(c) to an inertial frame, the initial state (green) and thrust direction (red) are rotated into the Moon-centered J2000 reference frame, assuming an initial epoch of Jan. 1, 2025 (thrust direction is now inertially-fixed). If no maneuver is performed, the uncontrolled trajectory path quickly departs the NRHO, as depicted in Figure 6.52(a); departure occurs more rapidly in this higher-fidelity simulation compared to the CR3BP uncontrolled analog in Figure 6.5(a). The single thrust-arc CR3BP recovery plan (Figure 6.5(c)) is simulated in the low-thrust augmented $\mathcal{N}$-body ephemeris force mode by propagating the J2000-fixed-direction thrust arc (red) for 31.4 hours, depicted in the rotating frame in Figure 6.52(b). As apparent in this visualization, a fixed inertial orientation implies subtle direction changes in a rotating frame. The resulting final state is then propagated ballistically in the ephemeris model for an additional 32.78 days (blue), or five periods of the 9:2 NRHO.

Directly implementing the CR3BP maneuver plan in the $\mathcal{N}$-body ephemeris force model results in a trajectory that initially resembles the target NRHO. However, unlike the CR3BP simulation, the trajectory in Figure 6.52(b) begins to deviate after several revolutions of the orbit and has departed by 32.78 days. While this recovery plan likely achieves the objective of re-entering the region of applicability of the employed SK algorithm, it fails to obtain long-term adherence to the desired reference motion, thus revealing several limitations of this direct CR3BP-to-ephmeris recovery transformation approach. In particular, the higher-fidelity behavior is epoch-dependent, and it is challenging to predict the higher-fidelity behavior of the CR3BP maneuver plan prior to simulation. Hence, a more consistent

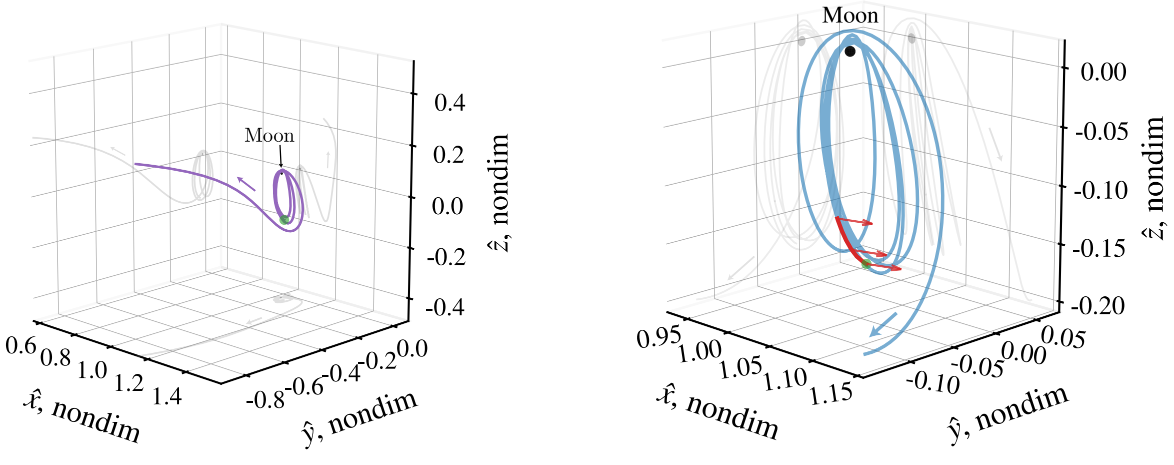


Figure 6.52. Higher fidelity simulation of NRHO recovery scenario at 46.4 days (epoch: Jan. 1, 2025), propagated for an additional five periods of the NRHO. The uncontrolled departure path (a) and simulated fixed-J2000 maneuver (b) correspond to the CR3BP recovery scenario in Figure 6.5.

and robust method of transitioning the lower-fidelity NNIT maneuver plans is explored in this investigation.

A differential corrections algorithm is employed to aid in transitioning CR3BP maneuver plans into $\mathcal{N}$-body ephemeris force model. The utility of targeting algorithms is demonstrated in the NNIT framework, enabling the creation of robust maneuver plans in the presence of errors introduced by NN estimation. In more realistic mission scenarios, lower-fidelity modeling errors are similarly addressed by implementing a direct multiple shooting algorithm, as detailed in Section 3.5. To illustrate this process, consider the NNIT-generated maneuver plan depicted in Figure 6.5(b). Dividing this maneuver plan into multiple segments, transforming each arc individually to the Moon-centered J2000 reference frame, and propagating in the $\mathcal{N}$-body ephemeris model, yields the discontinuous maneuver plan depicted in Figure 6.53. Although clearly discontinuous in state variables, this initial guess includes ten revolutions of the underlying NRHO (65.5 days), thus producing a startup solution that remains in the vicinity of the target orbit for a significantly longer duration of time than the simulation depicted in Figure 6.52(b). The targeting algorithm employs a minimum norm update that seeks to identify the feasible trajectory that is closest to the initial guess; thus, a longer term recovery plan is potentially available by correcting the series of trajectories in Figure 6.53.

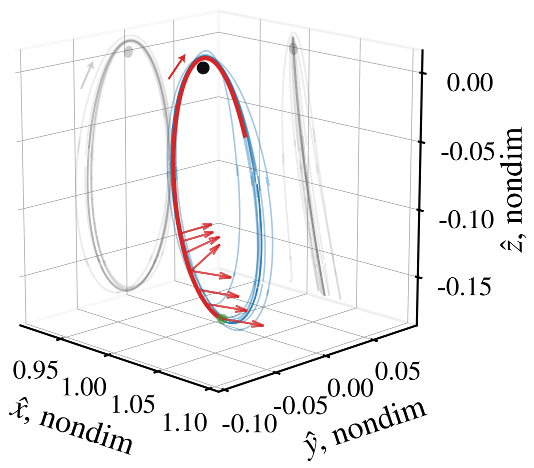


Figure 6.53. Sample initial guess for N-body ephemeris targeting algorithm

The N-body ephemeris targeting process ensures all segments are continuous in position, velocity, mass, and time. Similar to the NNIT implementation in Section 6.2.5, this multiple shooting algorithm incorporates the realistic constraint that low-thrust engines possess a lower bound of allowable thrust. As an alternative approach to transformation heuristics (Section 6.2.5), this NRHO recovery application implements the f_{min} threshold in the targeting process by applying the constraint,

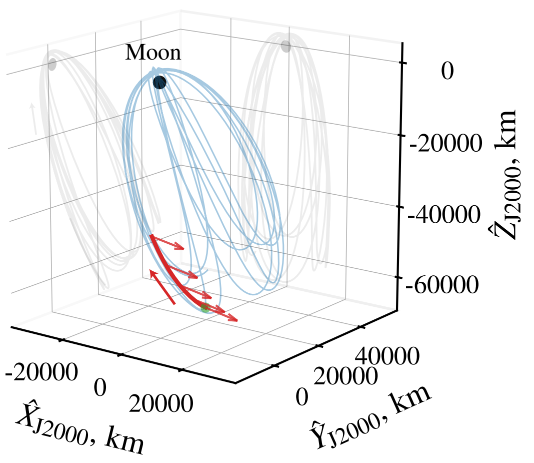
$$F_j(\boldsymbol{X}) = \begin{cases} f_i & f_i \leq f_{\text{min}} \\ f_i - f_{\text{max}} & f_i > f_{\text{min}} \end{cases} \tag{6.31}$$

along each thrusting segment. Satisfying the constraint in Equation (6.31) ensures that each arc is either purely ballistic or at maximum throttle. Furthermore, the implemented targeting algorithm optionally includes the ability to enforce thrust direction continuity. Thrust continuity is previously achieved in this investigation through initial guess transformation heuristics, Section 6.2.5. As an alternative method to identify a fixed thrust direction that satisfies recovery requirements, a simple continuity constraint is included,

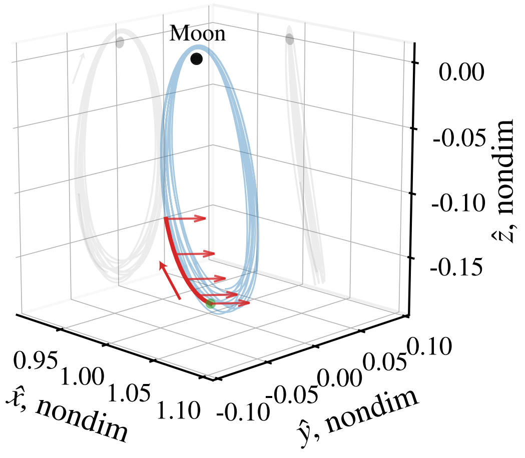
$$\boldsymbol{F}_j(\boldsymbol{X}) = \hat{u}_i - \hat{u}_{i-1} \tag{6.32}$$

where $(i-1)$ and (i) are thrusting segments, and $\hat{u}_i = (\hat{u}_{\mathrm{J2000}})\Big|_i$ is inertially-fixed in the Moon-centered J2000 reference frame. Satisfying Equation (6.32) ensures all sequential thrust segments possess the same J2000-oriented thrust direction. An additional consideration when applying thrust direction continuity is to only enforce the unit magnitude constraint on a single orientation vector from the sequence, $|\hat{u}_i| = 1$; the continuity constraint implicitly ensures all equal direction vectors possess a magnitude of one.

Iterating on the initial guess depicted in Figure 6.53 yields the converged maneuver plan depicted in Figure 6.54, with Figures 6.54(a) and 6.54(b) illustrating the recovery plan in the Moon-centered J2000 and Earth-Moon rotating reference frames, respectively. This converged solution differs from the uncorrected simulation in several notable ways. First, the single-thrust arc simulation in Figure 6.52(b) significantly deviates from the NRHO over the 32.8-day ballistic propagation segment. The converged solution, however, remains in the close vicinity of the NRHO for 70-days following the initial recovery segment. This longer-term boundedness is achieved by thrusting for 9.6 additional hours (1.7 days compared to 1.3 days), and subtliy adjusting the thrust direction by 20°. Thus, leveraging an iterative numerical method significantly increases the effectiveness of the resulting solution for this specific simulation, and introduces a greater degree of consistency in achieving recovery plans in a higher-fidelity model. Furthermore, ephemeris targeting offers the ability to enforce mission-specific criteria, e.g., constraints on eclipses or allowable attitude orientations. The inclusion of differential corrections at various stages of the transition to a higher-fidelity model is common in trajectory design; however, further analysis of incorporating NNIT into higher-fidelity models remains future work.



