Trajectory Design Using Sampling-Based Kinodynamic Planning in Multi-Body Systems

by

Kristen Lee Bruchko

B.S., University of Illinois Urbana-Champaign, 2017

M.S., University of Colorado Boulder, 2022

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Ann and H.J. Smead Department of Aerospace Engineering Sciences

2024

Committee Members:

Prof. Natasha Bosanac, Chair

Prof. Daniel Scheeres

Prof. Morteza Lahijanian

Prof. Marcus Holzinger

Dr. Donald Dichmann

Bruchko, Kristen Lee (Ph.D., Aerospace Engineering Sciences)

Trajectory Design Using Sampling-Based Kinodynamic Planning in Multi-Body Systems

Thesis directed by Prof. Natasha Bosanac

Missions designed to explore cislunar and deep space are imperative to expanding humanity's knowledge of the solar system and our scientific understanding of the universe. However, multi-body systems have complex gravitational environments. Constructing a trajectory within chaotic systems while sufficiently exploring the solution space is time-consuming, requires an adequate preliminary solution, and expert knowledge of the environment. This becomes more complex when mission requirements and spacecraft hardware impose a variety of constraints. To overcome these challenges, the development of new innovative technologies for trajectory design in chaotic regimes is vital to enabling successful and sustainable future space exploration missions.

The focus of this research is to develop a new approach for constructing trajectories in chaotic multi-body systems using sampling-based kinodynamic motion planning. First, probabilistic roadmap generation is used to define nodes and weighted, directed edges that produce an efficient and discrete graph representation of the chaotic, continuous solution space governed by the Circular Restricted Three-Body Problem (CR3BP). Next, Dijkstra's and Yen's search algorithms are used to repeatedly query the graph to automatically construct a variety of unique initial guess trajectories for feasible, low cost transfers between periodic orbits. Lastly, each initial guess is corrected via collocation and optimized to recover a nearby continuous, end-to-end trajectory in a multi-body system. This research enables an effective and computationally feasible approach to explore the solution space and construct initial guesses while reducing the burden on a human-in-the-loop. Mission planning in cislunar and deep space is also supported via autonomous trajectory design and accommodate changing conditions or unforeseen scenarios by enabling rapid design and redesign. This trajectory design framework is demonstrated by constructing a variety of unique transfers between periodic orbits near the Moon in the Earth-Moon CR3BP.

Dedication

To my amazing husband and most loving supporter, Vasyl. I'd never be where I am today without you. I'd give you an honorary degree if I could.

Acknowledgements

First, I would like to thank my parents for their endless support and love throughout all aspects of my life and teaching me to never stop learning. I am eternally grateful to have you both as my parents. I also want to thank my sisters who always listen to me and answer my endless phone calls. And for teaching me I'm more capable than I realized, to my son William. Thank you for making me a mother.

Next, I want to thank my advisor Dr. Natasha Bosanac for her guidance throughout my years at CU and helping me explore all the wild ideas that have somehow come together in the end. I would also like to thank my past lab mates, Jack, Ian, Stefano, and Thomas, as well as my current lab mates, Renee, Giuliana, Sai, Max, and Cole, for all the laughs, advice, and thought-provoking questions about my research. You all helped me grow so much as a researcher. A special thanks to Renee for always lending a listening ear and staying so positive during the stressful times, and who never got tired looking at the same plots over and over. Additionally, I'd like to thank my friends outside of my research group, Anna, Damennick, Sam, Jesse, David, and Jordan, for making graduate school during a pandemic as amazing as it could have been. Life was never dull with all of you around.

Finally, I would like to acknowledge the NASA Space Technology Graduate Research Opportunity (NSTGRO Grant 80NSSC20K1214) for funding this research. Also thank you to my mentors, Dr. Robert Pritchett and Dr. Donald Dichmann, who have provided me endless support and amazing feedback as my NASA collaborators.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Due to the chaotic nature of the solution space in a multi-body gravitational environment, designing a trajectory that delivers a spacecraft to a specific target location subject to hardware, path, and operational constraints can be a challenging task. Current state-of-the-art approaches often rely on using dynamical systems theory for preliminary analysis of motion in a low-fidelity model such as the CR3BP. However, designing a trajectory using these approaches often relies heavily on a human with expert knowledge of the environment. Furthermore, rapid exploration of a solution space for efficient trajectory design is currently limited by the state of the art trajectory design strategies.

Motion planning algorithms are used to overcome similar challenges associated with rapidly constructing constrained trajectories and autonomously exploring a solution space in many other fields, such as robotics. Sampling-based kinodynamic motion planning relies on sampling valid configurations and paths to construct a summarized and efficient discrete graph representation of a continuous solution space. Then, graph search algorithms are used to identify various solution paths between user-defined start and goal configurations.

The work presented in this dissertation focuses on rapid trajectory design that minimizes the reliance on a human-in-the-loop. A graph representation of the continuous, chaotic solution space in the CR3BP is first constructed. Then, using graph search algorithms, the graph is queried to search for unique initial guesses for spacecraft trajectories to recover geometrically distinct transfers between periodic orbits.

1.1 Motivation

Missions designed to explore cislunar and deep space are imperative for expanding humanity's knowledge of the solar system and our scientific understanding of the universe. The future of space exploration will require regularly operating in the chaotic regimes of multi-body environments. However, these multi-body systems have complex gravitational environments while the mission parameters and spacecraft hardware impose a variety of constraints such as communication requirements, propulsion capabilities, and limitations on maneuvers. Constructing constrained trajectories within chaotic systems while sufficiently exploring the entire design space is time-consuming and often requires expert knowledge of the environment. Designing an individual, end-to-end trajectory suitable for a specific mission generally requires an adequate preliminary solution and is critical to the success and feasibility of any spacecraft mission.

Current trajectory design methods heavily rely on the intuition of mission analysts to design feasible, yet complex trajectories. Astrodynamicists currently leverage lower fidelity models such as the CR3BP as well as dynamical systems theory to gain insight into the underlying dynamics of the system. However, it is challenging and time-consuming for a human-in-the-loop to manually and sufficiently explore the complex solution space to design efficient solutions. Furthermore, if mission constraints or requirements change, redesigning the trajectory is a nontrivial process. To overcome these limitations, the development of new innovative technologies for trajectory design in chaotic regimes is vital to support future spacecraft missions required to explore and understand our solar system and the universe.

Motion planning techniques, a term used in robotics to describe finding a sequence of collision free paths that connect an initial configuration to a desired goal, often address similar challenges in the fields of robotics and artificial intelligence [68]. Given the set of all possible configurations for a robot, denoted the free configuration space, and knowledge of any obstacles in the environment or dynamics the robot must adhere to, a path can be automatically recovered using path planning techniques. In the fields of robotics, artificial intelligence, and control theory, path planning

techniques have been used to construct collision-free solution paths by planning a sequence of local paths between two configurations where all paths and configurations adhere to any environment dynamics and known constraints [45]. Path planning, also referred to as motion planning, algorithms were originally designed for simple linear environments, such as the common piano mover problem of moving a piano from one room to another without damaging anything and fitting through any doorways [45]. These algorithms have since evolved to handle more challenging problems, such as planning paths with uncertainty, adhering to physical laws, and geometric constraints. Current challenges being addressed include problems such as utilizing rovers for planetary exploration that compute the environment uncertainty with sensors or cars that travel autonomously on highways that are required to follow the car's dynamics [71]. Kinodynamic planning attempts to solve a motion planning problem subject to kinematic and dynamic constraints, e.g. finding a path from a given start position and velocity to a goal position and velocity while avoiding obstacles and respecting constraints on velocity and acceleration [22]. Most common path planning algorithms typically fall into three categories: potential functions, cell decompositions, or sampling-based algorithms. The focus for this research is on sampling-based algorithms for the flexibility in solving path planning problems in high-dimensional, complex environments [47].

Sampling-based motion planning algorithms aim to discretize the environment by constructing a summarized representation through sampling valid configurations, called nodes, and local paths or motion, called edges. Two key algorithms are rapidly-exploring random trees (RRT), a planner that grows a tree to explore the environment between two desired configurations to recover a single solution path, and probabilistic roadmap generation (PRM), a planner that first explores the environment to construct a graph that can be later searched for many solution paths. Due to the growth of the tree for a RRT algorithm and the required knowledge of a start and goal configuration a priori, it is often used for constructing a single solution path in an environment. Conversely, PRM is more useful for recovering multiple solution paths since the graph can be constructed autonomously with limited knowledge of the environment. Once this graph is constructed, a graph search algorithm such as Dijkstra's algorithm can be used to recover a variety of solution

paths between similar or unique start and goal configurations.

Using the current state of the art in trajectory design, it can be challenging to both identify a suitable itinerary for a trajectory that will optimally satisfy mission constraints and to select appropriate segments within each phase of the itinerary to construct a feasible solution. These challenges are similar to path planning problems in chaotic, dynamic environments where searching for an optimal solution path can be challenging when various constraints and dynamics along the solution path must be satisfied. By leveraging sampling-based kinodynamic planning techniques and graph search algorithms, the process of designing a trajectory can be divided and parameterized into distinct phases that are consistent for any trajectory design problem in a given system. Together, these approaches enable automatic and rapid sampling of the solution space needed to efficiently construct an initial guess for a feasible trajectory within a complex dynamical environment, such as a multi-body system. This new trajectory design model can support mission planning in cislunar and deep space and accommodate changing conditions or unforeseen scenarios by enabling rapid redesign.

1.2 Previous Contributions

1.2.1 Spacecraft Trajectory Design in Multi-Body Systems

Decades of past and future spacecraft missions have followed motions that are derived from dynamical structures admitted by the CR3BP. Early studies of the dynamics within the CR3BP by Farquhar and Breakwell, as well as the numerous subsequent studies of periodic orbit families about the Lagrange points, lead to the important use of libration point orbits for trajectory design and ultimately led to the discovery of halo orbits that enable a large number of past, present, and future successful spacecraft missions [24, 8]. Additionally, the development of ballistic lunar transfers in 1987 enabled the Japanese spacecraft Hiten to reach the Moon in 1991 and are also being used as part of the trajectory to the northern rectilinear halo orbit for the Lunar Gateway program [2, 3, 26]. Weather observation satellites such as Solar and Heliospheric Observatory

(SOHO) launched in 1995 by the European Space Agency (ESA) and the Deep-Space Climate Observatory (DSCOVR) launched in 2015 by the National Aeronautics and Space Administration (NASA) operated near the Sun-Earth Lagrange points [21, 11]. Additionally, missions such as the International Sun-Earth Explorer-3 (ISEE-3) launched in 1978 as the first mission to a halo orbit, the more recently launched James Webb Space Telescope (JWST) in 2021 designed to observe the first galaxies and supernovae, or the Lunar gateway planned for launch in 2025 as a support outpost for sustainable future human missions are designed to operate in various halo orbits [1, 29, 26].

In chaotic gravitational environments, trajectory designers frequently use dynamical systems theory to gain insight into the underlying dynamics and construct a preliminary trajectory. The bounded solutions admitted in the CR3BP such as periodic and quasi-periodic orbits as well as invariant manifolds offer potential preliminary trajectories that can be useful in designing an initial guess for a trajectory and explore the solution space [41]. For libration point missions, early analysis aimed to numerically explore the solution space and the types of trajectory arcs available within the design space through computation of invariant manifolds [30, 34]. To analyze the design space and these dynamical structures, tools such as Poincaré mapping have been studied extensively and can be expanded to analyze higher-dimensional problems [31]. However, these tools, as well as the numerous other methods, often rely on knowledge and decomposition of the design space that may be challenging for a human-in-the-loop. Furthermore, in scenarios where the CR3BP is a poor approximation of the dynamical environment, these methods may be insufficient. With the increase of missions designed to operate in chaotic multi-body systems, new trajectory design strategies that reduce the dependency on a human analyst and address these challenges are valuable.

1.2.2 Recent Applications of Motion Planning to Spacecraft Trajectory Design

To simplify, explore, and better understand chaotic multi-body systems, motion planning techniques have been applied to trajectory design scenarios. Discretizing a set of known fundamental solutions, such as periodic orbits, quasi-periodic orbits, and invariant manifolds, has been applied to database generation to simplify and summarize the solution space in various multi-body

systems [64, 25, 17]. Folta, Bosanac, Guzzetti, and Howell created a trajectory design reference catalog by characterizing families of periodic and quasi-periodic solutions for efficient orbit design [25]. Rapid search methods have been implemented by Das-Stuart, Howell, and Folta to search the accessible regions of a spacecraft with a database of possible trajectories to construct initial guesses for trajectories using graph search algorithms [17]. Additionally, Trumbauer and Villac have constructed weighted and directed graph representations from precomputed databases of known dynamical structures in lower-fidelity models, and heuristically searched the graph for transfers during on-board trajectory redesign [73]. Sampling-based motion planning, in particular a tree algorithm, has been applied for real-time, propellant-optimized autonomous spacecraft trajectory generation by Starek et. al. [67]. In the two-body problem, Parrish computed near-optimal paths from a graph-based approach to enable a continuous-thrust trajectory optimization framework [57]. Using Dijkstra's algorithm, Tsirogiannis constructed maneuver-enabled transfers between periodic orbits in the Saturn-Titan system using a discretization of the solution space [74]. More recently, Deka and McMahon used a tree-based kinodynamic motion planning approach for safe relative spacecraft motion in a classical two-body model [18]. Additionally, Smith and Bosanac have used clustering to generate a set of motion primitives and a graph that reflects their potential for composability; searching these graphs to produce primitive sequences supports rapid trajectory design and design space exploration [64]. While these applications apply various motion planning techniques to trajectory design problems, a majority of these methods rely on a human-in-the-loop to construct a sufficient representation of the solution space by defining a discretized database of known fundamental solutions or potential trajectory motion.

1.3 Dissertation Overview

Mission analysts aim to optimize multiple design objectives, such as propellant usage and time of flight, while adhering to mission constraints. However, in a chaotic environment, the complexity of the nonlinear dynamics due to the gravitational forces from multiple bodies make global optimization difficult to achieve. To reduce the complexity of exploring the design space for a variety

of feasible solutions and the overall burden on a human-in-the-loop, the goal of this dissertation is to develop a new approach for constructing initial guesses for complex trajectories in chaotic multi-body systems by using sampling-based kinodynamic planning techniques. Probabilistic roadmap generation and graph theory are used to construct an efficient, discrete graph representation that summarizes a desired portion of the continuous, chaotic solution space in the CR3BP. This graph representation is then rapidly and repeatedly searched using graph search algorithms to automatically recover a variety of unique initial guesses for spacecraft trajectories without heavy reliance on a human-in-the-loop. These initial guesses are then corrected via collocation and optimized for propellant usage to produce geometrically distinct trajectories between user-defined initial and final configurations along periodic orbits.

1.3.1 Organization

The research presented in this dissertation is organized as follows:

Chapter 2: The dynamical model used in this research is the circular restricted three body problem. This model approximates the motion of a spacecraft under the gravitational influence of two primary bodies and admits a variety of fundamental solutions advantageous for preliminary trajectory design. This chapter presents a detailed derivation of this model and methods for computing fundamental solutions such as equilibrium points, periodic orbits, and hyperbolic invariant manifolds. Additionally, the corrections methods used to compute and recover continuous families of periodic orbits are presented.

Chapter 3: To recover continuous trajectories in a multi-body system, a two-point boundary problem is often solved. In this work, collocation is used to numerically correct discontinuous initial guesses for trajectories in the circular restricted three body problem. Then, the corrected solutions are optimized to minimize the total maneuver costs using sequential quadratic programming. The algorithms and derivations necessary for these methods are detailed in this chapter.

Chapter 4: Kinodynamic planning techniques are leveraged to plan a collision-free path, a sequence of valid configurations and local motion between configurations, in an environment with known obstacles and dynamic constraints. This chapter introduces probabilistic roadmap generation, a sampling-based path planning algorithm used to construct a discrete graph of an environment, called the roadmap. This roadmap is then rapidly searched using graph search methods to produce a variety of unique solution paths. These algorithms and the accompanying graph theory that are leveraged in this dissertation for initial guess construction for trajectory design are presented in detail.

Chapter 5: This chapter provides the framework for using sampling-based kinodynamic planning to construct unique transfers in the CR3BP. The planning and graph search algorithms as well as the required parameters necessary to adapt these techniques for use in a chaotic solution space are discussed and analyzed in detail. This framework is extensively demonstrated in a planar transfer design scenario by exploring natural and impulsive trajectories between an L_1 and L_2 Lyapunov orbit.

Chapter 6: In this chapter, multiple transfer design scenarios in the lunar regime are presented using the trajectory design framework presented in Chapter 5. Additional transfers are further explored for the planar Lyapunov design scenario presented in Chapter 5. Next, transfers between Lyapunov orbits at distinct energy levels are generated to explore the solution space across multiple Jacobi constants. Finally, a roadmap is constructed to summarize spatial motion to recover transfers between halo orbits at a single energy level.

Chapter 7: A summary of the methods and technical results completed in this dissertation are presented. Recommendations for future work and potential applications for utilizing kinodynamic planning and graph search methods for trajectory design in multi-body systems are also presented.

1.3.2 Contributions

The technical contributions to the community are summarized as follows:

(1) Development of methodology for constructing a discrete graph representation of a segment of the continuous, chaotic solution space modeled by the circular restricted three body problem using sampling-based kinodynamic planning. The graph is constructed to capture natural flow through local regions of the solution space while limiting the number of configurations sampled and local paths constructed, creating an efficient representation the reduces the reliance on a human-in-the-loop. This contribution enables a rapid and computationally feasible approach to automatically explore the solution space by capturing viable states in regions of interest with distinct sensitivity and discretely representing their connectivity.

(2) A sampling-based kinodynamic planner is formulated for initial guess construction for transfers between libration point orbits in a multi-body system using graph search methods. Using Dijkstra's search algorithm and Yen's search algorithm with the addition of a dissimilarity measure, unique initial guesses for trajectories are recovered automatically. These initial guesses are then corrected via collocation and optimized with respect to specific trajectory design parameters, such as maneuver capabilities and time of flight, to produce geometrically distinct transfers between boundary conditions. Additional unique initial guesses are recovered for distinct boundary conditions using the same graph, reducing the computational time required to explore the same solution space. This enables a more efficient, rapid exploration of the trajectory design space that reduces the required knowledge and burden of a human trajectory analyst.

(3) Demonstration of combining motion planning techniques and a robust corrections method for recovering natural and locally optimal maneuver-enabled trajectories in multiple trajectory design scenarios in the lunar regime.

Chapter 2

Dynamical Model and Fundamental Solutions

Trajectory design and initial guess construction often use a variety of dynamical models and numerical methods. This research leverages the CR3BP as a low fidelity model and dynamical systems theory to support verification of the trajectories constructed in the later chapters. First, the equations of motion are presented for the CR3BP that model the approximate motion of a spacecraft under the gravitational influence of two primary bodies. In the CR3BP, a variety of fundamental solutions exist, including five equilibrium points, continuous families of periodic and quasi-periodic orbits, and their associated hyperbolic invariant manifolds [41]. These natural transport mechanisms are often used as the basis for constructing initial guesses for complex trajectories. The methods used to compute these dynamical structures are presented in this chapter.

2.1 Circular Restricted Three-Body Problem

The CR3BP is an autonomous dynamical model that approximates the motion of a third body P_3, commonly a spacecraft, under the gravitational influence of two primary bodies, P_1 and P_2. The two larger primary bodies are assumed to have constant mass with $\tilde{M}_1 > \tilde{M}_2$. In this derivation, the tilde notation represents a dimensional quantity. The spacecraft is assumed to possess a comparatively negligible mass and as such, does not influence the motion of the primaries. These two primaries are also assumed to travel on circular orbits about their mutual barycenter that lies on the line between P_1 and P_2 [41].

Newtonian mechanics and a rotating reference frame centered around the system's origin is used to derive a set of autonomous equations for the motion of the spacecraft. Figure 2.1 displays the configuration and orbit of each primary, shown as the Earth and Moon, and the spacecraft in the inertial reference frame, $\{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{Z}}\}$, and the rotating reference frame, $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$, with both origins at the system's barycenter. The position vector of P_1 and P_2 with respect to the origin are written as $\tilde{\boldsymbol{R}}_1 = \tilde{X}_1\hat{\boldsymbol{X}} + \tilde{Y}_1\hat{\boldsymbol{Y}} + \tilde{Z}_1\hat{\boldsymbol{Z}}$ and $\tilde{\boldsymbol{R}}_2 = \tilde{X}_2\hat{\boldsymbol{X}} + \tilde{Y}_2\hat{\boldsymbol{Y}} + \tilde{Z}_2\hat{\boldsymbol{Z}}$, respectively. The position vector of P_3 with respect to the origin is written as $\tilde{\boldsymbol{R}}_3 = \tilde{X}_3\hat{\boldsymbol{X}} + \tilde{Y}_3\hat{\boldsymbol{Y}} + \tilde{Z}_3\hat{\boldsymbol{Z}}$. The position of the spacecraft with respect to the system's origin, $\tilde{\boldsymbol{R}}_3$, can move freely within the system and the position of the spacecraft with respect to each primary is calculated as $\tilde{\boldsymbol{R}}_{i,3} = (\tilde{X}_3 - \tilde{X}_i)\hat{\boldsymbol{X}} + (\tilde{Y}_3 - \tilde{Y}_i)\hat{\boldsymbol{Y}} + (\tilde{Z}_3 - \tilde{Z}_i)\hat{\boldsymbol{Z}}$, $i = [1, 2]$ in the inertial frame.

2.1.1 Equations of Motion

The derivation begins by writing the scalar potential function for P_3, per unit mass $\tilde{M}_3$, in the inertial frame

$$\tilde{U}_3 = \frac{\tilde{G}\tilde{M}_1}{\tilde{R}_{1,3}} + \frac{\tilde{G}\tilde{M}_2}{\tilde{R}_{2,3}} \tag{2.1}$$

where $\tilde{G}$ is the universal gravitational constant equal to $6.674 \times 10^{-11}\ m^3\ kg^{-1}\ s^{-2}$. Using Newton's second law of motion, the force per unit mass acting on the spacecraft, $\boldsymbol{F}_3$, is equal to the gradient of the potential function differentiated with respect to an inertial observer, denoted by $(\cdot)''$, such

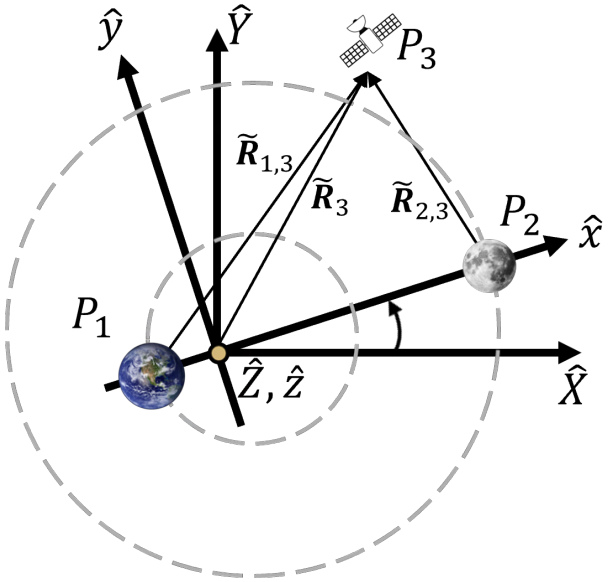


Figure 2.1: Configuration of P_1, P_2, and P_3 in the inertial, $\{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{Z}}\}$, and rotating, $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$, reference frames for the CR3BP.

that

$$\boldsymbol{F}_3 = \tilde{\boldsymbol{R}}_3 = \boldsymbol{\nabla}\tilde{U}_3 \tag{2.2}$$

The partial derivatives of $\tilde{U}_3$ with respect to $\tilde{\boldsymbol{R}}_3$ can be written in each of the inertial reference frame components as

$$\frac{\partial \tilde{U}_3}{\partial \tilde{X}} = \frac{\partial}{\partial \tilde{X}}\left(\frac{\tilde{G}\tilde{M}_1}{\tilde{R}_{1,3}} + \frac{\tilde{G}\tilde{M}_2}{\tilde{R}_{2,3}}\right) = -\left(\frac{\tilde{G}\tilde{M}_1(\tilde{X}_3 - \tilde{X}_1)}{\tilde{R}_{1,3}^3} + \frac{\tilde{G}\tilde{M}_2(\tilde{X}_3 - \tilde{X}_2)}{\tilde{R}_{2,3}^3}\right) \tag{2.3}$$

$$\frac{\partial \tilde{U}_3}{\partial \tilde{Y}} = \frac{\partial}{\partial \tilde{Y}}\left(\frac{\tilde{G}\tilde{M}_1}{\tilde{R}_{1,3}} + \frac{\tilde{G}\tilde{M}_2}{\tilde{R}_{2,3}}\right) = -\left(\frac{\tilde{G}\tilde{M}_1(\tilde{Y}_3 - \tilde{Y}_1)}{\tilde{R}_{1,3}^3} + \frac{\tilde{G}\tilde{M}_2(\tilde{Y}_3 - \tilde{Y}_2)}{\tilde{R}_{2,3}^3}\right) \tag{2.4}$$

$$\frac{\partial \tilde{U}_3}{\partial \tilde{Z}} = \frac{\partial}{\partial \tilde{Z}}\left(\frac{\tilde{G}\tilde{M}_1}{\tilde{R}_{1,3}} + \frac{\tilde{G}\tilde{M}_2}{\tilde{R}_{2,3}}\right) = -\left(\frac{\tilde{G}\tilde{M}_1(\tilde{Z}_3 - \tilde{Z}_1)}{\tilde{R}_{1,3}^3} + \frac{\tilde{G}\tilde{M}_2(\tilde{Z}_3 - \tilde{Z}_2)}{\tilde{R}_{2,3}^3}\right) \tag{2.5}$$

By applying the gradient to Equation 2.2, this derivation produces 6 equations of motion for P_3 (as well as 6 equations of motion for P_1 and P_2) in the inertial frame with respect to an inertial observer that are written as

$$\tilde{X}'' = -\frac{\tilde{G}\tilde{M}_1(\tilde{X}_3 - \tilde{X}_1)}{\tilde{R}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{X}_3 - \tilde{X}_2)}{\tilde{R}_{2,3}^3} \tag{2.6}$$

$$\tilde{Y}'' = -\frac{\tilde{G}\tilde{M}_1(\tilde{Y}_3 - \tilde{Y}_1)}{\tilde{R}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{Y}_3 - \tilde{Y}_2)}{\tilde{R}_{2,3}^3} \tag{2.7}$$

$$\tilde{Z}'' = -\frac{\tilde{G}\tilde{M}_1(\tilde{Z}_3 - \tilde{Z}_1)}{\tilde{R}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{Z}_3 - \tilde{Z}_2)}{\tilde{R}_{2,3}^3} \tag{2.8}$$

Modeling the motion for each body P_1, P_2, and P_3, produces a system of 18 differential equations. However, only 10 integrals of motion exist: 6 from the conservation of linear momentum, 3 from the conservation of angular momentum, and 1 from the conservation of mechanical energy. As a result, no analytical solution exists due to requiring 18 integrals of motion.

Characteristic length, mass, and time quantities are introduced for nondimensionalization and one governing system parameter is introduced [41, 69]. The characteristic length, l^*, is selected as the assumed constant distance between the two primaries:

$$l^* = \tilde{R}_1 + \tilde{R}_2$$

where $\tilde{R}_i$ is the distance from the system barycenter to body P_i. This formulation of the characteristic length along with the primaries orbit provides the constant nondimensional distance between the primaries equal to unity in the rotating frame. The characteristic mass quantity, m^*, is equal to the total mass of the system:

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

Using the characteristic mass, the mass parameter of the system, μ, is defined as:

$$\mu = \frac{\tilde{M}_2}{m^*} \tag{2.9}$$

where the nondimensional mass value for P_1 and P_2 are then equal to $(1-\mu)$ and μ, respectively. Since the motion of the spacecraft does not influence the motion of the primaries, the primaries can be modeled as an isolated two-body problem allowing time to be nondimensionalized such that the mean motion of the primaries is equal to unity:

$$t^* = \left(\frac{(\tilde{R}_1 + \tilde{R}_2)^3}{\tilde{G}(\tilde{M}_1 + \tilde{M}_2)}\right)^{1/2} = \left(\frac{l^{*3}}{\tilde{G}m^*}\right)^{1/2}$$

The characteristic quantities and mass parameter computed for the Earth-Moon system are provided in Table 2.1 [52, 53]. Using this nondimensionalization scheme and the characteristic quan-

Table 2.1: Characteristic quantities and mass parameter for the Earth-Moon CR3BP.

μ	l^* (km)	m^* (kg)	t^* (s)
1.215059×10^{-2}	384400	6.047176×10^{24}	3.751903×10^{5}

tities, Eqs. 2.6- 2.8 can be rewritten as:

$$X'' = -\frac{(1-\mu)(X-X_1)}{R_{1,3}^3} - \frac{\mu(X-X_2)}{R_{2,3}^3} \tag{2.10}$$

$$Y'' = -\frac{(1-\mu)(Y-Y_1)}{R_{1,3}^3} - \frac{\mu(Y-Y_2)}{R_{2,3}^3} \tag{2.11}$$

$$Z'' = -\frac{(1-\mu)(Z-Z_1)}{R_{1,3}^3} - \frac{\mu(Z-Z_2)}{R_{2,3}^3} \tag{2.12}$$

Using the assumption that each primary travels on a circular orbit in the orbit plane about the system's barycenter, the position of each primary is known at all times. Equations 2.10- 2.12 are then rewritten to only depend on the position and velocity of P_3:

$$X'' = -\frac{(1-\mu)(X+\mu\cos(t))}{R_{1,3}^3} - \frac{\mu(X-(1-\mu)\cos(t))}{R_{2,3}^3} \tag{2.13}$$

$$Y'' = -\frac{(1-\mu)(Y+\mu\sin(t))}{R_{1,3}^3} - \frac{\mu(Y-(1-\mu)\sin(t))}{R_{2,3}^3} \tag{2.14}$$

$$Z'' = -\frac{(1-\mu)Z}{R_{1,3}^3} - \frac{\mu Z}{R_{2,3}^3} \tag{2.15}$$

However, these nondimensional equations of motion for P_3 written in the inertial frame are still time-dependent.

To recover autonomous equations of motion for the CR3BP, a rotating coordinate system is defined with the origin at the system's barycenter and axes $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$: $\hat{\boldsymbol{x}}$ is directed from P_1 to P_2, $\hat{\boldsymbol{z}}$ is aligned with the system's orbital angular momentum vector, and $\hat{\boldsymbol{y}}$ completes the right-handed orthogonal triad [41]. The position of P_3 can then be written in the rotating frame as a transformation of the nondimensional inertial quantities:

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

The fixed nondimensional positions of P_1 and P_2 located on the $\hat{\boldsymbol{x}}$-axis are then equal to $-\mu$ and $(1-\mu)$, respectively. The nondimensional position of P_3 can then be expressed in the rotating frame with respect to each primary as $\boldsymbol{r}_1 = (x+\mu)\hat{\boldsymbol{x}} + y\hat{\boldsymbol{y}} + z\hat{\boldsymbol{z}}$ and $\boldsymbol{r}_2 = (x-1+\mu)\hat{\boldsymbol{x}} + y\hat{\boldsymbol{y}} + z\hat{\boldsymbol{z}}$. Using Eqs. 2.13- 2.15, the motion of P_3 can be expressed in the rotating frame with respect to an inertial observer:

$$X'' = -\frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-(1-\mu))}{r_2^3} \tag{2.16}$$

$$Y'' = -\frac{(1-\mu)(y+\mu)}{r_1^3} - \frac{\mu(y-(1-\mu))}{r_2^3} \tag{2.17}$$

$$Z'' = -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} \tag{2.18}$$

where the nondimensional distances of P_3 from P_1 and P_2 are $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}$, respectively.

The equations of motion for P_3 in the rotating frame with respect to an observer in the rotating frame can be obtained using transport theorem. First, the time derivative of the position in the inertial frame can be written as:

$$\boldsymbol{r}' = \dot{\boldsymbol{r}} + \boldsymbol{\omega}_{R/I} \times \boldsymbol{r} \tag{2.19}$$

where a time derivative with respect to an observer in the rotating frame is denoted by $\dot{(\cdot)}$ and $\boldsymbol{\omega}$ is the angular velocity of the rotating frame, R, with respect to the inertial frame, I. Since the direction of the angular velocity in the rotating frame is aligned with the $\hat{\boldsymbol{z}}$-axis and the mean motion of the system was defined such that it is equal to unity, the angular velocity rotation can be defined as:

$$\boldsymbol{\omega}_{R/I} = 1\hat{\boldsymbol{z}} \tag{2.20}$$

As a result of the expansion of Eqn. 2.19, the second derivative reveals the acceleration of P_3 with respect to an inertial observer as:

$$\boldsymbol{r}'' = (\ddot{x} - 2\dot{y} + x)\hat{\boldsymbol{x}} + (\ddot{y} + 2\dot{x} - y)\hat{\boldsymbol{y}} + \ddot{z}\hat{\boldsymbol{z}} \tag{2.21}$$

Finally, equating Eqn. 2.21 along with Eqns. 2.16- 2.18 recover the equations of motion of P_3 in the CR3BP and are written as:

$$\ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} \tag{2.22}$$

$$\ddot{y} = -2\dot{x} + y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \tag{2.23}$$

$$\ddot{z} = -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} \tag{2.24}$$

By defining a nondimensional pseudo-potential function, U^*, the equations may be written in a compact form as:

$$\ddot{x} = 2\dot{y} + \frac{\partial U^*}{\partial x}, \qquad \ddot{y} = -2\dot{x} + \frac{\partial U^*}{\partial y}, \qquad \ddot{z} = \frac{\partial U^*}{\partial z} \tag{2.25}$$

where

$$U^* = \frac{1}{2}(x^2 + y^2) + \frac{(1-\mu)}{r_1} + \frac{\mu}{r_2} \tag{2.26}$$

While there is still no analytical solution to this system of equations, these three second-order differential equations can be transformed into six first-order differential equations. Then, given an initial state $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$, the motion of P_3 in the CR3BP can be computed numerically using Eqn. 2.25.

2.1.2 Jacobi Constant

The CR3BP admits an integral of motion in the rotating frame, commonly referred to as the Jacobi constant, due to the autonomous Hamiltonian structure of the equations of motion. To derive the Jacobi constant from Equations 2.25, the dot product of the acceleration and velocity vectors of P_3 in the rotating frame equals:

$$\begin{aligned} \ddot{x}\dot{x} + \ddot{y}\dot{y} + \ddot{z}\dot{z} &= \left(2\dot{y} + \frac{\partial U^*}{\partial x}\right)\dot{x} + \left(-2\dot{x} + \frac{\partial U^*}{\partial y}\right)\dot{y} + \frac{\partial U^*}{\partial z}\dot{z} \\ &= \dot{x}\frac{\partial U^*}{\partial x} + \dot{y}\frac{\partial U^*}{\partial y} + \dot{z}\frac{\partial U^*}{\partial z} \end{aligned} \tag{2.27}$$

which when integrated is equivalent to

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

Since the pseudo-potential function is autonomous, its derivation with respect to time is always zero. Thus, the Jacobi constant is a constant of integration and Equation 2.28 can be rewritten to recover the equation for the Jacobi constant, C_J:

$$\begin{aligned} C_J &= 2U^* - v^2 \\ &= (x^2 + y^2) + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - \dot{x}^2 - \dot{y}^2 - \dot{z}^2 \end{aligned} \tag{2.29}$$

where $v^2 = \dot{x}^2 + \dot{y}^2 + \dot{z}^2$. Analyzing this expression reveals an inversely proportional relationship between C_J and the total energy of P_3. For natural motion in the CR3BP, this expression remains constant along a single trajectory and supplies insight into the accessible regions of the system.

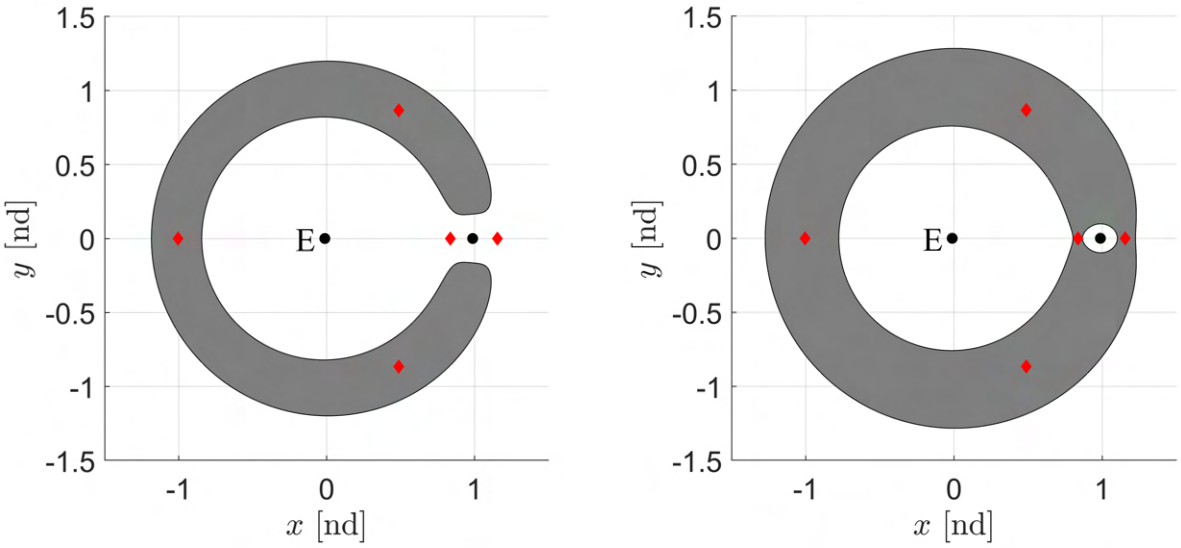
2.1.3 Zero Velocity Surfaces

At a single energy level in the CR3BP, the motion of P_3 can be divided into allowable and forbidden regions that are categorized by real and imaginary velocities. Allowable positions in the configuration space are bounded by surfaces when $\dot{x} = \dot{y} = \dot{z} = 0$, that is the when the velocity transitions from real to imaginary, with a nonzero acceleration. These boundaries, called a zero velocity surface (ZVS), at a given energy level are computed given the following:

$$\left\{ (x, y, z) \in \mathbb{R}^3 \mid 0 = (x^2 + y^2) + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - C_J \right\} \tag{2.30}$$

This infinite set of position vectors with zero velocity form a surface (or curve) in three-(two-)dimensional configuration space. Analyzing these surfaces can provide valuable insight into the potential motion of P_3 at various energy levels. Figure 2.2 depicts the zero velocity curve (ZVC) for two distinct energy levels for the Earth-Moon CR3BP, where the equilibrium points are shown as red diamonds and the primaries are shown as black circles. These surfaces depict the forbidden regions in grey for a spacecraft that possess a specific Jacobi constant and can be useful in trajectory design by characterizing where natural motion is allowable. By analyzing the ZVC displayed in Figure 2.2a, a spacecraft near the Earth must pass through the lunar region in order to reach the exterior region of the ZVC. On the contrary, the ZVS in Figure 2.2b show that a spacecraft with

a higher Jacobi constant near the Earth cannot reach the lunar or exterior region with natural motion, its energy must first be altered to access new allowable configurations.