(a) Moon-centered J2000



(b) Rotating frame

Figure 6.54. Higher-fidelity NNIT recovery path at 46.4 days (summarized in Table 6.13).

Additional Higher-Fidelity Recovery Plans

To demonstrate further applicability of the proposed ephemeris targeting step, recovery trajectories at various locations along the unintended departure simulation in Section 6.1.3 are transformed and corrected in an $\mathcal{N}$-body ephemeris force model. Recall that time along the departing segment is measured from the final SK maneuver, as visualized in Figure 6.2. Several previously-analyzed recovery plans at various points in time are transformed to a Moon-J2000 inertial reference frame, and corrected for continuity using the targeting methodology discussed in Section 6.4.1. These simulations are listed in Table 6.13 with figure numbers corresponding to visualizations in specified models and reference frames.

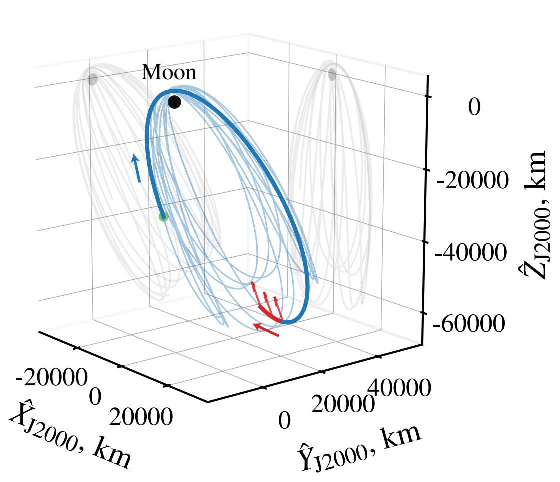
Each NNIT-determined CR3BP maneuver plan along the departing segment quickly converges to a higher-fidelity $\mathcal{N}$-body ephemeris counterpart. At 29 days post-thrust failure event, the CR3BP maneuver plan performs comparably to an XAC-generated SK maneuver. Similar to the sample application visualized Figure 6.54, thrust direction continuity is constrained, resulting in the maneuver plans depicted in Figure 6.55. However, in achieving longer-term NRHO motion, the $\mathcal{N}$-body ephemeris targeting algorithm results an additional 10 hours of thrust time. As previously noted in the CR3BP recovery simulation, the pro-

Table 6.13. Summary of higher-fidelity maneuver plans stemming from NNIT-generated NRHO recovery paths (epoch: Jan. 1, 2025). Thrust directions fixed in J2000 frame along each integration segment

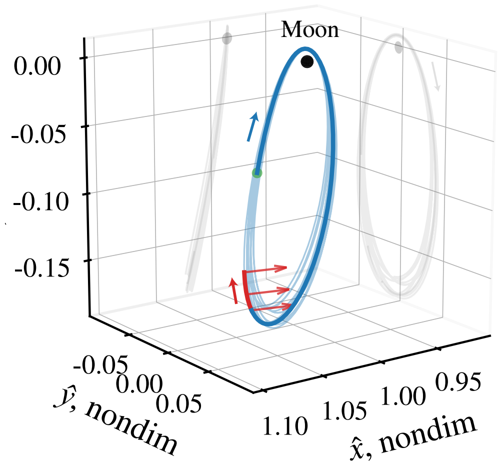
	CR3BP	$\mathcal{N}$-body Ephemeris	
Departure, days	Rotating frame	Moon-centered J2000	Rotating
29	Figure 6.4	Figure 6.55(a)	Figure 6.55(b)
46.4	Figure 6.5	Figure 6.54(a)	Figure 6.54(b)
49	Figure 6.6(a)	Figure 6.56(a)	Figure 6.56(b)
51	Figure 6.6(b)	Figure 6.57(a)	Figure 6.57(b)
55	Figure 6.6(c)	Figure 6.58(a)	Figure 6.58(b)

posed NNIT framework tends to expend excessive propellant at lower levels of error; thus, the transformed solution reinforces the conclusion that the proposed NNIT recovery algorithm from Section 6.1 has greater utility at larger error levels, and its use in SK applications is likely limited by propellant efficiency considerations.

Recovery plans in the CR3BP that originate further into the inadvertent departure duration are depicted in Figure 6.6. Unlike prior ephemeris corrections simulations, these segments involve significantly longer thrust durations and, thus, a single fixed-orientation thrust segment is unavailable from the NNIT maneuver plan. However, consistent with the original NNIT simulation, relaxing the fixed-orientation constraint to allow attitude to instantaneously shift between segments in the Moon-centered J2000 reference frame produces the low-thrust trajectories in Figure 6.56 (49 days), Figure 6.57 (51 days), and Figure 6.58 (55 days). Convergence is rapidly obtained in each case, demonstrating that similar motion exists in a higher-fidelity model. The initial location for each simulation stems for a CR3BP state and, thus, analyzing the proposed methodology in a higher-fidelity simulation that fixes a J2000 initial state remains future work.

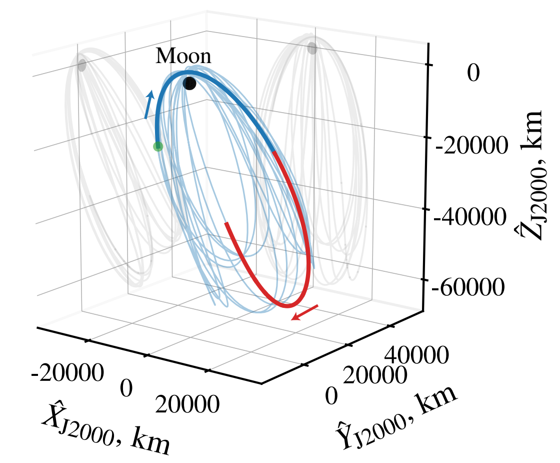


(a) Moon-centered J2000

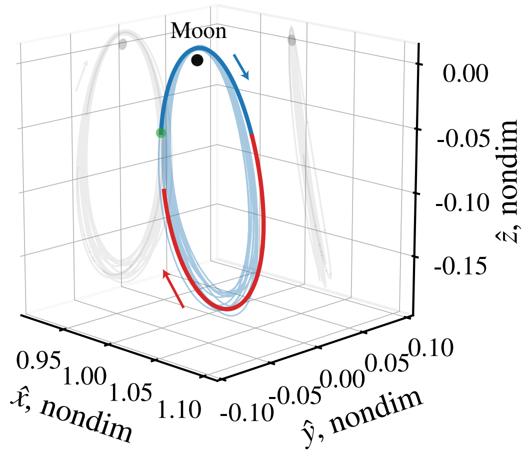


(b) Rotating frame

Figure 6.55. Higher-fidelity NNIT recovery path at 29 days (summarized in Table 6.13).

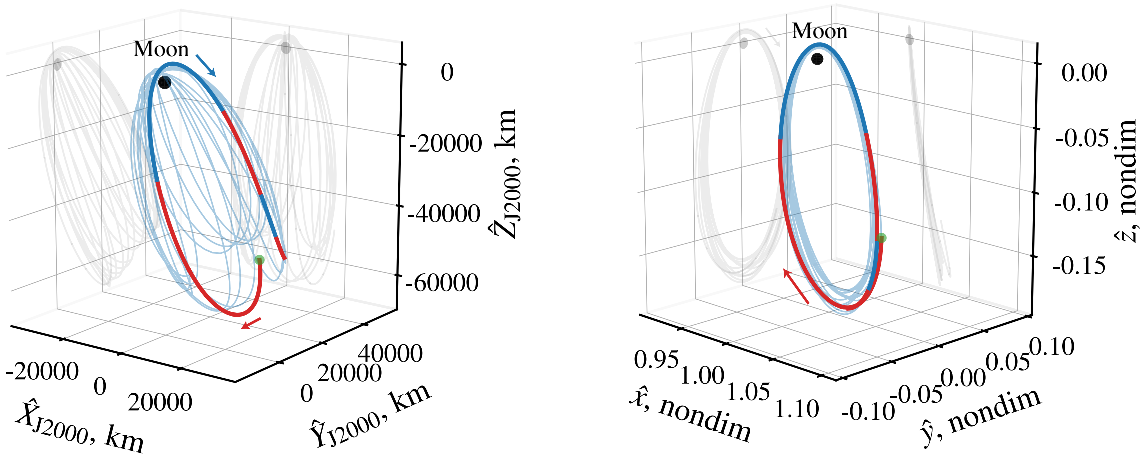


(a) Moon-centered J2000



(b) Rotating frame

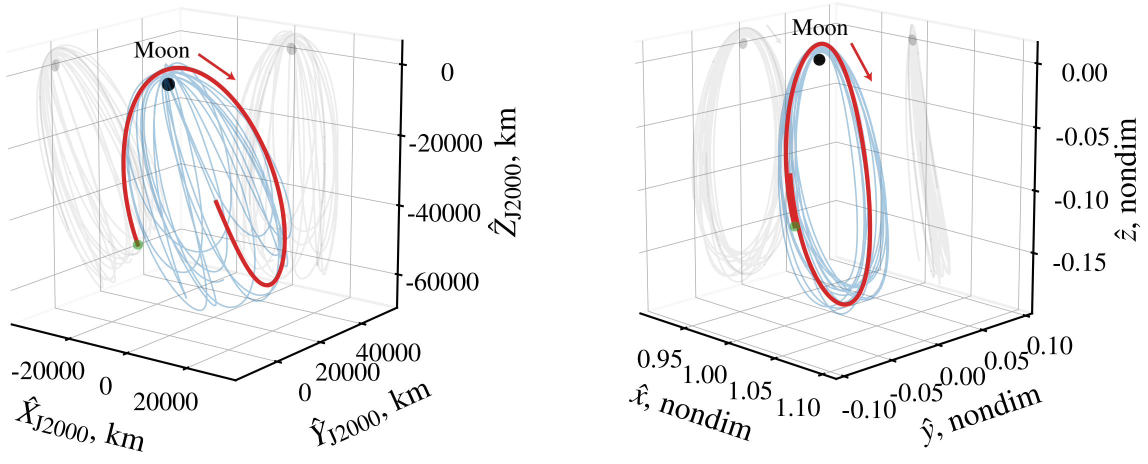
Figure 6.56. Higher-fidelity NNIT recovery path at 49 days (summarized in Table 6.13).



(a) Moon-centered J2000

(b) Rotating frame

Figure 6.57. Higher-fidelity NNIT recovery path at 51 days (summarized in Table 6.13).



(a) Moon-centered J2000

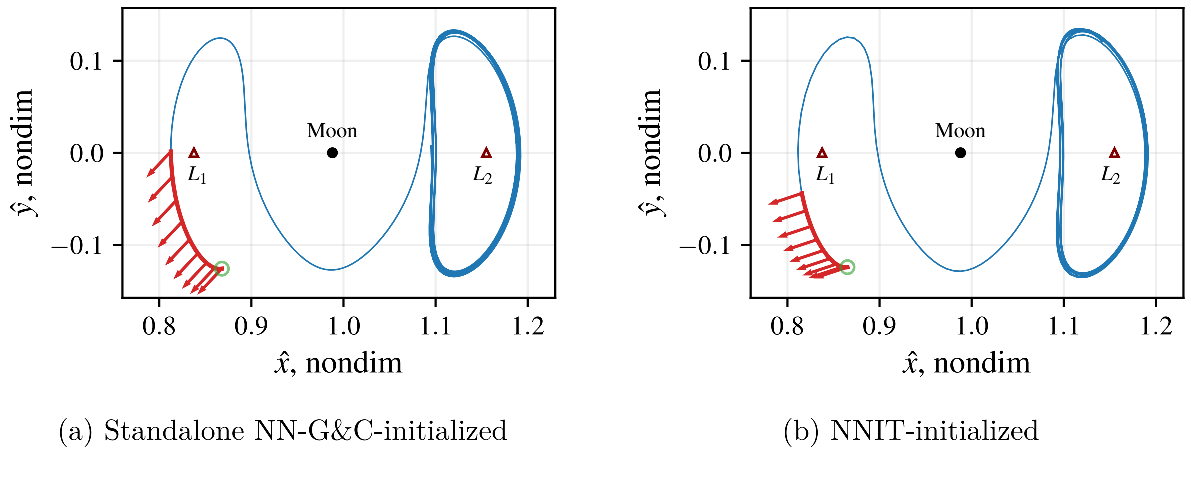
(b) Rotating frame

Figure 6.58. Higher-fidelity NNIT recovery path at 55 days (summarized in Table 6.13).

6.4.2 Heteroclinic Transfer Recovery Maneuver Plans

The transfer recovery mission application, detailed in Section 6.2, demonstrates remarkable flexibility of the proposed NN-G&C techniques in the presence of several simplifying assumption. In particular, motion throughout the CR3BP analysis is assumed fixed in the $\hat{x}$-$\hat{y}$ plane. Hence, a higher-fidelity analysis provides insight into applicability of the proposed RL training techniques and NN-G&C framework in a more realistic scenario (a similar analysis for the NRHO recovery task is presented in Section 6.4.1). Several sample simulations are corrected in the $\mathcal{N}$-body ephemeris force model by leveraging the targeting process detailed in Section 6.4.1.

The heteroclinic transfer sample mission application is presented in this investigation through the single representative scenario depicted in Figure 6.13, where the corresponding NN-G&C recovery plans are visualized in Figure 6.16 (Standalone NN-G&C) and Figure 6.38(a) (NNIT). These two simulations are leveraged as initial guesses for $\mathcal{N}$-body corrections, with converged ephemeris maneuver plans plotted in Figure 6.59. For the standalone NN-G&C case, thrust direction continuity is enforced via orientation constraints, Equation (6.32), and coasting segments emerge from the thresholding approach detailed in Equation (6.31). Alternatively, the NNIT process achieves similar continuity through heuristic initial guess transformations, Section 6.2.5, that combine sequential thrust arcs into a

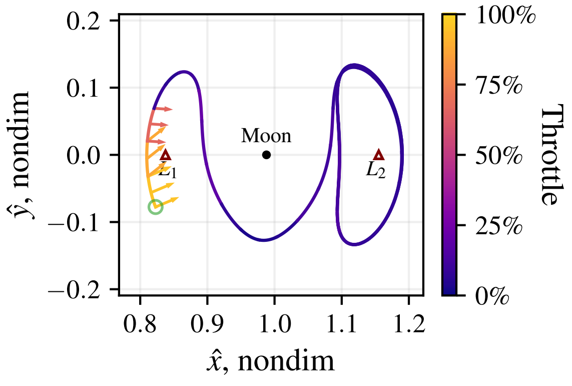


(a) Standalone NN-G&C-initialized (b) NNIT-initialized

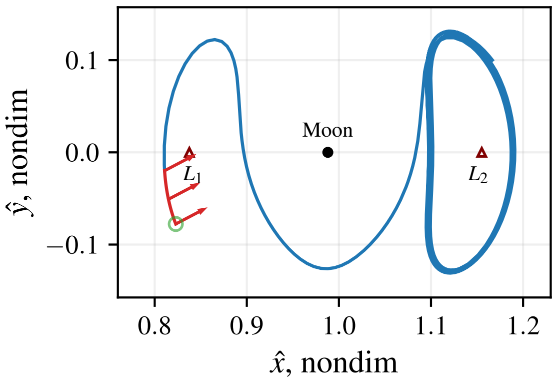
Figure 6.59. Higher-fidelity maneuver plans for the sample perturbation and departure path depicted in Figure 6.13.

single unified segment. While convergence is rapidly obtained in both scenarios, the NNIT-originated recovery response exhibits a shorter thrust segment and, thus, expends less fuel to perform the same recovery task. This inefficiency stems from an overestimation of thrust time that is characteristic of the employed standalone NN-G&C algorithm, and is addressed in the NNIT implementation through the thrust time adjustment, Equation (6.22), performed during the initial guess transformation procedure. Furthermore, by first enforcing low-thrust constraints on magnitude and direction in the CR3BP, the NNIT approach yields an $\mathcal{N}$-body ephemeris targeting algorithm with an order of magnitude fewer free variables and constraints than its analog standalone NN-G&C process. Thus, this scenario demonstrates the potential benefits gained through NNIT in the context of the CR3BP by simplifying the targeting problem and achieving more efficient solutions by reducing the thrust time over-estimation.

Several additional higher-fidelity maneuver transfer recovery paths are simulated to evaluate the proposed NN-G&C algorithm and $\mathcal{N}$-body-ephemeris transition and corrections processes. First, the proposed ephemeris targeting techniques are applied to the sample representative NNIT scenarios in Figures 6.35(a) and 6.35(c) (visualized in Figures 6.60 and 6.61, respectively. In both cases, the targeter rapidly converges on a maneuver plan that incorporates additional gravitational forces. Next, Figure 6.62 further exemplifies the



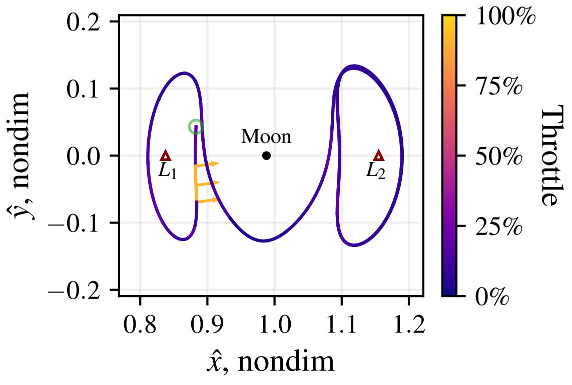
(a) CR3BP standalone NN-G&C simulation



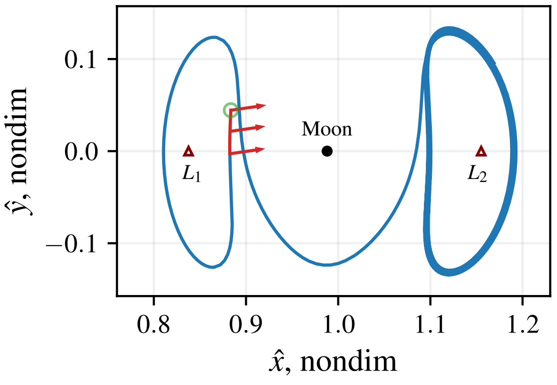
(b) NNIT-initialized ephemeris trajectory

Figure 6.60. Maneuver plans for the sample NN-G&C simulation depicted in Figures 6.32 and 6.35(a).

utility of the NNIT approach to NN-G&C. In this simulation, inaccuracies in the NN estimations lead to multiple larger thrust segments prior to entering the L_2 Lyapunov orbit. While this behavior poses practical challenges to standalone implementation, an iterative approach is capable of overcoming such inaccuracies, leading to the converged solution visualized in Figure 6.62(b). Thus, while additional computational overhead is introduced compared to a standalone NN-G&C approach, significant benefit is observed in the blended NNIT framework.