(a) Zero velocity curve at $C_J = 3.10$.

(b) Zero velocity curve at $C_J = 3.20$.

Figure 2.2: Zero velocity curves depicting the allowable (white) and forbidden (gray) regions in the xy-plane of the Earth-Moon CR3BP at two distinct energy levels.

2.2 Fundamental Solutions

In chaotic gravitational environments, trajectory designers frequently use dynamical systems theory to gain insight into the underlying dynamics and to construct a preliminary trajectory. In the CR3BP, a variety of fundamental solutions exist, including five equilibrium points, continuous families of periodic and quasi-periodic orbits, and their associated hyperbolic invariant manifolds, that can be computed using dynamical systems theory [41]. Selected segments along these natural transport mechanisms are often used as the basis for constructing initial guesses for complex trajectories in the CR3BP and can be crucial to successful mission planning.

2.2.1 Equilibrium Points

In the CR3BP, there are five equilibrium points, or Lagrange points, that lie in the plane of the motion of the primaries: three collinear points that lie along the $\hat{x}$ axis, commonly denoted L_1, L_2, and L_3, and two triangular points that form an equilateral triangle with the primaries, commonly denoted L_4 and L_5. These points in an autonomous dynamical model correspond to stationary points in the pseudo-potential function where the time derivative of the state is equal to the zero vector [69]. As such, a spacecraft modeled by the CR3BP dynamics that is located at an equilibrium point will stay there indefinitely until acted on by an external force.

The locations of the five equilibrium points in the CR3BP are calculated by setting the velocity and acceleration to zero in Equation 2.25. This result confirms the equilibrium points are the critical points in the pseudo-potential function where $\partial U^*/\partial \boldsymbol{r} = \mathbf{0}$. The locations of each point solve the following equations:

$$\frac{\partial U^*}{\partial x} = x - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} = 0 \tag{2.31}$$

$$\frac{\partial U^*}{\partial y} = y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} = 0 \tag{2.32}$$

$$\frac{\partial U^*}{\partial z} = -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} = 0 \tag{2.33}$$

By inspection of Equation 2.33 in order for acceleration to be the zero vector, $z_i = 0 \, \forall L_i \in [1, 5]$; this places all equilibrium points in the plane of the motion of the primaries. The x and y coordinates for each equilibrium point may be computed by examining two potential cases: when $y = 0$ and $y \neq 0$.

To compute the x and y coordinates for the collinear points, y is set to zero. Assuming this, the x-coordinate can be found by examining the roots of Equation 2.31, which is simplified to:

$$0 = x - \frac{(1-\mu)(x+\mu)}{((x+\mu)^2)^{\frac{3}{2}}} - \frac{\mu(x-1+\mu)}{((x-1+\mu)^2)^{\frac{3}{2}}} = f(x) \tag{2.34}$$

Upon inspection, there are three intervals where $f(x) = 0$:

$$(x+\mu) > 0, (x-1+\mu) > 0$$

$$(x+\mu) < 0, (x-1+\mu) < 0$$

$$(x+\mu) > 0, (x-1+\mu) < 0$$

Since Equation 2.34 cannot be solved for analytically, 3 solutions are numerically calculated using a root-finding method given an initial guess. Three different regions can be used to select three distinct initial guesses for x in the CR3BP: 1) $-\mu < x < 1-\mu$, 2) $x > (1-\mu)$, and 3) $x < -\mu$. By solving Equation 2.34 for a root in each region, the x-coordinate of each of the collinear points is computed.

To calculate the location of the triangular equilibrium points, the case of $y \neq 0$ is explored by examining the system of equations given by Equations 2.31- 2.32. For this equation to be solved, $r_1 = r_2 = 1$ which means the L_4 and L_5 equilibrium points form an equilateral triangle with the primaries and are symmetric across the x-axis. Therefore, the location of the triangular equilibrium points is given by:

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

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

where L_4 is located at the positive y-coordinate and L_5 is located at the negative y-coordinate. For the Earth-Moon system, the xy-coordinates of the five equilibrium points in the CR3BP are listed in Table 2.2 and are depicted by red diamonds in the xy-plane of the rotating frame in Figure 2.3. The location of the Earth is depicted by a grey circle at $x = -\mu$ with annotation 'E' while the location of the Moon is depicted by a grey circle at $x = 1-\mu$ with annotation 'M'.

Table 2.2: Location of the equilibrium points in the Earth-Moon CR3BP.

Equilibrium Point	$x[nd]$	$y[nd]$
L_1	0.836915127047076	0
L_2	1.155682164448510	0
L_3	−1.005062645702342	0
L_4	0.487849414649438	0.866025403784439
L_5	0.487849414649438	−0.866025403784439

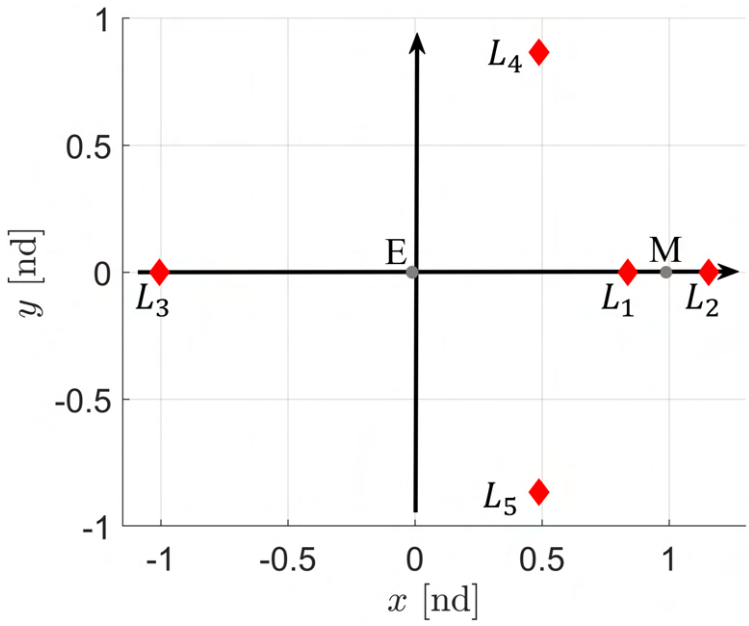


Figure 2.3: Location of the five equilibrium points in the Earth-Moon CR3BP.

2.2.2 Periodic Orbits

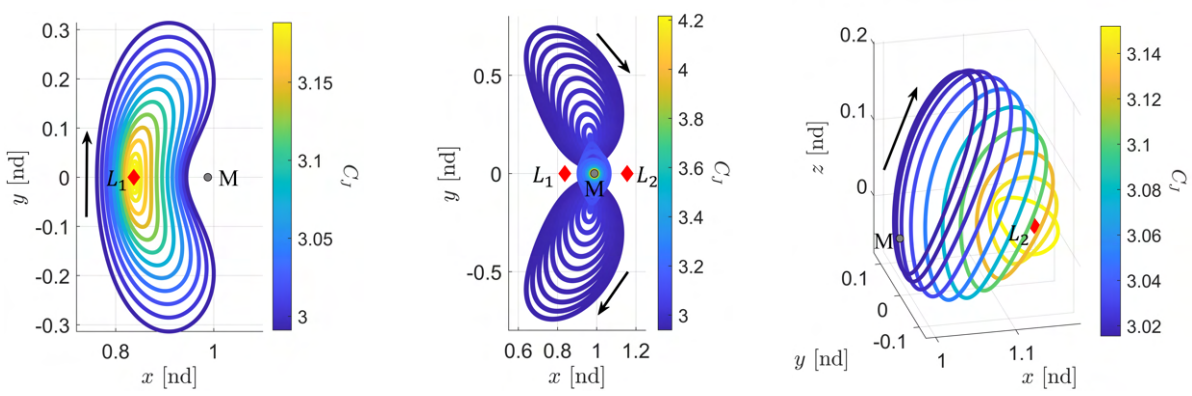
Periodic orbits are paths that repeat exactly after one orbital period in the rotating frame and exist in continuous, single-parameter families throughout the system. Periodic orbits generated in the CR3BP are often used in trajectory design because, in many cases, they indicate the existence of similar bounded motions in the rotating frame in higher-fidelity models. A one-dimensional periodic orbit satisfies the following condition:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t+T) \tag{2.35}$$

where $\boldsymbol{x}$ is an initial state and T is the orbital period of the orbit. In the CR3BP, an infinite variety of periodic orbits emanate from stationary solutions in the system or bifurcate from other periodic orbit families [59].

A variety of periodic orbit families exist centered around the equilibrium points, called libration point orbits. These periodic orbits can be characterized by parameters such as Jacobi constant or stability, and vary geometrically in the configuration space. As such, computing and understanding these periodic orbits are necessary for trajectory design for future complex space

missions [41]. Examples of these periodic orbit families near the equilibrium points include Lyapunov, halo, vertical, and axial orbits. Lyapunov orbits are planar periodic orbits, while halo and vertical orbits are spatial orbits. Furthermore, halo orbits are symmetric about the xz plane and are denoted northern and southern orbits; the southern orbit family results for $z_0 < 0$ while the northern orbit family results for $z_0 > 0$ for the initial state vector of each orbit [27]. Axial orbits bifurcate from a member of the Lyapunov orbit family and a member of the vertical orbit family, and exist as a continuous family in between these two bifurcations. Additionally, there are orbit families that are primary-centered orbits, meaning they orbit around one of the primaries in the system. Families such as distant prograde orbits and distant retrograde orbits are centered around P_2. Near the triangular equilibrium points, there are planar periodic orbit families, such as $L_{4,5}$ short or long period orbits. Additional resonant orbit families also exist within the CR3BP. Figure 2.4 depicts selected members of three periodic orbit families near the Moon in the Earth-Moon CR3BP. These orbits are colored by Jacobi constant. To compute periodic orbits in the CR3BP, a multiple-shooting corrections algorithm is used by constructing a free variable and constraint vector formulation described in the following section.



(a) L_1 Lyapunov orbits (b) Distant prograde orbits (c) L_2 Northern halo orbits

Figure 2.4: Selected members of families of periodic orbits in the Earth-Moon CR3BP.

2.2.2.1 Differential Corrections

To compute periodic orbits in the CR3BP from an initial guess, a multiple-shooting corrections algorithm is used to solve a two-point boundary value problem [41]. First, an initial guess trajectory computed from an initial state $\boldsymbol{x}_1$ and orbital period T is evenly discretized in time into n arcs. The initial state along each arc as well as the integration time along an arc are combined to construct the free variable vector. The initial state on the i-th arc, $\boldsymbol{x}_i$, is defined via discretization of the initial guess. The time along each arc, assuming each arc has the same integration time, is defined as

$$\Delta t = T/n$$

This decomposes the initial guess into n arcs, each with the same integration time. The state vector at each node and integration time along an arc are used to define a free variable vector, $\boldsymbol{X}$, as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T & \boldsymbol{x}_2^T & \dots & \boldsymbol{x}_n^T & \Delta t \end{bmatrix}^T \tag{2.36}$$

that has dimensions $6n + 1$. Next, to form the constraint vector, continuity, periodicity, and energy level constraints are constructed to recover a continuous periodic orbit at a specified Jacobi constant. First, state continuity constraints are enforced between neighboring arcs:

$$\boldsymbol{F}_c = \begin{bmatrix} \boldsymbol{x}_1(\Delta t) - \boldsymbol{x}_2 \\ \boldsymbol{x}_2(\Delta t) - \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_{n-1}(\Delta t) - \boldsymbol{x}_n \end{bmatrix}^T \tag{2.37}$$

where $\boldsymbol{x}_i(\Delta t)$ is equal to the final state on the i-th arc by integrating the initial state, $\boldsymbol{x}_i$, for time Δt. Next, a periodicity constraint is enforced to ensure continuity between the final and initial arc. Since the Jacobi constant is conserved for a natural trajectory, only five state components are required to ensure full state continuity. Additionally, to constrain the initial state to the y-axis, a constraint on the first node y-coordinate is enforced. This periodicity constraint vector is then

defined as:

$$\boldsymbol{F}_{PO} = \begin{bmatrix} x_n(\Delta t) - x_1 \\ y_n(\Delta t) - y_1 \\ z_n(\Delta t) - z_1 \\ \dot{x}_n(\Delta t) - \dot{x}_1 \\ \dot{z}_n(\Delta t) - \dot{z}_1 \\ y_1 \end{bmatrix} \tag{2.38}$$

Last, a constraint on Jacobi constant may be included in the constraint vector formulation to compute an orbit at a specific energy level as

$$\boldsymbol{F}_{C_J} = \begin{bmatrix} C_J(\boldsymbol{x_1}) - C_{J,desired} \end{bmatrix} \tag{2.39}$$

where $C_J(\boldsymbol{x_1})$ is the Jacobi constant evaluated at the initial state on the first arc. These constraints form the total constraint vector $\boldsymbol{F}$:

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_c^T & \boldsymbol{F}_{PO}^T & \boldsymbol{F}_{C_J} \end{bmatrix}^T \tag{2.40}$$

Using these definitions, the free variable vector is then updated iteratively using Newton's method until the magnitude of the constraint vector equals zero within a desired tolerance, which is set to 10^{-12} for this research. Given an initial guess, X_1, the update equation at the i-th iteration is equal to:

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i - \boldsymbol{DF}(\boldsymbol{X}_i)^T(\boldsymbol{DF}(\boldsymbol{X}_i) \cdot \boldsymbol{DF}(\boldsymbol{X}_i)^T)^{-1}\boldsymbol{F}(\boldsymbol{X}_i) \tag{2.41}$$

where $DF(\boldsymbol{X})$ is the Jacobian matrix. With the addition of the Jacobi constant constraint to compute a single periodic orbit, the Jacobian is rectangular and, therefore not invertible, so the minimum norm solution is used in the update equation.

The Jacobian for this problem formulation is calculated analytically by leveraging the state transition matrix, $\boldsymbol{\Phi}$; a linear mapping between state variations to a reference trajectory from time t_0 to t [58, 41]. A linear approximation of a perturbed trajectory with respect to a reference trajectory can be computed using the first-order variational equations of motion computed using

$$\boldsymbol{\delta \dot{x}}(t) \approx \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{ref}} \boldsymbol{\delta x}(t) \tag{2.42}$$

where f are the system dynamics evaluated along the reference trajectory. Solutions to this equation take the following form:

$$\boldsymbol{\delta x}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\delta x}(t_0) \tag{2.43}$$

While these variational equations are linear, they are time-dependent by evaluation long the reference trajectory. This equation reveals a relationship that is a linear mapping from one time t_0 to another time t. Substitution Equation 2.43 into Equation 2.42 produces the first-order differential equation for the STM

$$\dot{\boldsymbol{\Phi}}(t, t_0) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{ref}} \boldsymbol{\Phi}(t, t_0) \tag{2.44}$$

with the initial condition for $\boldsymbol{\Phi}(t_0, t_0) = I_{6x6}$, a 6-dimensional identity matrix [41]. These additional 36 elements of the state transition matrix can be numerically propagated along with a state vector to analyze motion near a reference trajectory in the CR3BP.

Then, the continuity constraints of the Jacobian can be written in terms of the state transition matrix. The derivative of the continuity at the end of the i-th arc and at the beginning of the i+1-th arc is then written as:

$$\frac{\partial(\boldsymbol{x}_i(\Delta t) - \boldsymbol{x}_{i+1})}{\partial \boldsymbol{x}_i} = \boldsymbol{\Phi}_i \tag{2.45}$$

where $\boldsymbol{\Phi}_i$ can be computed by numerical propagation along with a state vector in the equations of motion using an identity matrix as the initial condition $\Phi_i = \boldsymbol{I}_{6x6}$ [41]. To compute the Jacobian of the energy level constraint, the partials of the Jacobi constant evaluated at the initial state along the first arc are explicitly written as:

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{x}} = \begin{bmatrix} 2x_1 - \frac{2((1-\mu)(x_1+\mu)}{r_1^3} - \frac{2\mu(x_1-1+\mu)}{r_2^3} \\ 2y_1 - \frac{2y_1(1-\mu)}{r_1^3} - \frac{2\mu y_1}{r_2^3} \\ \frac{-2z_1(1-\mu)}{r_1^3} - \frac{2z_1\mu}{r_2^3} \\ -2\dot{x}_1 \\ -2\dot{y}_1 \\ -2\dot{z}_1 \end{bmatrix}^T \tag{2.46}$$

where $r_1 = \sqrt{(x_1+\mu)^2+y_1^2+z_1^2}$ and $r_2 = \sqrt{(x_1-1+\mu)^2+y_1^2+z_1^2}$. Finally, the completed Jacobian for use in Equation 2.41 can be written as:

$$DF(\boldsymbol{X}) = \begin{bmatrix} & \boldsymbol{\Phi}_1 & & -\boldsymbol{I}_{6x6} & \boldsymbol{0}_{6x6} & \dots & \boldsymbol{0}_{6x6} & \dot{\boldsymbol{x}}_1(\Delta t) \\ & \boldsymbol{0}_{6x6} & & \boldsymbol{\Phi}_2 & -\boldsymbol{I}_{6x6} & \dots & \boldsymbol{0}_{6x6} & \dot{\boldsymbol{x}}_2(\Delta t) \\ & \vdots & & & \ddots & & & \vdots \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & & & \boldsymbol{0}_{5x6} & \boldsymbol{0}_{5x6} & \dots & \boldsymbol{\Phi}_n\{1:4,6\} & \dot{\boldsymbol{q}}(\Delta t) \\ & [0,1,0,0,0,0] & & \boldsymbol{0}_{1x6} & \boldsymbol{0}_{1x6} & \dots & \boldsymbol{0}_{1x6} & 0 \\ & \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{x}} & & \boldsymbol{0}_{1x6} & \boldsymbol{0}_{1x6} & \dots & \boldsymbol{0}_{1x6} & 0 \end{bmatrix} \tag{2.47}$$

where $\boldsymbol{q} = [x_1, y_1, z_1, \dot{x}_1, \dot{z}_1]^T$ and $\boldsymbol{\Phi}_n$ has the fifth row removed since the $\dot{y}$ component was removed from the periodicity constraint. Using this method, the converged solutions represent a numerical approximation of periodic orbits in the CR3BP that satisfies the periodicity constraint as well as the Jacobi constant constraint, if utilized, in the problem formulation.

2.2.2.2 Pseudo-Arclength Continuation

To find additional members in a periodic orbit family, a pseudo-arclength continuation method is employed by stepping along the tangent of the solution curve to identify potential nearby members [40]. Given a previously corrected periodic orbit with free variable vector, X_{prev}, a nearby periodic orbit, X_{new}, is found along the parameter curve, p. The initial guess for the nearby orbit is constructed by stepping a specified distance, δs, along the solution curve:

$$\boldsymbol{X}_{new} = \boldsymbol{X}_{prev} + \hat{\boldsymbol{n}}_{curr}\delta s \tag{2.48}$$

where $\hat{\boldsymbol{n}}_{curr}$ is the unit vector tangent to the family's solution curve, lying in the nullspace of the Jacobian evaluated at $\boldsymbol{X}_{curr}$. By removing the Jacobi constant constraint for periodic orbits

computed using Newton's method outlined in the previous section, the formulation of the Jacobian becomes computed such that the nullspace of the matrix is 1-dimensional, ensuring a straightforward computation of the other members along the periodic orbit family. Instead, an additional constraint is appended to the constraint vector $\boldsymbol{F}$ to ensure the new converged solution is indeed δs away from the previous solution along the solution curve. The new constraint is written as:

$$F_{fam} = [(\boldsymbol{X} - \boldsymbol{X}_{prev})^T \cdot \hat{\boldsymbol{n}}_{curr} - \delta s] \tag{2.49}$$

Due to the additional constraint, the Jacobian matrix is also appended with an additional row:

$$DF_{fam} = [\hat{\boldsymbol{n}}_{curr}^T] \tag{2.50}$$

With this updated formulation, members along a periodic family may be computed given an initial member in the family. This process can be repeated until a set of termination conditions are satisfied, such as the solution failing to converge at each correction step, impacting a primary, or computing a specified number of members within the orbit family.

2.2.3 Invariant Manifolds

A stability analysis of a periodic orbit reveals the behavior of nearby motion; orbits that are unstable possess unstable and stable hyperbolic invariant manifolds that flow asymptotically away from and towards the periodic orbit as $t \to \infty$ [58]. Specifically, the monodromy matrix, i.e., the state transition matrix propagated for exactly one orbital period $\Phi(T, 0)$, is decomposed into reciprocal pairs of eigenvalues [41]. A pair of real, reciprocal eigenvalues indicates the existence of global stable and unstable hyperbolic invariant manifolds, W^S and W^U [41]. Each global manifold structure can be composed into positive and negative half-manifold structures, where each half-manifold flows in one direction away from the periodic orbit.

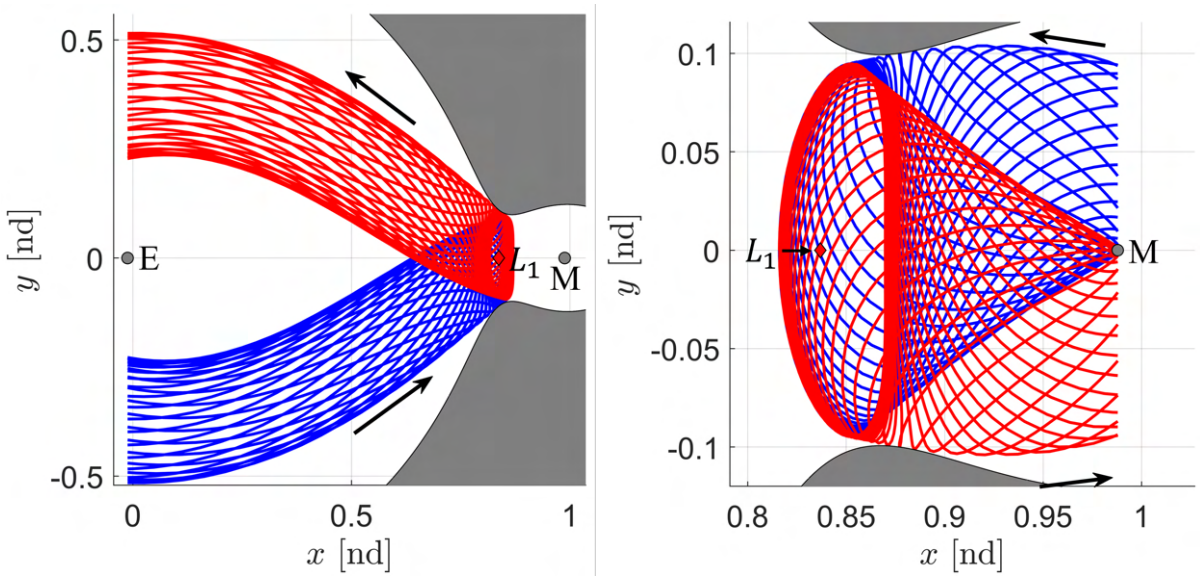
To numerically compute trajectories that lie along the invariant manifold of an unstable periodic orbit, first the orbit is discretized into a sequence of states. Then, for each state, a small step of magnitude d is taken off the periodic orbit into the direction of the stable (or unstable) eigenspace.

Given the stable (unstable) eigenvectors, $\boldsymbol{e_s}$ ($\boldsymbol{e_u}$), that are computed from the monodromy matrix, an initial state for a trajectory that lies on an invariant manifold structure is computed as:

$$\boldsymbol{x}_{mani} = \boldsymbol{x}_{po} \pm d\boldsymbol{v} \tag{2.51}$$

where $\boldsymbol{v}$ denotes the stable (or unstable) eigenvector normalized by the magnitude of the position components and $\pm$ denotes the direction of the manifold [41]. The perturbed state, $\boldsymbol{x}_{mani}$, is then numerically integrated backward (or forward) in time, respectively, to produce a trajectory along the stable (or unstable) half manifold. This process is repeated for each discretized state along the periodic orbit and perturbations in each direction within the eigenspace, producing a numerical approximation of the stable (or unstable) manifold [41].

Figure 2.5 depicts the global stable (blue) and unstable (red) invariant manifolds associated with a periodic orbit at $C_J = 3.15$ in the Earth-Moon CR3BP. Each trajectory is propagated until it possess an x-coordinate that is the same as a primary, but other termination criteria such as impacting a primary body or a specified number of states or apses with respect to a primary body can be of interest as well [41]. These stable and unstable invariant manifolds govern natural transport throughout the system and are, therefore, often used to design free or low-cost transfers between their associated periodic orbits [41].



(a) Stable and unstable half-manifolds directed towards the Earth.

(b) Stable and unstable half-manifolds directed towards the Moon.

Figure 2.5: Selected trajectories that lie along the stable (blue) and unstable (red) invariant manifolds associated with an L_1 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP.

Chapter 3

Trajectory Corrections and Optimization

To compute a continuous trajectory in the CR3BP, a two-point boundary value problem is solved using a free variable and constraint vector formulation of collocation. Although many other corrections schemes exist, collocation is a common corrections scheme that tends to be robust to discontinuities and supports the application of path constraints. Collocation is a numerical technique for implicitly integrating differential equations of a dynamical system where a solution is recovered by computing piecewise polynomials that approximate a trajectory and satisfy the system dynamics at collocation points [60]. Then, to improve the accuracy of the solution, a mesh refinement process is used in combination with collocation to increase or decrease the number of segments in the mesh. The collocation framework described in the following section closely follows the generalized odd-degree formulation used by Grebow and Pavlak as well as Pritchett to recover continuous trajectories in a multi-body system [60, 28]. Once a continuous trajectory is recovered using collocation, a constrained nonlinear optimization problem is solved using the 'interior-point' algorithm within the *fmincon* tool in MATLAB® in combination with collocation to minimize the sum of the squares of the maneuver magnitudes across the trajectory and recover locally optimal impulsive solutions. The methods used to recover locally optimal, continuous trajectories are presented in this chapter.

3.1 Trajectory Corrections via Collocation

Given an initial guess for a trajectory that is composed of discontinuous segments, collocation is a corrections algorithm used to recover a continuous solution in the CR3BP. First, an initial guess for a trajectory is discretized into a mesh of boundary nodes along the trajectory consisting of s segments. This discretization results in $s+1$ boundary nodes. Each boundary node is classified by a state and time. Then, the i-th segment is discretized into p_i arcs, where each arc will be later sampled to produce several nodes that describe the collocation problem. For clarity, the following notation will be used for this research: the state and time measured from the beginning of the trajectory at the k-th node on the i-th segment and j-th arc are denoted as $\boldsymbol{x}_i^{j,k}$ and $t_i^{j,k}$, respectively. This discretization scheme constructs the resulting mesh, Π, as discrete points in time:

$$\Pi : t_1 < t_2 < \dots t_s < t_{s+1} \tag{3.1}$$

where each mesh point is also described by an associated 6-dimensional initial state in the CR3BP.

The simplest collocation problem can be solved using a first-order approximation of the dynamics, e.g. Euler's integration rule. Euler's integration predicts the state at a later time step, $\boldsymbol{x}_{i+1}$, using the state at the current time step, $\boldsymbol{x}_i$, and vector field information provided by evaluating the system's dynamics using the current state and time step, thus providing the "slope" of the problem. The state at a later time is computed by:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x} + \Delta t_i \boldsymbol{f}(t_i, \boldsymbol{x}_i) \tag{3.2}$$

where $\boldsymbol{f}(t_i, \boldsymbol{x}_i) = \dot{\boldsymbol{x}}_i$ are the system dynamics evaluated at t_i and $\boldsymbol{x}_i$. To evaluate how well the integration rule predicted the next state, the difference in the predicted state and the actual state can be computed via the following defect equation:

$$\boldsymbol{\Delta}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i+1} + \Delta t_i \boldsymbol{f}(t_i, \boldsymbol{x}_i) = \boldsymbol{0} \tag{3.3}$$

The collocation problem is solved when $\boldsymbol{\Delta}_i = \boldsymbol{0}, \forall i$, ensuring continuity between all segments in the predefined mesh. The accuracy of solution solved with this collocation scheme is a direct result

of the numerical integration method chosen; since Euler's rule approximates the solution with a first order integration scheme, the resulting solution has first order accuracy. Alternatively, the trapezoidal method which achieves second order accuracy, can be used to increase the accuracy of the solution and decrease the number of segments required to achieve an accurate solution.

To achieve a solution with an accuracy sufficient for real-life applications, higher-order collocation schemes are implemented by increasing the degree of the polynomial used to represent the solution. The accuracy of the final solution depends on the selection of the polynomial as well as the node spacing strategy: lower order polynomials may be used in conjunction with a finer mesh, while higher order polynomials may achieve a higher order accuracy with a coarser mesh. First, the time along each arc is normalized to $[-1, 1]$ for numerical simplicity. The normalized time τ along the j-th arc on the i-th segment is computed using the following transformation:

$$\tau = \frac{2}{\Delta t_i^j}(t - t_i^{j,1}) - 1 \tag{3.4}$$

where the integration time along the arc is $\Delta t_i^j = t_i^{j,n} - t_i^{j,1}$.

Using the above mesh discretization and time normalization scheme, each arc is discretized into boundary nodes and collocation nodes using an n^{th} order polynomial and a node spacing strategy. For an n^{th} order polynomial, n discretized nodes, called collocation nodes, are placed along each arc in the initial guess. Typically, n is selected to be an odd degree polynomial since it will yield the same accuracy with a less complex formulation as the $n+1$ even degree formulation [60]. A 7^{th} order polynomial is used in this research, consistent with previously demonstrated trajectory design implementations in multi-body systems [32, 33, 28]. Therefore, 7 collocation nodes are placed along each arc and the location of these nodes is selected by the node spacing strategy.

The node spacing strategy determines where the collocation nodes are located along a selected arc that correspond to the roots of a Legendre polynomial, $P_n(\tau)$ [28]. Legendre-Gauss-Lobatto (LGL) nodes are collocated at -1 and 1 (i.e. the boundaries of the arc) as well as the roots of $\dot{P}_{n-1}(\tau)$, providing an accuracy of $2n-2$ [28]. Legendre-Gauss-Radau reversed points are located

at the roots of $P_{n-1}(\tau)$ and $P_n(\tau)$, providing an accuracy of $2n-1$, while Legendre-Gauss points are only located at the roots of $P_{n-1}(\tau)$, providing an accuracy of $2n$ [28]. For this implementation, an LGL node spacing strategy is employed due to its higher order of accuracy and simplified design problem due to placing collocation nodes at the boundary nodes of each segment [28]. If the boundary nodes were not previously defined as collocation nodes, they would need to be included in the problem formulation, increasing the dimension of the problem. For this strategy, collocation nodes are placed at the boundary nodes of each segment as well as at the roots of the derivative of the $(n-1)$th polynomial at time τ. Additionally, a weighting term, w_k, is computed at time τ_k for each node $k = [1, n]$ according to the LGL node spacing strategy. The node locations and associated quadrature weights using the LGL node spacing strategy can be computed for $n = 7$ and are displayed in Table 3.1[42].

Table 3.1: Truncated Legendre-Gauss-Lobatto node locations and quadrature weights computed for $n = 7$.

Node number (k)	Normalized time, τ_k	Weight, w_k
1	-1.000	0.0476
2	-0.8302	0.2768
3	-0.4688	0.4317
4	0.0000	0.4876
5	0.4688	0.4317
6	0.8302	0.2768
7	1.0000	0.0476

Along each arc, the collocation nodes are categorized into free nodes and defect nodes. The free nodes are the odd-numbered nodes for a given arc while the defect nodes are the even-numbered nodes. The free nodes, labeled $\boldsymbol{x}$, are used to construct the polynomial representation for each state component along an arc while the defect nodes, labeled $\boldsymbol{p}$, are the locations where the polynomials are evaluated and compared to the system dynamics. Figure 3.1 conceptually depicts a segment discretized into two arcs using 7 collocation nodes described by the 7th order polynomial representation with LGL node spacing [28]. Free nodes are depicted as blue triangles, defect nodes are depicted as red circles, and the arc boundaries are the free nodes outlined in black. Since the i-th

segment is discretized into p_i arcs, the final state along arc j and the initial state along arc $j+1$ are shared since they share a common boundary node, but consecutive segments do not share boundary nodes.

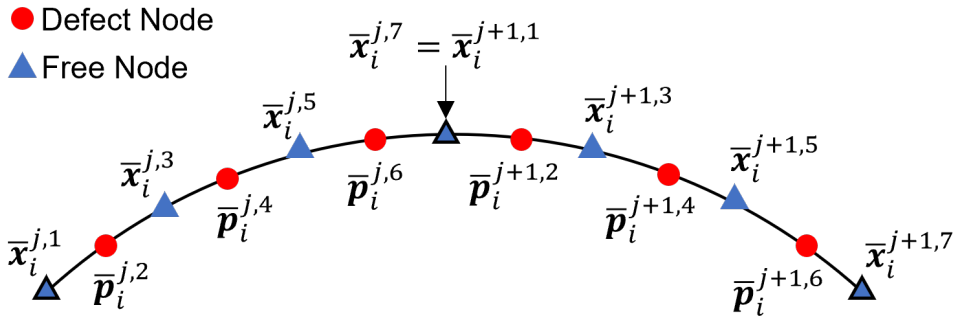


Figure 3.1: A conceptual example of the collocation nodes for arc j and $j+1$ along segment i using a 7th degree polynomial and a LGL node spacing strategy.

At each location of τ, the polynomial is evaluated to compute an approximate state along the arc, $\boldsymbol{p}$. The states along arc j on segment i are approximated by the n^{th} order polynomial as

$$\boldsymbol{p}_i^j(\tau) = \boldsymbol{C}_i^j \begin{bmatrix} 1 & \tau & \tau^2 & \dots & \tau^n \end{bmatrix}^T \tag{3.5}$$

where $\boldsymbol{C}_i^j$ is a unique matrix for each arc composed of polynomial coefficients and has dimensions $6 \times (n+1)$ for the CR3BP. Each row of $\boldsymbol{p}_i^j(\tau)$ provides the coefficients of the polynomial at each node along the arc at time τ_k. Then, these polynomial coefficients are used to evaluate the state and first time derivative of the state at each τ_k, denoted $\boldsymbol{p}_i^{j,k}$ and $\dot{\boldsymbol{p}}_i^{j,k}$ respectively. The time derivative of the states and polynomials at each free node with respect to normalized time are computed as:

$$\dot{\boldsymbol{x}}_i^{j,k} = \frac{\Delta t_i^j}{2} \boldsymbol{f}(\tau_k, \boldsymbol{x}_i^{j,k}) \tag{3.6}$$

$$\dot{\boldsymbol{p}}_i^{j,k} = \frac{\Delta t_i^j}{2} \boldsymbol{f}(\tau_k, \boldsymbol{p}_i^{j,k}) \tag{3.7}$$

where Δt_i^j is the non-normalized time for the arc and $\boldsymbol{f}(\cdot)$ represents the evaluation of the system dynamics, e.g. the dynamics of the CR3BP.

Using the states and their derivatives of the free nodes, a matrix representation along arc j

on segment i can be written as

$$\boldsymbol{C}_i^j \begin{bmatrix} \boldsymbol{\tau} & \dot{\boldsymbol{\tau}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_i^{j,1}, & \boldsymbol{x}_i^{j,3}, & \dots & \boldsymbol{x}_i^{j,n} & | & \dot{\boldsymbol{x}}_i^{j,1}, & \dot{\boldsymbol{x}}_i^{j,3}, & \dots & \dot{\boldsymbol{x}}_i^{j,n} \end{bmatrix} \tag{3.8}$$

where the matrices $\boldsymbol{\tau}$ and $\dot{\boldsymbol{\tau}}$ are

$$\boldsymbol{\tau} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \tau_1 & \tau_3 & \dots & \tau_n \\ \tau_1^2 & \tau_3^2 & \dots & \tau_n^2 \\ \vdots & \vdots & \dots & \vdots \\ \tau_1^n & \tau_3^n & \dots & \tau_n^n \end{bmatrix} \tag{3.9}$$

$$\dot{\boldsymbol{\tau}} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \\ 2\tau_1 & 2\tau_3 & \dots & 2\tau_n \\ 3\tau_1^2 & 3\tau_3^2 & \dots & 3\tau_n^2 \\ \vdots & \vdots & \dots & \vdots \\ n\tau_1^{n-1} & n\tau_3^{n-1} & \dots & n\tau_n^{n-1} \end{bmatrix} \tag{3.10}$$

Finally, the states and derivatives at the locations of the free nodes are computed to form the matrix representation $\boldsymbol{C}_i^j$, which can be written as:

$$\boldsymbol{C}_i^j = \begin{bmatrix} \boldsymbol{x}_i^{j,1}, & \boldsymbol{x}_i^{j,3}, & \dots & \boldsymbol{x}_i^{j,n} & | & \dot{\boldsymbol{x}}_i^{j,1}, & \dot{\boldsymbol{x}}_i^{j,3}, & \dots & \dot{\boldsymbol{x}}_i^{j,n} \end{bmatrix} \boldsymbol{A}^{-1} \tag{3.11}$$

where $\boldsymbol{A} = [\boldsymbol{\tau}, \dot{\boldsymbol{\tau}}]$.