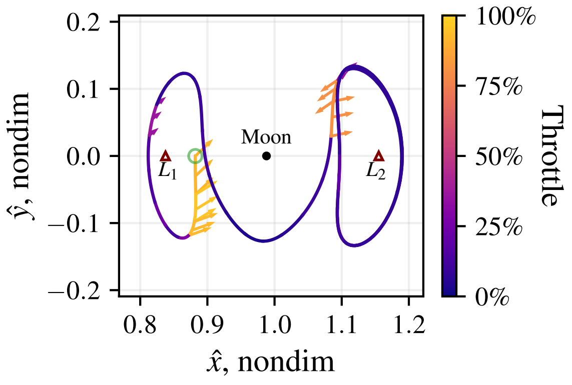


(a) CR3BP standalone NN-G&C simulation

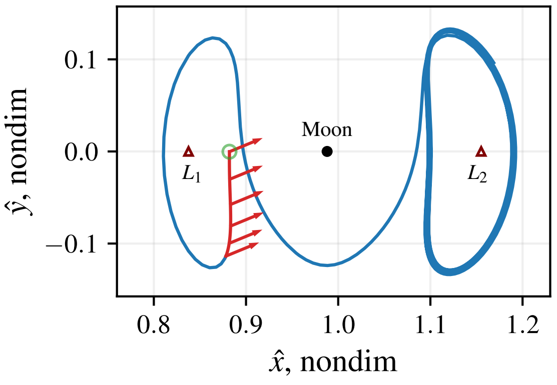


(b) NNIT-initialized ephemeris trajectory

Figure 6.61. CR3BP and $\mathcal{N}$-body ephemeris trajectories for the NN-G&C recovery paths visualized in Figures 6.34 and 6.35(c).



(a) CR3BP standalone NN-G&C simulation



(b) NNIT-initialized ephemeris trajectory

Figure 6.62. NN-G&C recovery simulation demonstrating the capability of NNIT's employed targeting process to eliminate unneeded later thrust segment and instead construct a simple fixed-orientation thrust arc.

7. CONCLUSION

As expeditions into space continue to increase in both quantity and ambition, onboard autonomy has emerged as a critical component in enabling the success of future missions. This investigation leverages recent advancements in artificial neural networks to enable computationally efficient closed-loop G&C processes in challenging, nonlinear dynamical regimes with potential applicability for onboard use. In particular, a class of hybrid methods, denoted NNIT, is developed through this research, demonstrating significant promise in leveraging NNs to improve the range of applicability and performance of traditional G&C algorithms, thus offering a potentially valuable framework for autonomous decision-making for future low-thrust space missions. The proposed NN-G&C frameworks exhibit a remarkable ability to autonomously overcome significant deviations from reference trajectories, as evidenced through several simulated mission applications. This investigation's sample mission applications showcase the potential practicality of implementing an NN-aided G&C framework without demanding excessive computational resources, thus demonstrating the considerable promise for future NN-G&C concepts and RL training procedures. Onboard maneuver planning will enable future spacecraft to respond more effectively and efficiently to unexpected events in challenging regions of space, and RL-trained NNs provide an exciting potential component in facilitating this autonomy.

7.1 Neural Network-aided Guidance and Control

The demonstrated NN-G&C techniques offer computationally efficient closed-loop G&C in a multi-body regime, both as a standalone process or as an NNIT approach in conjunction with an onboard targeting scheme. Many state-of-the-art low-thrust G&C methodologies require access to a high-performance computing resources, rendering them unsuitable for a flight computer. Conversely, many onboard approaches are computationally efficient, yet they are unable to take advantage of the modern abundance of large-scale computers. With RL, the computationally intensive training process capitalizes on ground-based computing resources, while nevertheless generating a computationally efficient closed-loop controller. Furthermore, RL approaches all separate the agent from the environment, which renders learning schemes applicable to multiple spacecraft and mission scenarios, as evidenced in this investigation through the variation mission application objectives, trajectory characteristics, low-thrust propulsive capability, and G&C frameworks.

The performance of standalone NN-G&C is assessed in this investigation through the heteroclinic transfer recovery mission application; in the context of this analysis, 'standalone' refers to framework of leveraging an NN to directly generate thrust commands without an intermediate targeting step. In the transfer recovery analysis, RL is leveraged to train a NN controller to estimate low-thrust control states to return a spacecraft to a desired transfer path between libration point orbits. In sample simulations, assuming perturbations three orders of magnitude greater than potential navigation errors, the standalone NN-G&C approach successfully completes the given transfer in more than 99 % of cases. Furthermore, additional scenarios demonstrate remarkable controller reconfigurability and generalization, where the RL-trained NN controller adapts to nearby geometries not encountered during training. The relationship between performance on nearby reference paths and orbital energy suggests that energy observations and variation are important during training. This generalization ability indicates potential of a NN controller continuing to perform the requisite G&C task in real-time despite subtle shifts in mission objectives. The success of the standalone NN-G&C method across many reference geometries and thrust levels in this analysis demonstrates the flexibility of the underlying RL training scheme.

Reinforcement learning offers a powerful approach in this investigation for uncovering insight in support of SK for timing and control along challenging, nonlinear multi-body orbits. This investigation details a general approach for training a NN-parameterized SK controller that does not rely on individual orbit characteristics or domain-specific knowledge. Furthermore, RL is demonstrated as remarkably effective in determining maneuver placement strategies for NN-G&C and XAC algorithms alike, suggesting future applicability for complex destination orbits and alternative control schemes. While computational efficiency and accuracy motivate NN control applications, further research is necessary to address current propellant consumption limitations in orbit maintenance applications. Onboard SK planning will enable future spacecraft to operate autonomously, and NNs provide an exciting component in facilitating autonomy despite challenging nonlinear regions of space.

This investigation explores the validity and potential limitations of the employed model's simplifying assumptions by transitioning NN-G&C-generated CR3BP maneuver plans to a higher-fidelity $\mathcal{N}$-body ephemeris force model. An additional targeting step is introduced to facilitate this transition process, and higher-fidelity analogs are successfully identified for maneuver plans in both the NRHO and heteroclinic transfer recovery mission applications. While this analysis demonstrates promise for future applicability of the proposed learning methods and representing the NN-G&C low-thrust trajectories in a higher-fidelity model, the current investigation does not evaluate performance is the context of more realistic mission scenarios that include, for example, more accurate error models. Due to the demonstrated efficacy of NN-G&C approaches in the CR3BP, applying the NNIT framework to a higher-fidelity mission application remains an exciting potential avenue for future work

7.2 Neural Network-Initialized Targeting (NNIT)

The inherent complexity of NN behavior presents a significant risk when assessing their suitability for onboard G&C tasks. In the context of spaceflight, NN speed and accuracy must be weighed against interpretability challenges when evaluating G&C options. Traditional iterative methods, unlike NN-G&C schemes, produce exact solutions but require an accurate initial guess and are computationally expensive. The proposed NNIT approach mitigates the risk incurred by the NN by employing a differential corrections process to ensure convergence and, conversely, the NN speeds up the targeter by rapidly offering accurate startup solutions. Thus, the NNIT framework simultaneously expands the solution space of the targeting algorithm while reducing the impact of chaotic dynamics on the NN's ability to perform challenging tasks.

In merging targeting and NN control paradigms, the proposed NNIT method offers a realistic G&C architecture with demonstrated functionality in simulated low-thrust orbit recovery scenarios. In particular, the 'integrated' NNIT architecture demonstrates promising results in the inadvertent departure scenario from a planned NRHO trajectory, resulting in a multi-week extension of the recovery window compared to a conventional crossing-control SK method. Notably, the recovery maneuver plans generated by NNIT consistently avoid the perilune region for larger thrusting segments, thereby corroborating existing understanding from prior investigations that placing maneuvers near the Moon in NRHO application results in an ineffective G&C policy. To achieve this objective, the NN implements an effective maneuver timing procedure by typically assigning a throttle value of zero when in proximity to perilune, thus simultaneously achieving successful low-thrust control and maneuver positioning policies. The value of the NNIT scheme is demonstrated through this mission application, and the current results suggest future applicability in applying the integrated NNIT framework to increasingly challenging tasks in complex dynamical regions of space.

The concept of implementing an NNIT framework given a pre-trained NN controller is evaluated in this investigation through the heteroclinic transfer recovery mission application. In this context, given the stated goal of transfer recovery, NNIT simulations demonstrate the efficacy of the proposed NNIT initial guess transformation processes. In this context, simple transformation heuristics are employed to enable the rapid combination and scaling of successive thrust segments to facilitate accurate startup solution generation despite errors present in the NN-G&C simulation. By allowing infeasible preliminary estimates, the NNIT initial guess transformation process yields simple startup solutions that are rapidly iterated on to produce feasible maneuver plans. The proposed transformation process mitigates the effect of observed NN-G&C overestimations in thrust time, and enables control variable continuity without requiring hundreds of additional design variables in the targeting process. While simulations demonstrate the implemented NNIT method as successful in the vast majority of simulations, situations emerge such that targeter is not able to identify a solution, and future work is necessary to acquire a more comprehensive understanding of the limitations of the proposed transformation method, and establishing an effective response in such cases.

7.3 Recommendations for Future Work

Many exciting avenues for future work emerge from the current body of research in applying RL techniques in spacecraft G&C tasks. Specifically, for the present investigation, recommendations for future research areas are broadly categorized between the application area (multi-body G&C for low-thrust spacecraft) and the employed ML framework.

Application Areas

The current investigation focuses on G&C tasks within the cislunar region. Several opportunities for future work exist in expanding the current analysis to other domains and in evaluating the proposed NNIT approach in light of alternative computational G&C strategies.

Optimization initialization This investigation demonstrates the efficacy of leveraging a NN to initialize components of a conventional G&C strategy through the proposed NNIT framework. While a direct multiple shooting algorithm is employed in this research, this targeting algorithm serves as a stand-in for any such computational G&C method that requires initialization. Thus, an interesting future research direction involves evaluating the NNIT concept for initializing other G&C schemes. In particular, optimization methods are ubiquitous in spaceflight, with several investigations suggesting the potential for onboard use. A NNIT approach may improve this type of G&C architecture, and future work is necessary to understand the potential improvements such a strategy might yield.