Last, the state and time derivative of the state with respect to normalized time are approximated at the location of the defect nodes using the constructed polynomials and the matrices $\boldsymbol{B}$ and $\boldsymbol{D}$:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & 1 \\ -1 & \tau_2 & \tau_4 & \dots & \tau_{n-1} & 1 \\ 1 & \tau_2^2 & \tau_4^2 & \dots & \tau_{n-1}^2 & 1 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ -1 & \tau_2^n & \tau_4^n & \dots & \tau_{n-1}^n & 1 \end{bmatrix} \tag{3.12}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \\ 2\tau_2 & 2\tau_4 & \dots & 2\tau_n \\ \vdots & \vdots & \dots & \vdots \\ n\tau_2^{n-1} & n\tau_4^{n-1} & \dots & n\tau_n^{n-1} \end{bmatrix} \tag{3.13}$$

where the $\boldsymbol{B}$ matrix includes the first and last columns to compute the states at the boundary nodes as well. Since matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{D}$ are only dependent on the node spacing strategy selected, they are constant for every arc and are only required to be computed once at the initialization of the collocation problem. Finally, the states at the defect nodes and the initial and final state along arc j on segment i can be computed as:

$$\begin{bmatrix} \boldsymbol{p}_i^{j,0} & \boldsymbol{p}_i^{j,2} & \boldsymbol{p}_i^{j,4} & \dots & \boldsymbol{p}_i^{j,n-1} & \boldsymbol{p}_i^{j,f} \end{bmatrix} = \boldsymbol{C}_i^j \boldsymbol{B} \tag{3.14}$$

while the derivatives of the polynomials at the defect nodes can be computed as:

$$\begin{bmatrix} \dot{\boldsymbol{p}}_i^{j,0} & \dot{\boldsymbol{p}}_i^{j,2} & \dot{\boldsymbol{p}}_i^{j,4} & \dots & \dot{\boldsymbol{p}}_i^{j,n-1} & \dot{\boldsymbol{p}}_i^{j,f} \end{bmatrix} = \boldsymbol{C}_i^j \boldsymbol{D} \tag{3.15}$$

These approximated states and time derivatives of the polynomials located at the defect nodes are later used to construct the defect equations that compare these evaluations to the states and time derivatives of the states computed using the system dynamics in Equation 3.7.

Now the collocation problem can be formulated using a free variable vector and a constraint vector. The design variables are the collection of all unique states at the free nodes along each arc as well as the nondimensional time for each arc so that the total time of flight may vary. Assuming each segment is discretized into p_i arcs, the free variable vector is the defined as:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^{1,1} & \dots & \boldsymbol{x}_1^{1,n-2} & \Delta t_1^1 & \dots & \boldsymbol{x}_1^{p_1,1} & \dots & \boldsymbol{x}_1^{p_1,n} & \Delta t_1^{p_1} \\ \boldsymbol{x}_2^{1,1} & \dots & \boldsymbol{x}_2^{1,n-2} & \Delta t_2^1 & \dots & \boldsymbol{x}_2^{p_2,1} & \dots & \boldsymbol{x}_2^{p_2,n} & \Delta t_2^{p_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{x}_s^{1,1} & \dots & \boldsymbol{x}_s^{1,n-2} & \Delta t_s^1 & \dots & \boldsymbol{x}_s^{p_s,1} & \dots & \boldsymbol{x}_s^{p_s,n} & \Delta t_s^{p_s} \end{bmatrix} \tag{3.16}$$

and $\boldsymbol{X}$ is then reshaped into a single column vector.

The constraint vector captures both the continuity constraints between segments and the defect constraints for every defect node along the trajectory. First, the continuity constraint between the end of the i-th segment and the start of the $i+1$-th segment is defined as

$$\boldsymbol{F}_{c_i} = \begin{cases} \boldsymbol{x}_i^{p_i,7} - \boldsymbol{x}_{i+1}^{1,1}, \text{if no maneuver} \\ \boldsymbol{r}_i^{p_i,7} - \boldsymbol{r}_{i+1}^{1,1}, \text{if maneuver} \end{cases} \tag{3.17}$$

where $\boldsymbol{r} = [x, y, z]^T$. Then, the constraint vector that enforces continuity between neighboring segments along the entire trajectory is defined as

$$\boldsymbol{F}_c = \begin{bmatrix} \boldsymbol{F}_{c_1} & \boldsymbol{F}_{c_2} & \dots & \boldsymbol{F}_{c_{s-1}} \end{bmatrix}^T \tag{3.18}$$

Next, the defect constraints evaluate how well the polynomial satisfies the system dynamics at the defect nodes. The defect constraints for the j-th arc along the i-th segment for a 7th order polynomial are defined as

$$\boldsymbol{F}_{d_i}^j = \begin{bmatrix} \boldsymbol{\Delta}_i^{j,1} \\ \boldsymbol{\Delta}_i^{j,2} \\ \boldsymbol{\Delta}_i^{j,3} \end{bmatrix} = \begin{bmatrix} (\dot{\boldsymbol{p}}_i^{j,2} - \dot{\boldsymbol{x}}_i^{j,2}) w_2 \\ (\dot{\boldsymbol{p}}_i^{j,4} - \dot{\boldsymbol{x}}_i^{j,4}) w_4 \\ (\dot{\boldsymbol{p}}_i^{j,6} - \dot{\boldsymbol{x}}_i^{j,6}) w_6 \end{bmatrix} \tag{3.19}$$

where w_i are the quadrature LGL node weights and $\dot{\boldsymbol{p}}$ is the derivative of the polynomial state with respect to normalized time. Then, the defect constraints for the i-th segment are written as $\boldsymbol{F}_{d_i} = [\boldsymbol{F}_{d_i}^1, \boldsymbol{F}_{d_i}^2, \dots, \boldsymbol{F}_{d_i}^p]^T$. Finally, the complete constraint vector is defined as

$$\boldsymbol{F}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{F}_c & \boldsymbol{F}_{d_1} & \boldsymbol{F}_{d_2} & \dots & \boldsymbol{F}_{d_s} \end{bmatrix}^T \tag{3.20}$$

To recover a solution, the free variable vector is iteratively updated from an initial guess by applying Newton's method using Equation 2.41. The nonzero components of the Jacobian matrix of the constraint vector with respect to the free variables for this formulation are computed analytically. These iterations continue until the norm of the constraint vector equals zero within a selected tolerance of 10^{-12} for this research.

3.1.1 Mesh Refinement

Once the initial collocation problem is solved with a prespecified user-defined discretization, the dynamics at each defect node are sufficiently satisfied by the polynomial constructed, but the trajectory may not be accurately approximated at other locations. Therefore, to improve the accuracy of the final solution, a mesh refinement process is used to remove and/or add arcs and adjust the location of the boundary points in the initial mesh. Following the approach presented by Pritchett, control with explicit propagation (CEP) is used in this research to update the number of arcs in the mesh [60].

The CEP mesh refinement method involves two phases: arc removal and arc addition. After the initial collocation problem is solved, consecutive arcs are examined to determine if they can be combined using explicit propagation; removing arcs initially will reduce the size of the collocation problem [28]. First, the initial boundary node on the i-th arc is propagated for $t = \Delta t_i^j + \Delta t_i^{j+1}$. The error is then defined as the difference between the state at the end of this propagation time and the final boundary node on the subsequent $j+1$-th arc. If the norm of this error is below 10^{-12}, then the shared boundary point between the two arcs is removed and the two arcs are combined into a single arc. Then, the free collocation nodes are recomputed using the LGL node spacing strategy using numerical propagation. If any arcs were merged during this first phase of mesh refinement, the collocation problem is solved again using the updated mesh. This process is repeated until the number of arcs does not change.

Next, arcs are added where the accuracy of the solution is not met to within some tolerance. The initial boundary node of each arc is numerically propagated for the integration time associated with the arc. The error for a specific arc is then computed as the difference between the final state from the numerical propagation and the final boundary node of the arc. If the norm of the error is above 10^{-12}, then the arc is split into two separate arcs of equal integration time. The collocation problem is then solved with the new mesh, repeating until no new arcs are added. Using mesh refinement in collocation results in a higher level of accuracy along the entire solution, verified via

explicit propagation, and the accuracy of the final solution is no longer dependent on a user-defined initial mesh [60].

3.2 Constrained Nonlinear Optimization

Given an initial guess for a trajectory, a natural or maneuver-enabled trajectory may be recovered using the collocation algorithm described in the previous section. However, for maneuver-enabled trajectories, the recovered solution may only be one of many possible solutions and may not be an optimal solution for propellant usage or other objectives. Therefore, a constrained nonlinear optimization problem is formulated using a free variable vector and constraint vector with collocation to recover locally optimal maneuver-enabled transfers between periodic orbits.

Many algorithms and programs exist for solving constrained nonlinear optimization problems. Algorithms such as the 'interior-point' method are well-suited for large scale problems, e.g. a large free variable vector or constraint vector. This is due to leveraging the sparsity of the Jacobian and Hessian matrices when computed analytically to reduce computational time [13, 12]. For this research, the 'interior-point' algorithm within the *fmincon* tool in MATLAB® is employed to compute constrained optimal maneuver-enabled trajectories in the CR3BP.

First, a continuous solution is recovered using collocation with the mesh refinement approach described in the previous section; this supplies the initial guess for the constrained optimization problem. In addition to the continuity and defect constraints, two additional free variables and constraints are incorporated to ensure the optimal solution departs from the initial orbit and arrives to the final orbits. Two additional free variables that represent a time propagated along an orbit, τ_{depart} and τ_{arrive}, are included to specify a constrained state along the initial and final orbits, respectively. These variables are used to define a new state, e.g. $\boldsymbol{x}_{depart}$, by propagating an initial state that defines the periodic orbit for time τ, e.g. $\boldsymbol{x}_{depart} = \boldsymbol{f}(\tau_{depart}, \boldsymbol{x}_{PO,init})$. Similarly for the final orbit, τ_{arrive} is used to define the state $\boldsymbol{x}_{arrive}$.

Using the departure and arrival states along the orbit, two new constraints, $\boldsymbol{F}_{depart}$ and $\boldsymbol{F}_{arrive}$, are introduced to constraint the transfer to the departure and arrival states. In addition to

any maneuvers placed along the transfer by a human trajectory designer, two impulsive maneuvers are placed at the departure and arrival positions for each orbit. This constrains only the position of the initial state along the transfer to the initial orbit and the position of the final state along the transfer to the final orbit. These constraints are defined using the states associated with the free variable nodes as:

$$\begin{aligned} \boldsymbol{F}_{depart} &= \boldsymbol{r}_1^{1,1} - \boldsymbol{r}_{depart} \\ \boldsymbol{F}_{arrive} &= \boldsymbol{r}_s^{p,n} - \boldsymbol{r}_{arrive} \end{aligned} \tag{3.21}$$

where $\boldsymbol{r}_{depart}$ and $\boldsymbol{r}_{arrive}$ are the position components of $\boldsymbol{x}_{depart}$ and $\boldsymbol{x}_{arrive}$, respectively. The new free variable vector and constraint vector for the constrained nonlinear optimization problem are written as:

$$\begin{aligned} \boldsymbol{X}_{opt} &= \begin{bmatrix} \boldsymbol{X}^T & \tau_{depart} & \tau_{arrive} \end{bmatrix}^T \\ \boldsymbol{F}_{opt} &= \begin{bmatrix} \boldsymbol{F}^T & \boldsymbol{F}_{depart} & \boldsymbol{F}_{arrive} \end{bmatrix}^T \end{aligned} \tag{3.22}$$

where $\boldsymbol{X}$ and $\boldsymbol{F}$ are the same free variable vector and constraint vector defined in Equation 3.16 and Equation 3.20. These two free variables and constraints allow the initial departure position of the transfer to occur anywhere along the initial orbit and the final arrival position of the transfer to occur anywhere along the final orbit.

Using the free variable vector and constraint vector defined above, the constrained nonlinear optimization problem can be solved to minimize the sum of the magnitude of impulsive maneuvers squared placed along the transfer. Minimizing the squared maneuver magnitudes rather than the sum of maneuver magnitudes can help aid in convergence of the optimization function. This objective function is the defined as:

$$J = \sum_{i=1}^{m} \Delta v_i^2 \tag{3.23}$$

where Δv_i is the i-th maneuver along the transfer and $m \geq 2$ is the number of maneuvers along the transfer, including the maneuvers placed at the departure ($i = 1$) and arrival ($i = m$) states. The converged solution is then an approximated optimal maneuver-enabled transfer between two

periodic orbits with the same accuracy of the 7^{th} order polynomial and LGL node spacing strategy utilized within the collocation formulation.

Chapter 4

Motion Planning

Path planning, also called motion planning, is a term used originally in robotics that describes the process of automatically constructing a sequence of discrete paths, or motion, for a robot. These techniques often focus on efficiently exploring a complex solution space and rapidly constructing valid solution paths, or sequences of motion, within an environment that adhere to all known obstacles or constraints. Additionally, the solution paths may be constructed to recover optimal paths or motion with respect to an environment heuristic, such as path length or cost. Traditional motion planners consist of graph-search, sampling-based, or interpolating curve algorithms, that aim to plan a trajectory between an origin and destination [76]. A subfield of motion planning, called kinodynamic planning, is characterized by the explicit consideration of a robot's dynamics during the planning process by obeying any kinematic and dynamic constraints [22]. In this research, sampling-based kinodynamic planning is used to first discretize the solution space to construct a directed, weighted graph and then recover valid paths using graph search algorithms. This chapter presents an overview of the motion planning methods and graph search algorithms, as well as the necessary graph theory, required to summarize a solution space and rapidly construct solution paths from the graph.

4.1 Sampling-Based Kinodynamic Planners

Sampling-based planning algorithms aim to solve a path planning problem by sampling nodes, which are collision-free valid configurations in an environment, and by connecting the nodes with collision-free, valid local paths called edges [47]. The result of the planner is a solution path which is a sequence of nodes and edges between user-defined start and goal nodes. Each configuration sampled and path constructed during the planner is considered valid if it also obeys all kinematic and dynamic constraints in the environment in addition to being collision-free with any known obstacles. Since these planners rely on sampling the environment to determine what configurations and paths are valid, they do not construct an explicit discretization or boundary representation of the environment. This method of exploring the environment can greatly speed up the performance of solving a planning query for a particular solution rather than mapping out an entire environment using methods such as grid or cell-based discretizations.

Additionally, sampling-based planners achieve some form of completeness that depends on the selected method of sampling; the completeness characteristic requires the planner always answer a path planning query correctly in asymptotically bounded time or terminate when there is no solution [47]. If the sampling is performed randomly, the planner achieves probabilistic completeness, meaning the planner will eventually return a solution if one exists as the number of samples drawn increases. Conversely, if the sampling is performed deterministically, the planner achieves a resolution completeness that depends on the resolution utilized during sampling.

Two popular sampling-based planning algorithms are PRM and RRT. The PRM planner focuses on generating a graph, called the roadmap, that sufficiently and discretely summarizes the environment. Nodes and edges are added to the graph until the desired graph completeness criteria is met, such as sampling a predetermined number of nodes or satisfying a coverage or density metric. Then a graph search algorithm is used to recover a solution path based on start and goal nodes that can be added to the graph after the roadmap has already been constructed. A main benefit of PRM is that once a roadmap is constructed for an environment, it can be searched multiple times

for a variety of configurations and constraints, rapidly producing a variety of valid paths based on user defined criteria.

Conversely, a RRT requires knowledge of the start and goal configurations prior to exploring the environment. This planner is completed by generating a randomly constructed space-filling tree from a given start configuration until the tree reaches a desired goal configuration. The tree, rooted at the start, grows by adding local paths and configurations to the tree in the direction of a randomly sampled configuration in the environment until the goal is reached to within a tolerance. As such, it is most often used for single query problems since it relies on knowledge of where the tree is needed to grow and terminates exactly once a goal is reached. Additionally, this may limit the tree exploration to the region of the solution space in between the start and the goal configurations.

Both of these planners are popular for solving higher-dimensional problems while being probabilistically complete. Although each of these planners have certain individual desirable traits, a combination of these methods is also possible and may offer more advantages in exploring certain environments, such as sampling-based roadmap of trees which connects multiple local trees to form a global roadmap that can be used for multiple search queries [47]. While the main focus of the techniques demonstrated in the following chapters will rely heavily on traditional PRM methods, some steps, such as the local sampling of the solution space, will be inspired by general discrete search algorithms [45].

4.1.1 Probabilistic Roadmap Generation

Motion planning with PRM is performed in two key phases: the learning phase and the query phase. During the learning phase, a directed weighted graph containing nodes and edges is constructed until the environment is sufficiently discretely summarized; this graph is referred to as the roadmap [19]. Nodes represent valid configurations in the environment and edges represent valid paths, or motion, between neighboring nodes. Valid nodes and edges must adhere to all system dynamics without intersecting any known obstacles. Nodes added to the graph are determined by the selected sampling scheme, such as uniform, random, or informed sampling, or a combination

of techniques [45, 47]. While the ideal number of nodes sampled is often one of the most difficult parameters to select, especially in unknown or complex environments, common methods include sampling a fixed number of nodes or sampling until a coverage or density metric is successfully evaluated. When a node evaluation metric is combined with incremental sampling, the resulting roadmap produced is often an efficiently summarized representation of an environment. These two methods combined can help determine when to stop sampling nodes and may limit unnecessary or redundant information being added to the roadmap by preventing oversampling which can directly effect the number of edges constructed as well [56, 55, 50].

After the desired number of nodes are added to the graph, determined by the chosen node sampling and/or evaluation metric, edges are constructed between neighboring nodes. For each node, edges may be constructed leaving or arriving to the node. A local planner is used to determine which neighboring nodes to attempt to connect through edges by selecting neighbors that would construct "promising" or low-cost edges between the two nodes; an ideal local planner is one that is likely to succeed [47]. Choosing an appropriate technique for determining the best neighbors for a given node may help increase the connectivity within the graph, ideally capturing all sensitive or narrow regions within an environment. Common techniques to identify neighbors include attempting to connect all nodes in simple environments, connecting nodes within a specified radius, or connecting nodes to only their k-nearest neighbors where k is either pre-defined or determined by local environment parameters [47]. These edges can either be directed, meaning the path can only be traversed in one direction, or undirected. Edges can also be weighted to reflect either a desired characteristic of the environment or the likelihood or difficulty of a specific edge being traversed in the lowest cost path recovered by a graph search algorithm. Edge weights are often weighted with a non-negative edge weight so graph search algorithms that recover the shortest or lowest cost path, such as Dijkstra's algorithm, do not fail. If edge weights are negative, other graph search algorithms that are not as computationally efficient such as the Bellman-Ford algorithm may be used, but Dijkstra's algorithm will only find the lowest cost path with non-negative edge weights [16]. Selecting a heuristic to assign edge weights for a given environment may aid the search algo-

rithm in finding optimal paths with respect to this heuristic, e.g. if time along an edge is selected as the edge weight, a path with the shortest total time from the start node to the goal node may be found.

Graph-based metrics may also be used to dictate the number of edges in the graph, such as constructing edges until the graph is either weakly or strongly connected that may be determined by evaluating the components of the graph. A component is a subgraph, or portion, of the graph that is not part of a larger connected subgraph [19]. Connected components are a way to determine the reachability of the nodes in a graph [19]. A weakly connected graph contains all nodes in the same component and there exists a path to every node starting from any other node, assuming each edge can be traversed in any direction (i.e. all edges are undirected). A strongly connected graph also contains all nodes in the same connected component, but there must exist a path between all nodes while still adhering to the direction of edges. In Figure 4.1, a graph is depicted that contains two connected components. The connected component on the left containing 'LMNOP' is only weakly connected since no nodes in the subgraph can reach node 'L', but if all the edges in the subgraph were undirected, there exists a path between every pair of nodes. Whereas the connected component on the right containing 'QRS' is strongly connected since a directed path exists between every pair of nodes. These two components might be joined to form a single weakly connected component if for example there was a path between 'P-Q'. To evaluate if a component

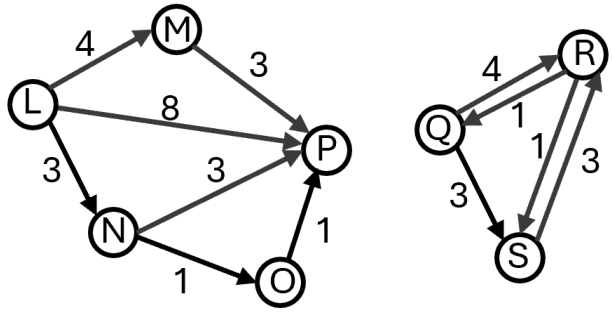


Figure 4.1: Example of a graph with two components: a weakly connected graph component (left) and a strongly connected graph component (right).

or graph is strongly connected, a simple graph traversal algorithm such as breadth-first search or

depth-first search can be used. These search algorithms explore the graph by visiting all nodes and categorize the nodes into two lists: visited or not visited. For any graph, the graph is first traversed starting from a single node, n_0, and the results are examined. If all nodes in a graph were explored and marked visited, then the transpose of the graph can be checked where the transpose of the graph is simply just flipping the direction of the edges. If all nodes are again visited starting from node n_0 on the transpose of the graph, then the graph is strongly connected. If the graph was not strongly connected, then it may be checked if it is weakly connected instead. To check if the graph is weakly connected, a directed graph can be made 'undirected' by adding an additional edge in the opposite direction for every directed edge. Then, if all nodes are marked as visited with a graph traversal algorithm, the graph is weakly connected. Determining the reachability of the nodes in a graph can help ensure the success of recovering solution paths after the roadmap is completed. If a desired start node exists in one connected component and the desired goal node exists in a different connected component, a path will never be recovered and the roadmap must be improved.

In challenging environments, an additional step of refining the roadmap may be included to expand the graph into sensitive or under-represented areas of the solution space, thereby improving the likelihood of success for multiple path planning queries. These areas of the solution space may be determined by the density of nodes across the roadmap, increased sampling near the boundaries of the environment or along obstacle boundaries, or using a visibility-based sampling scheme [47]. One method to evaluate the dispersion, or empty regions, within a graph is using Voronoi diagrams or its dual counterpart of a Delaunay Triangulation (DT). From a DT, the dispersion of the graph is defined by the largest empty sphere in the configuration space [38]. This method is commonly used in path planning problems to identifying areas of an environment that are insufficiently approximated by a discrete set of nodes or edges [35, 36]. Given a set of points, commonly the nodes from a graph, a DT is constructed by creating a set of tetrahedra that do not contain any nodes within their circumcircle. An example of the triangulation resulting from 30 randomly sampled points is shown in Figure 4.2; the black circles represent the points, the blue lines depict the triangulation, and the pink circles depict the resulting circumcircles. Algorithms

such as the Bowyer–Watson algorithm can compute the DT for a finite set of points for any number of dimensions [7]. These tetrahedra can then be examined to determine where empty regions in the graph may exist that require additional graph information. After these areas are identified,

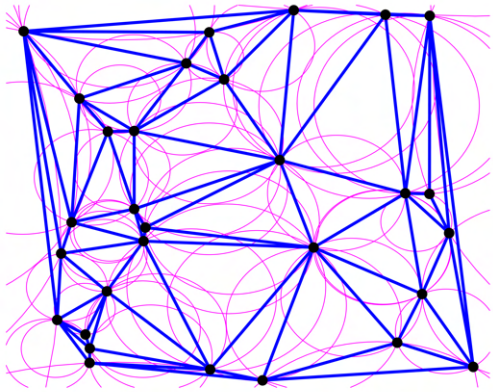


Figure 4.2: Example of a Delaunay Triangulation for a set of 30 points.

additional nodes are sampled and edges are constructed to increase the graph node coverage and edge connectivity in those areas to represent the environment better. Once the desired number of nodes and edges are added to the graph, or other graph-based metrics are satisfied, the roadmap is considered complete.

Next, the query phase occurs where the completed roadmap is searched rapidly and repeatedly for valid solution paths between user-specified start and goal configurations; if these configurations do not exist in the graph, they are added to the roadmap as nodes and connected to their neighbors with edges. To search the roadmap, the graph is queried using a graph search method, such as a graph traversal or shortest path search algorithm, for one or more paths between given start node(s) and goal node(s) if any paths exist.

A conceptual example of the PRM algorithm appears in Figure 4.3, where nodes are indicated by circles, obstacles are indicated by squares, and directed, weighted edges are indicated with a dashed arrow and the weight indicated with a non-negative number. First, nodes are sampled within the environment and added to the graph if they are valid, i.e. do not intersect any obstacles

such as the two nodes that are crossed out in the left most figure since they intersect with the two known obstacles. The start and goal nodes, indicated by the blue and red nodes respectively, are also added to the roadmap. Next, directed edges that do not intersect any obstacles are constructed between neighboring nodes and assigned an edge weight. Since the goal of this planning problem is to recover the lowest cost path, each edge is weighted with a non-negative weight. During the query, a search algorithm is used to recover a solution path from the start node to the goal node. In this example, a best first algorithm, such as Dijkstra's algorithm, is used to compute the lowest cost path by minimizing the sum of the edge weights along the solution path depicted by the solid arrows. A completed roadmap may be queried as many times as needed to produce a variety of solution paths between the same start and goal node(s) or any other identified start and goal node(s).

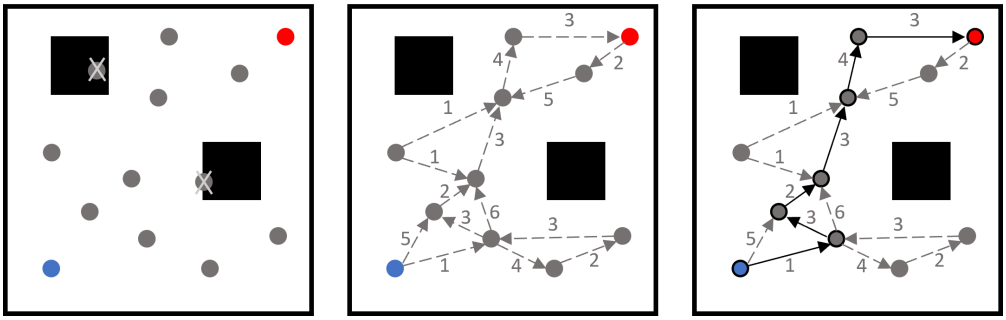


Figure 4.3: Conceptual example of the probabilistic roadmap generation planner.

4.2 Graph Search Algorithms

Discrete search algorithms are used to search the roadmap summarizing the continuous solution space for possible paths. A wide variety of discrete search algorithms, each searching the solution space uniquely, are suited to different environments [37]. Search algorithms are assessed for completeness and their computational efficiency. An efficient search minimizes the total number of nodes visited, improving computationally efficiency because it does not necessarily search the entire space and visit every node. A complete search algorithm will always terminate in finite time,

either providing a solution path if one exists or determining one does not exist. An incomplete search algorithm will continue searching for infinite time, such as searching an unbounded solution space.

There are three main categories of search algorithms: breadth first searches, depth first searches, and best first searches [66]. Each of these algorithms generates a search tree, but explores the path options differently. Breadth first searches explore all children of the root node first, then explores all of the grandchildren, and repeats until a desired node is found or the graph was entirely searched [66]. A child node is a node connected to the parent node through an edge, whereas a grandchild node is connected to the parent node through two edges, and so on for the remaining levels of children. Breadth first searches explore vertically first, then moving through the graph left to right [66]. These types of search algorithm produces the shortest path in terms of the number of edges linked together, but it does not guarantee an efficient search. Beginning again at the root of the tree, depth first searches select one parent node, continuously searching through that node's children, until all children are explored or the desired goal node is reached. When all children have been explored, the algorithm returns to the parent node and repeats the process of searching through the next set of children along the next parent node. Depth first searches are typically well-suited for scenarios where it is important to search the entire space. Depth first searches explore horizontally first, then moving through the graph top to bottom [66]. In environments where it is not important to search the entire graph, but instead produce the best path according to a predefined objective, best first search algorithms are useful. Best first searches, also called informed searches, explore the graph efficiently using predefined environment knowledge. In this research, best first search algorithms are used.

4.2.1 Dijkstra's Algorithm

Dijkstra's algorithm is a best-first search algorithm that explores a graph by expanding in 'promising' directions; 'promising' nodes and edges have smaller heuristics, e.g. weights, costs, or lengths. Exploring the roadmap in this manner ensures Dijkstra's algorithm is computationally

efficient because it does not need to explore every node in the graph and still guarantees it will find the optimal path with respect to a given heuristic [45]. It is also a complete algorithm, meaning it will terminate in finite time, even if no solution exists [37]. Dijkstra's algorithm is used to identify the shortest, or lowest cost path if a heuristic is selected, between two nodes, or two sets of nodes, in a weighted, directed graph [37].

The algorithm searches the graph beginning from the start node(s) by creating two lists: the priority list and the closed list [37]. The priority list contains the nodes currently being explored at a single iteration and the closed list contains all nodes previously explored, ensuring an efficient algorithm that does not explore any node more than once. To start the priority list, any directed edge away from the start node(s) are explored first and the start node(s) is added to the closed list; this results in all neighboring nodes from the start node(s) being contained within the priority list. Additionally, the total cost, or weight, of each path from its respective start node to the current node being searched is retained within the priority list. All edges are expanded and neighbors are explored in one iteration. The total cost of all potential paths is then used to sort the priority list and find the neighbor with the lowest cost path away from its respective start node, indicating the next best node to explore. The next best node is then removed from the priority list and added to the closed list, while all of its neighboring nodes and their associated total weights from the start node are added to the priority list. The priority list is then sorted again to find the next best node to explore. This process is repeated until the goal node is contained as the next best node within the priority list, indicating a solution has been found, or until all nodes have been explored.

The result of this algorithm is the optimal shortest path, or lowest cost path based on the heuristic selected for the edge weights. By changing the desired edge weight or creating a cost function to explore a variety of environment parameters, a variety of optimal solution paths may be found. This process is conceptually depicted in Figure 4.4, where the directed weighted graph is searched for the lowest cost path between node 'L' and node 'P'. Starting from node 'L', the first iteration explores is neighbors through directed edges away from 'L': node 'M' and node 'N'. This creates two entries in the priority list, which is then sorted to reveal node 'N' is the best promising

node to continue to explore. Next, node ‘N’ is added to the closed list and it’s neighbors are added to the priority list. At this iteration, the goal node ‘P’ is contained within the priority list, but since it is not the best next node to explore, the algorithm continues. After the fourth iteration, the lowest cost path ‘L-N-O-P’ is recovered.

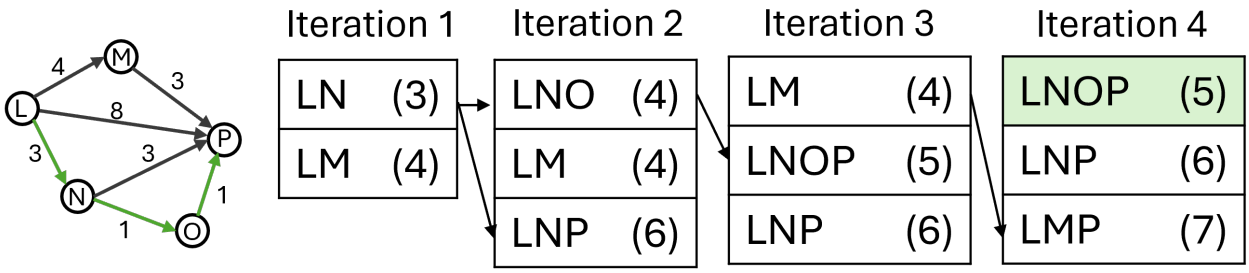


Figure 4.4: Dijkstra’s search algorithm to recover the lowest cost path (green) from the start node ‘L’ to the goal node ‘P’ by minimizing the sum of the edge weights in the directed, weighted graph.

4.2.2 Yen’s Algorithm

Once a graph is completed and an optimal path is found if one exists, it may be useful for additional sub-optimal paths to be explored from the same roadmap using the same start and goal nodes. In a variety of path planning problems, using a repeating or modified graph search algorithm to search for a specific number of paths is common. For example, in a driving application a second-optimal path that avoids traffic present on the time-optimal route might be of interest to a driver if the second route is only one minute longer. Searching for these additional paths is commonly denoted as the k-shortest paths problem. Developing algorithms for k-shortest paths has been of great interest over the past few decades with most algorithms improving upon Yen’s algorithm; one of the first algorithms to search a directed and weighted graph for additional simple paths [75]. In this approach, additional paths of increasing sum of the desired heuristic, such as length or edge weight, are identified using Dijkstra’s search algorithm on subgraphs of the roadmap. Subgraphs, defined as graphs containing a subset of the nodes and edges of a completed graph, are repeatedly searched until the desired number of paths is returned or until all paths have been explored [46].

While Yen's algorithm is not a computationally efficient search algorithm, it is utilized within this research due to its simplicity as a proof of concept to demonstrate the success of a k-shortest path algorithm. Other k-shortest path algorithms exist that improve upon the computational and storage requirements, such as Lawler's algorithm.

Yen's algorithm searches for additional paths by assuming that the next lowest cost path is a subset of the previous best path. As such, the algorithm searches various subgraphs by removing edges from the previous best path to create the subgraphs and searching the new subgraph for the shortest or lowest cost path, e.g. with Dijkstra's algorithm. To search for k paths, Yen's algorithm creates each subgraph using the concept of spur nodes, spur paths, and root paths. A spur node is the location along the previous best path from which to remove the next subsequent edge and start a new search. A spur path is the new path found from the subgraph by searching from the spur node to the goal node using Dijkstra's algorithm. The root path is the portion of the previous best path from the start node to the spur node. Additionally, to ensure previous shortest paths are not repeated, if the current root path being explored is also contained within a portion of any of the previous shortest paths, the next subsequent edge from that previous shortest path is removed from the current subgraph to prevent the same path from being recovered. After the spur path is found, the total path from the subgraph is the combination of the root path and the spur path. If no spur path is found, then no path is recovered from the subgraph and the next subgraph with a new spur node is searched. This process is detailed below for an example graph where the shortest four paths are recovered.

Given a roadmap, a start node, and a goal node, k paths are identified from the same graph, or until no additional paths are found. A conceptual example of Yen's k-shortest paths algorithm for a directed and weighted graph is depicted in Figure 4.5, where the start node is labeled 'L'; and the goal node is labeled 'P'. Yen's algorithm searches the graph by maintaining two lists, A and B. List A contains the k-shortest paths while list B contains potential candidate paths to be added to list A. First, the shortest path is identified using Dijkstra's algorithm on the completed graph and saved as the first shortest path, A^1. This path, 'L-N-O-P', is highlighted in green in the first

iteration of Figure 4.5. To identify the second shortest path, three subgraphs are created because the first shortest path has three edges resulting in three spur node locations: 'L', 'N', and 'O'. These subgraphs are created by removing one edge from the first shortest path at a time; the edge that is removed is the subsequent edge from the spur node in the first shortest path. The edges that are removed from the graph have their edge weight set to infinity and are depicted in Figure 4.5 with dashed red arrows. Then, Dijkstra's graph search algorithm is used to search each subgraph from the spur node to the goal node for three candidate paths. Since the subgraph created from the spur node 'O' has no possible spur path, this subgraph produces no feasible total path. Therefore, the first two subgraphs produce paths B^1 and B^2, which are sorted in the B list based on total cost. This results in the second shortest path, 'L-N-P', being removed from the B list and added to the A list as path A^2. Path 'L-M-P' still remains in the B list.

For the next iteration, two subgraphs are created since the second shortest path has two edges. The first subgraph is created from spur node 'L'. While this subgraph does result in a valid root path and spur path, path 'L-M-P' is already contained within list B so it is not added again. The second subgraph is created from spur node 'N'. Since the root path 'L-N' exist in both of the previous shortest paths, two edges must be removed: edge 'N-O' is removed because it is the subsequent edge from node 'N' in the first shortest path and edge 'N-P' is removed because it is the subsequent edge from node 'N' in the second shortest path. If only edge 'N-P' was removed from the graph, the first shortest path would be repeated as the next shortest path; this check ensures the top two shortest paths are not repeated and there exists no duplicate paths in list A. With these two edges removed, there are no spur paths that exist, so no additional paths are added to list B. This results in the third shortest path, A^3, being 'L-M-P', and no paths are currently in the B list after this iteration. To identify the fourth shortest path (and subsequent $k > 4$ paths if they exist), the same process of creating subgraphs and producing candidate shortest paths is repeated. This entire process is repeated until k paths are computed or until no more paths exist. For this example, no additional paths exist and the algorithm would terminate with the shortest 4 paths as a result.

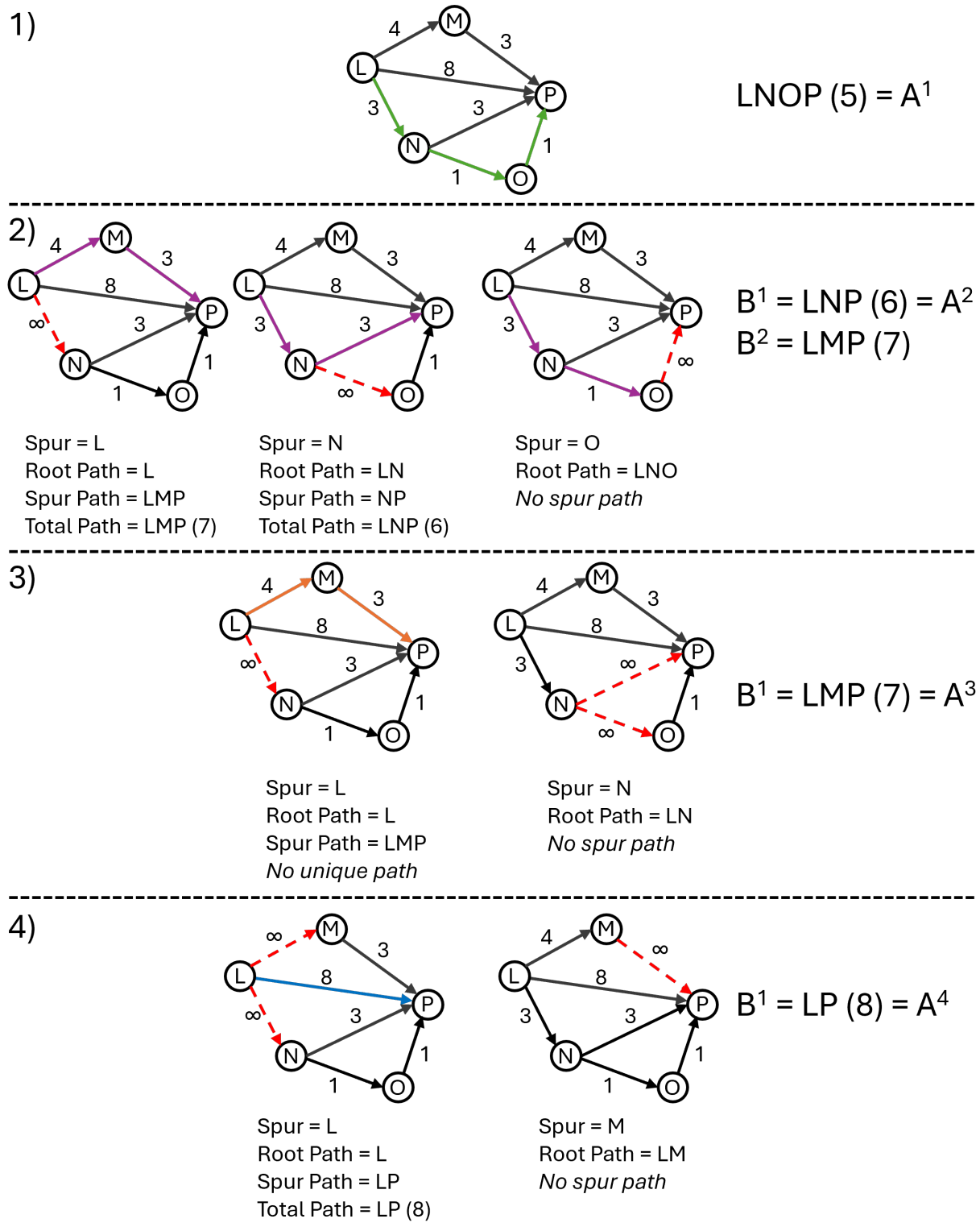


Figure 4.5: A conceptual depiction of Yen's k-shortest paths algorithm for a directed, weighted graph with $k = 4$.

4.2.3 k-Unique Paths Algorithm

With large or densely connected graphs along with the k-shortest paths algorithm, the k-shortest paths may share a large number of edges that are the same. While these paths are in fact distinct paths, they may not support efficient exploration of the available paths within a graph. These returned paths may not be useful for certain path planning problems, such as a driver interested in comparing two distinct routes that take separate highways due to common traffic patterns rather than two routes that take the same highway and only vary by one exit, for example. To overcome this challenge, it is useful to find the k-shortest *unique* paths. While the criteria that determine a unique, or sufficiently different, path may be different for each path planning problem, common approaches focus on identifying paths with limited overlapping edges. These approaches often use similarity metrics that compare the number of shared edges to the number of unique edges for each new path: paths that share no edges have a similarity metric of 0, while paths that share every edge have a similarity metric of 1 [46]. Alternative approaches may incorporate the geometric or topological differences between paths based on partitioning the graph into cells to determine which region the path lies in or which obstacles have been intersected [4]. Then, if the goal is to search for geometrically distinct paths, a designer may require that the k-th shortest unique path must not share, for example, 50% of its edges with a previous shortest unique path or be a specified distance away from the previous shortest unique path.

For the example presented in the previous section, the purple path in Figure 4.5 would not be the second shortest unique path because it shares one of its two edges (edge 'L-N') with the first shortest path. Therefore, the second shortest unique path would be the orange path, path A_3, in the third iteration of Figure 4.5. To incorporate this uniqueness measure into Yen's algorithm, a third list C is created to contain all unique paths, a subset of the shortest paths in list A [46]. The first optimal path A_1 is also added to list C and labeled C_1. Next, as additional paths are found and saved within A, the similarity measure for each subsequent path in A is computed, e.g. $Sim(A_2, C_1)$. If the similarity between path A_2 and C_1 is below a user-specified threshold, path

A_2 is saved as path C_2. Otherwise, path A_3 is the found and compared to C_1. The similarity between the last path in list C and the most recent shortest path in list A is continuously checked as the algorithm searches the graph. The addition of the similarity measure only creates a third list to be saved during Yen's algorithm, but does not modify the algorithm any further where list C will always be a subset of list A. This process is repeated until k paths are found in list A, or a desired k_{unique} paths are found in list C. Additionally, if list C is not unique enough after the algorithm terminates, list C can be created again with a stricter tolerance given a completed list A. Once complete, this approach returns only the subset of paths generated via Yen's algorithm that are deemed sufficiently different according to a specified similarity measure and user-defined uniqueness threshold.

Chapter 5

Kinodynamic Planning for Trajectory Design in a Multi-Body System

As mission requirements, hardware capabilities, or path constraints become increasingly complex, an expert trajectory designer may struggle to both examine the space of feasible motions or identify a viable initial guess. Furthermore, the array of possible motions may deviate significantly from natural dynamical structures generated in low-fidelity models. As a result, there are scenarios where it may be difficult to use existing approaches for initial guess construction and to examine a broader solution space.

In this research, sampling-based kinodynamic planning is used to construct initial guesses for transfers between periodic orbits in the CR3BP while reducing the burden on a human-in-the-loop. First, a graph that efficiently summarizes a portion of the solution space is constructed. Next, the graph is searched for unique solution paths between periodic orbits that will be used as initial guesses in a collocation corrections algorithm. Last, geometrically distinct optimal maneuver-enabled transfers are recovered using a nonlinear constrained optimization algorithm with collocation. This chapter outlines the trajectory design process using sampling-based kinodynamic planning and each step is described in extensive detail throughout this chapter.

5.1 Trajectory Design Process Overview

Sampling-based kinodynamic planning is used to construct initial guesses for transfers between periodic orbits in the Earth-Moon CR3BP with minimal input from a trajectory designer. The technical approach for this research is divided into three key phases: 1) the learning phase, 2) the application phase, and 3) the query phase. First, a generalized graph, called the roadmap, is constructed using randomly sampled nodes and edges during the learning phase which can be divided into 3 key steps:

(1) Construct locally, weakly connected neighborhoods from randomly sampled configurations until the desired solution space is covered

(2) Minimize dispersion, or empty regions, within the configuration space by constructing additional nodes identified through a Delaunay Triangulation

(3) Construct edges between nodes until the roadmap is strongly connected

Next, any additional desired trajectory arcs or spacecraft states along dynamical structures are added as additional edges and nodes to the roadmap during the application phase. The start and goal nodes are also added to the roadmap during this phase if they are not already contained within the roadmap. Last, the query phase occurs where the roadmap is repeatedly searched for unique initial guesses that can be corrected and optimized to recover continuous trajectories. This phase is divided into 3 steps:

(1) Select a search heuristic to optimize along initial guesses

(2) Search roadmap to recover k solution paths that are sequences of natural and maneuver enabled edges that supply unique initial guesses for trajectories

(3) Correct and optimize initial guesses via collocation to recover geometrically distinct maneuver-enabled transfers

Once the learning phase is completed for a desired portion of the solution space, the application phase and/or the query phase may be repeated as many times as necessary to recover additional unique transfers between a new set of start and goal nodes or transfers that optimize a different search heuristic. This process is demonstrated for designing maneuver-enabled transfers between an L_1 and L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP in the remainder of this chapter.

5.2 Phase 1: The Learning Phase

5.2.1 Roadmap Definitions

The learning phase constructs an efficient directed, weighted graph, G, containing nodes and edges, $G = \{N, E\}$, that supplies a discrete summary of a desired region of the continuous solution space in the CR3BP. In this graph, also called the roadmap, nodes $\boldsymbol{n} \in N$ are defined as nondimensional states in the rotating frame, $\boldsymbol{n} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$, where a spacecraft could potentially be located at discrete time intervals. Edges, $\boldsymbol{e} \in E$, represent valid connections between two nodes. Each edges either 1) connects two nodes with the same position vector but distinct velocity vectors via one impulsive maneuver to instantaneously change velocity or 2) connects two nodes with distinct position vectors via a natural trajectory segment. This results in edges that either have an impulsive maneuver to change the velocity at a single location or edges that correspond to a natural arc. The following attributes define each edge between neighboring nodes: $e = \{[(\boldsymbol{n}_i, \boldsymbol{n}_j), t, \boldsymbol{\Delta V}, e_w] \mid \boldsymbol{f}(\boldsymbol{n}_i, \boldsymbol{\Delta V}, t) = \boldsymbol{n}_j\}$. Each edge connects node $\boldsymbol{n}_i$ to $\boldsymbol{n}_j$ using the system's dynamics, $\boldsymbol{f}$, along with either an impulsive maneuver, $\boldsymbol{\Delta V}$, or an integration time, t.

Additionally, edge weights are computed as e_w to reflect the desired objective function that will be optimized during the query phase across all solution paths. In this work, edge weights reflect the cost associated with traversing each edge; this parameter can reflect either any applicable impulsive maneuvers, only the time of flight, or a weighted function that considers both edge weight

attributes:

$$e_w = w_m \|\boldsymbol{\Delta V}\| + w_t t \tag{5.1}$$

where w_m and w_t are tunable weights selected by the user. For this research, by default, the edge weight cost is equal to the total norm of any impulsive maneuvers along an edge normalized by the minimum speed of the nodes in the node pair or a zero edge weight for time-varying edges. For the maneuver weight term, the normalization by speed ensures that edge weights capture the change in direction and relative change in speed for all states within the graph. If the normalization is not included, nodes in sensitive areas of the solution space that possess higher speeds, e.g. close to the Moon, tend to be avoided by the search algorithms due to the larger cost associated with the edges. This edge weight parameter can later be modified to prioritize any other desired trajectory design parameter in the edge weight structure, such as optimizing time of flight only to recover the shortest initial guess trajectories or a weighted function between impulsive maneuvers and time of flight. These terms and the methods used to sample and construct the key components for the roadmap are described within the remainder of this section.

5.2.2 Neighborhood Construction

To begin constructing the roadmap, position vectors, velocity vectors, and Jacobi constants for nodes are independently sampled to construct neighborhoods using randomized sampling in position space, randomized sampling for energy levels, and the concept of 1-neighbors in velocity space [45]. For this application of kinodynamic planning, a neighborhood is considered a local, weakly connected graph that covers a specific spherical region of the configuration, or position, space defined from a center position vector. Neighborhoods are constructed such that a spacecraft that possesses a state vector associated with a single node at the boundaries of a neighborhood can traverse through its local neighborhood for free or with a very low-cost, e.g. a car entering in a neighborhood can drive through it in a short amount of time or using minimal gas. Sampling small local graphs or bundles of continuous edges, rather than individual nodes, that are later globally connected is an efficient method for estimating the connectivity of dynamic environments by elim-

inating the need for a steering method to identify nearby neighbors that may produce a successful edge, a challenge in most kinodynamic motion planning problems [63]. For the neighborhoods utilized within this work, constructing continuous edges within a neighborhood reduces the number of required two-point boundary value problems that need to be solved using numerical methods, such as shooting methods, in order to construct a sufficiently connected roadmap, which can be computationally expensive or unsuccessful in challenging environments [54]. Each neighborhood is constructed similar to a small tree-based graph containing a root node and one branch per direction intended to explore: a position vector is randomly sampled to define the centroid and then subsequent nodes and edges are constructed forward and backward in time to explore the local solution space.

To define the centroid of a neighborhood, a valid position vector is randomly sampled from a uniform distribution within the desired region of the solution space. Random sampling, and subsequently neighborhood construction, is employed in this research as the desired method of node construction. Other common node sampling schemes, such as grid or cell based discretization, require the selection of how coarse or fine to sample nodes. This additional parameter selection may require knowledge of the environment and its dynamics in order to sufficiently construct enough nodes in locations that summarize the solution space. The configuration is only valid if the position vector does not intersect with any known obstacles in the configuration space. Valid configurations must lie within the allowable regions of motion at a given Jacobi constant, i.e. the zero velocity surfaces. For this research, sampled configurations must also satisfy a 500km altitude constraint around the Moon, an obstacle that may be changed depending on mission requirements. At this altitude, motion is found to be dominated by the Earth's gravity and require the use of a three-body model [23]. Additionally, to facilitate an efficient roadmap, valid configurations must not lie within the covered region of a previously constructed neighborhood to ensure minimal overlap between neighborhoods. Therefore, neighborhoods are constructed iteratively to ensure multiple neighborhoods do not cover the same configuration space.

Once the centroid is sampled, a set of 1-neighbors, defined by distinct velocity directions and

randomly sampled Jacobi constants, is constructed for a position vector to span the velocity space and range of available Jacobi constants in the roadmap to complete the full-state node definition. A set of 1-neighbors is constructed for a unique point by defining a set of nearby points for which an edge may be constructed [45]. Neighbors are employed in this research for a specific position vector by creating a graph that explores the local solution space with edges that flow both forward and backward in time. This effectively creates natural flow through the centroid by constructing bundles of edges by combining each backward edge with its naturally continued forward edge. Given a valid position vector in the graph, $\boldsymbol{r}$, its set of 1-neighbors is constructed by varying the instantaneous velocity direction and Jacobi constant; this results in a set of nodes with the same position vector but with distinct velocity directions and speeds. This set of nodes, the 1-neighbors, will then be able to explore the nearby states reachable by propagating each full state, i.e. the position vector along with each associated velocity vector, forward and backward in time. Assuming a valid configuration is sampled for the centroid of a neighborhood, $\boldsymbol{r}$, a set of unit velocity vectors for the 1-neighbors $N(\boldsymbol{r})$ that span the local velocity space is defined as

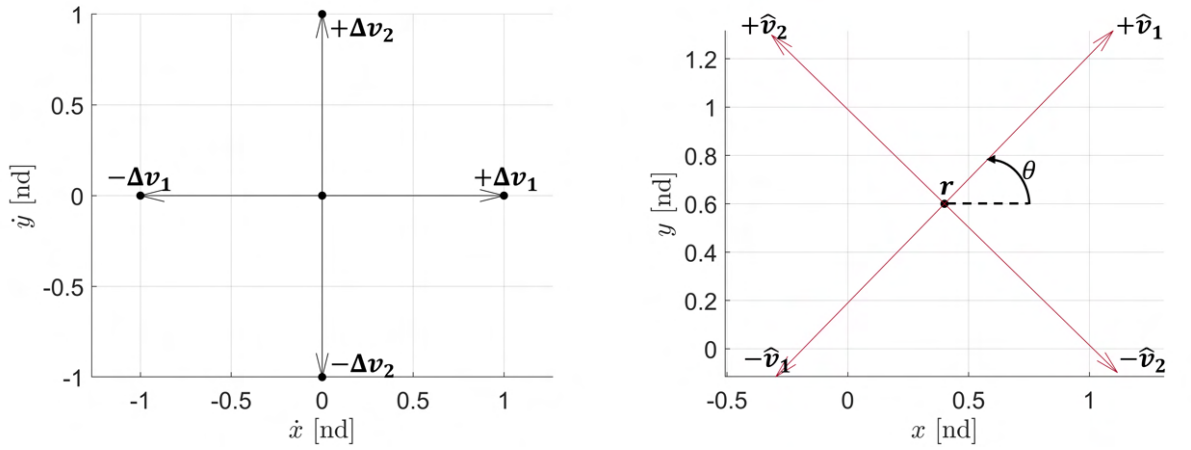
$$N(\boldsymbol{r}) = \{\pm \boldsymbol{\Delta v_i}\} \;\; \forall i = \{1, n\} \tag{5.2}$$

where $\boldsymbol{\Delta v_i}$ represents a unique unit velocity vector direction and the $2n$ vectors are computed such that the angle between each vector is evenly distributed. However, for spatial scenarios, velocity directions can only be evenly distributed within a unit sphere if the number of directions selected is also the number of vertices of a platonic solid, i.e. $n_p = 4, 6, 8, 12, 20$. Thus, in order to distribute any number of velocity directions as uniformly as possible, the Fibonacci sphere is used for cases where $2n \notin n_p$. The Fibonacci sphere supports generating a near-uniform sampling on a sphere by maximizing the smallest neighboring distances among the desired number of points [39].

In an n-dimensional space for a linear path planning problem, there are at most $2n$ 1-neighbors [45]. Therefore, $n \geq \{2, 3\}$ for this implementation for exploring the 2-dimensional planar and 3-dimensional spatial velocity space. However, the value of n selected is a design choice for this dynamic environment; a minimum value of n may construct an efficient roadmap but provide a

coarse representation of the velocity space, whereas a larger value of n will increase the number of nodes and edges in the graph, which will increase the computational time required to construct and search the graph. In this work, n is selected as the minimum recommended value of $n = \{2, 3\}$ for all scenarios explored as a demonstration of the initial representation of velocity space.

For the selected value of n, $2n$ unit velocity directions are constructed along with $2n$ randomly sampled Jacobi constants. In Figure 5.1a, for $n = 2$, there are four unit velocity vector computed by $\pm\Delta\boldsymbol{v_1} = [1, 0]$ and $\pm\Delta\boldsymbol{v_2} = [0, 1]$, resulting in 4 1-neighbors of $\boldsymbol{r}$ denoted $\pm\hat{\boldsymbol{v}}_i$ depicted in Figure 5.1b. Therefore, to explore the velocity space for a unique position vector in the planar



(a) 4 unit velocity directions constructed for planar velocity space.

(b) 4 unique rotated velocity directions for a set of 1-neighbors of $\boldsymbol{r}$.

Figure 5.1: A set of 1-neighbors constructed for $n = 2$ to span a planar velocity space for a given position vector.

CR3BP there are a minimum of 4 velocity directions and 6 velocity directions in the spatial CR3BP created using this set of 1-neighbors. Additionally, to ensure each neighborhood constructed for a position vector and the local solution space it explores is unique, the set of velocity directions $\Delta\boldsymbol{v_i}$ are then rotated from a nominal velocity direction, selected as the x-axis, using randomly selected spherical coordinates. In Figure 5.1b, the set of 1-neighbors are depicted in position space by the black circle, their individual velocity directions are depicted by the red arrows, each rotated by θ from the x-axis.

Finally, to complete each full-state node definition, the speed of velocity needs to be determined. For a given position vector with a unique unit velocity direction, the speed is randomly selected from the allowable energy levels of the roadmap, computed using Eq. 2.29. Each unit vector is then multiplied by its unique randomly sampled speed, all calculated at the centroid's position vector, to produce the velocity vector at a specified Jacobi constant which completes the state definition for each node. This method creates $2n$ nodes with the same position vector, but unique velocity directions and energy levels. This set of completed nodes with a full state definitions will be referred to as the 1-neighbors of a given position vector.

To construct a full neighborhood, each state associated with a node in the 1-neighbor set is then propagated both forward and backward in time for a specified arc length to construct natural trajectory segments that flow through the centroid's position vector, creating bundles of edges that span unique velocity directions and energy levels [63]. By constructing edges both forward and backward in time from a centroid, the neighborhood is constructed such that natural (i.e. free to traverse) motion through a neighborhood occurs from $2n$ unique directions along the boundaries of a local neighborhood. Each edge within a neighborhood is propagated for a specified arc length such that all neighborhoods in the graph are on a similar size, rather than propagating edges for a duration of time, which can lead to larger neighborhoods for states with faster speeds and smaller neighborhoods for states with a slower speed. The arc length each edge is propagated for, ℓ, is selected empirically but may be adjusted as a tunable parameter of the roadmap generation process. For the example demonstrated in this section, $\ell = 0.05$. However the process of selecting this parameter, as well as other governing roadmap parameters, will be described further in Section 5.2.6.

The final state at the end of each edge generated from the 1-neighbor set is additionally saved as a node if the state is valid under the same criteria as described previously. If the node is added, a directed edge that possesses a default edge weight of zero, corresponding to the natural flow of motion between these configurations, is defined forward in time between each pair of nodes; from a final state that was propagated backward in time to the center node or from the center node to

a final state that was propagated forward in time. Ensuring each edge is directed forward in time will assist the search algorithm later by ensuring the final solution paths all flow forward in time as well.

Last, to ensure each new node constructed within the neighborhood from the final state of each new edge will also span its future velocity space and the desirable energy levels, a set of 1-neighbors with randomly selected Jacobi constants is constructed for each node using the same method as described for the center set of 1-neighbors in a neighborhood with one minor difference. Rather than randomly sampling an angle θ to rotate the velocity directions, the velocity direction of the final state along each trajectory arc is chosen to be the first velocity direction in the set and every other velocity direction in the set of 1-neighbors is rotated from this vector.

Thus, the neighborhood consists of a set of $2n$ nodes at the centroid of the neighborhood, $2n$ edges that represent natural trajectory motion forward in time from each center node, $2n$ edges that represent natural trajectory motion backward in time from each center node, and $4n^2$ additional nodes at the final state of each edge that are saved if they are valid in the desired solution space. A neighborhood's radius is then equal to the minimum distance to all of the natural neighboring nodes, i.e. the position vectors associated with the nodes at the end of each of the $2n$ natural edges. By selecting the minimum distance, the local region each neighborhood covers is a conservative estimate of the total area. This radius, along with the position vector associated with the center nodes, define the spherical local region each neighborhood covers. Finally, to ensure a spacecraft can reach each node within each set of 1-neighbors and the 1-neighborhood is weakly connected, a maneuver-enabled edge is constructed between each node with the same position vector to represent a spacecraft being able to instantaneously change velocity with an impulsive maneuver.

An example neighborhood is displayed in Figure 5.2; the solid blue lines depict edges propagated forward in time from its center node with the final state depicted by a blue circle, the solid red lines depict edges propagated backward in time from its center node with the final state depicted by a red circle, and the purple arrows depict each unique velocity direction at a given position vector. For this example, all nodes and edges were constructed at a single Jacobi constant. Once

all nodes and edges are constructed, the area of the configuration space the neighborhood covers is computed; this is depicted by the dashed dark red circle.

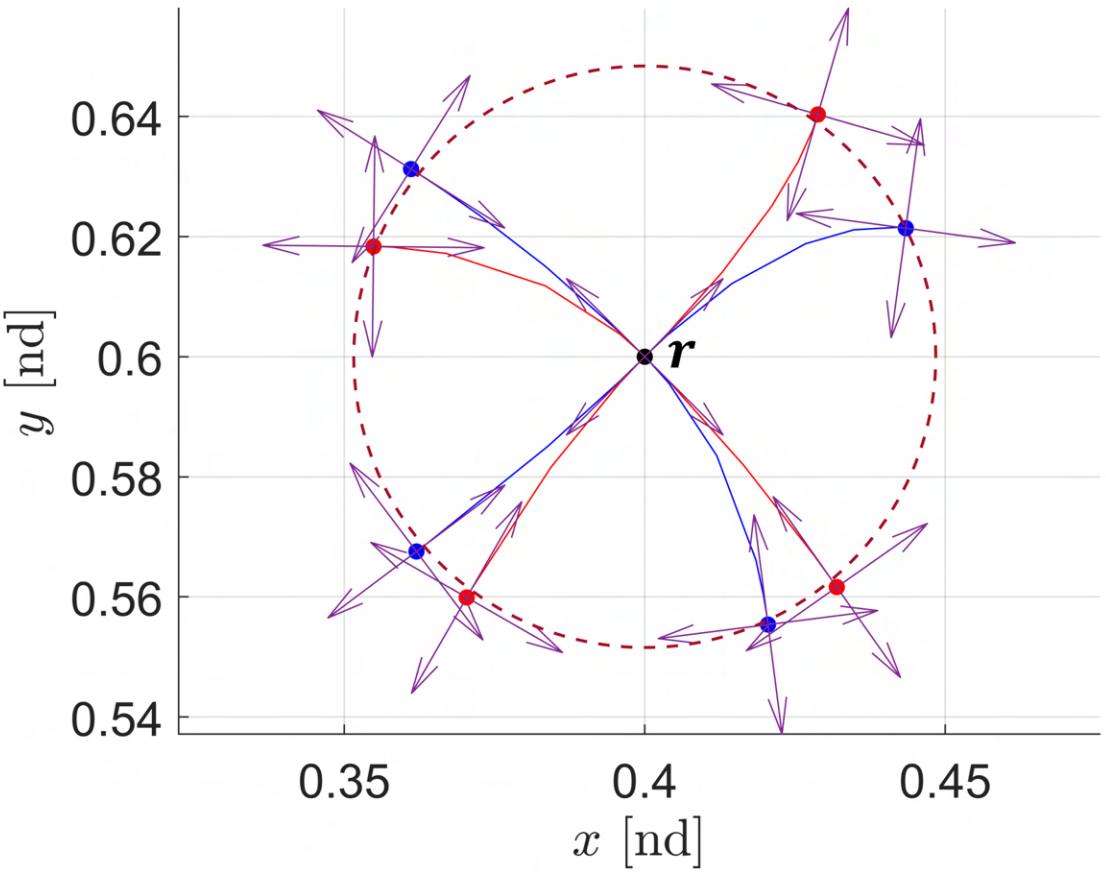


Figure 5.2: An example neighborhood constructed at a single Jacobi constant for $n = 2$ depicting the span of velocity directions defined by the 1-neighbors computed at each node, and the natural edges propagated forward (blue) and backward (red) in time for each 1-neighbor defined at the centroid, $\boldsymbol{r}$.

5.2.3 Graph Coverage

Neighborhoods are iteratively added to the roadmap until the desired region of the solution space is sufficiently covered. For this construction of roadmaps in the CR3BP, coverage is defined by the number of neighborhoods that overlap with any other neighborhood in the graph, n_c. Accordingly, neighborhoods are iteratively constructed until all neighborhoods intersect with at

least one other neighborhood to increase the likelihood of nodes associated with one neighborhood connecting to nodes within one other neighborhood during the global edge construction step; this number is selected empirically but may be adjusted as a tunable parameter of the roadmap generation process. An example roadmap is displayed in Figure 5.3, where the area each neighborhood covers is depicted by a red circle. For this example, neighborhoods are constructed until each neighborhood overlaps with two other neighborhoods, $n_c = 2$. The influence of selecting the value for this parameter will be described further in Section 5.2.6. By constructing neighborhoods and evaluating the coverage iteratively, the resulting graph is determined to be well-covered with very little redundant centroids and natural edges. Furthermore, this approach facilitates an adaptive selection of the total number of nodes and edges within the graph.

5.2.4 Minimize Dispersion in Configuration Space

Once all neighborhoods are added to the graph and the roadmap is deemed covered, additional nodes are added to the graph to minimize the dispersion in the configuration space. The dispersion of the graph is defined by the largest empty sphere in the configuration space [38], i.e. no nodes exist in those regions, which indicates areas that are unreachable by a spacecraft. Empty spheres, or under-sampled regions, in the configuration space are identified using a Delaunay triangulation constructed using the set of all of the unique position vectors associated with the nodes in the graph. A Delaunay triangulation is a geometric spanner that is the dual to a Voronoi diagram; the centers of all circumcircles produced by the triangulation form the vertices of the Voronoi diagram. While a Voronoi diagram can be used to identify a region around each point that is closer to that point than any other point in the set, the Delaunay triangulation is utilized here to capture the empty regions within the space. This method is commonly used in path planning problems to identify areas of a solution space that are insufficiently approximated by a discrete set of nodes or edges [35, 36]. Additionally, Delaunay triangulations have been successfully demonstrated in re-planning path problems where more expansive coverage of the environment is desired or to improve on a first coarse sampling in dynamic environments [43, 61].

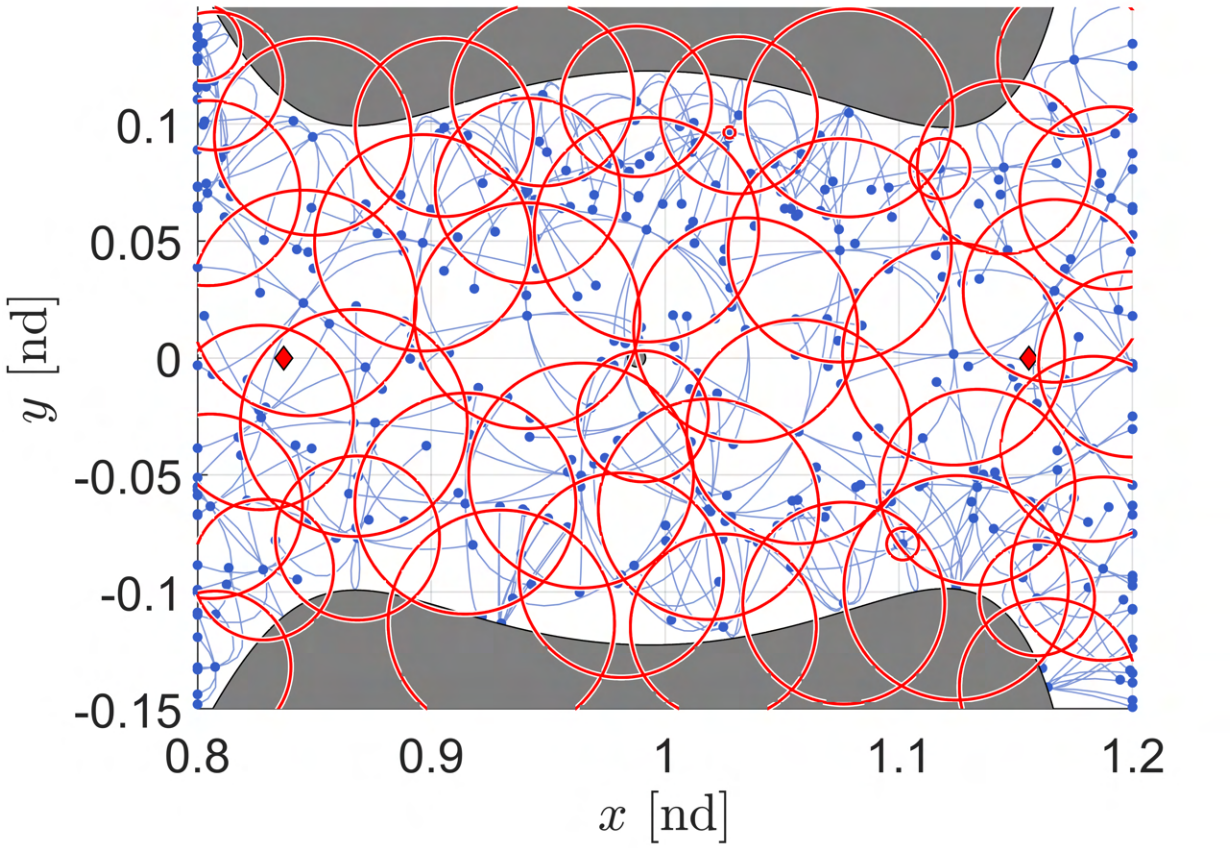


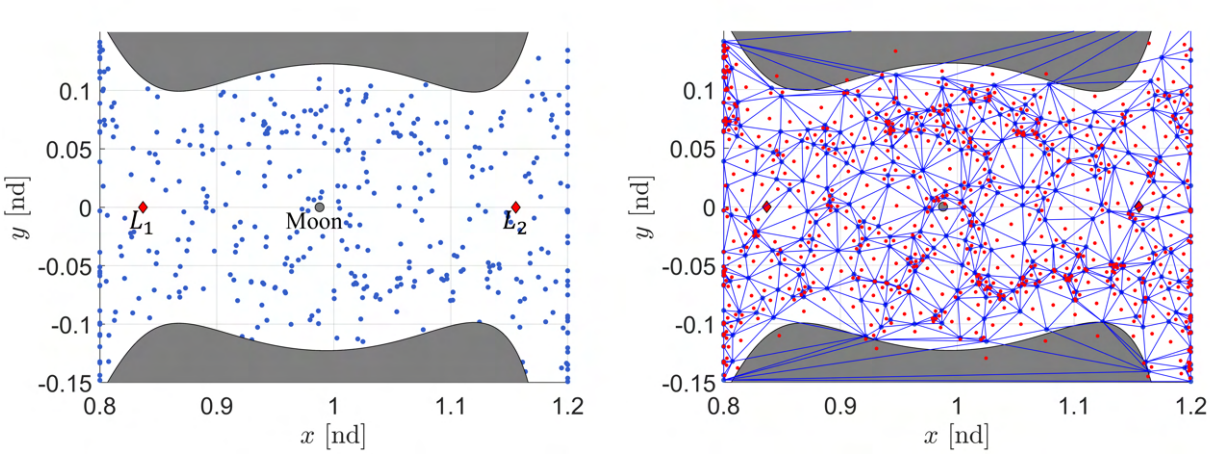
Figure 5.3: An example covered roadmap constructed at $C_J = 3.15$ where each neighborhood intersects with two additional neighborhoods.

An initial Delaunay triangulation is constructed by creating a set of tetrahedra that do not contain any nodes within their circumcircle, where the vertices of each tetrahedron are all current position vectors associated with the nodes in the roadmap. Once the initial triangulation is constructed, the volume of each tetrahedron is computed to order the set of tetrahedra from largest to smallest and the centroids of all tetrahedra are computed. The largest tetrahedron is then used to identify a region within the configuration space that indicates the largest empty region within the graph. Using the largest tetrahedron, the centroid is a candidate for the configuration of a new node. If the configuration is not a valid position vector, e.g. it lies within the zero velocity surface or intersects with any obstacles, the next largest tetrahedron and its centroid is used to seed a new

configuration.

Once a valid configuration is selected, a set of nodes is constructed using the same methods previously discussed; a set of 1-neighbors is constructed to span the velocity space and the speed is randomly selected from the allowable energy levels in the roadmap. This again constructs a set of $2n$ nodes at the configuration within the largest tetrahedra with a valid centroid. Then, the average of the minimum distances to every node's nearest neighbor in configuration space is computed. Additional nodes are added to the graph until the average of all minimum distances stabilizes to within a selected tolerance, τ_{DT}. If this tolerance has not been achieved, a new Delaunay triangulation is constructed after the new configuration, and subsequently the set of nodes, is added to the graph to repeat the process of identifying the next valid configuration to add to the graph. For this example, the tolerance is selected to be 10^{-7} and the influence of this parameter will be described further in Section 5.2.6. If this tolerance is not met between the previous and current iteration, the Delaunay triangulation is recomputed to identify the next largest tetrahedra and construct a new set of 1-neighbor nodes.

A triangulation is recomputed at each iteration to continuously identify the largest empty areas within the configuration space while also taking into account the new nodes added during previous iterations. Through this approach, the resulting graph better covers the configuration space by minimizing the dispersion without the need for a user to select the number of additional nodes to add to the roadmap. An example of a Delaunay triangulation is displayed in Figure 5.4, where the nodes in the graph used to construct the tetrahedra are depicted by blue circles, the triangulation is depicted by blue lines, and the centroid of each tetrahedron is depicted by a red circle. For this example, all nodes are constructed at $C_J = 3.15$ and the zero velocity curve is depicted by the shaded grey areas; centroids located within this area are not be valid configurations for new nodes. If the largest tetrahedron has a centroid that lies within the zero velocity curve, the next largest tetrahedron's centroid is selected as a potential new configuration for the current iteration. This process is repeated until a valid configuration is selected. Figure 5.5 depicts the distribution of all nodes within the graph after the dispersion is minimized; the nodes constructed



(a) Distribution of nodes (blue) after neighborhood construction.

(b) Initial Delaunay triangulation and all centroids (red).

Figure 5.4: Initial Delaunay triangulation and the centroids for a set of nodes at $C_J = 3.15$.

from neighborhood construction are depicted in blue and the nodes constructed from the Delaunay triangulation are depicted in pick. Given the 412 unique position vectors associated with the nodes from neighborhood construction, 129 new position vectors were identified through the Delaunay triangulation process to construct 516 additional nodes. These additional Delaunay triangulation nodes minimize the empty regions within the graph, allowing a spacecraft to access more states. This will also aid in more successful edges constructed during the following step by identifying neighbors that are closer together.