Higher-fidelity models and applications A key advantage gained through RL is the clear delineation between the agent and the environment. This learning approach is independent of the underlying dynamical model, making it applicable across multiple domains. The present study utilizes the CR3BP to demonstrate controller effectiveness within a complex dynamical model and showcases the persistence of these maneuver plans in a higher-fidelity $\mathcal{N}$-body force model. Numerous potential research avenues exist for applying the methods from this investigation to other challenging dynamical environments, such as four-body problems, the $\mathcal{N}$-body ephemeris model, and small body operations. One such possibility involves implementing a 'transfer learning' approach that involves converging a NN on desired behavior in a lower-fidelity model and subsequently leveraging the NN's trainable parameter to initialize the corresponding NN in a higher-fidelity problem.

Machine Learning Framework

Several possibilities for future work involve examining the present investigation's RL and NN implementation decisions. The current research seeks to leverage an established RL framework to evaluate options for implementing learning environments and to understand how the resulting NN may be used to support G&C tasks. Alternatively, analyzing the present work from an ML perspective involves seeking to improve machine learning capability through improved NN model decisions, RL algorithm selection and implementation, and state/action feature design. Advancing the understanding of effectively selecting and implementing the fundamental machine learning mechanisms that enable this research would provide a significant step forward toward eventual onboard implementation, and three such possible areas for future work are enumerated below.

Evaluate RL implementation options The recent surge of interest in RL has generated an extensive array of options when implementing an RL training method. For the NN model, this variation encompasses aspects such as size, depth, and activation functions. Furthermore, recent studies have highlighted the effectiveness of recurrent neural networks in spaceflight applications in accounting for temporal correlations in observations [14], [19], [20], [144]. Given a selected NN model with a given set of hyperparameters, multiple state-of-the-art actor-critic RL training procedures are available, including PPO, TD3, and SAC, each with their own set of possible customizations. Moreover, RL methods are notoriously difficult to benchmark as the performance of each algorithm varies greatly based on implementation [145], and numerous customizations to the original algorithms are ubiquitously employed despite not appearing in the original source. Certain algorithms and configurations do not always follow predictable trends in trying to predict which will perform best for a given application. The present investigation largely bases its RL implementations, NN models, and tuning parameter selections on prior work, and a more comprehensive analysis would likely suggest that a more effective RL approach is available. While this investigation focuses on environment definitions, obtaining a better understanding of the broad array of possible machine learning configurations would likely lead to improved performance throughout this investigation.

Improve RL signal design This investigation proposes multiple options for designing observation and action signals in support of low-thrust G&C applications. An interesting avenue of future work involves performing a more comprehensive analysis of the relative importance of each state feature and to evaluate alternative coordinate options. This sort of feature engineering is an increasingly common area of ML research and would likely increase NN performance in the simulated tasks. Furthermore, an analysis of the available options for low-thrust control encoding could provide a useful baseline for future work, as low-thrust is becoming an increasingly common area for ML applications in spaceflight. This type of signal analysis is beginning to appear in the literature, e.g., in a relative motion application [146], and forms an important step toward potential future onboard deployment of NN models.

Investigation Hierarchical Reinforcement Learning (HRL) options One obstacle faced in constructing a NN to aid in spaceflight G&C tasks is accounting for scaling issues that accompany the long simulation times. Specifically, it is challenging to simultaneously achieve a sufficient degree of precision on both a minute-by-minute and year-by-year basis. This investigation observes a similar trend, where a high degree of short-term precision is often sacrificed to achieve longer-term objectives. Recent advancements in RL suggest HRL as a potentially promising approach [147], [148] in decomposing near-term and overarching objectives. One recent study demonstrates the usage of HRL in designing spaceflight campaigns [149]. Investigating possible benefits of an HRL framework for G&C applications presents a potentially interesting avenue for expanding on the present investigation.

REFERENCES

[1] M. Tipaldi, R. Iervolino, and P. R. Massenio, “Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges,” *Annual Reviews in Control*, vol. 54, pp. 1–23, 2022, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2022.07.004.

[2] D. Davis, S. Bhatt, K. Howell, *et al.*, “Orbit maintenance and navigation of human spacecraft at cislunar near rectilinear halo orbits,” in *27th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, San Antonio, Texas, Feb. 2017.

[3] B. G. Marchand, M. W. Weeks, C. W. Smith, and S. Scarritt, “Onboard autonomous targeting for the trans-earth phase of orion,” *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 3, pp. 943–956, 2010. DOI: 10.2514/1.42384. eprint: https://doi.org/10.2514/1.42384.

[4] M. A. Shoemaker, S. Hur-Diaz, A. J. Liounis, *et al.*, “Terrain relative navigation in a lunar landing scenario using autongc,” in *AIAA SciTech Forum*, AIAA, San Diego, CA, Jan. 2022. [Online]. Available: https://ntrs.nasa.gov/citations/20210024481.

[5] D. Dunham and C. Roberts, “Stationkeeping techniques for libration-point satellites,” *The Journal of the Astronautical Sciences*, vol. 49, no. 1, pp. 127–144, Mar. 2001. DOI: 10.1007/bf03546340.

[6] D. C. Folta, T. A. Pavlak, A. F. Haapala, K. C. Howell, and M. A. Woodard, “Earthmoon libration point orbit stationkeeping: Theory, modeling, and operations,” *Acta Astronautica*, vol. 94, no. 1, pp. 421–433, 2014, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2013.01.022.

[7] C. P. Newman, D. C. Davis, R. J. Whitley, J. R. Guinn, and M. S. Ryne, “Stationkeeping, orbit determination, and attitude control for spacecraft in near rectilinear halo orbits,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, 2018, pp. 1–20.

[8] D. Izzo, D. Tailor, and T. Vasileiou, “On the stability analysis of deep neural network representations of an optimal state feedback,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 1, pp. 145–154, 2021. DOI: 10.1109/TAES.2020.3010670.

[9] H. Li, H. Baoyin, and F. Topputo, "Neural networks in time-optimal low-thrust interplanetary transfers," *IEEE Access*, vol. 7, pp. 156 413–156 419, 2019. DOI: 10.1109/ACCESS.2019.2946657.

[10] L. Cheng, Z. Wang, F. Jiang, and J. Li, "Fast generation of optimal asteroid landing trajectories using deep neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 4, pp. 2642–2655, 2020. DOI: 10.1109/TAES.2019.2952700.

[11] N. L. O. Parrish, "Low thrust trajectory optimization in cislunar and translunar space," Ph.D. dissertation, University of Colorado Boulder, 2018.

[12] T. A. Estlin, B. J. Bornstein, D. M. Gaines, *et al.*, "Aegis automated science targeting for the mer opportunity rover," eng, *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–19, 2012, ISSN: 21576904.

[13] R. Francis, T. Estlin, G. Doran, *et al.*, "Aegis autonomous targeting for chemcam on mars science laboratory: Deployment and results of initial science team use," English, *Science Robotics*, vol. 2, no. 7, 2017, ISSN: 2470-9476.

[14] L. Federici, B. Benedikter, and A. Zavoli, "Deep learning techniques for autonomous spacecraft guidance during proximity operations," *Journal of Spacecraft and Rockets*, vol. 58, no. 6, pp. 1774–1785, 2021. DOI: 10.2514/1.A35076. eprint: https://doi.org/10.2514/1.A35076. [Online]. Available: https://doi.org/10.2514/1.A35076.

[15] J. Broida and R. Linares, "Spacecraft rendezvous guidance in cluttered environments via reinforcement learning," in *29th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, Ka'anapali, Hawaii, Jan. 2019, pp. 1–15.

[16] Y.-h. Zhu and Y.-z. Luo, "Fast approximation of optimal perturbed long-duration impulsive transfers via artificial neural networks," eng, *IEEE transactions on aerospace and electronic systems*, vol. 57, no. 2, pp. 1123–1138, 2021, ISSN: 0018-9251.

[17] A. Huang and S. Wu, "Neural network-based approximation model for perturbed orbit rendezvous," eng, *Mathematics (Basel)*, vol. 10, no. 14, 2022, ISSN: 2227-7390.

[18] A. Scorsoglio, R. Furfaro, R. Linares, and M. Massari, "Relative motion guidance for near-rectilinear lunar orbits with path constraints via actor-critic reinforcement learning," *Advances in Space Research*, vol. 71, no. 1, pp. 316–335, 2023, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2022.08.002.

[19] L. Federici, A. Scorsoglio, A. Zavoli, and R. Furfaro, "Meta-reinforcement learning for adaptive spacecraft guidance during finite-thrust rendezvous missions," *Acta Astronautica*, vol. 201, pp. 129–141, 2022, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2022.08.047.

[20] B. Gaudet, R. Linares, and R. Furfaro, "Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations," eng, *Acta Astronautica*, vol. 171, pp. 1–13, 2020, ISSN: 0094-5765.

[21] B. Gaudet, R. Linares, and R. Furfaro, "Six degree-of-freedom hovering over an asteroid with unknown environmental dynamics via reinforcement learning," in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1910.

[22] T. Uriot, D. Izzo, L. F. Simões, *et al.*, "Spacecraft collision avoidance challenge: Design and results of a machine learning competition," *Astrodynamics*, Apr. 2021, ISSN: 2522-0098. DOI: 10.1007/s42064-021-0101-5.

[23] J. Jiang, X. Zeng, D. Guzzetti, and Y. You, "Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures," *Acta Astronautica*, vol. 171, pp. 265–279, 2020, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2020.03.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576520301351.

[24] J. G. Elkins, R. Sood, and C. Rumpf, "Bridging reinforcement learning and online learning for spacecraft attitude control," *Journal of Aerospace Information Systems*, vol. 19, no. 1, pp. 62–69, 2022. DOI: 10.2514/1.I010958. eprint: https://doi.org/10.2514/1.I010958. [Online]. Available: https://doi.org/10.2514/1.I010958.

[25] G. Viavattene, E. Grustan-Gutierrez, and M. Ceriotti, "Multi-objective optimization of low-thrust propulsion systems for multi-target missions using anns," *Advances in Space Research*, vol. 70, no. 8, pp. 2287–2301, 2022, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2022.07.039.