5.2.5 Construct Edges

After all the neighborhoods are constructed and additional nodes are dispersed, additional edges are constructed until the graph is globally strongly connected. A strongly connected graph guarantees that every node within the graph can reach every other node within the graph using the directed edges [15]. Accordingly, a spacecraft that possesses a state vector associated with a single node can reach any other node within the roadmap. For every node in the graph, one neighbor forward in time and one neighbor backward in time is identified. Neighbors are identified using a distance measure that captures both the distance between two nodes in the configuration

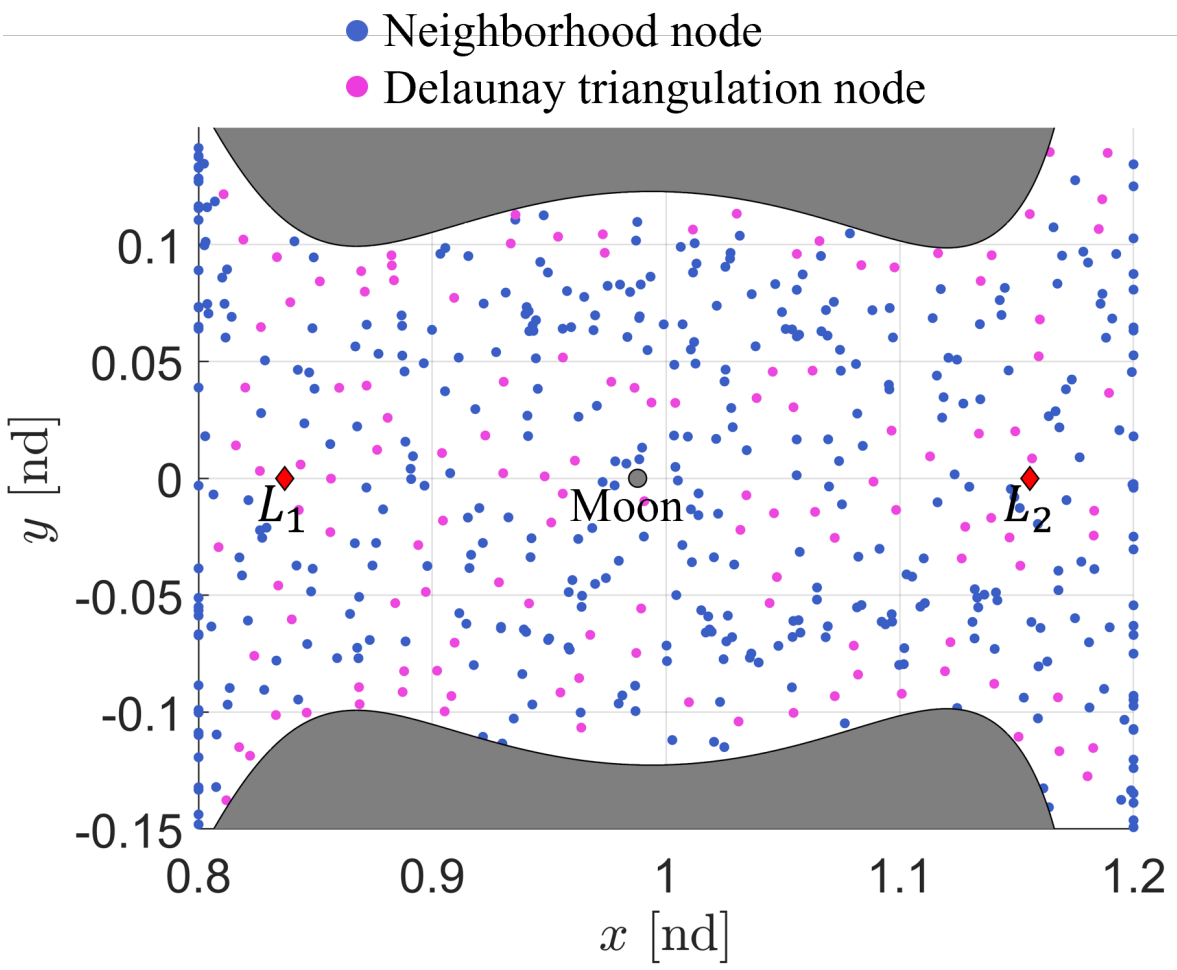


Figure 5.5: Distribution of nodes after additional nodes are added to minimize dispersion.

space as well as the angle between their velocity vectors; the goal is to locate nearby nodes with motion that flows in similar directions. First, to find the best neighbor for node i, the state is first naturally propagated for $\pm\ell$ using the same arc length used to compute the natural edges within each neighborhood. Then, the distance between the final state along this temporary trajectory arc and every other node in the graph is computed:

$$\begin{aligned} d_{i,j} &= \|\boldsymbol{d}_i(t_f) - \boldsymbol{d}_j(t_0))\| \\ D_i &= [d_{1,2}, d_{1,3}, ..., d_{1,j}] \forall j \end{aligned} \tag{5.3}$$

where $\boldsymbol{d}_i(t_f)$ is the position vector produced by propagating node i until $\ell_f = \ell$ and $\boldsymbol{d}_j(t_0)$ is the position vector of the j-th node. Next, the angle between the velocity vector of the final state along

the temporary trajectory arc and the velocity vector of every other node is computed:

$$\begin{aligned} \theta_{i,j} &= \cos^{-1}\left(\frac{\boldsymbol{v}_i(t_f)\cdot\boldsymbol{v}_j(t_0)}{\|\boldsymbol{v}_i(t_f)\|\|\boldsymbol{v}_j(t_0)\|}\right) \\ \Theta_i &= [\theta_{1,2},\theta_{1,3},...,\theta_{1,j}]\forall j \end{aligned} \tag{5.4}$$

Additionally, since the maximum value of the angular difference between two velocity vectors is π, Θ ranges between $[0,\pi]$. Therefore, the distance vector to all neighboring nodes for node i, D_i, is also normalized by scalar values a_i and b_i to compute D_i^* such that the new minimum and maximum bounds are $[0,\pi]$. Then, the node that minimizes the following function is the current best neighbor of node i:

$$n_{best} = \arg\min_j \left(D_i^* + \Theta_i\right) \tag{5.5}$$

If a node is already a neighbor of node i, its value in D_i and Θ_i are set to ∞ to prevent the same neighbor from being selected again.

To connect a node with its identified neighbor, a single shooting corrections algorithm is used to construct a valid trajectory segment between two distinct position vectors, $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$, associated with the node, n_1, and its identified neighbor, n_2. For this formulation, the velocities at the initial state, $\boldsymbol{v}(t_0)$, and final state, $\boldsymbol{v}(t_f)$, along the corrected trajectory segment are allowed to vary, as well as the integration time along the trajectory segment, t_f. The final position vector along the trajectory segment, $\boldsymbol{r}(t_f)$, is constrained to equal the final position vector, $\boldsymbol{r_2}$. Additionally, the Jacobi constant of the initial state along the trajectory segment is constrained to the range of energy levels contained within the roadmap, $C_{J_{max}} > C_J(x_0) > C_{J_{min}}$. This additional constraint ensures useful edges are constructed between neighboring nodes and minimizes the change in energy between nodes and edges; this will aid the search algorithm later in selecting edges that are more 'promising' and will limit the size of impulsive maneuvers at the beginning and end of the trajectory segment. Slack variables, β_{min} and β_{max}, are used to enforce the inequality constraints of the Jacobi constant as equality constraints in the constraint vector. If the roadmap is constructed at a single energy level, these two slack variables and constraints may be turned into one equality constraint.

The free variable vector, $\boldsymbol{X}$, is then constructed as

$$\boldsymbol{X} = [v_x(t_0), v_y(t_0), v_z(t_0), t_f, \beta_{min}, \beta_{max}]^T \tag{5.6}$$

Next, the constraint vector, $\boldsymbol{F}$, is constructed as

$$\boldsymbol{F} = [(\boldsymbol{r}(t_f) - \boldsymbol{r_2})^T, C_J(x_0) - C_{J_{min}} - \beta_{min}^2, C_{J_{max}} - C_J(x_0) - \beta_{max}^2]^T \tag{5.7}$$

Using these definitions, the free variable vector is updated using Newton's method in Equation 2.41 until the norm of the constraint vector equals zero within a desired tolerance, selected as 10^{-12} for this research. While the Jacobian for this formulation is calculated analytically, it is rectangular and, therefore, not invertible, so the minimum norm solution is used in the update equation.

To construct an initial guess for the single-shooting scheme, the selected node is propagated naturally forward or backward in time until the arc length is ℓ, the same arc length used to construct neighborhoods and identify neighboring nodes. The elapsed time at the state along this arc that is closest in position space to the neighboring node seeds the integration time for the initial guess while the original velocity of the first node is used to seed the velocity at the first state along the arc for the initial guess. Additionally, similar to the nodes, each edge is also subject to a 500km altitude constraint around the Moon. If any periapsis state along this trajectory segment lies below this altitude, the edge is not saved and a new neighbor is identified.

If a maneuver-enabled edge is successfully computed via single-shooting, the impulsive maneuver magnitude at the beginning and end of the trajectory segment is further reduced via local optimization implemented using sequential quadratic programming within the fmincon tool in MATLAB® [49]. Using the optimized solution, the maneuver magnitude serves as the edge weight for every edge that changes distinct velocities between identical positions. Since the resulting trajectory segment from edge construction directly influences the edges and edge weights added to the graph, optimizing each solution for the sum of maneuver magnitudes squared is critical to ensure the discrete graph representation is as accurate as possible. If optimization were not included as a necessary step during edge construction, the computational time would be reduced, however, the

total cost of all maneuvers across the solution paths would increase. This may result in coarser initial guesses that could be more difficult to recover continuous solutions during trajectory corrections. If the single shooting corrections scheme fails, the corresponding terms in the neighbor distance function, D_i and Θ_i, are set to infinity and a new neighbor may be identified with new edges attempted to be constructed.

This process of identifying a successful neighbor may be attempted as many times as desired, but to limit the computational time and increase the likelihood of constructing useful edges, a limit on the number of neighbor attempts should be set. For this research, a limit of 20 neighboring nodes may be identified; if all 20 attempts are unsuccessful, no neighbors will be connected and these nodes will be labeled as 'failed'. If a significant number of nodes are 'failed' nodes, then this may indicate a poor selection of the roadmap parameters and an insufficient number of nodes in specific or sensitive regions.

The trajectory segment computed as a result of this corrections algorithm is used to connect the positions of the node with the position of its neighboring node using three edges; two single impulse edges to change the velocity at each node and one natural edge to connect the position vectors. The first maneuver-enabled edge is used to change the velocity from the initial node to the initial state along the new trajectory segment. The second edge is a natural trajectory segment that is equal to the solution of the single shooting corrections algorithm, resulting in a time-varying edge. The third edge is another maneuver-enabled edge that is used to change the velocity from the final state along the new trajectory segment to the velocity of the associated center node. The neighbor identification and edge construction process are depicted in Figure 5.6 for node n_i. First, the state associated with node n_i is propagated forward in time, depicted by the black arc. Then, the best neighbor is the node that minimizing the distance objective function from the end of the temporary trajectory arc, depicted by the small open black circle. This results in node n_j being selected as the best candidate for a neighbor. To construct the edges from node n_i to node n_j, the optimized trajectory arc is denoted by the blue arc. The first edge changes the velocity from the velocity of node n_i to the velocity of the initial state along the trajectory arc, n_a, which is

additionally saved as a new node. The next edge is a time-varying edge between node n_a and node n_b, the final state also saved as a new node. The third edge changes the velocity from node n_b back to node n_j.

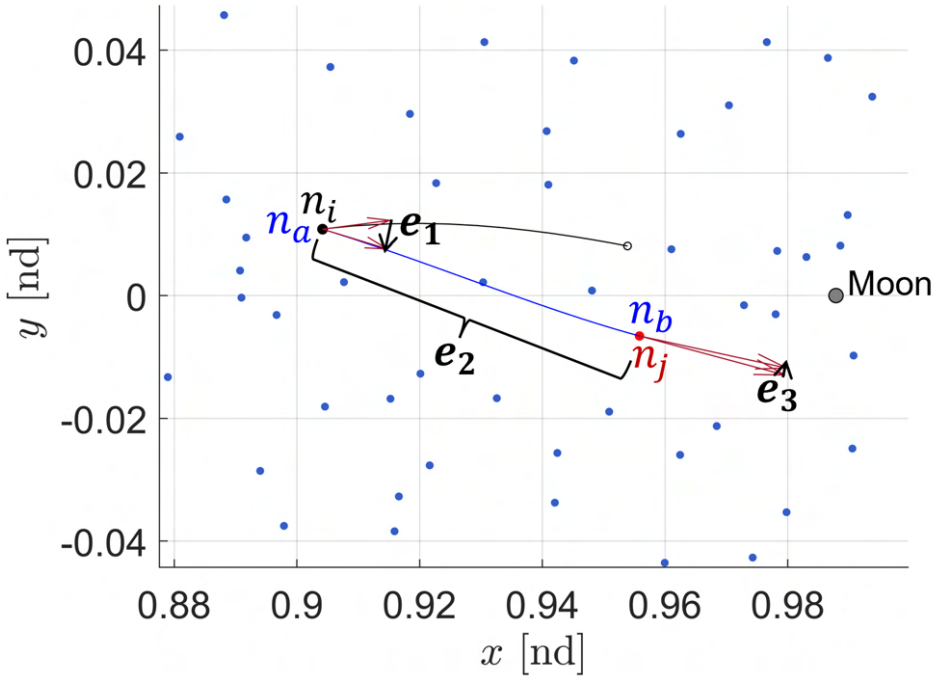


Figure 5.6: New node and edge construction to connect neighboring nodes, n_i and n_j.

After every node gains one new neighbor forward and backward in time, the connectivity of the graph is checked. If the graph is not strongly connected, the number of components within the graph are identified. Any nodes not contained within the largest component, i.e. the component with the largest number of nodes, and not labeled as 'failed' nodes, gain one additional neighbor forward and one additional neighbor backward in time. This process reduces the number of redundant edges being constructed in the largest component within the graph and encourages additional connections within the smaller weaker components of the graph. This process is repeated until the entire graph is strongly connected, facilitating an adaptive number of edges and a strongly connected, yet efficient graph.

A completed roadmap that summarizes the planar solution space in the Earth-Moon CR3BP

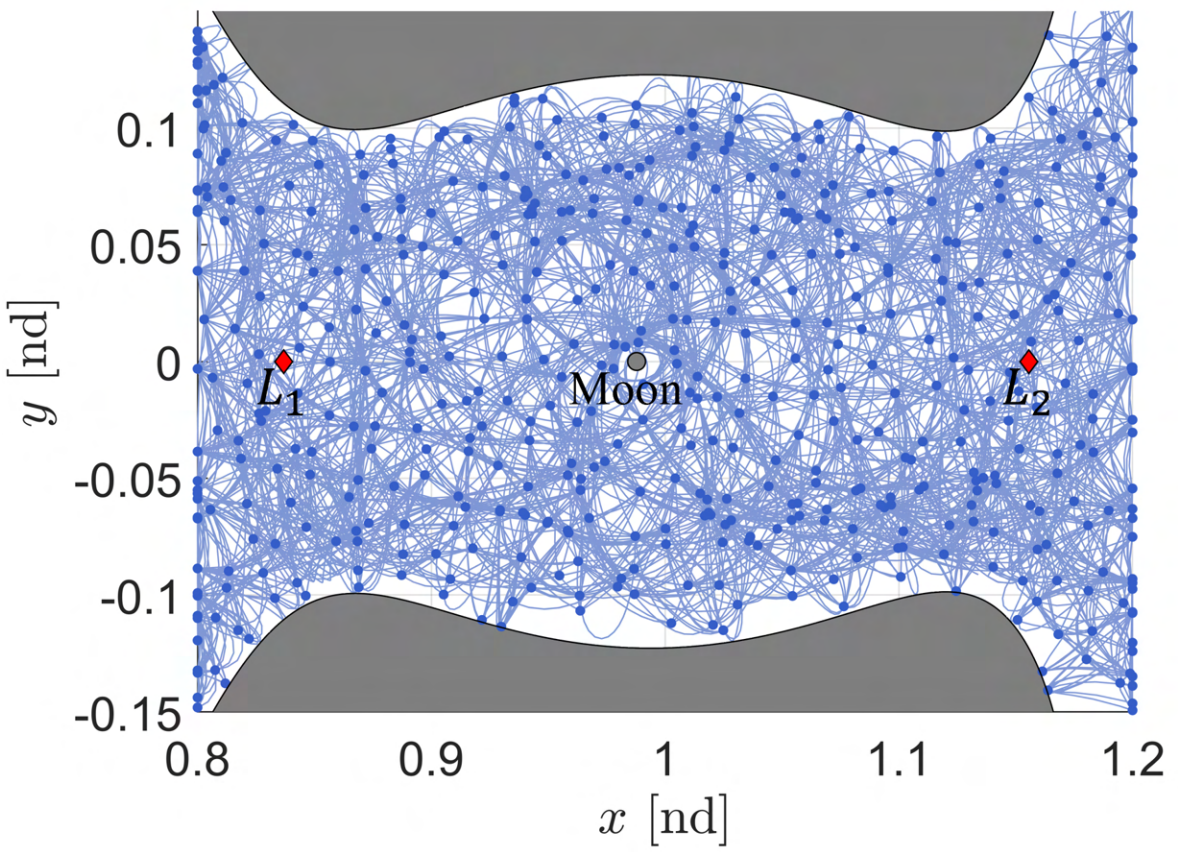


Figure 5.7: A completed roadmap generated to summarize the planar solution space in the lunar vicinity at $C_J = 3.15$.

at $C_J = 3.15$ appears in Figure 5.7. This graph contains 541 unique position vectors, 10,295 nodes, and 215,006 edges. Since the roadmap is constructed at a single value of Jacobi constant, all nodes and edges are constrained to $C_J = 3.15$. Therefore, no nodes or edges enter within the zero velocity curve, depicted by the grey region in Figure 5.7. Additionally, while the methods presented in this subsection are intended to construct an efficient discrete representation, there is no guarantee for an optimal graph or guaranteed number of nodes and edges constructed. There are still denser and sparser regions of nodes throughout the configuration space, which also leads to regions containing a various number of edges, e.g. nodes do not extend all the way towards the zero velocity curves therefore there are fewer edges in this region. However, the randomness of

the nodes and natural bundles of edges constructed for neighborhoods were found to improve the smoothness of the recovered solution paths while overall reducing the number of human selected parameters and limiting the required knowledge of the solution space.

5.2.6 Influence of Roadmap Parameters

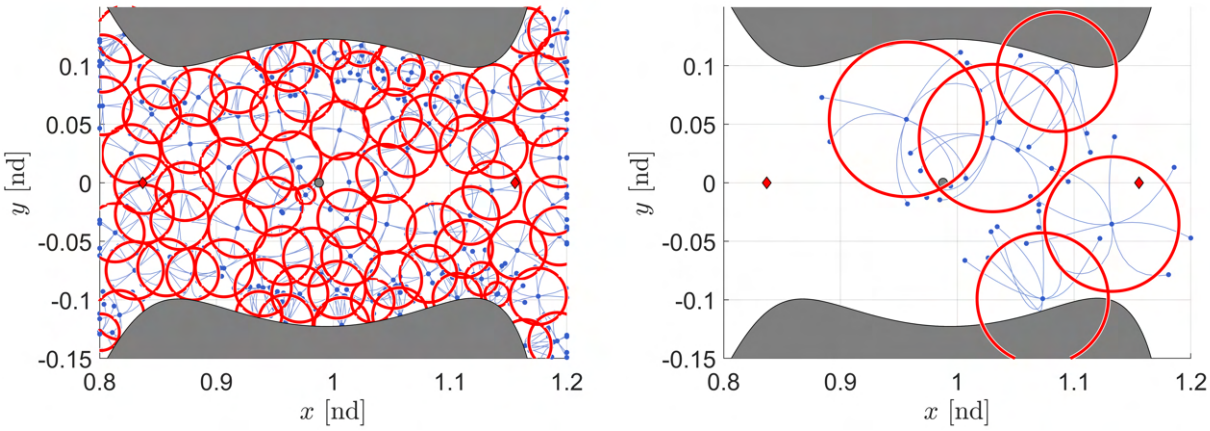
To construct an efficient graph, three roadmap parameters need to be selected by a user: the arc length parameter for all natural edges, ℓ, the number of neighborhoods coverage parameter, n_c, and the stopping tolerance for the Delaunay triangulation, τ_{DT}. All three parameters are dependent on each other, modifying one parameter will influence the selection of the other two parameters. The arc length parameter dominates the selection of the graph influencing multiple steps throughout the roadmap construction process; this parameter not only dictates how much area each neighborhood covers, but also how long each edge is naturally propagated for which is used to find neighbors during edge construction as well as constructing the initial guess for edge construction. Additionally, these parameters should be selected with the desired results in mind. A selection of these parameters that results in fewer nodes in the graph will result in a shorter computation time, but also more discrete representation of the solution space; this roadmap may result in a rougher approximation of feasible trajectories recovered from the initial guesses searched for during the query searches.

When selecting the arc length parameter, there are two major considerations: 1) how many bundles of edges of natural flow through the graph are desired and 2) how large of an area in the configuration space is covered by each neighborhood. By selecting a smaller ℓ, the number of neighborhoods required to cover the graph increases regardless of the selection of n_c since each neighborhood covers a smaller portion of the configuration space. This will result in a graph that contains a larger number of bundles of edges that flow through each neighborhood which can help capture the flow through the solution space and also result in 'smoother' initial guesses recovered, i.e. trajectories with fewer discontinuities. Furthermore, the nodes from all neighborhoods will already be closer together, most likely creating a denser distribution of nodes in the configuration

space. This may result in fewer nodes added during the triangulation step to minimize dispersion, depending on the stopping tolerance τ_{DT} selected.

When a larger value of ℓ is selected, fewer neighborhoods will be required to cover the graph since each neighborhood will locally cover a larger portion of the configuration space but this will result in a sparser distribution of nodes constructed during this step. While a larger value of ℓ may result in longer bundles of edges, leading to traversing the roadmap quicker using longer trajectory arcs of natural motion, too large a value for ℓ will result in bundles of edges that traverse the graph too quickly and may skip over potential smaller paths of natural motion. Two examples of the influence on arc length parameter selection during neighborhood construction are displayed in Figure 5.8 with $n_c = 1$. In Figure 5.8a, an arc length of $\ell = 0.025$ is selected. This results in smaller neighborhoods and a denser distribution of nodes which will require a larger computation time to complete the graph since there are more nodes that require edges constructed during the final global edge construction step.

Nevertheless, the desired region of the solution space is well covered by a majority of the neighborhoods and there are few empty regions in the configuration space, such as the small empty region between the Moon and L_2. In Figure 5.8b, an arc length of $\ell = 0.075$ is selected. This results in fewer, but larger neighborhoods constructed and a sparser distribution of nodes. In this example, only 5 neighborhoods were constructed since technically each neighborhood is intersecting with one other neighborhood, the value selected for n_c. This demonstrates the importance of selecting parameters ℓ and n_c together; this example produces a poor representation of neighborhood construction and even with the addition of nodes through a Delaunay triangulation, there are large regions of the solution space that do not contain the critical bundles of edges to allow natural flow through the space. The selection of ℓ will also directly affect the selection of τ_{DT} as well as n_c. Other deterministic methods for constructing nodes, such as sampling along a grid, or coverage evaluation, such as coverage-based path planning algorithms, may provide an alternative representation of the solution space [70]. However, these methods may increase the dependency on a human-in-the-loop.



(a) $\ell = 0.025$. (b) $\ell = 0.075$.

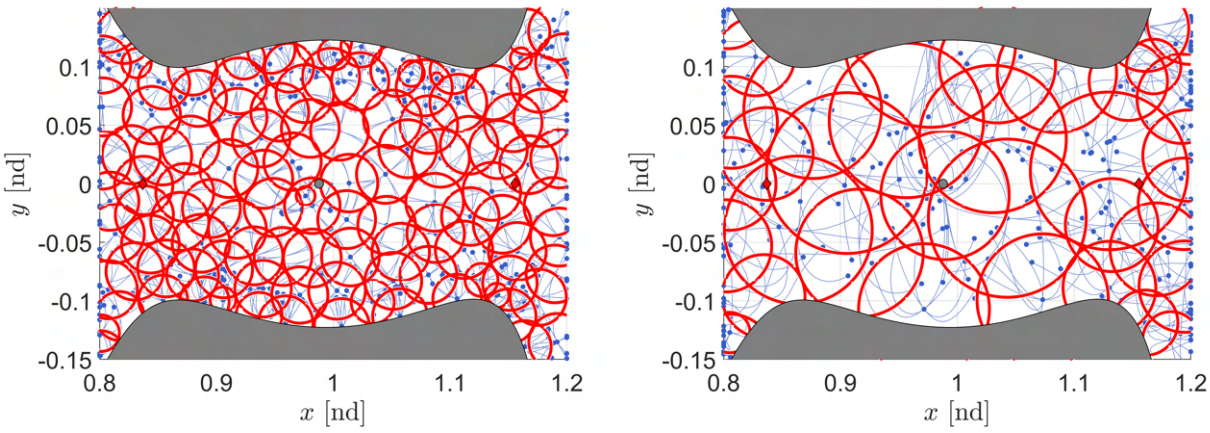
Figure 5.8: Neighborhood construction for various sizes of ℓ with $n_c = 1$.

While the selection of ℓ influences how large each neighborhood will be, the selection of n_c influences how densely the roadmap is covered by these neighborhoods by determining when to stop constructing neighborhoods. When a lower value of n_c is selected along with a smaller arc length, there may be portions of the configuration space that are not well-covered; this may lead to areas unreachable by the spacecraft unless nodes are added in these areas through minimizing dispersion. If a lower value of n_c is selected along with a larger arc length, there many only be a few neighborhoods constructed if the neighborhoods were all large enough to overlap with each other early on in the neighborhood construction process.

There also may be a limit on the number of overlapping neighborhoods possible for a desired region of the solution space. During neighborhood construction, there may be instances where every neighborhood but one has the desired number of overlapping neighborhoods. Often, this last remaining neighborhood lies near the boundaries of either the desired region of the solution space or along the boundaries of feasible motion, e.g. the zero velocity curves. When this occurs, it is recommended to limit the number of samples attempted for a centroid of a neighborhood. For this research, if 1000 samples are drawn that are not valid, then the roadmap is deemed covered. This is to limit the case in which there are truly no additional neighborhoods that can be added to

the roadmap that do not already intersect with other neighborhoods or lie within the zero velocity curves producing imaginary motion. In this instance, it would be similar to not selecting a value of n_c and simply constructing neighborhoods until no more valid samples were drawn.

In Figure 5.9, two selections of n_c are shown using the example neighborhood distributions shown in Figure 5.8. Since the roadmap constructed using a smaller value of ℓ already had a denser sampling of neighborhoods, a value of $n_c = 2$ was sufficient to fill out the remaining noticeable empty regions within the solution space while the roadmap constructed using the larger value of ℓ required a selection of $n_c = 3$ to cover the same approximate area. Despite being covered, the graph in Figure 5.9b still contains larger regions of empty space that do not contain any nodes. This roadmap would benefit from a stricter tolerance on τ_{DT} whereas the graph in Figure 5.9a might not require the addition of as many new nodes to achieve the same distribution.



(a) $n_c = 2$.

(b) $n_c = 3$.

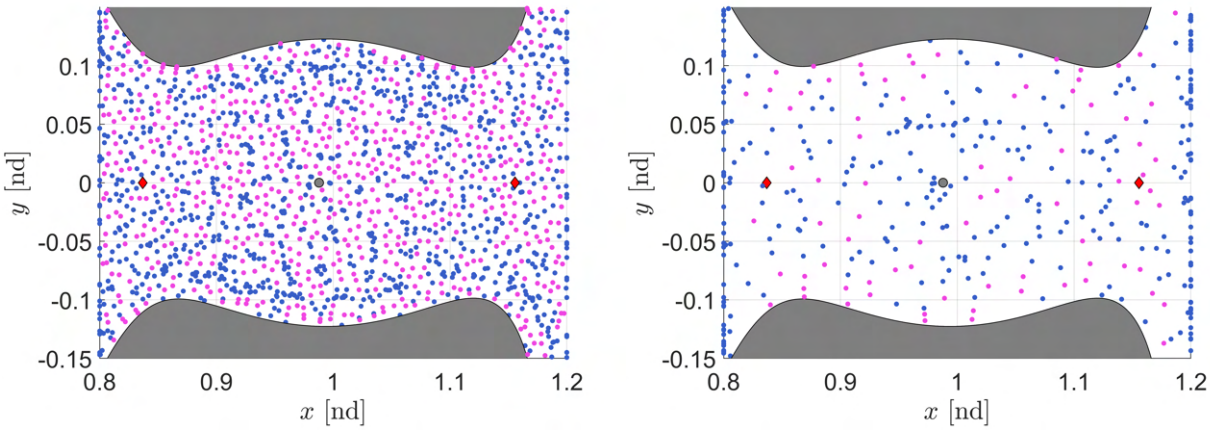
Figure 5.9: Number of overlapping neighborhoods, n_c, required for the roadmap to cover the desired region in the solution space.

Once the roadmap is considered covered based on the selection of ℓ and n_c, the Delaunay triangulation step minimizes dispersion through the configuration space by adding addition nodes at identified position vectors in regions of empty space. Depending on the size of neighborhoods and the selection of τ_{DT}, the nodes from the neighborhoods might already meet the tolerance selected

without the need to add any additional nodes. Most often, as the size of the neighborhoods decreases and the number of overlapping neighborhoods increases, the fewer nodes will be added during the minimizing dispersion step. Therefore, this parameter selection mainly influences the density of nodes and ensures there are no larger empty regions within the configuration space that may have been missed during neighborhood construction. If a larger density of nodes is desired or if computational time is not a factor in the consideration for parameter selection, a stricter tolerance may be set for τ_{DT}.

In Figure 5.10, two selections of τ_{DT} are shown using the example neighborhood distribution and coverage criteria in Figure 5.9 where the nodes added from the neighborhood distribution are depicted by blue circles and the nodes added from the Delaunay triangulation are depicted by pink circles. For the smaller arc length neighborhood distribution, there was already a higher density of nodes since there are more neighborhoods in this graph. When selecting the stopping tolerance for this roadmap, $\tau_{DT} = 10^{-5}$ was already achieved and resulted in no additional nodes added to the graph whereas $\tau_{DT} = 10^{-8}$ results in a very dense graph and minimal empty regions left in the graph, as shown in Figure 5.10a. This graph will likely require a larger computational time for both graph construction and search queries due to the increase in number of nodes and edges in the graph. This graph may also produce a larger number of useful initial guesses, i.e. initial guesses with fewer discontinuities, because the graph may better capture the connectivity of the solution space by spanning the position space and velocity space with more nodes and edges. In Figure 5.10b, even though the selected parameters for neighborhood distribution and coverage resulted in fewer neighborhoods and larger empty portions of the configuration space, a stopping tolerance of $\tau_{DT} = 10^{-6}$ greatly increases the distribution of nodes throughout the configuration space while creating a more efficient, sparser graph. This selection of parameters may result in a coarse construction of initial guesses for transfers as well as fewer useful solution paths recovered since there is overall a fewer number of nodes and edges within the roadmap.

When selecting all three roadmap parameters, it is important to consider how all parameters affect each other and their influence on the roadmap together. In Figure 5.11, the number of



(a) $\tau_{DT} = 10^{-8}$.

(b) $\tau_{DT} = 10^{-6}$.

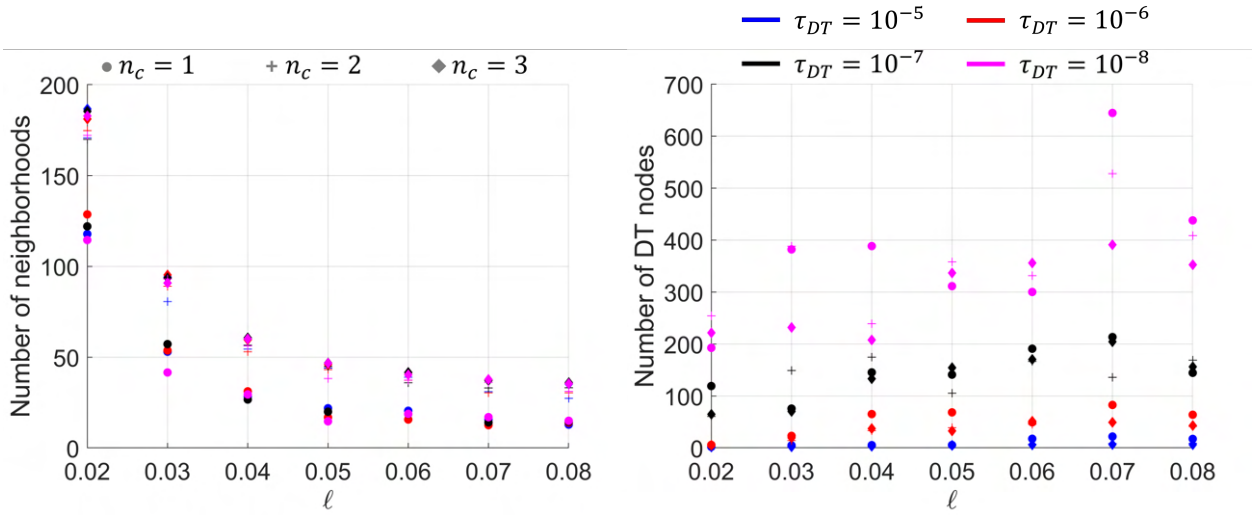
Figure 5.10: Stopping tolerance, τ_{DT}, for adding additional nodes to minimize dispersion in the configuration space.

neighborhoods added to the graph and the number of additional nodes added to the graph through the Delaunay triangulation process are explored for various values of arc length ℓ, coverage metric n_c, and stopping criteria τ_{DT}. Arc length is explored along the x-axis of both figures, while the coverage metric and stopping criteria are denoted using various markers and colors using the following scheme. Three values of n_c are explored and denoted using three marker types: $n_c = 1$ is denoted by a filled in circle, $n_c = 2$ is denoted by a plus sign, and $n_c = 3$ is denoted by a filled in square. Four values of τ_{DT} are explored and denoted using four colors: $\tau_{DT} = 10^{-5}$ is denoted using blue, $\tau_{DT} = 10^{-6}$ is denoted using red, $\tau_{DT} = 10^{-7}$ is denoted using black, and $\tau_{DT} = 10^{-8}$ is denoted using magenta. For each combination of parameters, 10 simulations were ran to compute the average number of neighborhoods constructed and the average number of additional nodes added to minimize dispersion.

In Figure 5.11a, the parameter curves along $n_c = 2$ and $n_c = 3$ begin to overlap for the examples shown in this section due to the maximum number of neighborhoods constructed for the desired portion of the solution space A selection of $n_c \geq 3$ would likely result in the same distribution of neighborhoods based on the selected arc length value. Additionally, as the value of

arc length increases, the number of neighborhoods constructed levels out. This is again due to the maximum number of neighborhoods allowed to be constructed within the desired solution space. When this occurs, the value selected for τ_{DT} becomes more relevant based on how dense the user requires the graph to be.

In Figure 5.11b, the distribution of number of nodes added to minimize dispersion has less of a defined trend. The selection of this parameter will largely depend on the overall goals of the graph, e.g. how dense the node distribution is, and the desired results of the query phase. Generally, the smaller the neighborhoods, i.e. small ℓ, and more neighborhoods constructed, i.e. larger n_c, the less nodes are added through a Delaunay triangulation. For this parameter, the trend is simply as the tolerance decreases, the number of nodes increases. Due to the rapid construction of neighborhoods due to random sampling and relative small propagation time, it is recommended to explore these three parameters before constructing a roadmap for a new environment and selecting all three parameters simultaneously.



(a) Arc length versus number of neighborhoods. (b) Arc length versus number of new nodes dispersed.

Figure 5.11: Number of neighborhoods and number of nodes added to minimize dispersion for various parameter selections of $\ell = \{0.02 - 0.08\}$, $n_c = \{1, 2, 3\}$, and $\tau_{DT} = \{10^{-5} - 10^{-8}\}$.

5.3 Phase 2: The Application Phase

During the application phase, additional nodes and edges that are relevant to the search query and/or mission parameters are connected to an existing roadmap. This phase may be repeated as necessary to alter the information contained within the roadmap and recover different results from the query phase described in the following section.

5.3.1 Connect States Along Relevant Dynamical Structures

To complete the roadmap for a specific search query, states along the desired initial and final orbits are projected onto the roadmap as new nodes that will be used as the set of start and goal nodes, respectively. These new nodes are first connected to its nearest neighboring node along the same orbit using a natural edge, such that the initial and final orbit are free to transverse and the initial guess can connect to any location along the orbit. Next, a set of 1-neighbors is constructed for each state along the orbit; this is to vary the geometries of edges departing and arriving to the initial and final orbits. Then, each new node with a position vector along the periodic orbits is connected to its nearest neighboring node already contained within the roadmap. Nodes added along the initial orbit are propagated forward in time to identify neighbors while nodes added along the final orbit are propagated backward in time to identify neighbors to ensure all initial guesses will depart along edges leaving the initial orbit and end on edges arriving onto the final orbit. If the graph will be searched for solution paths that transfer between both orbits, each node will gain a neighbor both forward and backward in time. Neighbors are identified using the same distance measure used in the generalized roadmap described in the previous section. Similar to increasing the number of velocity directions for a given position vector in the graph, increasing the number of edges departing or arriving to the initial and final orbits, respectively, can increase the available departure or arrival geometries for the final solution paths recovered during Phase 3 if desired.

Next, any additional mission specific information relevant to the search query can also be projected onto the generalized roadmap. If specific trajectory arcs must be included within a final

solution path, these segments can be added during this step and later used within the query phase as additional start and goal nodes. Otherwise, if the desired initial or final orbit possesses stable or unstable manifolds, states and small trajectory segments that lie on these manifolds may be added to the roadmap as nodes and time-varying edges, respectively, to improve the natural flow of the graph.