[26] J. A. Reiter, D. B. Spencer, and R. Linares, "Spacecraft maneuver strategy optimization for detection avoidance using reinforcement learning," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Portland, Maine, Aug. 2019, pp. 1–19.

[27] J. A. Reiter and D. B. Spencer, “Augmenting spacecraft maneuver strategy optimization for detection avoidance with competitive coevolution,” in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1910.

[28] D. Izzo, M. Märtens, and B. Pan, “A survey on artificial intelligence trends in spacecraft guidance dynamics and control,” *Astrodynamics*, vol. 3, no. 4, pp. 287–299, 2019.

[29] M. Shirobokov, S. Trofimov, and M. Ovchinnikov, “Survey of machine learning techniques in spacecraft control design,” eng, *Acta Astronautica*, vol. 186, pp. 87–97, 2021, ISSN: 0094-5765.

[30] C. Sánchez-Sánchez and D. Izzo, “Real-time optimal control via deep neural networks: Study on landing problems,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018. DOI: 10.2514/1.G002357. eprint: https://doi.org/10.2514/1.G002357. [Online]. Available: https://doi.org/10.2514/1.G002357.

[31] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Tupputo, and R. Linares, “Deep learning for autonomous lunar landing,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, 2018, pp. 1–22.

[32] L. Cheng, Z. Wang, Y. Song, and F. Jiang, “Real-time optimal control for irregular asteroid landings using deep neural networks,” *Acta Astronautica*, vol. 170, pp. 66–79, 2020, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2019.11.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576520300151.

[33] R. Furfaro, A. Scorsoglio, R. Linares, and M. Massari, “Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach,” eng, *Acta Astronautica*, vol. 171, pp. 156–171, 2020, ISSN: 0094-5765.

[34] B. Gaudet and R. Linares, “Integrated guidance and control for pinpoint mars landing using reinforcement learning,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, Aug. 2018, pp. 1–20.

[35] B. Gaudet, R. Linares, and R. Furfaro, “Deep reinforcement learning for six degree-of-freedom planetary landing,” eng, *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020, ISSN: 0273-1177.

[36] A. Scorsoglio, R. Furfaro, R. Linares, and B. Gaudet, “Image-based deep reinforcement learning for autonomous lunar landing,” in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910.

[37] X. Jiang, S. Li, and R. Furfaro, "Integrated guidance for mars entry and powered descent using reinforcement learning and pseudospectral method," *Acta Astronautica*, vol. 163, pp. 114–129, 2019, Fourth IAA Conference on Dynamics and Control of Space Systems (DYCOSS2018), ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2018.12.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576518313043.

[38] L. Cheng, Z. Wang, and F. Jiang, "Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm," eng, *Astrodynamics*, vol. 3, no. 4, pp. 375–386, 2019, ISSN: 2522-008X.

[39] B. Dachwald, "Evolutionary neurocontrol: A smart method for global optimization of low-thrust trajectories," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Providence, Rhode Island, Aug. 2004, pp. 1–16.

[40] B. Dachwald, "Optimization of very-low-thrust trajectories using evolutionary neurocontrol," *Acta astronautica*, vol. 57, no. 2, pp. 175–185, 2005, ISSN: 0094-5765.

[41] L. Cheng, Z. Wang, F. Jiang, and C. Zhou, "Real-time optimal control for spacecraft orbit transfer via multiscale deep neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 5, pp. 2436–2450, 2019. DOI: 10.1109/TAES.2018.2889571.

[42] E. Schiassi, A. DAmbrosio, K. Drozd, F. Curti, and R. Furfaro, "Physics-informed neural networks for optimal planar orbit transfers," *Journal of Spacecraft and Rockets*, 2022, Published Online: 4 Jan 2022. DOI: 10.2514/1.A35138. [Online]. Available: https://doi.org/10.2514/1.A35138.

[43] D. Izzo and E. Öztürk, "Real-time guidance for low-thrust transfers using deep neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 2, pp. 315–327, 2021. DOI: 10.2514/1.G005254.

[44] A. Zavoli and L. Federici, "Reinforcement learning for robust trajectory design of interplanetary missions," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 8, pp. 1440–1453, 2021. DOI: 10.2514/1.G005794. [Online]. Available: https://doi.org/10.2514/1.G005794.

[45] D. Miller, J. A. Englander, and R. Linares, "Interplanetary low-thrust design using proximal policy optimization," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Portland, Maine, Aug. 2019, pp. 1–16.

[46] A. Das-Stuart, K. C. Howell, and D. C. Folta, “Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies,” *Acta Astronautica*, vol. 171, pp. 172–195, 2020, ISSN: 0094-5765.

[47] A. Das-Stuart and K. Howell, “Contingency planning in complex dynamical environments via heuristically accelerated reinforcement learning,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Portland, Maine, Aug. 2019, pp. 1–21.

[48] D. Miller and R. Linares, “Low-thrust optimal control via reinforcement learning,” in *29th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, Ka’anapali, Hawaii, Jan. 2019, pp. 1–18.

[49] S. De Smet and D. J. Scheeres, “Identifying heteroclinic connections using artificial neural networks,” *Acta Astronautica*, vol. 161, pp. 192–199, Aug. 2019.

[50] N. L. Parrish and D. J. Scheeres, “Optimal low-thrust trajectory correction with neural networks,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, 2018, pp. 1–20.

[51] J. Yan, H. Yang, and S. Li, “Ann-based method for fast optimization of jovian-moon gravity-assisted trajectories in cr3bp,” *Advances in Space Research*, 2022, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2022.01.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117722000394.

[52] C. J. Sullivan, N. Bosanac, and R. L. Anderson, “Designing low-thrust transfers near earthmoon l2 via multi-objective reinforcement learning,” *Journal of Spacecraft and Rockets*, 2023. DOI: 10.2514/1.A35463.

[53] C. J. Sullivan, N. Bosanac, R. L. Anderson, and A. Mashiku, “Exploring the low-thrust transfer design space in an ephemeris model via multi-objective reinforcement learning,” in *AIAA SciTech Forum*, AIAA, San Diego, California, 2022. DOI: 10.2514/6.2022-1887. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2022-1887. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-1887.

[54] L. Federici, A. Scorsoglio, A. Zavoli, and R. Furfaro, “Autonomous guidance for cislunar orbit transfers via reinforcement learning,” in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), Aug. 2021.

[55] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares, "Guidance for closed-loop transfers using reinforcement learning with application to libration point orbits," in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910.

[56] N. B. LaFarge, K. C. Howell, and R. Linares, "A hybrid closed-loop guidance strategy for low-thrust spacecraft enabled by neural networks," in *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), Feb. 2021.

[57] N. B. LaFarge, K. C. Howell, and D. C. Folta, "Adaptive closed-loop maneuver planning for low-thrust spacecraft using reinforcement learning," in *73rd International Astronautical Congress*, IAF, Paris, France, Sep. 2022.

[58] C. J. Sullivan and N. Bosanac, "Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system," in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1910.

[59] C. J. Sullivan, N. Bosanac, R. L. Anderson, A. K. Mashiku, and J. R. Stuart, "Exploring transfers between earth-moon halo orbits via multi-objective reinforcement learning," in *2021 IEEE Aerospace Conference*, IEEE, 2021, pp. 1–13. DOI: 10.1109/AERO50100.2021.9438267.

[60] D. Guzzetti, "Reinforcement learning and topology of orbit manifolds for station-keeping of unstable symmetric periodic orbits," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Portland, Maine, Aug. 2019, pp. 1–20.

[61] A. B. Molnar, "Hybrid station-keeping controller design leveraging floquet mode and reinforcement learning approaches," M.S. thesis, Purdue University, Dec. 2020.

[62] S. Bonasera, N. Bosanac, C. J. Sullivan, I. Elliott, N. Ahmed, and J. W. McMahon, "Designing sunearth l2 halo orbit stationkeeping maneuvers via reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 2, pp. 301–311, 2023. DOI: 10.2514/1.G006783.

[63] N. B. LaFarge, K. C. Howell, and D. C. Folta, "An autonomous stationkeeping strategy for multi-body orbits leveraging reinforcement learning," in *AIAA SciTech Forum*, AIAA, San Diego, CA, Jan. 2022. DOI: 10.2514/6.2022-1764.

[64] F. E. Laipert and J. M. Longuski, "Automated missed-thrust propellant margin analysis for low-thrust trajectories," *Journal of Spacecraft and Rockets*, vol. 52, no. 4, pp. 1135–1143, 2015, ISSN: 0022-4650.

[65] A. Rubinsztejn, R. Sood, and F. E. Laipert, "Neural network optimal control in astrodynamics: Application to the missed thrust problem," eng, *Acta Astronautica*, vol. 176, pp. 192–203, 2020, ISSN: 0094-5765.

[66] P. A. Witsberger, "A neural-network-based controller for missed-thrust interplanetary trajectory design," Ph.D. dissertation, Purdue University, Apr. 2022. DOI: 10.25394/PGS.19658136.v1.

[67] G. E. Box, "Robustness in the strategy of scientific model building," in *Robustness in Statistics*, R. L. LAUNER and G. N. WILKINSON, Eds., Academic Press, 1979, pp. 201–236, ISBN: 978-0-12-438150-6. DOI: 10.1016/B978-0-12-438150-6.50018-2.

[68] V. G. Szebehely, *Theory of Orbits: The Restricted Problem of Three Bodies*, eng. New York: Academic Press, 1967, ISBN: 1-299-45434-8.

[69] C. Acton, N. Bachman, B. Semenov, and E. Wright, "A look towards the future in the handling of space science mission geometry," *Planetary and Space Science*, vol. 150, pp. 9–12, 2018, ISSN: 0032-0633. DOI: https://doi.org/10.1016/j.pss.2017.02.013.

[70] B. Park and K. C. Howell, "Leveraging intermediate dynamical models for transitioning from the circular restricted three-body problem to an ephemeris model," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Charlotte, North Carolina, Aug. 2022.

[71] J. R. Brophy, "Perspectives on the success of electric propulsion," *Journal of Electric Propulsion*, vol. 1, no. 1, 2022, ISSN: 2731-4596.

[72] A. D. Cox, "A dynamical systems perspective for preliminary low-thrust trajectory design in multi-body regimes," Ph.D. dissertation, Purdue University, 2020.

[73] H. Kuninaka, K. Nishiyama, Y. S. I. Funaki, and H. Koizumi, "Hayabusa asteroid explorer powered by ion engines on the way to earth," in *31st International Electric Propulsion Conference*, Ann Arbor, Michigan, Sep. 2009, pp. 1–6.

[74] K. Nishiyama, S. Hosoda, K. Ueno, R. Tsukizaki, and H. Kuninaka, "Development and testing of the hayabusa2 ion engine system," eng, *Transactions of the Japan Society for Aeronautical and Space Sciences*, vol. 14, no. 30, pp. 131–140, Jul. 2016, ISSN: 1884-0485.

[75] P. E. Clark, B. Malphrus, K. Brown, *et al.*, "Lunar ice cube mission: Determining lunar water dynamics with a first generation deep space cubesat," in *47th Lunar and Planetary Science Conference*, The Woodlands, Texas, Mar. 2016.

[76] N. Bosanac, A. D. Cox, K. C. Howell, and D. C. Folta, "Trajectory design for a cislunar cubesat leveraging dynamical systems techniques: The lunar icecube mission," eng, *Acta Astronautica*, vol. 144, pp. 283–296, 2018, ISSN: 0094-5765.

[77] *Bit-3 rf ion thruster*, 70010819F, Busek Co. Inc., 2019.

[78] C. Russell and C. Raymond, *The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres*, eng, 1st ed. Springer, 2012, ISBN: 1-4614-4902-2.

[79] M. D. Rayman, P. Varghese, D. H. Lehman, and L. L. Livesay, "Results from the deep space 1 technology validation mission," eng, *Acta Astronautica*, vol. 47, no. 2, pp. 475–487, 2000, ISSN: 0094-5765.

[80] T. Statler, E. Adams, E. Dotto, A. Rivkin, and A. Cheng, "Overview of the dart mission seven months to launch," in *7th IAA Planetary Defense Conference*, UNOOSA, Online Event, Apr. 2021.

[81] D. A. Herman, "Nasa's evolutionary xenon thruster (next) project qualification propellant throughput milestone: Performance, erosion, and thruster service life prediction after 450 kg," in *57th Joint Army-Navy-NASA-Air Force (JANNAF) Propulsion Meeting*, Colorado Springs, CO, Nov. 2010.

[82] W. Hart, G. M. Brown, S. M. Collins, *et al.*, "Overview of the spacecraft design for the psyche mission concept," in *2018 IEEE Aerospace Conference*, IEEE, Big Sky, Montana, 2018, pp. 1–20.

[83] J. S. Snyder, G. Lenguito, J. D. Frieman, T. W. Haag, and J. A. Mackey, "Effects of background pressure on spt-140 hall thruster performance," *Journal of Propulsion and Power*, vol. 36, no. 5, pp. 668–676, 2020, ISSN: 1533-3876.

[84] S. L. McCarty, L. M. Burke, and M. L. McGuire, "Analysis of cislunar transfers from a near rectilinear halo orbit with high power solar electric propulsion," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, 2018, pp. 1–14.

[85] C. M. Spreen, "Automated patch point placement capability for hybrid trajectory targeting," Ph.D. dissertation, Purdue University, 2017.

[86] E. Zimovan, "Characteristics and design strategies for near rectilinear halo orbits within the earth-moon system," M.S. thesis, Purdue University, May 2017.

[87] T. A. Pavlak, "Trajectory design and orbit maintenance strategies in multi-body dynamical regimes," Ph.D. dissertation, Purdue University, 2013.

[88] H. B. Keller, *Numerical solution of two point boundary value problems*, eng. Philadelphia, Pa.: Society for Industrial and Applied Mathematics, 1976, ISBN: 0-89871-021-9.

[89] A. D. Cox and E. M. Zimovan-Spreen, "Low-thrust multiple shooting parameterizations," Unpublished internal research document, Apr. 2018.

[90] N. Bosanac, "Leveraging natural dynamical structures to explore multi-body systems," Ph.D. dissertation, Purdue University, 2016.

[91] S. K. Scarritt, B. G. Marchand, A. J. Brown, W. H. Tracy, and M. W. Weeks, "Finite-burn linear targeting algorithm for autonomous path planning and guidance," eng, *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1605–1615, 2012, ISSN: 0731-5090.

[92] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *35th International Conference on Machine Learning (ICML)*, 2018. [Online]. Available: https://arxiv.org/pdf/1802.09477.pdf.

[93] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction* (Springer Series in Statistics), eng, 2nd ed. Springer, 2009, ISBN: 0-387-84857-6.

[94] C. Wilson and A. Riccardi, "Enabling intelligent onboard guidance, navigation, and control using reinforcement learning on near-term flight hardware," *Acta Astronautica*, vol. 199, pp. 374–385, 2022, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2022.07.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576522003502.

[95] S. S. Karri, *Nasa sbir 2019 phase i solicitation: Deep neural net and neuromorphic processors for in-space autonomy and cognition*, online, 2019. [Online]. Available: https://sbir.nasa.gov/printpdf/61636.

[96] G. Bersuker, M. Mason, and K. L. Jones, "Neuromorphic computing: The potential for high-performance processing in space," The Aerospace Corporation, Tech. Rep., 2018.

[97] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[98] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-dimensional continuous control using generalized advantage estimation*, 2015. arXiv: 1506.02438. [Online]. Available: https://arxiv.org/pdf/1506.02438.pdf.

[99] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. arXiv: 1602.01783. [Online]. Available: http://arxiv.org/abs/1602.01783.

[100] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, *Continuous control with deep reinforcement learning*, 2015. arXiv: 1509.02971 `[cs.LG]`. [Online]. Available: https://arxiv.org/pdf/1509.02971.pdf.

[101] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholm, Sweden: PMLR, 2018, pp. 1861–1870. [Online]. Available: http://proceedings.mlr.press/v80/haarnoja18b.html.

[102] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015.

[103] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[104] C. J. C. H. Watkins and P. Dayan, "Q-learning," eng, *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992, ISSN: 0885-6125.

[105] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, Feb. 2015.

[106] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, PMLR, 2014, pp. 387–395. [Online]. Available: https://proceedings.mlr.press/v32/silver14.html.

[107] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. arXiv: 1509.06461. [Online]. Available: http://arxiv.org/abs/1509.06461.

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

[109] A. Adams and P. Vamplew, "Encoding and decoding cyclic data," *The South Pacific Journal of Natural Science*, vol. 16, pp. 54–58, Jan. 1998.

[110] E. Blazquez, L. Beauregard, S. Lizy-Destrez, F. Ankersen, and F. Capolupo, "Rendezvous design in a cislunar near rectilinear halo orbit," eng, *Aeronautical journal*, vol. 124, no. 1276, pp. 821–837, 2020, ISSN: 0001-9240.

[111] V. Muralidharan and K. C. Howell, "Leveraging stretching directions for stationkeeping in earth-moon halo orbits," *Advances in Space Research*, 2021, Available online 22 October 2021, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2021.10.028.

[112] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[113] M. Vaquero and J. Senent, "Poincare : A multi-body, multi-system trajectory design tool," in *7th International Conference on Astrodynamics Tools and Techniques*, Oberpfaffenhofen, Germany, Nov. 2018, pp. 1–12.

[114] R. Pritchett, K. C. Howell, and D. C. Folta, "Low-thrust trajectory design for a cislunar cubesat leveraging structures from the bicircular restricted four-body problem," in *70th International Astronautical Congress*, Washington D.C., USA, Oct. 2019, pp. 1–18.

[115] C. Ocampo, "Finite burn maneuver modeling for a generalized spacecraft trajectory design and optimization system," *Annals of the New York Academy of Sciences*, vol. 1017, no. 1, pp. 210–233, 2004, ISSN: 0077-8923.

[116] B. A. Conway, "A survey of methods available for the numerical optimization of continuous dynamic systems," *Journal of Optimization Theory and Applications*, vol. 152, no. 2, pp. 271–306, Sep. 2011. DOI: 10.1007/s10957-011-9918-z.

[117] R. E. Pritchett, "Strategies for low-thrust transfer design based on direct collocation techniques," Ph.D. dissertation, Purdue University, 2020.

[118] D. Guzzetti, E. M. Zimovan, K. C. Howell, and D. C. Davis, "Stationkeeping analysis for spacecraft in lunar near rectilinear halo orbits," in *27th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, San Antonio, Texas, Feb. 2017.

[119] S. Bonasera, I. Elliott, C. J. Sullivan, N. Bosanac, N. Ahmed, and J. McMahon, "Designing impulsive station-keeping maneuvers near a sun-earth l2 halo orbit via reinforcement learning," in *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), Feb. 2021.

[120] M. Shirobokov, S. Trofimov, and M. Ovchinnikov, "Survey of station-keeping techniques for libration point orbits," eng, *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 5, pp. 1085–1105, 2017, ISSN: 0731-5090.

[121] E. M. Zimovan-Spreen, D. C. Davis, and K. C. Howell, "Recovery trajectories for inadvertent departures from an nrho," in *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), Feb. 2021.

[122] K. K. Boudad, K. C. Howell, and D. C. Davis, "Departure and escape dynamics from the near rectilinear halo orbits in the earth-moon-sun system," *The Journal of the Astronautical Sciences*, Jul. 2022. DOI: 10.1007/s40295-022-00328-w.

[123] D. C. Davis, R. J. Power, K. C. Howell, and J. P. Gutkowski, "Lunar impact probability for spacecraft in near rectilinear halo orbits," in *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), Feb. 2021.

[124] D. C. Folta, N. Bosanac, A. Cox, and K. C. Howell, "The lunar icecube mission design: Construction of feasible transfer trajectories with a constrained departure," in *26th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, Napa Valley, CA, Feb. 2016.