Given a set of available trajectories, small arcs along randomly selected trajectories can be added to the graph. First, a random state along a trajectory from the available invariant manifolds or set of known trajectories is saved as a new node for the graph. Next, this state is propagated for the same arc length, ℓ, as the other natural edges constructed for the roadmap. Last, the final state along this trajectory segment is additionally saved as a new node for the graph. This process creates two new nodes and one natural, zero-weight edge between the two nodes for the graph. Next, each new node is then connected to the generalized roadmap's nodes using the same distance measure to identify neighboring nodes. For each trajectory segment added to the roadmap, the initial node and final node along this new segment are connected to its $k \geq 2n$ nearest backward and forward neighbors, respectively. This is selected to ensure the position vector associated with the state at the beginning of the trajectory segment is connected to at least $2n$ other nodes, the same minimum number of edges every other position vector in the graph receives. The number of neighbors used to connect the new manifold trajectory segments to the generalized graph is selected as a balance of computational power and memory requirements with how desirable the trajectory segments along the invariant manifolds are to utilize during the final graph searches during Phase 3. If these additional segments should be prioritized, additional backward and forward edges can be constructed such that a larger portion of the solution space may connect to the desired time-varying edge by identifying more neighbors within the vicinity of the initial and final states. Each of these nodes gains three edges per identified neighbor, using the same methods as previously described for constructing new edges during Phase 1. This process is repeated until the desired amount of information from the available information about transport mechanisms is connected to the graph.

5.3.2 Identify Start and Goal Nodes

Last, the desired start and goal nodes for the search queries are identified. The original states along the desired initial periodic orbit serve as the set of start nodes and the original states along the desired final periodic orbit serve as the set of goal nodes. If any additional states can be considered for a start or goal node, they must be added to the graph during this phase. Figure 5.12 shows the generalized roadmap added with 50 states along an L_1 Lyapunov orbit at $C_J = 3.15$ and 50 states along an L_2 Lyapunov orbit at $C_J = 3.15$ connected to the roadmap. This step added 1390 new nodes to the graph and 15443 new edges. If different periodic orbits were desired as different start or goal nodes, or small trajectory arcs were desired, this phase may be repeated to change the graph to fit the desired search parameters.

5.4 Phase 3: The Query Phase

5.4.1 Select Search Heuristic

After the generalized roadmap is constructed and any additional relevant mission information is connected to the roadmap, the query phase searches the graph for valid initial guesses for transfers between the desired periodic orbits. Dijkstra's search algorithm is used to find the optimal solution path within the roadmap. This path produces the lowest cost sequence of nodes and edges that minimizes the cumulative sum of the selected heuristic for all edge weights across the path. This path then supplies an initial guess for a transfer between the selected periodic orbits. Using Equation 5.1, the default search heuristic minimizes the cumulative sum of impulsive maneuvers normalized by the minimum speed of the nodes in each node pair, i.e. $w_m = 1$ and $w_t = 0$. This results in all edges that change velocity between a pair of nodes with the same position vector with a nonzero edge weight and all edges that are time-varying between two unique position vectors with a zero edge weight.

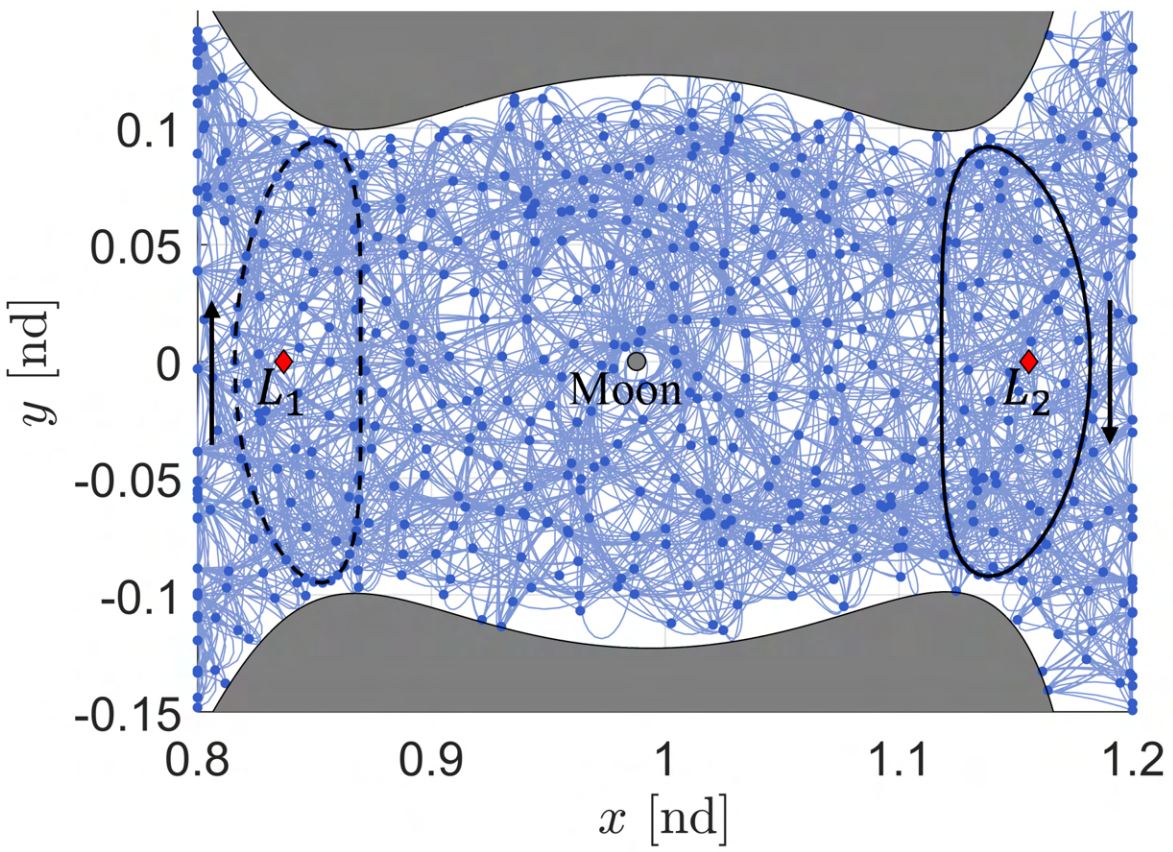


Figure 5.12: A completed roadmap generated to construct transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit at $C_J = 3.15$.

5.4.2 Search for Unique Initial Guesses

Typically in PRM, if no path exists between the selected start and goal node(s), the search algorithm will terminate with no solution. However, since the graph is constructed to be strongly connected, at least one path is guaranteed to exist since at least one path exists between every pair of nodes in the graph. In fact, if n_{PO} states along each initial and final orbit are added to the roadmap, this guarantees n_{PO}^2 solution paths exist within the graph, since there exists at least one path for every unique start and goal node selected. Once the roadmap is completed, Dijkstra's search algorithm is used to search the graph for the optimal initial guess for a transfer that can depart the L_1 Lyapunov orbit and arrive onto the L_2 Lyapunov orbit. This optimal initial guess,

depicted in Figure 5.13, is the lowest cost path in the completed graph that minimizes the sum of the scaled maneuvers. This initial guess corresponds to a path that is a sequence of nodes and edges that represent a sequence of instantaneous impulsive maneuvers and natural trajectory arcs. Each node location, depicted by circles, is colored by the maneuver magnitude required at that position vector. The initial guess has a total cost of $c(P_1) = 1.7630$, which corresponds to a total maneuver magnitude of 418.4201 m/s before corrections and further refinement. However, since the graph is only a discrete representation of the solution space, the cost of all initial guesses recovered are only an approximate cost and not indicative of the total maneuver requirements for a given transfer.

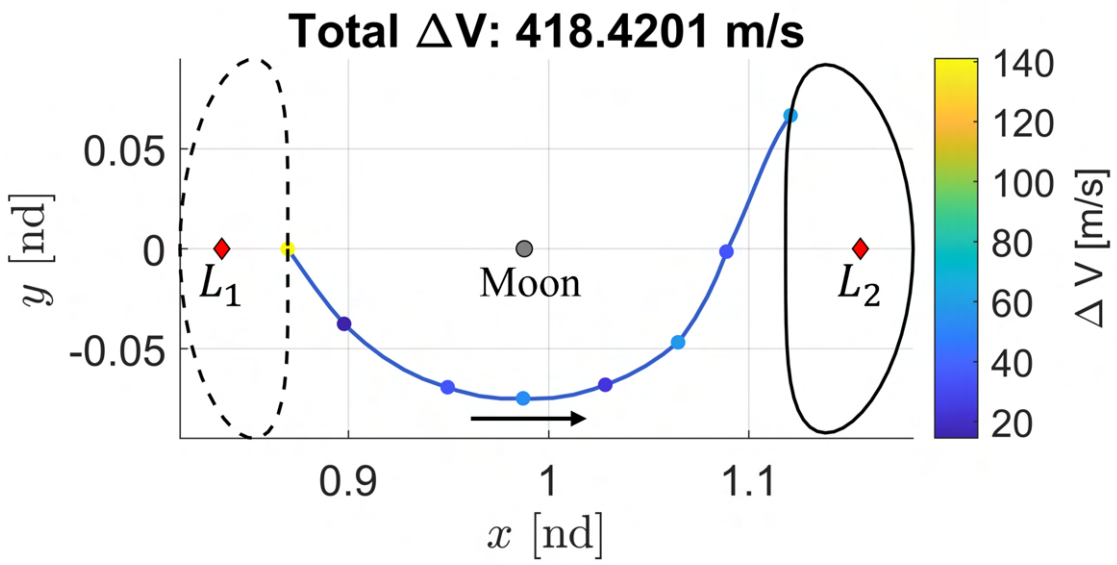


Figure 5.13: Optimal initial guess for a transfer from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system.

Commonly in trajectory design problems, a single initial guess may not produce a continuous trajectory that satisfies all the desired trajectory characteristics. A set of initial guesses that produce trajectories that are geometrically distinct or possess different trajectory characteristics, such as total maneuver magnitude, may be useful when exploring the design space. Therefore, once the optimal path is found, additional unique paths between the same start and goal node sets are identified by applying Yen's algorithm and a dissimilarity metric. Specifically, Dijkstra's search

algorithm is used in conjunction with a dissimilarity measure to search multiple subgraphs of the roadmap. However, since Yen's algorithm searches for the k shortest paths in a directed, weighted graph between a single pair of start and goal nodes, a small modification is made to search for all paths between the states associated with nodes along the initial and final orbits.

First, two additional imaginary nodes are added to the graph to represent a single start and goal location. The start node is a state with infinity for all 6-dimensional values and the goal node is a state with negative infinity for all 6-dimensional values. Then, the imaginary start node is connected using a zero weight edge to all the nodes associated with the states that lie on the initial periodic orbit. Last, all nodes associated with the states that lie on the final periodic orbit are connected using a zero weight edge to the imaginary goal node. This transforms the multi-start, multi-goal node search queries into a single start-goal node pair, required to ensure the implementation of Yen's algorithm is accurate. If the imaginary start and goal nodes were not included, Yen's algorithm does not guarantee to find the k shortest paths without potentially skipping over paths or finding paths that are not linearly increasing in cost.

Common unique paths are often identified using a similarity measure that is a ratio of the sum of edge weights of the shared edges between two paths and the distinct edges between two paths [46, 10]. While these similarity metrics often result in unique paths that have minimal overlap in the shared nodes and edges, they do not take into account any desired geometrical differences in the solution paths in the configuration space when analyzing initial guesses in the CR3BP. To recover initial guesses that result in geometrically distinct transfers, the distances between the solution paths in configuration space should be analyzed. Therefore, in this research, unique paths are identified with a dissimilarity measure that captures the distances between the position vectors of all nodes in two paths, P_i and P_j.

In order to capture the differences in position vectors of nodes located at different times along each initial guess, dynamic time warping (DTW) is used. Traditionally, DTW is used to align two time-dependent vectors, e.g. signals, such that the distance between each pair of elements within the vectors is minimized [51]. This is completed by stretching, or duplicating, elements of one vector

temporally to compare to another vector. Given two sets of K-dimensional data, $\boldsymbol{X}$ containing m samples and $\boldsymbol{Y}$ containing n samples, the euclidean distance between the m-th and n-th sample is computed as:

$$d_{mn}(\boldsymbol{X}, \boldsymbol{Y}) = \sqrt{\sum_{k=1}^{K}(x_{k,m} - y_{k,n})(x_{k,m} - y_{k,n})} \tag{5.8}$$

Initially, the values between every set of samples in both vectors is computed. Then, the vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ are warped onto a common set of instants such that the distance between the two vectors, d, computed as:

$$d = \sum_{m \in ix, n \in iy} d_{mn}(\boldsymbol{X}, \boldsymbol{Y}) \tag{5.9}$$

where ix and iy correspond to the warped vectors of $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively, is minimized [62]. The solution that minimizes the sum of distances between the samples in each vector is not allowed to skip or repeat samples, ensuring the warped vectors are compared at similar lengths. For example, in Figure 5.14, two paths are shown, a top path with only 4 data points and a bottom path with 6 data points. In order to compare these two paths, duplicate points along the top path must be used in order to determine the minimum distance between all points in the bottom path. By incorporating a time component to each path, all points along the bottom path are compared to their time counterparts along the top path, denoted by the black arrows, producing the minimum sum of all distances between the two paths.

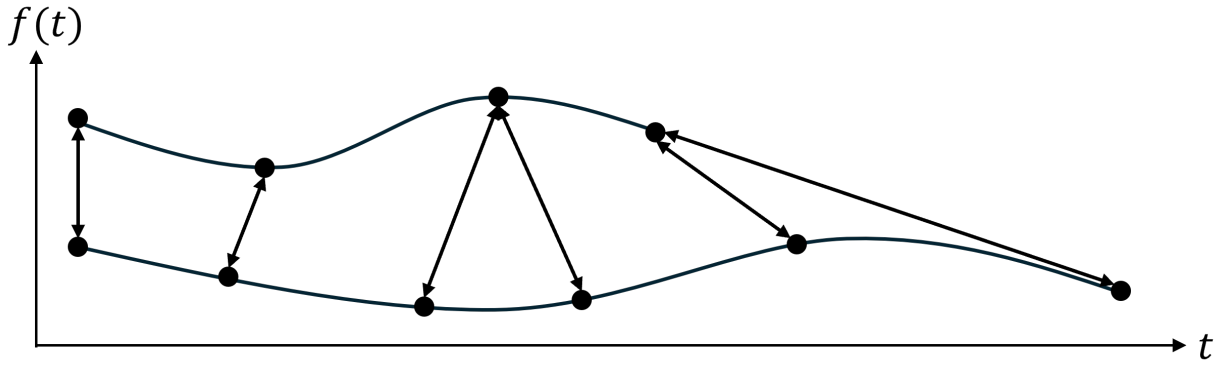


Figure 5.14: Comparing locations along two vectors using dynamic time warping to minimize the distances between data points.

To compute the dissimilarity between two paths, first, all unique position vectors associated with the nodes in each path are identified, R_i and R_j. Next, DTW stretches the two vectors such that the sum of the distances between corresponding points is minimized, resulting in a positive scalar distance $d(R_i, R_j)$. This distance value is used to determine how far apart two paths lie in configuration space for two solution paths found using Yen's algorithm. For a user-selected tolerance, δ, two paths are determined to be sufficiently different if $d(R_i, R_j) > \delta$. Multiple unique paths that are geometrically dissimilar may be identified by searching one completed graph repeatedly using this dissimilarity measure in conjunction with Yen's algorithm.

Initially, the value for the dissimilarity measure is set to $\delta = 0.5$ to distinguish unique paths. If this tolerance does not result in paths that are sufficiently dissimilar, δ can be increased after the desired k paths are identified using Yen's algorithm to produce a new subset of unique paths without needing to search the graph again for k paths. This is because the k paths list is not affecting by the dissimilarity tolerance, only the unique paths list is altered. Therefore, it is recommended to start with an initial value of δ that is smaller, and increase the value if necessary.

Using Yen's algorithm, additional but sub-optimal initial guesses are recovered. Within the list of k-shortest paths recovered and the selected value of δ, 423 unique solution paths are distinguished; producing a smaller set of initial guesses for a human trajectory designer to explore. Depicted in Figure 5.15 are 4 transfers of interest, including the optimal initial guess depicted in blue, where all nodes are again colored by their maneuver magnitudes. The 2^{nd} unique initial guess is denoted in orange; this path was originally $k = 60$ if the dissimilarity measure was not included. This initial guess has a total cost of 1.8760, corresponding to a total maneuver magnitude of 1011.0858 m/s. The 83^{rd} unique initial guess, $k = 2582$, is denoted in green. This initial guess has a total cost of 2.8461, corresponding to a total maneuver magnitude of 696.4495 m/s. Finally, the 186^{th} unique initial guess, $k = 5290$, is denoted in pink. This initial guess has a total cost of 3.0330, corresponding to a total maneuver magnitude of 974.5459 m/s.

While the 2^{nd} unique initial guess has the highest total maneuver magnitude associated with it, it is still found within the graph as the 60^{th} best path. This is due to the scaling of the maneuver

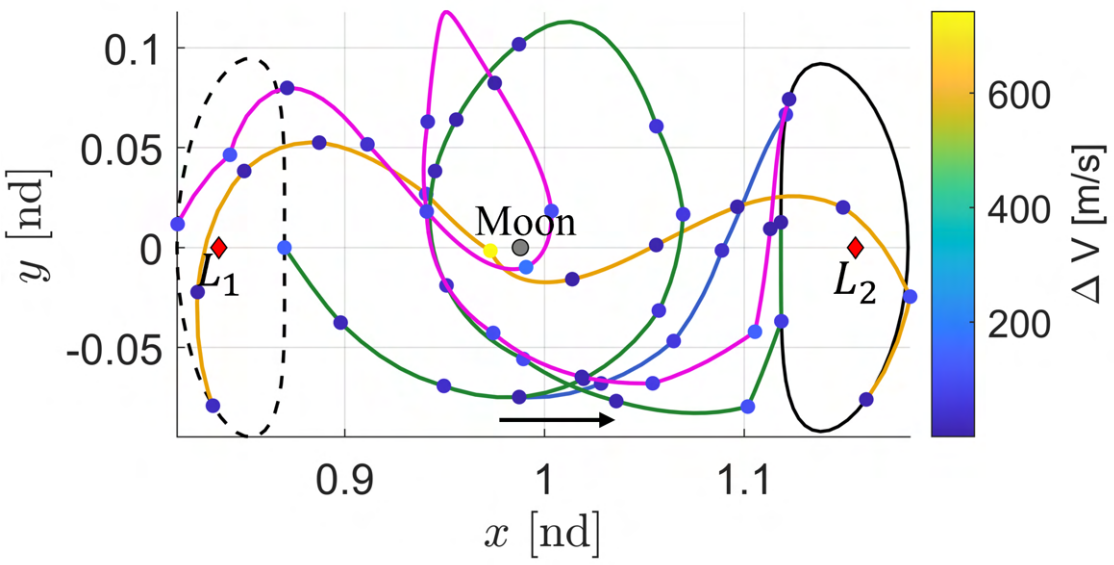


Figure 5.15: Four unique initial guess for transfers from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system.

magnitude by the minimum speed of the nodes within the edges selected as the search heuristic. If only maneuver magnitude was selected as the search heuristic, this orange solution path would be recovered much later within the list of sub-optimal paths, and subsequently the list of unique paths, due to its large cost. However, when the edge weights are scaled the largest maneuver along this path, denoted by the yellow circle along the orange path in Figure 5.15, reduces from 548.0709 m/s to an edge weight of 0.6089. The overall edge weights along this orange solution path are scaled from maneuver magnitudes of $[0.8676, 548.07079]$ m/s to $[0.0464, 0.6089]$, placing the larger maneuvers that occur at position vectors with higher speeds on a similar scale to other maneuvers along the transfer. This edge weight selection results in this solution path being recovered much earlier within the search process.

The initial guesses depicted in Figure 5.15 were selected to further analyze because they are geometrically similar to known trajectories that exist at this energy level of $C_J = 3.15$. Although each initial guess is a discrete representation, and therefore contains a maneuver at each location, the smoothness of each initial guess resembles potential natural motion throughout the solution space. This reveals that the discrete graph representation is sufficiently accurate for this trajectory

design scenario. Additionally, the various geometries selected here were selected because two initial guesses contain no revolutions around the Moon while the last two higher cost initial guesses contain one revolution around the Moon.

Exploring trajectories of various geometries is often an important aspect of trajectory design and this scenario demonstrates the success of using graph search algorithms to automatically recover unique initial guesses. Commonly with Dijkstra's algorithm, if the search heuristic is not selected appropriately for the environment, the resulting solution paths may not resemble feasible motion. In order for Dijkstra's algorithm to minimize the total cost of a solution path, it inherently also minimizes the number of total nodes and edges in each solution path. This could mean that unless the edges accurately represent the solution space, along with a carefully selected search heuristic, the search queries may produce a path that is a straight line from the initial nodes to the goal nodes, i.e. a path that minimizes the number of edges. However, this path may not be desirable or feasible in nonlinear, chaotic systems. For the two initial guesses with revolutions around the Moon, in the k shortest paths list, the green initial guess was recovered as $k = 2582$ while the pink initial guess was recovered as $k = 5290$. Even though these solution paths contain over twice the number of nodes and edges compared to the initial guesses with zero revolutions, each path was recovered relatively quickly within the k shortest path list. This demonstrates that the graph is constructed to be sufficiently accurate to resemble natural motion throughout the solution space by recovering multiple geometrically distinct initial guesses that have relatively low total costs for a discrete representation.

5.4.3 Recover Geometrically Distinct Transfers

Selected initial guesses are input into a collocation corrections algorithm with explicit propagation mesh refinement process and optimized within *fmincon* for the squares of their maneuver magnitudes to recover a set of geometrically distinct transfers with impulsive maneuvers between the selected periodic orbits. The state of each node and the integration time along the subsequent edge in the initial guess form the segments used in the initial mesh, so the total time of flight may

vary during corrections. If impulsive maneuvers are applied along the transfer between segments, only position constraints are included. The number of allowed maneuvers must be user-selected prior to corrections, although higher maneuver magnitudes along the initial guesses may be used to indicate locations where a maneuver may be beneficial in recovering a nearby transfer.

Using the optimal initial guess depicted in Figure 5.13, the solution path is corrected and optimized using 2 maneuvers placed at the beginning of the transfer and the end of the transfer. These maneuvers are placed such that the transfer can depart and arrive anywhere to the initial and final orbits. The recovered, continuous transfer possesses a final total maneuver magnitude of 1.4517 m/s and is depicted by the blue curves in Figure 5.16, while the initial guess for this transfer is depicted in light grey. This continuous trajectory lies very close to a natural heteroclinic connection for this energy level between these two orbits, depicted by the dashed black curves, demonstrating the applicability of the trajectory design process using sampling-based kinodynamic planning to recover feasible trajectories without relying on traditional trajectory design methods and a human-in-the-loop.

Using the second unique initial guess depicted in Figure 5.15, the initial guess is corrected and optimized again using 2 maneuvers placed at the beginning of the transfer and the end of the transfer, with 1 additional maneuver placed at the highest maneuver location closest to the Moon. The recovered, continuous transfer possesses a final total maneuver magnitude of 1.2871 m/s and is depicted by the blue curves in Figure 5.17, while the initial guess for this transfer is depicted in light grey. While the optimized solution varies noticeably from the geometry of the initial guess, this continuous trajectory lies close enough to another natural heteroclinic connection for this energy level between these two orbits, depicted by the dashed black curves, that with only one maneuver allowed along the transfer, a transfer that resembles that heteroclinic connection is recovered as the optimized solution. If the geometry of the initial guess were more desirable or prioritized over total maneuver costs, e.g. for a mission that desires a closer transfer to the Moon for scientific studies, additional maneuvers may be allowed along the transfer.

Similarly to the previous two solutions, the third unique initial guess selected is corrected

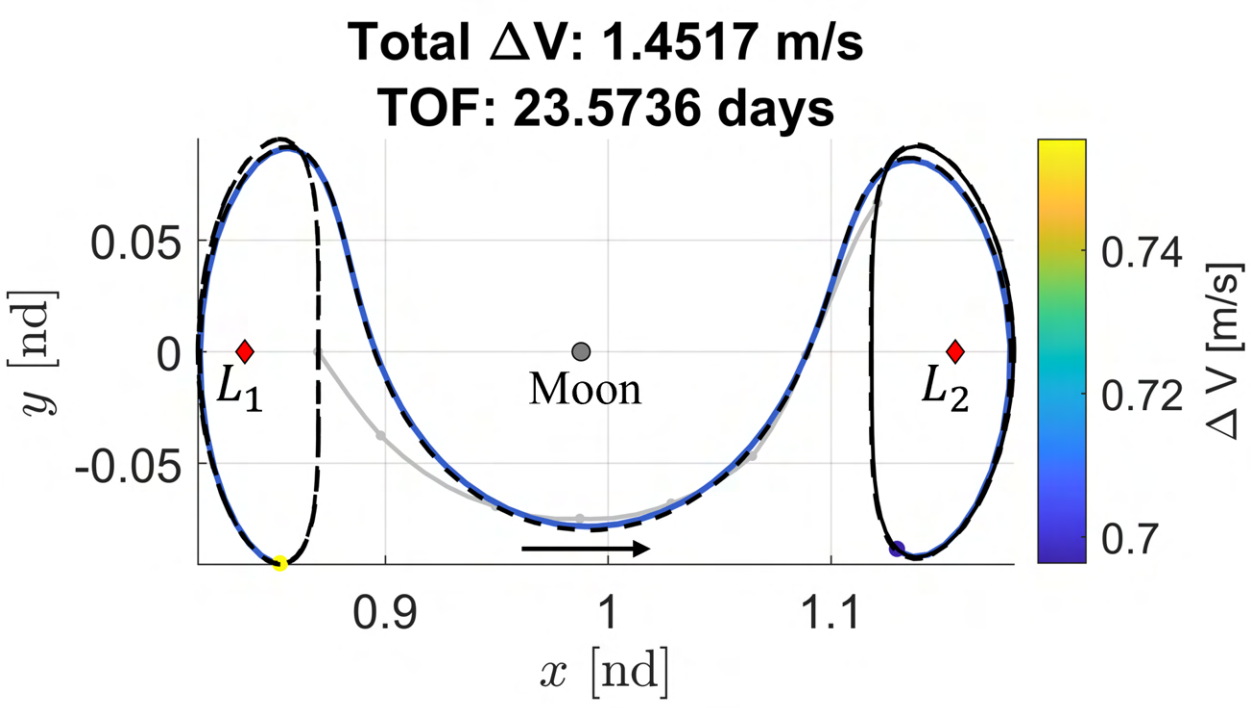


Figure 5.16: Continuous transfer from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the initial guess displayed in Figure 5.13.

and optimized using 2 maneuvers placed at the beginning and end of the transfer to depart and arrive to the Lyapunov orbits. The recovered, continuous transfer possesses a final total maneuver magnitude of 1.1396 m/s and is depicted by the blue curves in Figure 5.18, while the initial guess for this transfer is depicted in light grey. Additionally, this optimized continuous transfer resembles a natural 1 revolution heteroclinic connection for this energy level between these two orbits, depicted by the dashed black curves. While the final transfer varies sufficiently from the initial guess, the initial guess lies close enough to the natural heteroclinic connection that with minimal maneuvers allowed is what the initial guess is optimized to.

The last unique initial guess selected for a transfer, depicted in pink in Figure 5.15, is corrected using 2 maneuvers to depart and arrive to the Lyapunov orbits but also 9 maneuvers along the initial guess in order to retain geometry. The corrected solution recovered a continuous transfer with a cost of 341.9502 m/s and lies very closely to the initial guess. While this solution does not resemble any known natural transfers between these Lyapunov orbits at $C_J = 3.15$, the corrected

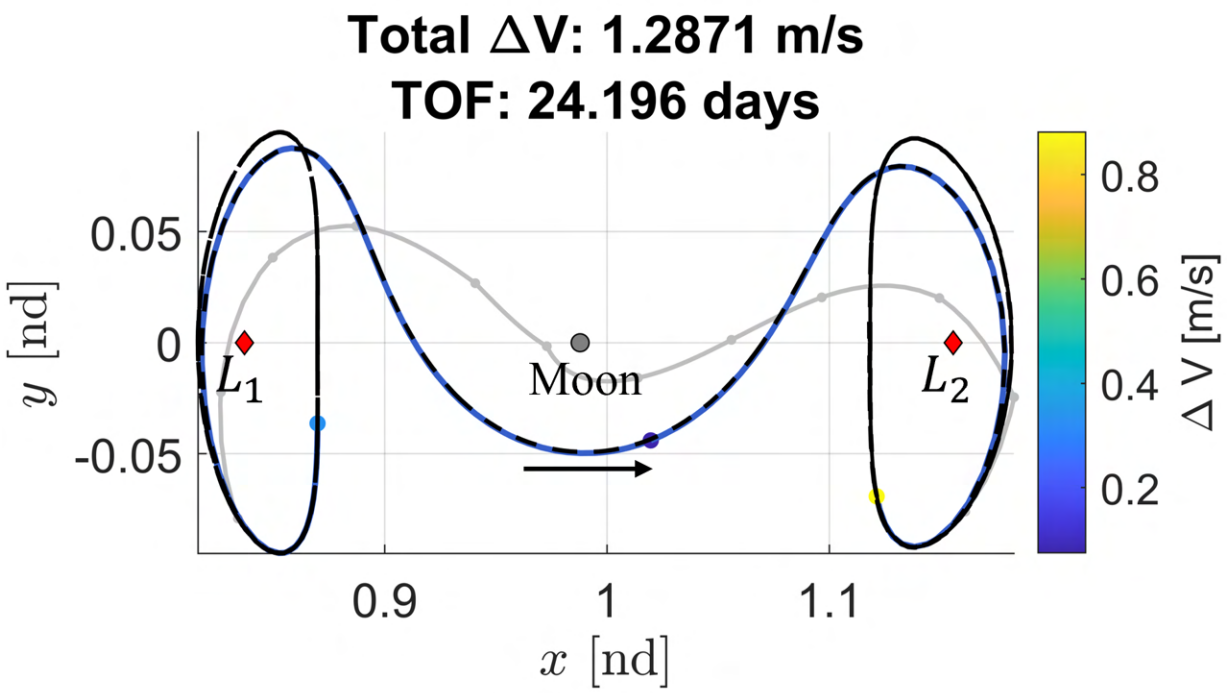


Figure 5.17: A second geometrically distinct transfer from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the orange initial guess displayed in Figure 5.15.

transfer is still a solution with a closer approach to the Moon, which may be desired for certain missions.

The trajectory design process demonstrated in this chapter for constructing geometrically distinct transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit is supported by the confirmation of recovering transfers that resemble various natural heteroclinic connections, as well as low-cost impulsive maneuver enabled trajectories. Additional unique initial guesses, potentially with multiple revolutions around the Moon, may be recovered by searching the roadmap for a larger number of k while using Yen's algorithm. Due to the discrete construction of the roadmap and maneuvers located at every node along an initial guess, longer time of flight initial guesses for transfers will inherently posses a larger total cost since more nodes and edges will be contained within the initial guess. This quality causes other potentially longer time of flight or multiple revolution unique initial guesses to require a larger computational time within Yen's algorithm and might be recovered for a larger value of k.

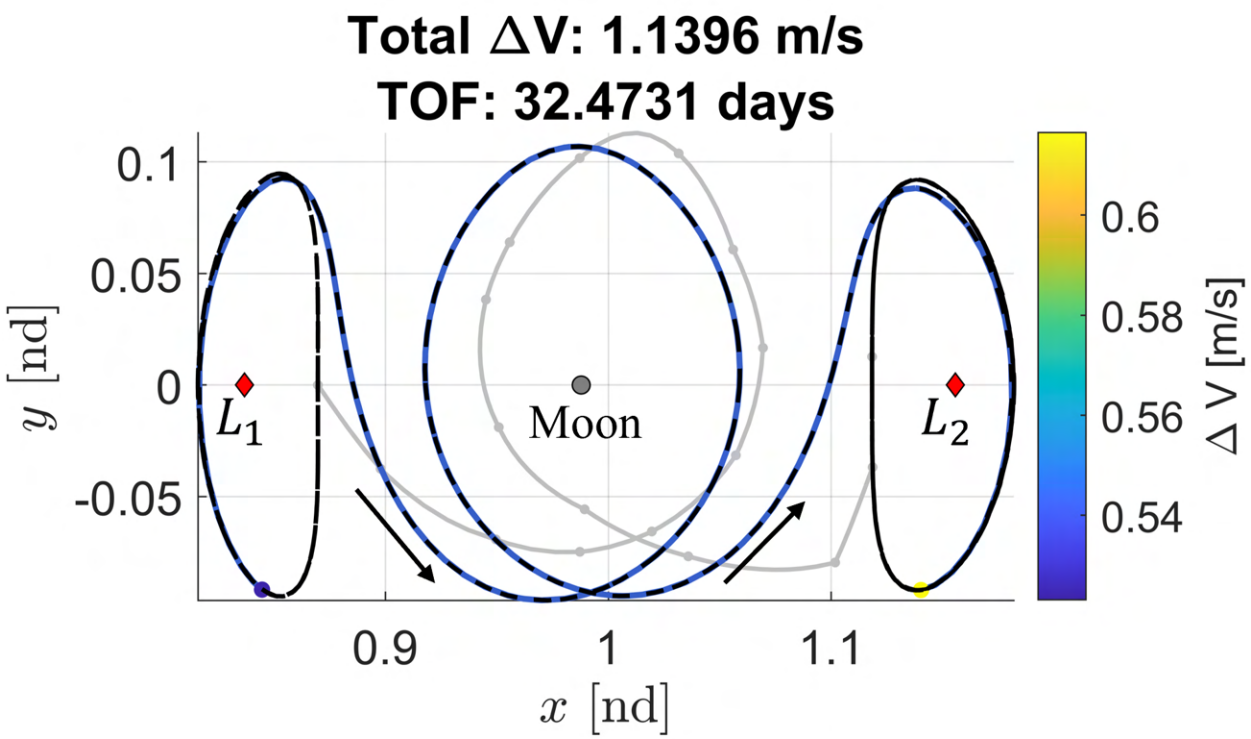


Figure 5.18: A third geometrically distinct transfer from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the green initial guess displayed in Figure 5.15.

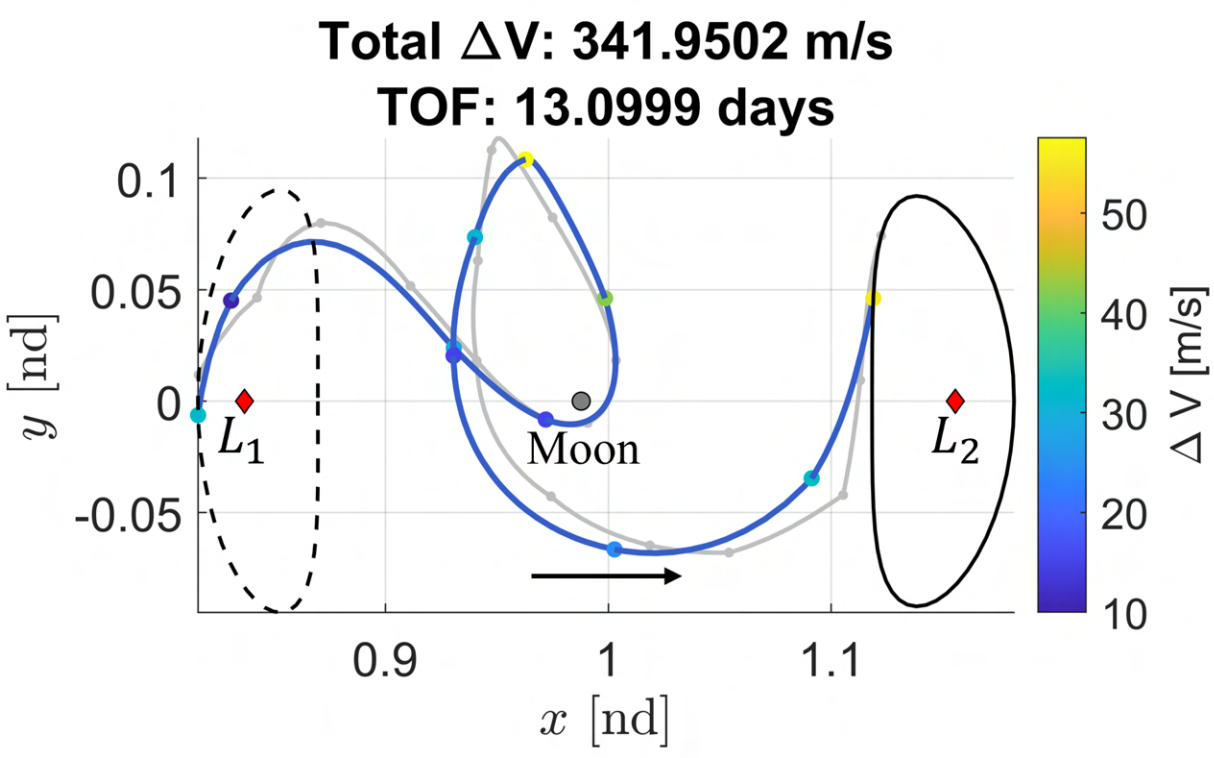


Figure 5.19: A fourth geometrically distinct transfer from an L_1 to L_2 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the green initial guess displayed in Figure 5.15.