[125] B. Park, K. C. Howell, and D. C. Folta, "Design of low-thrust transfers from an nrho to low lunar orbits: Applications for small spacecraft," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), Aug. 2021.

[126] D. E. Lee, "Gateway destination orbit model: A continuous 15 year nrho reference trajectory," NASA Johnson Space Center, White Paper, Aug. 2019.

[127] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares, "Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment," *Acta Astronautica*, vol. 186, Sep. 2021, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2021.05.014.

[128] C. J. Sullivan, N. Bosanac, A. K. Mashiku, and R. L. Anderson, "Multi-objective reinforcement learning for low-thrust transfer design between libration point orbits," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), Aug. 2021.

[129] A. F. Haapala and K. C. Howell, "A framework for constructing transfers linking periodic libration point orbits in the spatial circular restricted three-body problem," *International Journal of Bifurcations and Chaos*, vol. 26, no. 5, 2016.

[130] G. Bozis, "Zero velocity surfaces for the general planar three-body problem," eng, *Astrophysics and Space Science*, vol. 43, no. 2, pp. 355–368, 1976, ISSN: 0004-640X.

[131] P. Coady, *Proximal Policy Optimization with Generalized Advantage Estimation*, Nov. 2018. [Online]. Available: https://github.com/pat-coady/trpo.

[132] K. Williams, B. Barden, K. Howell, M. Lo, and R. Wilson, "Genesis halo orbit stationkeeping design," in *International Symposium: Space Flight Dynamics*, Biarritz, France, Jun. 2000.

[133] C. Forbes, M. Evans, N. Hastings, and B. Peacock, "Statistical distributions," in 4th ed. John Wiley & Sons, Ltd, 2010, ch. 11, pp. 69–73. DOI: 10.1002/9780470627242.ch11.

[134] J. M. Longuski, J. E. Prussing, and J. J. Guzmán, *Optimal Control with Aerospace Applications* (Space Technology Library), eng. New York, NY: Springer New York, 2013, vol. 32, ISBN: 146148944X.

[135] C. E. Roberts, "Long term missions at the sun-earth libration point l1: Ace, soho, and wind," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Girdwood, Alaska, Aug. 2011.

[136] A. Elwood, R. Hunter, and B. Cheetham, "Cislunar autonomous positioning system technology operations and navigation experiment (capstone)," in *2021 CubeSat Developers Workshop*, Virtual, Mar. 2021.

[137] D. C. Davis, S. M. Phillips, K. C. Howell, S. Vutukuri, and B. P. McCarthy, "Stationkeeping and transfer trajectory design for spacecraft in cislunar space," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Columbia River Gorge, Stevenson, Washington, Aug. 2017, pp. 1–20.

[138] C. Simó, G. Gómez, J. Llibre, and R. Martínez, "Station keeping of a quasi-periodic halo orbit using invariant manifolds," English, in *Second International Symposium on Spacecraft Flight Dynamics*, Darmstadt, Germany, Oct. 1986, pp. 65–70.

[139] K. Howell and T. Keeter, "Station-keeping strategies for libration point orbits: Target point and floquet mode approaches," in *AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Albuquerque, New Mexico, Feb. 1995.

[140] V. Muralidharan, "Stretching directions in cislunar space: Stationkeeping and an application to transfer trajectory design," Ph.D. dissertation, Purdue University, 2021.

[141] D. Folta and F. Vaughn, "A survey of earth-moon libration orbits: Stationkeeping strategies and intra-orbit transfers," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Providence, Rhode Island, Aug. 2004. DOI: 10.2514/6.2004-4741.

[142] M. W. Lo, B. G. Williams, W. E. Bollman, *et al.*, "Genesis mission design," eng, *The Journal of the Astronautical Sciences*, vol. 49, no. 1, pp. 169–184, 2001, ISSN: 0021-9142.

[143] D. Folta, M. Woodard, and D. Cosgrove, "Stationkeeping of the first earth-moon libration orbiters: The artemis mission," in *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Girdwood, Alaska, Aug. 2011.

[144] B. Gaudet, R. Linares, and R. Furfaro, "Adaptive guidance and integrated navigation with reinforcement meta-learning," eng, *Acta Astronautica*, vol. 169, pp. 180–190, 2020, ISSN: 0094-5765.

[145] J. W. Mock and S. S. Muknahallipatna, "A comparison of ppo, td3 and sac reinforcement algorithms for quadruped walking gait generation," *Journal of Intelligent Learning Systems and Applications*, vol. 15, no. 1, Feb. 2023. DOI: 10.4236/jilsa.2023.151003.

[146] L. Capra, A. Brandonisio, and M. Lavagna, “Network architecture and action space analysis for deep reinforcement learning towards spacecraft autonomous guidance,” *Advances in Space Research*, 2022, ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2022.11.048.

[147] A. G. Barto and S. Mahadevan, “Recent advances in hierarchical reinforcement learning,” *Discrete event dynamic systems*, vol. 13, no. 1, 2003, ISSN: 0924-6703.

[148] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, “Hierarchical reinforcement learning: A comprehensive survey,” *ACM computing surveys*, vol. 54, no. 5, pp. 1–35, 2021, ISSN: 0360-0300.

[149] Y. Takubo, H. Chen, and K. Ho, “Hierarchical reinforcement learning framework for stochastic spaceflight campaign design,” *Journal of spacecraft and rockets*, vol. 59, no. 2, pp. 421–433, 2022, ISSN: 0022-4650.

[150] N. B. LaFarge, “Autonomous guidance for multi-body orbit transfers using reinforcement learning,” M.S. thesis, Purdue University, May 2020.

A. Summary of Dynamical Models

The CR3BP (Section 2.2)and $\mathcal{N}$-body ephemeris (Section 2.3) force models are employed in this investigation; the details of the implemented reference frames, evolution functions, and evolution parameters are listed in Table A.1. The low-thrust model employed in this research is derived in Section 2.4.

Table A.1. Dynamical models simulated in this investigation

	CR3BP	**$\mathcal{N}$-body ephemeris**
Reference frame		
Name	Earth-Moon rotating	Ecliptic J2000
Center	Barycenter (B)	Moon
Symbol	$\mathcal{R}$	${}^{☾}\mathcal{I}_{\text{J2000}}$
Diagram	Figure 2.3	Figure 2.6
Evolution function ($\boldsymbol{\phi}$)		
Ballistic	Equation (2.27)	Equation (2.48)
Low-thrust	Equation (2.60)	Equation (2.61)
Evolution parameter ($\boldsymbol{x}$)		
Ballistic	$\boldsymbol{\rho} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$	${}^{\mathcal{I}}\boldsymbol{x} = \begin{bmatrix} X & Y & Z & \dot{X} & \dot{Y} & \dot{Z} \end{bmatrix}^T$
Low-thrust	$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\rho}^T & m \end{bmatrix}^T$	${}^{\mathcal{I}}\boldsymbol{x}^{\text{lt}} = \begin{bmatrix} \left({}^{\mathcal{I}}\boldsymbol{x}\right)^T & m \end{bmatrix}^T$

PUBLICATIONS

Several portions of this dissertation have been previously published in other academic venues. Specifically, portions of this work on RL-enabled low-thrust G&C appear in conference papers, a journal article, and the author's master's thesis. These are categorized by their application area as follows:

Mission Recovery Scenario: NRHO Inadvertent Departure

Section 6.1

- [57] N. B. LaFarge, K. C. Howell, and D. C. Folta, "Adaptive closed-loop maneuver planning for low-thrust spacecraft using reinforcement learning," in *73rd International Astronautical Congress*, IAF, Paris, France, Sep. 2022

Mission Recovery Scenario: Heteroclinic Transfers

Sections 6.2.2 to 6.2.4 (Standalone NN-G&C)

- [127] N. B. LaFarge, D. Miller, K. C. Howell, *et al.*, "Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment," *Acta Astronautica*, vol. 186, Sep. 2021, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2021.05.014

  Portions of the above article's content appear previously as:

  - [55] N. B. LaFarge, D. Miller, K. C. Howell, *et al.*, "Guidance for closed-loop transfers using reinforcement learning with application to libration point orbits," in *20th AIAA SciTech Forum*, AIAA, Orlando, Florida, Jan. 2020. DOI: 10.2514/6.2020-1910
  - [150] N. B. LaFarge, "Autonomous guidance for multi-body orbit transfers using reinforcement learning," M.S. thesis, Purdue University, May 2020

Section 6.2.5 (NNIT)

- [56] N. B. LaFarge, K. C. Howell, and R. Linares, "A hybrid closed-loop guidance strategy for low-thrust spacecraft enabled by neural networks," in *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), Feb. 2021

Multi-body Orbit Stationkeeping Scenario

Section 6.3

- [63] N. B. LaFarge, K. C. Howell, and D. C. Folta, "An autonomous stationkeeping strategy for multi-body orbits leveraging reinforcement learning," in *AIAA SciTech Forum*, AIAA, San Diego, CA, Jan. 2022. DOI: 10.2514/6.2022-1764

VITA

Nicholas Blaine LaFarge received a Bachelor of Arts in Mathematics and Japanese from the University of Colorado Boulder in 2014 and a Master of Science in Aeronautics and Astronautics from Purdue University in 2020. After his bachelors, he spent two years working in the software industry before finding his way toward the intersection of his many interests by studying astrodynamics at Purdue. Nick participated in an internship at the Jet Propulsion Laboratory in 2017, working on an interactive tool for computing multi-body orbit connections in MONTE. Starting in 2018, Nick participated in the Graduate Pathways Co-op program at NASA Johnson Space Center where he completed three work rotations focusing on automated onboard GN&C processes for Orion. Nick was awarded a NASA Space Technology Research Fellowship (NSTRF) in 2019, and has since completed three visiting technologist experiences as NASA Goddard Space Flight Center. Through the NSTRF program, Nick's PhD research focuses on furthering the understanding and development of reinforcement learning techniques in multi-body spaceflight domains. Nick will continue his career as a mission design engineer at the Johns Hopkins University Applied Physics Laboratory (APL) in the Space Exploration Sector.