Chapter 6

Application: Trajectory Exploration Near the Moon in the Earth-Moon CR3BP

To further demonstrate the applicability of the sampling-based kinodynamic planning method presented in Chapter 5, three additional trajectory design scenarios are explored. Each scenario analyzed in this chapter is explored within the Earth-Moon CR3BP, specifically exploring transfers between periodic orbits near the Moon. First, to continue the exploration of the transfer design scenario analyzed in Chapter 5 and to demonstrate that a single roadmap may be used for multiple search query parameters, transfers are constructing from an L_2 Lyapunov orbit to an L_1 Lyapunov orbit. Next, to examine the solution space across a range of energy levels, a planar roadmap is constructed for $C_J = [3.12 - 3.16]$ near the Moon. Multiple search queries for a variety of unique initial guesses for transfers between L_1 and L_2 Lyapunov orbits with distinct energy levels are generated to produce sets of geometrically distinct transfers for each search query. Last, a spatial roadmap is constructed at $C_J = 3.15$ to explore transfers between an L_1 and L_2 northern halo orbit. Each transfer design scenario expands on the roadmap capabilities presented earlier by constructing a roadmap across a range of energy levels and a spatial roadmap at a single energy level. The scenarios presented in this chapter demonstrate the feasibility of using sampling-based kinodynamic planning for a variety of trajectory design scenarios near the Moon.

6.1 Planar Transfers Between Lyapunov Orbits at $C_J = 3.15$

A key benefit of probabilistic roadmap generation is the capability of recovering multiple solution paths for a variety of boundary conditions and/or search parameters from the same roadmap by changing the search parameters. Since the construction of the roadmap is not dependent on the search query parameters, additional search queries can be explored. To demonstrate this benefit, in addition to the transfers constructed from the L_1 Lyapunov orbit to the L_2 Lyapunov orbit presented in Chapter 5, additional unique initial guesses are recovered for transfers from the L_2 Lyapunov orbit to the L_1 Lyapunov orbit using the completed roadmap in Figure 5.12.

6.1.1 Transfers from an L_2 to an L_1 Lyapunov Orbit at $C_J = 3.15$

To construct transfers from the L_2 Lyapunov orbit to the L_1 Lyapunov orbit at $C_J = 3.15$, the discretized states along the L_2 orbit must be connected to their nearest neighbors forward in time and the discretized states along the L_1 orbit must be connected to their nearest neighbors backward in time. This ensures all initial guesses recovered depart and arrive to the correct boundaries. If any additional states can be considered for a start or goal node, they must be added to the graph during this phase. Figure 6.1 shows the generalized roadmap along with 50 states along an L_2 Lyapunov orbit, depicted by the dashed black curve, and 50 states along an L_1 Lyapunov orbit, depicted by the solid black curve, connected the the roadmap. No additional trajectory segments from known hyperbolic invariant manifolds were added to this graph. This resulted in 1438 new nodes added to the graph and 29578 new edges.

Using the updated roadmap, unique initial guesses for transfers are recovered using Dijkstra's algorithm and Yen's algorithm. Again, a dissimilarity value of $\delta = 0.5$ is selected to distinguish sufficiently different initial guesses resulting from the list of k best paths. Three initial guesses are identified as potential candidates for transfers from the L_2 Lyapunov orbit to the L_1 Lyapunov orbit. These initial guesses are depicted in Figure 6.2 and the maneuvers at each node are colored by their respective maneuver magnitude. The first unique initial guess, $k = 9$, depicted by the blue curves,

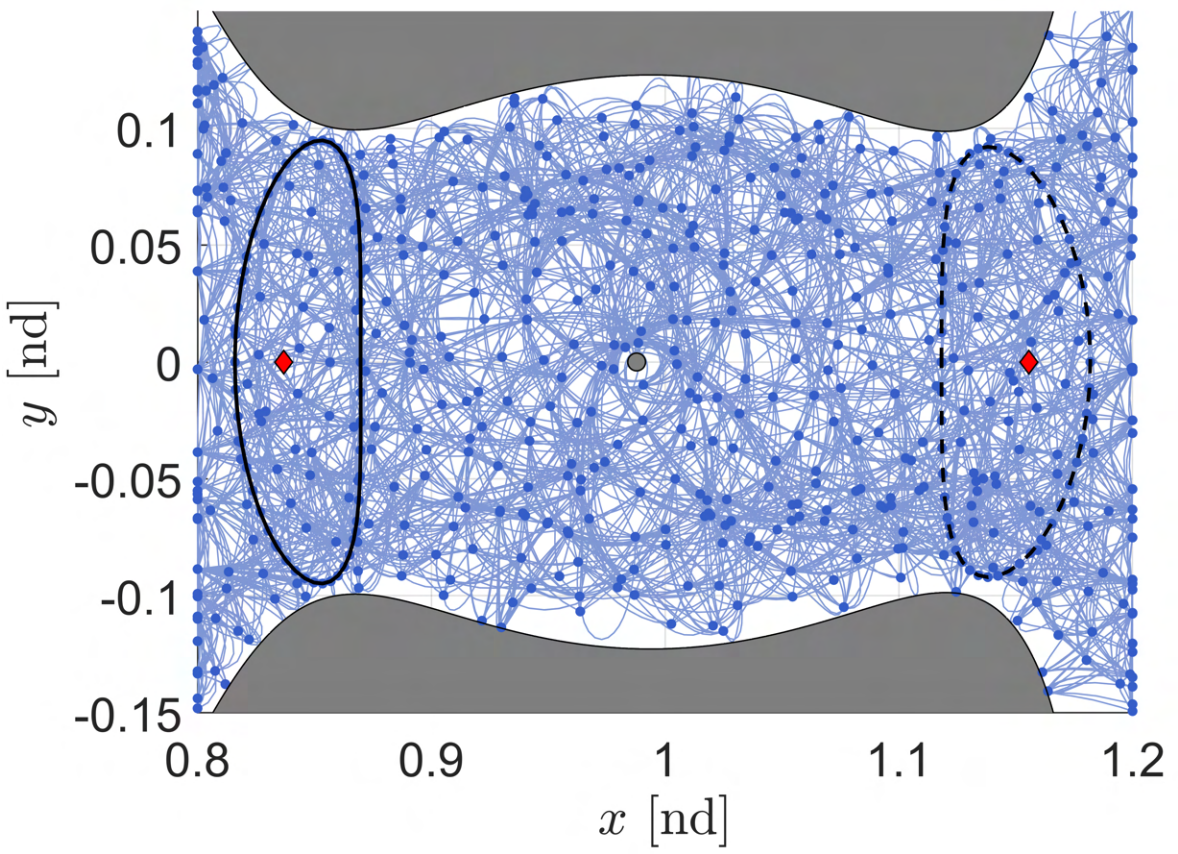


Figure 6.1: A completed roadmap generated to construct transfers from an L_2 Lyapunov orbit to an L_1 Lyapunov orbit at $C_J = 3.15$.

possesses a total path cost of 3.8575 and a total maneuver magnitude of 434.9531 m/s. The second unique initial guess, $k = 37$, depicted by the orange curves, possesses a total path cost of 4.5041 and a total maneuver magnitude of 1513.2954 m/s. The third unique initial guess, $k = 50$, depicted by the green curves, possesses a total path cost of 4.5914 and a total maneuver magnitude of 1292.6159 m/s. This third unique initial guess containing 1-revolution around the Moon demonstrates that while each initial guess is a coarse representation of a transfer, increasing the number of edges in the path does not automatically lead to an increase in total ΔV, e.g. the total maneuver cost of the orange initial guess is larger than the total maneuver cost of the green initial guess. This indicates the graph sufficiently captures the natural dynamics within the solution space and initial

guesses with multiple revolutions are possible to recover with graph search algorithms, provided a sufficient computational time to explore the desired number of solution paths. Additionally, while identifying sufficiently dissimilar initial guesses may be challenging for a trajectory design problem, Yen's algorithm along with the dissimilarity measure is able to automatically distinguish between unique initial guesses. Each of these solution paths are used as the initial guesses for the collocation corrections method to recover continuous, maneuver-enabled trajectories.

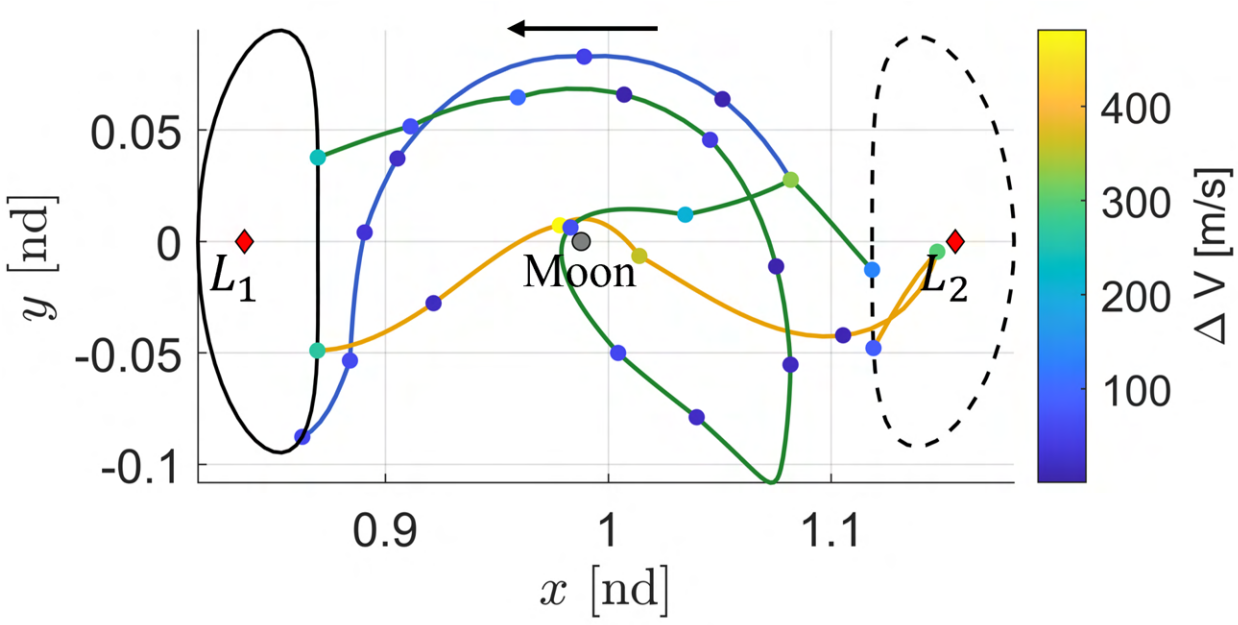


Figure 6.2: Three unique initial guess for transfers from an L_2 to L_1 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system.

The first selected initial guess from Figure 6.2 is corrected and optimized with respect to total maneuver magnitude with 2 maneuvers, one placed at the beginning of the transfer to depart the initial L_2 orbit and one placed at the end of the transfer to arrive to the L_1 orbit. Since the initial and final orbits possess the same Jacobi constant, the total change in energy along the transfer should be relatively small, therefore only the departure and arrival maneuvers are allowed. The final optimized transfer, depicted by the blue curves in Figure 6.3, has a total maneuver magnitude cost of 1.4607 m/s and a time of flight of 23.5221 days. This continuous transfer geometrically resembles a natural heteroclinic connection, depicted by the black dashed curves displayed in Figure 6.3.

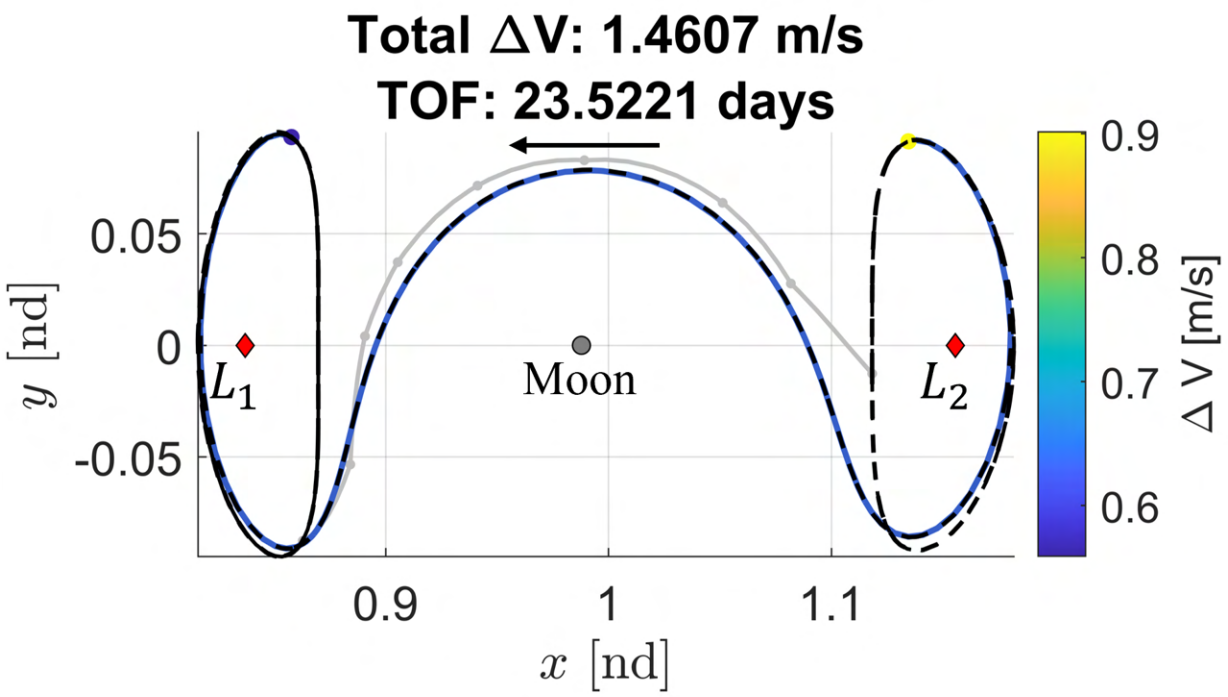


Figure 6.3: A geometrically distinct transfer from an L_2 to L_1 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the blue initial guess displayed in Figure 6.2.

The second selected initial guess from Figure 6.2 is corrected and optimized with respect to total maneuver magnitude with 4 maneuvers: one placed at the beginning of the transfer to depart the initial L_2 orbit, one placed at the end of the transfer to arrive to the L_1 orbit, and 2 maneuvers along the transfer at locations of the initial guess that correspond to higher edge weights. The additional maneuvers are placed along this transfer due to the sensitivity of corrections because the nodes with the largest edge weights along the initial guess are located near the Moon. However, each of these maneuvers along the transfer resulted in an optimized maneuver magnitude of less than 0.06 m/s for both locations. Each of these maneuver locations may not be required during optimization since this initial guess also lies very close to a second natural heteroclinic connection with this geometry. The final optimized transfer possesses a total maneuver magnitude cost of 1.5175 m/s, depicted by the blue curves, and the second natural heteroclinic connection, depicted by the black dashed curves, is displayed in Figure 6.4. The total time of flight for this optimized transfer is 23.6311 days. While the optimized transfer varies noticeably from the initial guess, with only 2

maneuvers allowed along the transfer, the initial guess is optimized to a transfer that resembles a heteroclinic connection due to the optimization objective function minimizing the sum of squared maneuvers. Additionally, the first two edges in the initial guess are considerably smoothed in the continuous solution. These two edges and high maneuver requirements may potentially indicate either a poor connection of the states along the initial orbit at the departure transfer location to the nodes in the roadmap or additional edges constructed to departure the states along the initial orbit. Furthermore, a different unique initial guess that shares a large portion of the nodes and edges with this initial guess may exist within the graph with a higher total cost, but may also remove the sharp changes in velocity at the first two edges of the transfer. This initial guess recovered was selected because it is the first initial guess, i.e. lowest total cost, of the graph with this geometry.

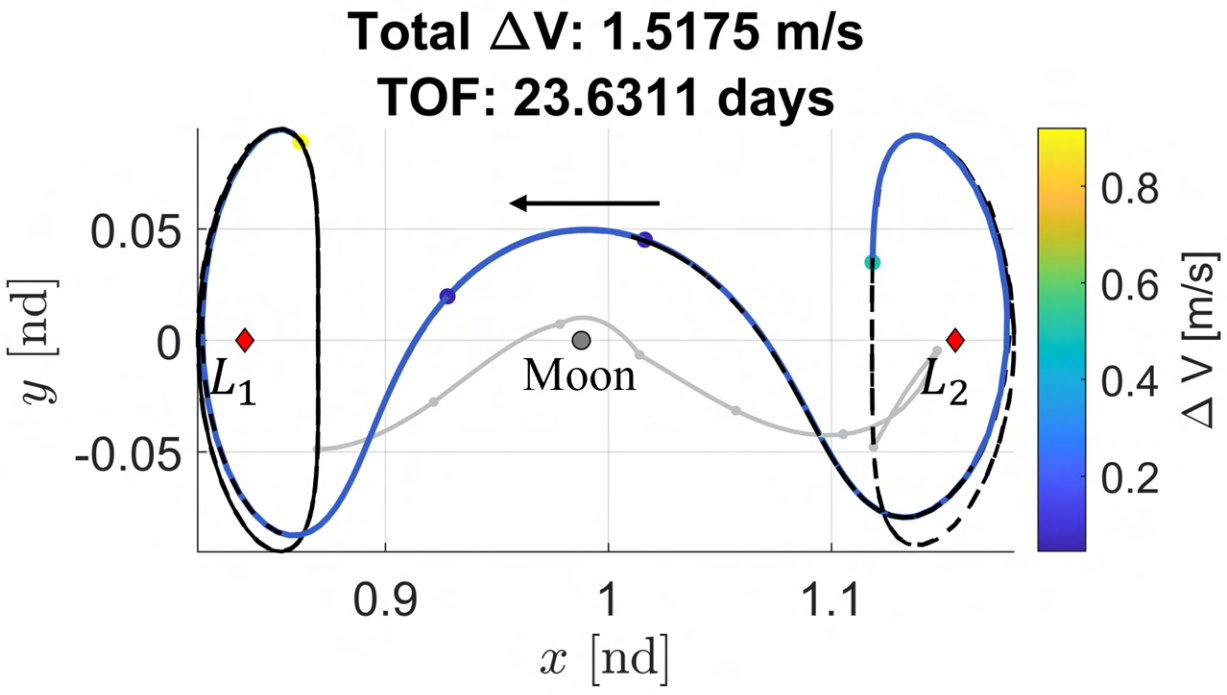


Figure 6.4: A second geometrically distinct transfer from an L_2 to L_1 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the orange initial guess displayed in Figure 6.2.

The third selected initial guess from Figure 6.2 is corrected with 7 maneuvers, one placed at the beginning of the transfer to depart the initial L_2 orbit, one placed at the end of the transfer to arrive to the L_1 orbit, and 5 maneuvers along the transfer at locations of the initial guess

that correspond to locations that possess higher edge weights. The additional maneuvers placed along this transfer were selected in order to preserve the geometry of the initial guess. The final optimized transfer has a total maneuver magnitude cost of 190.64 m/s, depicted by the blue curves in Figure 6.5 and has a total time of flight of 23.9204 days. The final transfer varies from the initial guess most notably at the departure and arrival segments. This is due to the coarse representation of the initial guess and again potentially may indicate a poor connection from the states along the initial and final orbit. While this is the lowest cost initial guess found within the roadmap with this geometry, a different initial guess that has a slightly higher cost but without the sharp departure and arrival segments may exist within the roadmap.

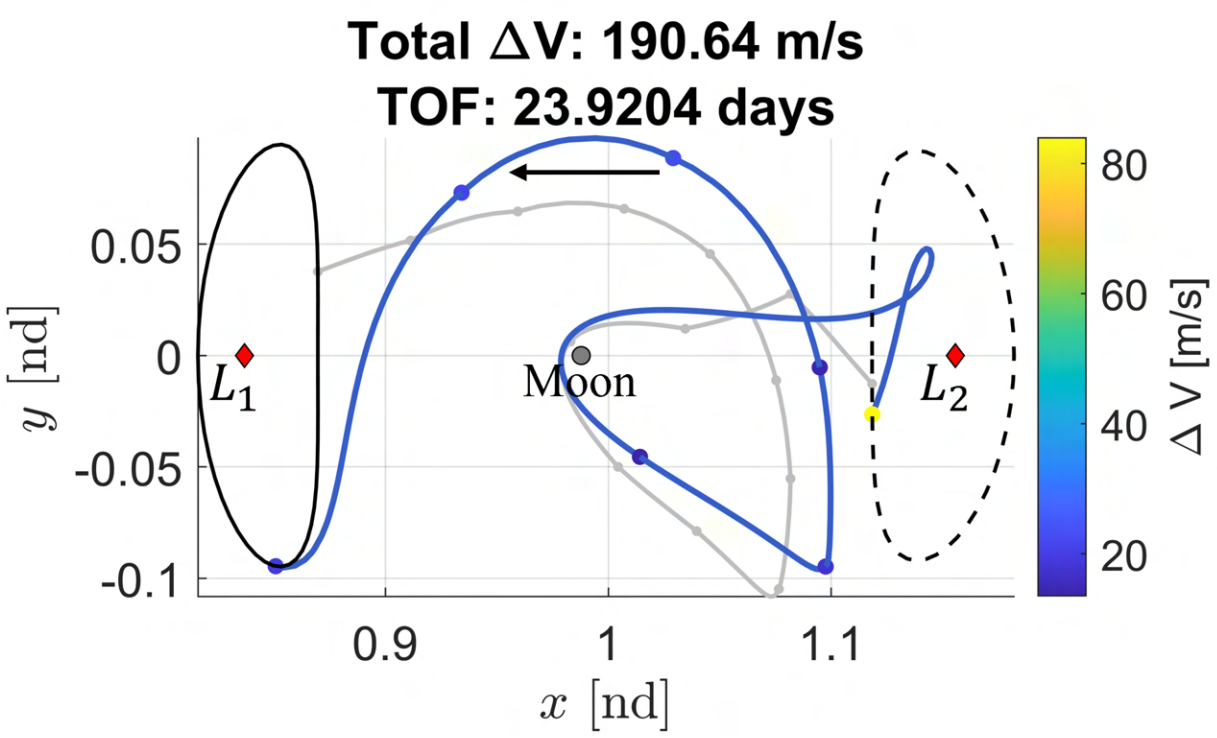


Figure 6.5: A third geometrically distinct transfer from an L_2 to L_1 Lyapunov orbit at $C_J = 3.15$ in the Earth-Moon CR3BP system, computed using the green initial guess displayed in Figure 6.2.

By modifying the initial and start nodes for the roadmap, additional unique transfers are recovered within the same solution space, enabling an automatic exploration of more possible transfers between the L_1 and L_2 Lyapunov orbits. If other locations within the available bounds

of the roadmap were desired as the start or goal nodes, or any potential locations within the environment where a spacecraft must pass through such as a close approach to the Moon, the graph searches can be quickly repeated with the new search parameters. Generating new solution paths by modifying the boundary conditions and regenerating solution paths may also be utilized for any potential re-planning scenarios required if the spacecraft's location deviates slightly; a new search may be used to rapidly generate an initial guess from the new location to the desired goal nodes.

6.2 Planar Transfers Between Lyapunov Orbits at $C_J = [3.12 - 3.16]$

6.2.1 Roadmap Construction for $C_J = [3.12 - 3.16]$

Often in mission design, various periodic orbits at different energy levels may be used and spacecraft may need to perform impulsive maneuvers along the trajectory, altering the spacecraft's energy level. Therefore, it is important to demonstrate the trajectory design process for transfers between periodic orbits at distinct energy levels by constructing a roadmap that spans a range of Jacobi constants. To demonstrate this, planar Lyapunov transfers between orbits at distinct energy levels are generated by first constructing a roadmap across the desired range of $C_J = [3.12 - 3.16]$. Using the sampling-based kinodynamic planning approach to trajectory design presented in the previous chapter, a graph is constructed using the parameters listed in Table 6.1.

Table 6.1: Graph construction parameters selected to construct a planar roadmap across energy levels $C_J = [3.12 - 3.16]$.

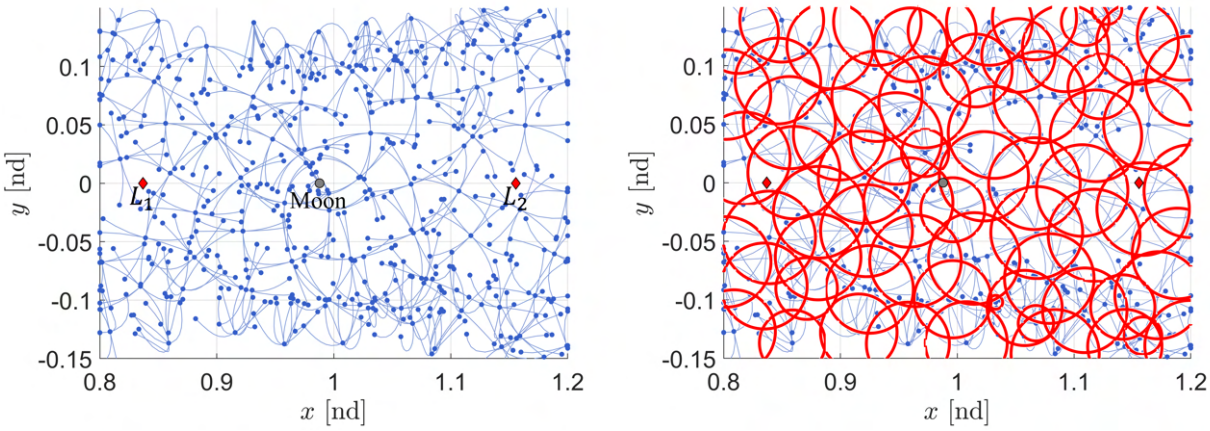
Parameter	Value
ℓ	0.04
n_c	3
τ_{DT}	10^{-8}
n	2

The selected value of ℓ and n_c were chosen for this scenario because a larger number of smaller neighborhoods produced the desired distribution of nodes and natural edges. Additionally, these

values produced a neighborhood distribution that contained minimal redundant natural edges or locally dense regions of nodes across the solution space. The number of desired velocity vectors computed for each set of 1-neighbors was selected as the minimum number of neighbors recommended to span the planar, 2-dimensional velocity space, e.g. since the desired velocity space has two dimensions to explore, the selected value of n should be 2 at a minimum. Since each neighborhood is designed to explore not only the local solution space through a range of velocity directions, but also at a range of energy levels, this could increase the dimension of the neighborhood sampling by one. A selection of $n = 3$ would also be recommended for any roadmap that spans multiple energy levels if computational time or memory storage requirements were not a consideration or limitation. However, a coarser representation of the solution space is desired for this demonstrative example, leading to the selection of $n = 2$. This results in every set of 1-neighbors containing 4 unique velocity directions for a given position vector.

In order to span the desired ranges of energy levels and satisfy the criteria for 3 neighborhoods overlapping for the solution space to be covered, 69 neighborhoods are constructed. The distribution of nodes, depicted by blue circles, and natural edges, depicted by blue curves, within each neighborhood are shown in Figure 6.6a, while the coverage of each neighborhood is depicted by the red circles in Figure 6.6b.

Using the selected roadmap construction parameters, the graph is an even distribution of neighborhoods without containing a large amount of nodes sampled within the same region or too many redundant natural edges. There is a larger number of nodes constructed from the neighborhoods near the top and bottom of the solution space, near where the zero velocity curves would lie. This is due to the neighborhood construction method. When a valid position vector sampled for the center of a neighborhood lies close to the boundaries of feasible motion for a specific energy level, it is more likely that a larger number of the natural edges in that neighborhood flow in similar directions in the solution space. This happens because when one centroid with a velocity vector with a positive y component is propagated forward, it produces a trajectory arc that extends away from the centroid in the positive y direction. However, when this same state is propagated back-



(a) Nodes (blue circles) and natural edges (blue curves) constructed for all neighborhoods.

(b) The covered region for each neighborhood depicted by a red circle.

Figure 6.6: The neighborhoods for a roadmap constructed for $C_J = [3.12 - 3.16]$ using graph parameters $\ell = 0.04$ and $n_c = 3$.

ward to construct an additional edge the trajectory arc may initially start off propagating in the negative y direction, but once it reaches the boundary of the zero velocity curve, it will curve back towards the centroid and flow in the positive y direction. This results in both edges that look very similar in position space and also contain final end states along each trajectory arc that lie very close together. These nodes will later also gain more edges during the global edge construction step, producing potentially redundant edges near the zero velocity curves in this region of the solution space.

When constructing the neighborhoods for this roadmap, one difference relative to the scenario presented in the previous chapter is the sampling of energy levels for each node. When constructing the set of 1-neighbors for each position vector within a neighborhood, a Jacobi constant within the bounds in the graph is randomly sampled to produce the speed of each node. This random sampling results in the natural edges within a neighborhood to be constructed at various energy levels. The distribution of energy levels for all the natural edges for all neighborhoods in the graph is shown in Figure 6.7. The result of this additional random sampling for the energy levels within the graph still produces bundles of natural edges that flow through each neighborhood, but in this graph each

set of bundles of edges are at a single energy level.

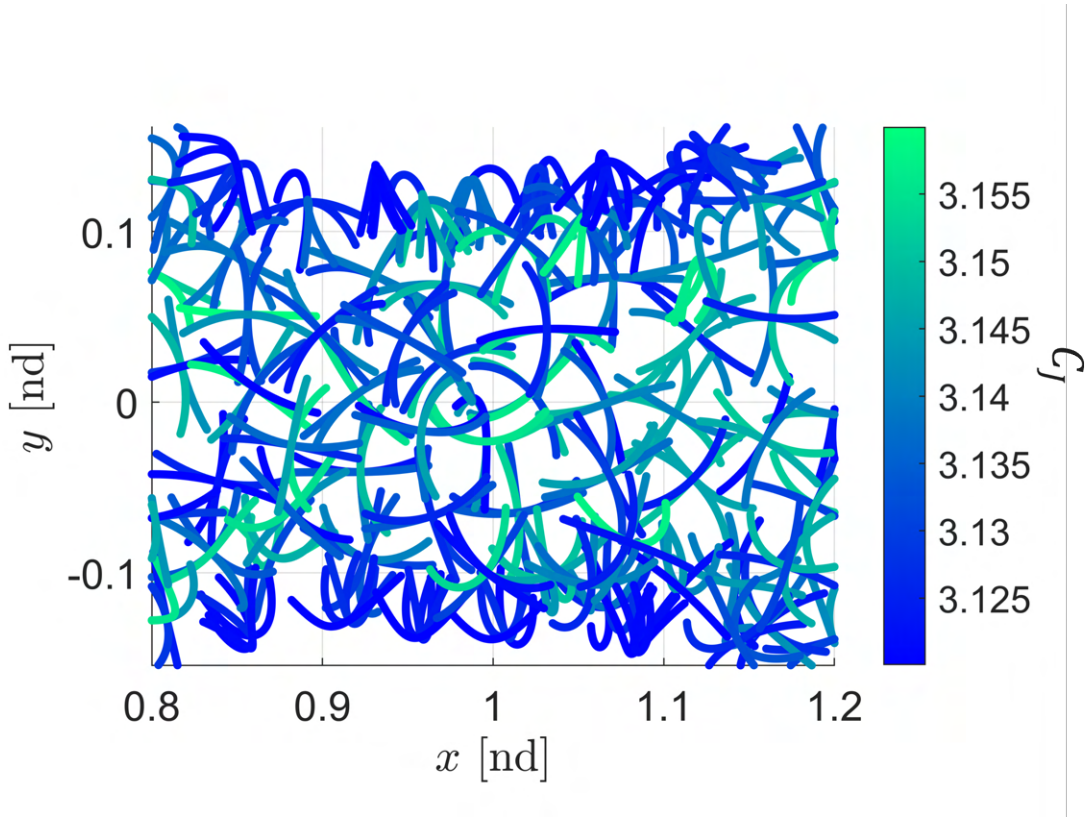


Figure 6.7: The natural edges of all neighborhoods shown in Figure 6.6 colored by Jacobi constant.

Each neighborhood within the graph should span a range of energy levels. However, for the neighborhoods constructed near the boundaries of feasible motion, these neighborhoods span a smaller or single energy level. This occurs because the zero velocity curves computed at each energy level have different regions of feasible motion. Therefore, when a valid position vector is sampled near the boundaries, it may only be a valid position vector at a smaller range of Jacobi constants. This situation can be seen in Figure 6.7, where the neighborhoods that are constructed near the top and bottom of the solution space are only constructed for energy levels that are closer to $C_J = 3.12$, the lower bound of Jacobi constant for the graph.

Since the graph is constructed across a range of energy levels, increasing the dimension of the sampling by one, a finer distribution of nodes was desired relative to the single energy level scenario demonstrated in the previous chapter. This is achieved by selecting a stricter tolerance of τ_{DT}. For

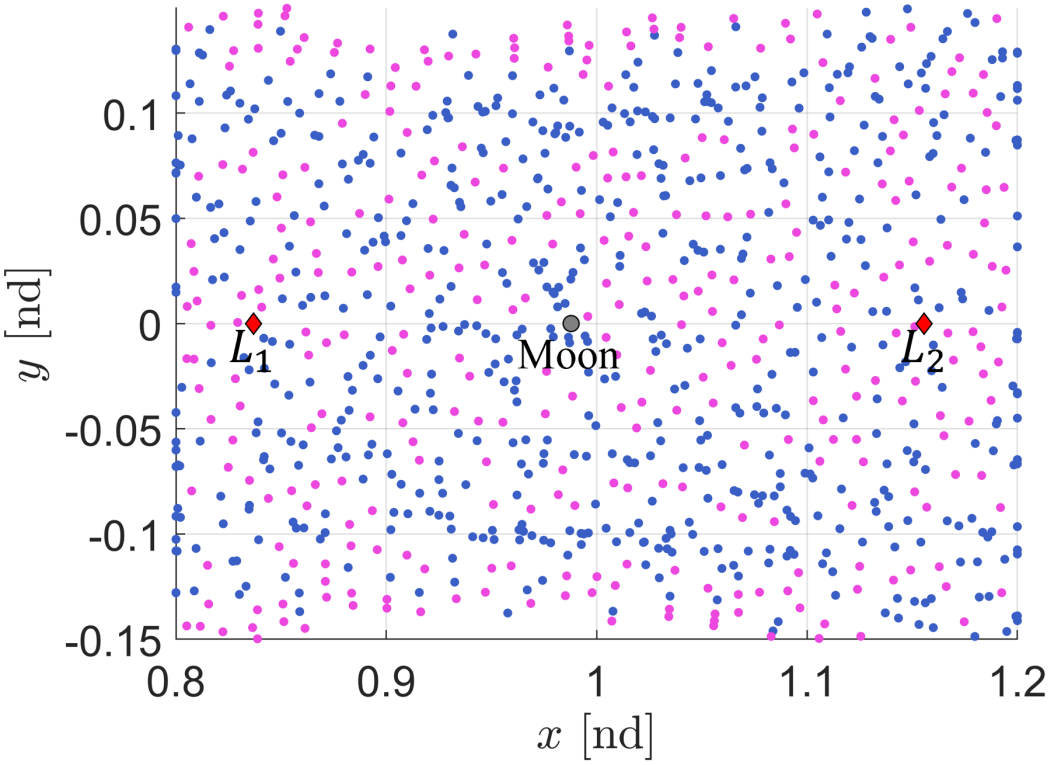


Figure 6.8: The distribution of nodes constructed from neighborhoods (blue) and Delaunay triangulations (pink).

this roadmap, 350 additional position vectors were identified using a Delaunay triangulation until the stopping tolerance was met. This resulted in 1400 additional nodes constructed for the graph, since every identified position vector gains its own set of 1-neighbors. The final distribution of nodes in the roadmap is displayed in Figure 6.8, where the nodes constructed from neighborhoods are depicted by blue circles and the nodes constructed from the Delaunay triangulation are depicted by the pink circles.

Finally, each edge gains a neighbor forward and backward in time until the graph is strongly connected. This results in 8246 time-varying edges to be constructed. The final completed roadmap is displayed in Figure 6.9. Now the roadmap can be used for various search queries by projecting different periodic orbits onto the graph as the set of initial and goal nodes.

To explore transfers between periodic orbits at distinct energy levels from this roadmap,

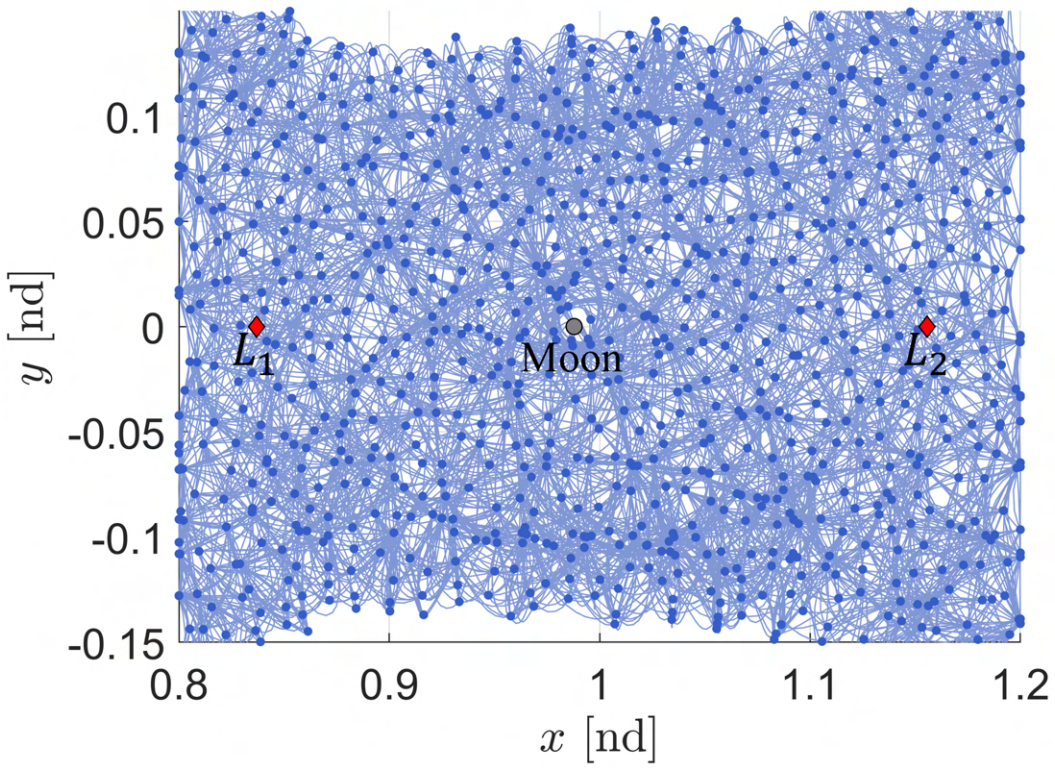


Figure 6.9: A planar roadmap constructed for $C_J = [3.12 - 3.16]$ to summarize the solution space near the Moon.

4 Lyapunov orbits are selected at Jacobi constants near the boundaries of the graph. Two L_1 Lyapunov orbits are computed at $C_J = \{3.125, 3.155\}$ and two L_2 Lyapunov orbits are computed at $C_J = \{3.125, 3.155\}$. These orbits are depicted in Figure 6.10 and are colored by Jacobi constant. Using the selected orbits, four transfer design scenarios are explored: two scenarios between Lyapunov orbits at distinct energy levels departing from an L_1 orbit and arriving to an L_2 orbit, and two scenarios between Lyapunov orbits at distinct energy levels departing from an L_2 orbit and arriving to an L_1 orbit. For each exploration of initial guesses for the four transfer design scenarios, 50 states along each desired orbit are added onto the roadmap displayed in Figure 6.9 to establish the set of initial and goal nodes for each scenario. The exploration of these four scenarios demonstrates using the single graph construction process across a range of energy levels to recover geometrically distinct transfers throughout different regions of the solution space at various Jacobi

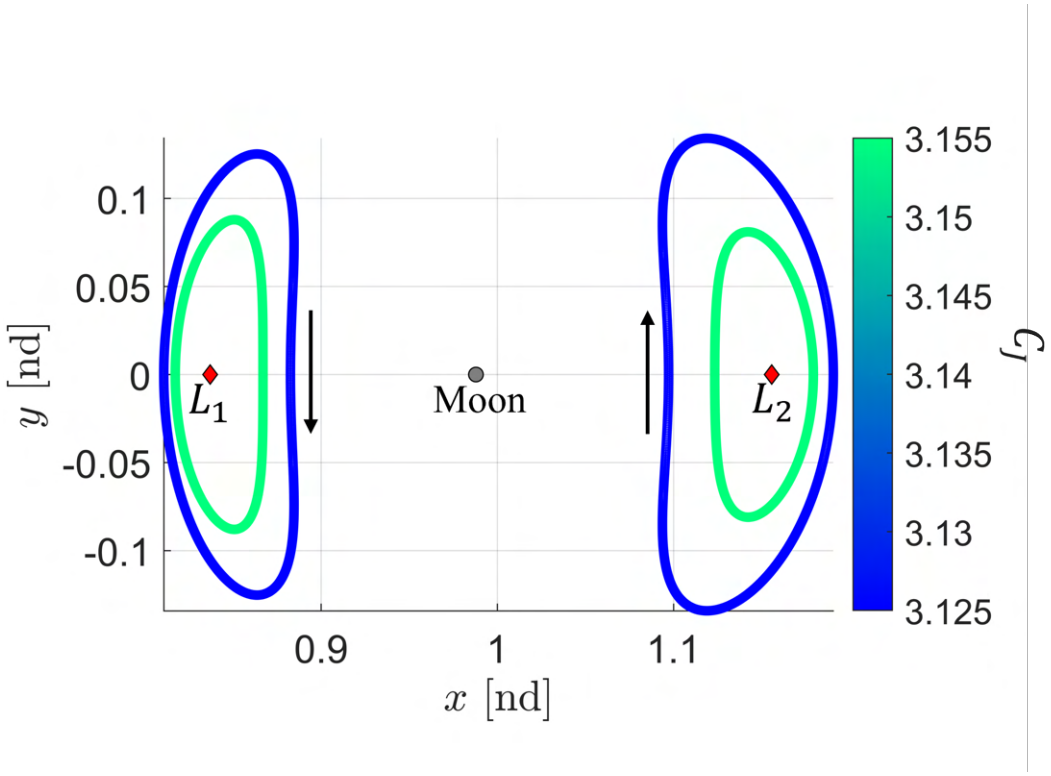


Figure 6.10: Two L_1 Lyapunov orbits computed at $C_J = \{3.125, 3.155\}$ and two L_2 Lyapunov orbits computed at $C_J = \{3.125, 3.155\}$, colored by Jacobi constant.

constants.

6.2.2 Planar Transfers from an L_1 Lyapunov Orbit at $C_J = 3.125$ to an L_2 Lyapunov Orbit at $C_J = 3.155$

Planar transfers are constructed by searching the graph from an L_1 Lyapunov Orbit at $C_J = 3.125$ to an L_2 Lyapunov Orbit at $C_J = 3.155$ using Dijkstra's and Yen's algorithm. To identify unique initial guesses, a dissimilarity tolerance of $\delta = 0.5$ is used. In Figure 6.11, 3 unique initial guesses are selected and the maneuvers along each initial guess are colored by the required maneuver magnitude at each node. The optimal initial guess, with a total maneuver cost of 600.3117 m/s, is depicted by the blue curves. The second initial guess selected, depicted by the orange curves, has a total maneuver cost of 606.8647 m/s. This second initial guess was the 4^{th} unique path in the list and the 164^{th} path in the k shortest path list. Finally, the last initial

guess selected is depicted by the green curves. This third initial guess has a total maneuver cost of 868.7853 m/s and was the 5^{th} unique path in the list and the 177^{th} path in the k shortest path list. These three initial guesses were selected as potential transfers between the desired orbits due to their difference in geometry and overall relative low-cost.

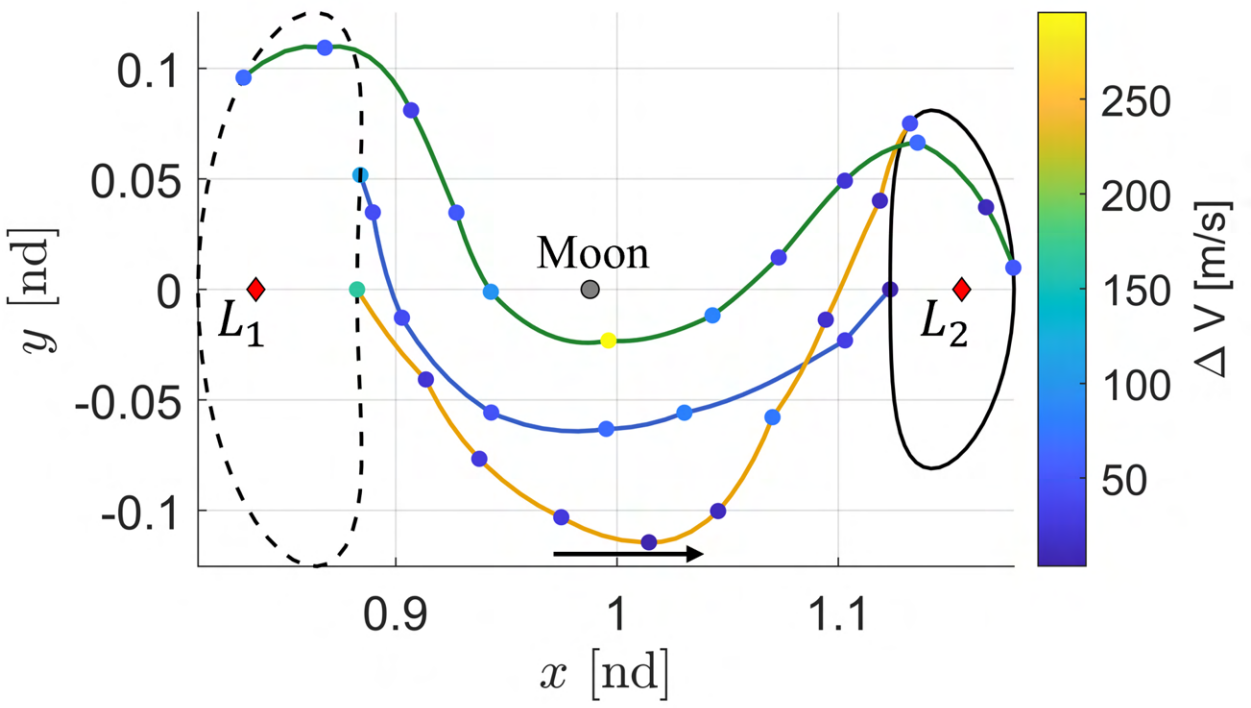
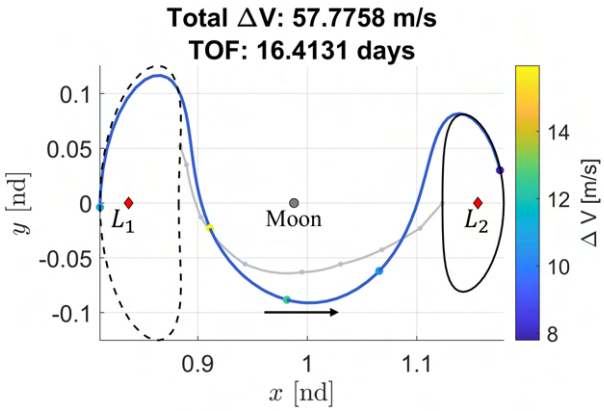


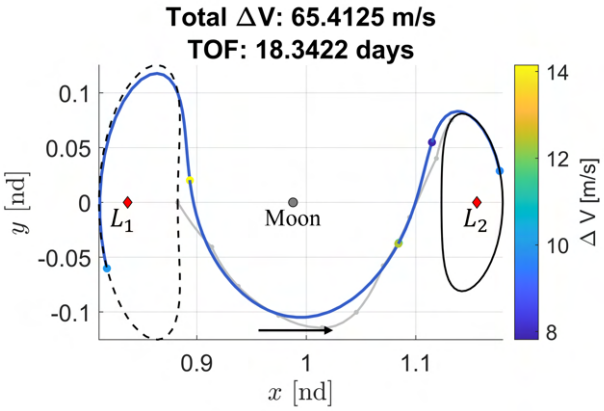
Figure 6.11: 3 unique initial guesses for transfers from an L_1 Lyapunov Orbit at $C_J = 3.125$ to an L_2 Lyapunov Orbit at $C_J = 3.155$.

Each initial guess is corrected and optimized for the sum of maneuver magnitudes squared using a collocation corrections algorithm and MATLAB's fmincon tool. The continuous transfers are displayed in Figure 6.12 where the correction solutions are depicted by the blue curves and each initial guess is depicted by the light grey curves. Each initial guess is corrected using one maneuver to depart the L_1 Lyapunov orbit, 3 maneuvers along the path at locations associated with the largest edge weights, and 1 additional maneuver to arrive onto the L_2 orbit. The first optimal initial guess is corrected to produce a continuous transfer, displayed in Figure 6.12a with a total maneuver cost of 57.7758 m/s and time of flight of 16.4131 days. The second initial guess selected

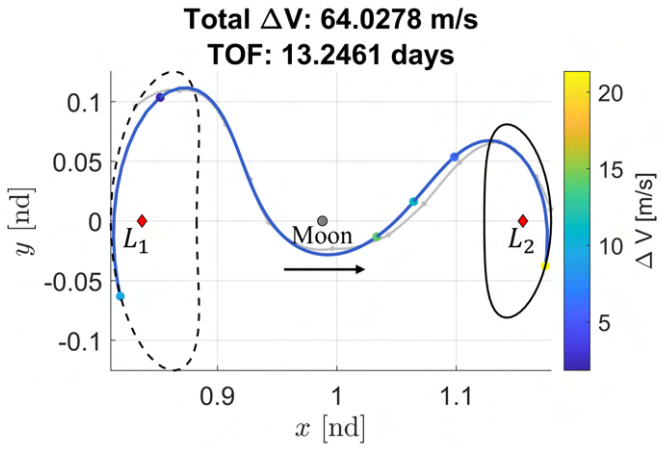
is corrected to produce a continuous transfer, displayed in Figure 6.12b, with a total maneuver cost of 65.4125 m/s and time of flight of 18.3422 days. The third initial guess selected is corrected to produce a continuous transfer, displayed in Figure 6.12c, with a total maneuver cost of 64.0278 m/s and time of flight of 13.2461 days.



(a) The corrected solution using the optimal (blue) initial guess in Figure 6.11.



(b) The corrected solution using the second selected (orange) initial guess in Figure 6.11.



(c) The corrected solution using the third selected (green) initial guess in Figure 6.11.

Figure 6.12: 3 continuous, maneuver-enabled transfers from an L_1 Lyapunov Orbit at $C_J = 3.125$ to an L_2 Lyapunov Orbit at $C_J = 3.155$

Each corrected initial guess results in a low-cost maneuver enabled transfer between the selected Lyapunov orbits. However, comparing the three corrected transfers, it is noted the first and

second selected initial guesses both optimize to transfers that are very geometrically similar. This demonstrates that while the dissimilarity measure used within Yen's algorithm recovers sufficiently unique initial guesses determined by the selected value of δ, there is no guarantee each initial guess will recover a sufficiently geometrically distinct transfer if only maneuver magnitude is optimized along the final transfer. If additional maneuvers were placed along each initial guess, the corrected solutions may slightly resemble their initial guess more, however the overall geometries of these two corrected solutions remains similar.

Comparing each initial guess with the respective optimized solution, it is noted that the second and third initial guess are corrected and optimized to recover transfers that are more geometrically similar than the first initial guess and optimized transfer. These sub-optimal initial guess, while possessing a slighter higher total cost, may be more accurate to the natural dynamics along the transfers, resulting in fewer differences during corrections. This demonstrates the benefit of generating multiple unique initial guesses; while the optimal initial guess is able to produce an optimized transfer, it may not be the best initial guess available within the graph.

6.2.3 Planar Transfers from an L_1 Lyapunov Orbit at $C_J = 3.155$ to an L_2 Lyapunov Orbit at $C_J = 3.125$

Planar transfers are constructed by searching the graph from an L_1 Lyapunov Orbit at $C_J = 3.155$ to an L_2 Lyapunov Orbit at $C_J = 3.125$ using Dijkstra's and Yen's algorithm. To identify unique initial guesses, a dissimilarity tolerance of $\delta = 0.5$ is used. In Figure 6.13, 3 unique initial guesses are selected and the maneuvers along each initial guess are colored by the required maneuver magnitude at each node. The optimal initial guess, with a total maneuver cost of 491.4463 m/s, is depicted by the blue curves. The second initial guess selected, depicted by the orange curves, has a total maneuver cost of 604.2817 m/s. This second initial guess was the 2^{nd} unique path in the list and the 84^{th} path in the k shortest path list. Finally, the last initial guess selected is depicted by the green curves. This third initial guess has a total maneuver cost of 957.3274 m/s and was the 8^{th} unique path in the list and the 206^{th} path in the k shortest path

list.

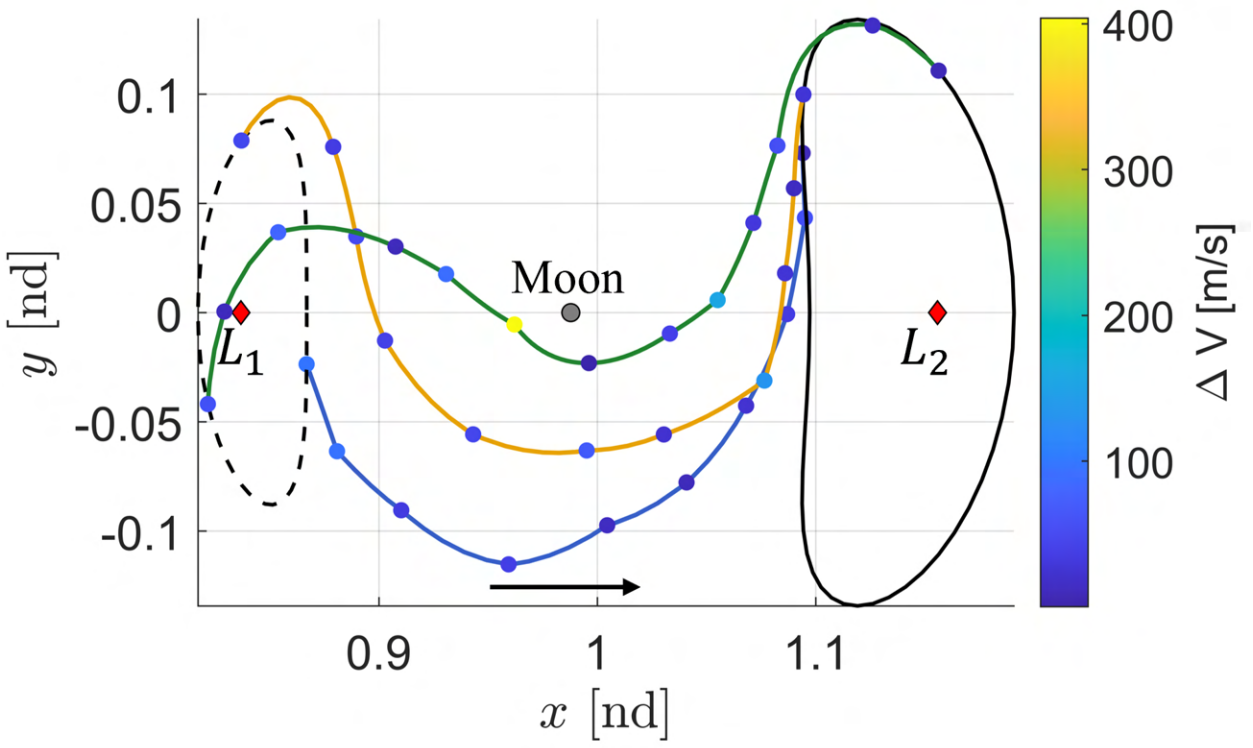


Figure 6.13: 3 unique initial guesses for transfers from an L_1 Lyapunov Orbit at $C_J = 3.155$ to an L_2 Lyapunov Orbit at $C_J = 3.125$.

Once again, these three initial guesses were selected as potential transfers between the desired orbits due to their difference in geometry and overall relative low-cost. If a larger value of δ was selected, each initial guess after $k_{unique} = 1$ would likely appear earlier in the unique paths list. However, the value of δ can be increased after Yen's algorithm is finished running. Therefore, a smaller value of δ is selected initially and can be increased if the resulting unique paths are not sufficiently different as desired. Additionally, these three initial guesses each depart the L_1 Lyapunov orbit at various locations. This demonstrates the success of including multiple states along each orbit used as the start and goal nodes for the search query and can increase the uniqueness of the solution paths recovered.

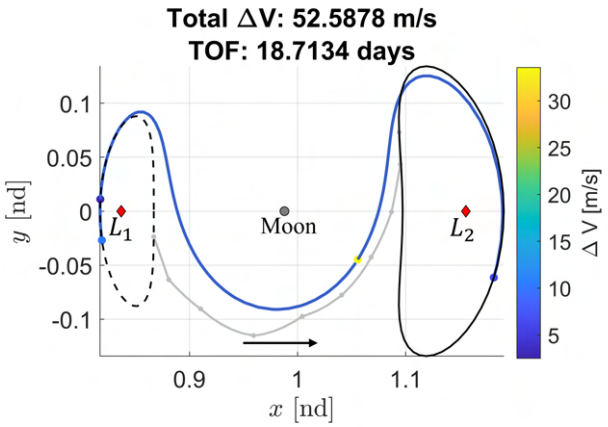
Each initial guess is corrected and optimized for the sum of maneuver magnitudes squared using a collocation corrections algorithm and MATLAB's fmincon tool. The continuous transfers

are displayed in Figure 6.14 where the correction solutions are depicted by the blue curves and each initial guess is depicted by the light grey curves. Each initial guess is corrected using one maneuver to depart the L_1 Lyapunov orbit, 2 maneuvers along the path at locations associated with the largest edge weights, and 1 additional maneuver to arrive onto the L_2 orbit. The first optimal initial guess is corrected to produce a continuous transfer with a total maneuver cost of 52.5878 m/s and time of flight of 18.7134 days displayed in Figure 6.14a, the second initial guess selected is corrected to produce a continuous transfer with a total maneuver cost of 48.9494 m/s and time of flight of 19.4977 days displayed in Figure 6.14b, and the third initial guess selected is corrected to produce a continuous transfer with a total maneuver cost of 41.0997 m/s and time of flight of 19.8926 days displayed in Figure 6.14c.

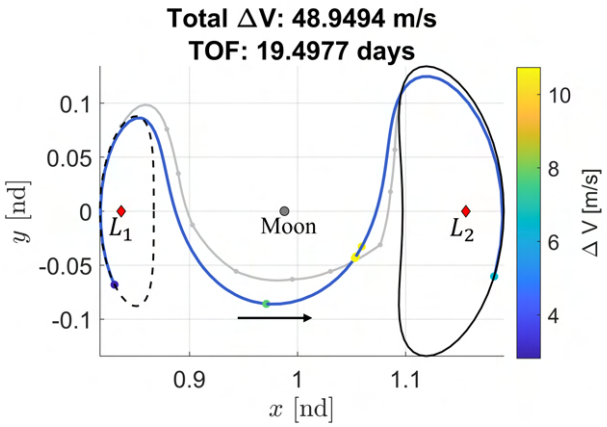
Similar to the previous search query, the first and second initial guess produce geometrically similar maneuver-optimal transfers, despite being sufficiently unique initial guesses based on the selected value of δ. While the two initial guesses are very similar as they approach the L_2 Lyapunov orbit, the corrected solutions both lie in between each initial guess for the first half of the transfer and contain slightly different departure locations from the L_1 Lyapunov orbit. Additionally, examining the third solution in Figure 6.14c, the two maneuvers allowed along the transfer both correct to locations very close together. These maneuvers could be combined into one larger maneuver to further reduce the cost of the initial guess without altering the geometry further. While each initial guess and corresponding transfer are valid solutions, additional initial guesses may be desirable to recover more geometrically distinct transfers by computing more solution paths with Yen's algorithm.

6.2.4 Planar Transfers from an L_2 Lyapunov Orbit at $C_J = 3.125$ to an L_1 Lyapunov Orbit at $C_J = 3.155$

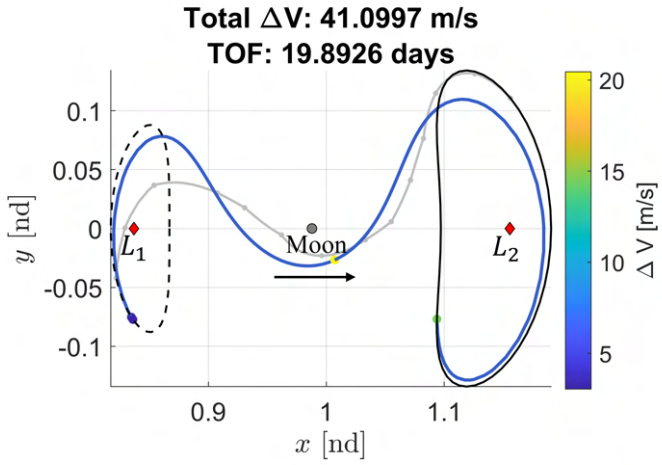
Since the roadmap was constructed without the dependence on the search queries, transfers that exist in the opposite direction can also be recovered by flipping the boundary conditions of each previous scenario. To explore the set of planar transfers from the L_2 Lyapunov orbit at $C_J = 3.125$



(a) The corrected solution using the optimal (blue) initial guess in Figure 6.13.



(b) The corrected solution using the second selected (orange) initial guess in Figure 6.13.



(c) The corrected solution using the third selected (green) initial guess in Figure 6.13.

Figure 6.14: 3 continuous, maneuver-enabled transfers from an L_1 Lyapunov Orbit at $C_J = 3.155$ to an L_2 Lyapunov Orbit at $C_J = 3.125$.

to the L_1 Lyapunov orbit at $C_J = 3.155$, initial guesses are constructed by searching the graph using Dijkstra's and Yen's algorithm. To identify unique initial guesses, a dissimilarity tolerance of $\delta = 0.5$ is used. In Figure 6.15, 2 unique initial guesses are selected and the maneuvers along each initial guess are colored by the required maneuver magnitude at each node. The optimal initial guess, with a total maneuver cost of 414.3157 m/s, is depicted by the blue curves. The second

initial guess selected, depicted by the orange curves, has a total maneuver cost of 676.8412 m/s. This second initial guess was the 2^{nd} unique path in the list and the 207^{th} path in the k shortest path list.

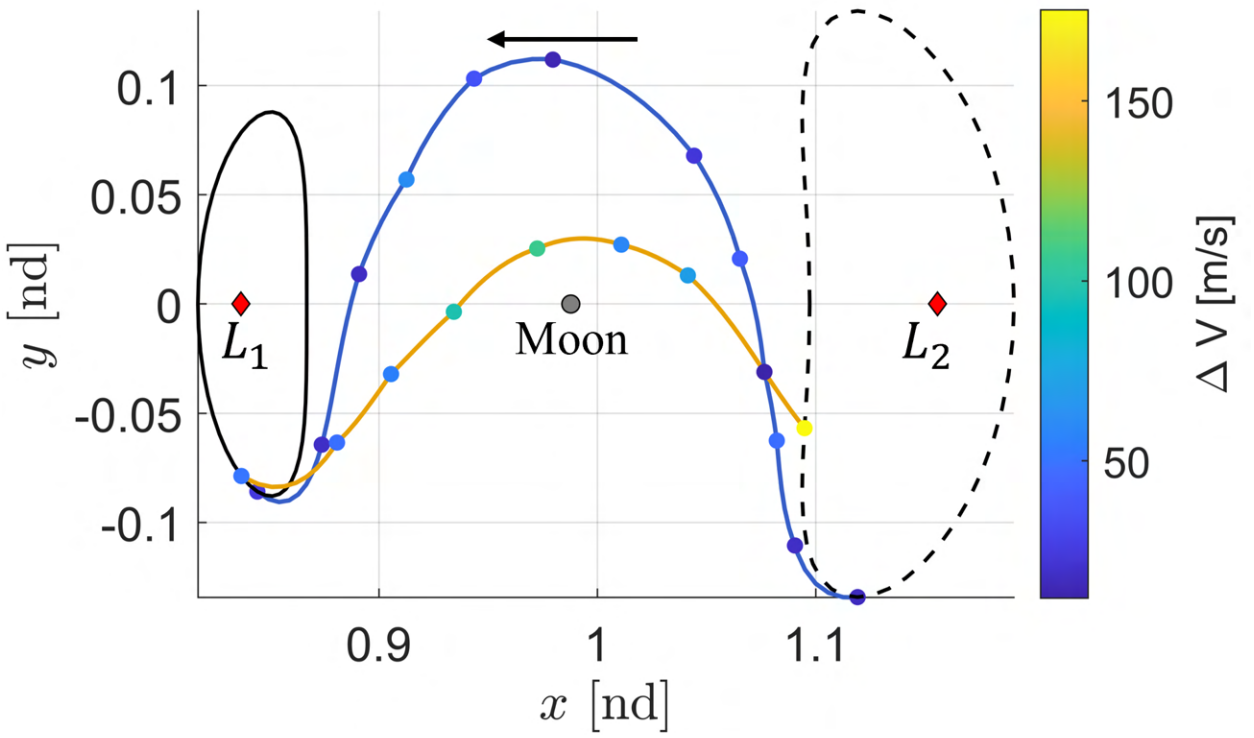


Figure 6.15: 2 unique initial guesses for transfers from an L_2 Lyapunov Orbit at $C_J = 3.125$ to an L_1 Lyapunov Orbit at $C_J = 3.155$.

Each initial guess is corrected and optimized for the sum of maneuver magnitudes squared using a collocation corrections algorithm and MATLAB's fmincon tool. The continuous transfers are displayed in Figure 6.16 where the correction solutions are depicted by the blue curves and each initial guess is depicted by the light grey curves. Each initial guess is corrected using one maneuver to depart the L_2 Lyapunov orbit and 1 additional maneuver to arrive onto the L_1 orbit. The first optimal initial guess is corrected with an additional 2 maneuvers along the initial guess to produce a continuous transfer with a total maneuver cost of 78.0839 m/s and time of flight of 19.6796 days, displayed in Figure 6.16a. The second initial guess selected is corrected with an additional 3 maneuvers along the initial guess to produce a continuous transfer with a total maneuver cost of

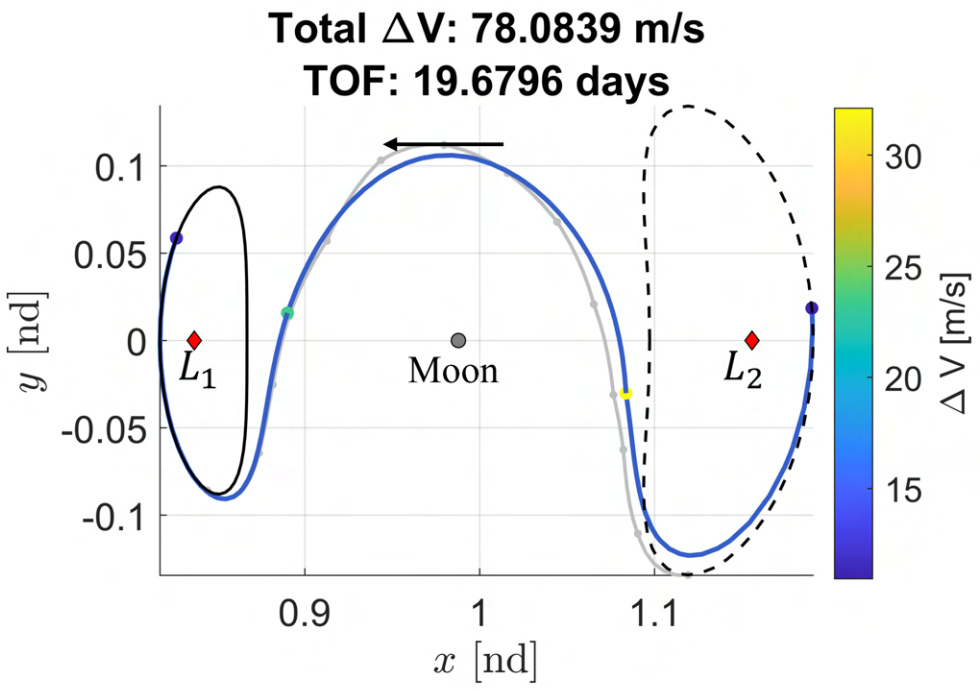
80.7641 m/s and time of flight of 25.7559 days displayed in Figure 6.16b.

For this scenario, the optimal and second sub-optimal unique initial guesses are corrected to recovered continuous transfers that very closely resemble their respective initial guesses. Each corrected transfer varies slightly in the departure and arrival locations for each orbit, however the second initial guess corrected to recover a transfer that varies noticeably in the departure segments. This is because the first node along the second initial guess was the location of the largest maneuver along the path. During corrections, the departure geometry of this transfer changed quite drastically, but produced a much smoother transfer and departure off the L_2 Lyapunov orbit. As a result, the first half of the transfer varies slightly from the initial guess.

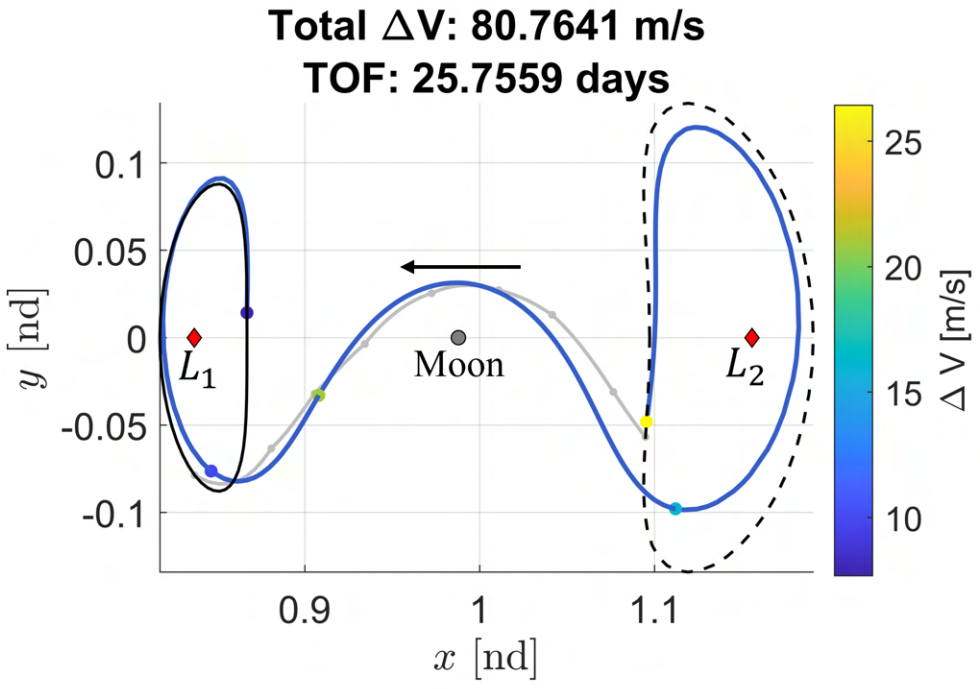
6.2.5 Planar Transfers from an L_2 Lyapunov Orbit at $C_J = 3.155$ to an L_1 Lyapunov Orbit at $C_J = 3.125$

Last, planar transfers are constructed by searching the graph for transfers from the L_2 Lyapunov orbit at $C_J = 3.155$ to the L_1 Lyapunov orbit at $C_J = 3.125$ using Dijkstra's and Yen's algorithm. To identify unique initial guesses, a dissimilarity tolerance of $\delta = 0.5$ is used. In Figure 6.17, 2 unique initial guesses are selected and the maneuvers along each initial guess are colored by the required maneuver magnitude at each node. The optimal initial guess, with a total maneuver cost of 553.8467 m/s, is depicted by the blue curves. The second initial guess selected, depicted by the orange curves, has a total maneuver cost of 729.7712 m/s. This second initial guess was the 2^{nd} unique path in the list and the 9^{th} path in the k shortest path list.

Each initial guess is corrected and optimized for the sum of maneuver magnitudes squared using a collocation corrections algorithm and MATLAB's fmincon tool. The continuous transfers are displayed in Figure 6.18 where the correction solutions are depicted by the blue curves and each initial guess is depicted by the light grey curves. Each initial guess is corrected using one maneuver to depart the L_2 Lyapunov orbit, 2 maneuvers along the path at locations associated with the largest edge weights, and 1 additional maneuver to arrive onto the L_1 orbit. The first optimal initial guess is corrected to produce a continuous transfer with a total maneuver cost of



(a) The corrected solution using the optimal (blue) initial guess in Figure 6.15.



(b) [The corrected solution using the second optimal (orange) initial guess in Figure 6.15.

Figure 6.16: 2 continuous, maneuver-enabled transfers from an L_2 Lyapunov Orbit at $C_J = 3.125$ to an L_1 Lyapunov Orbit at $C_J = 3.155$.

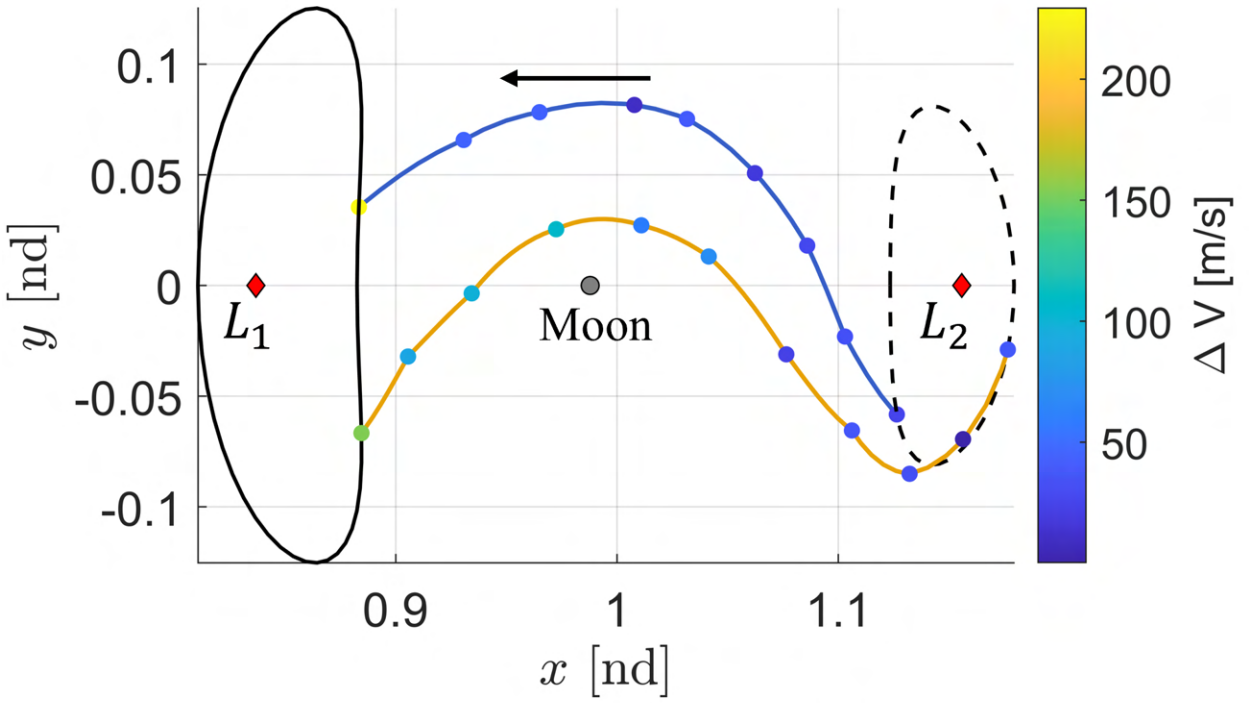
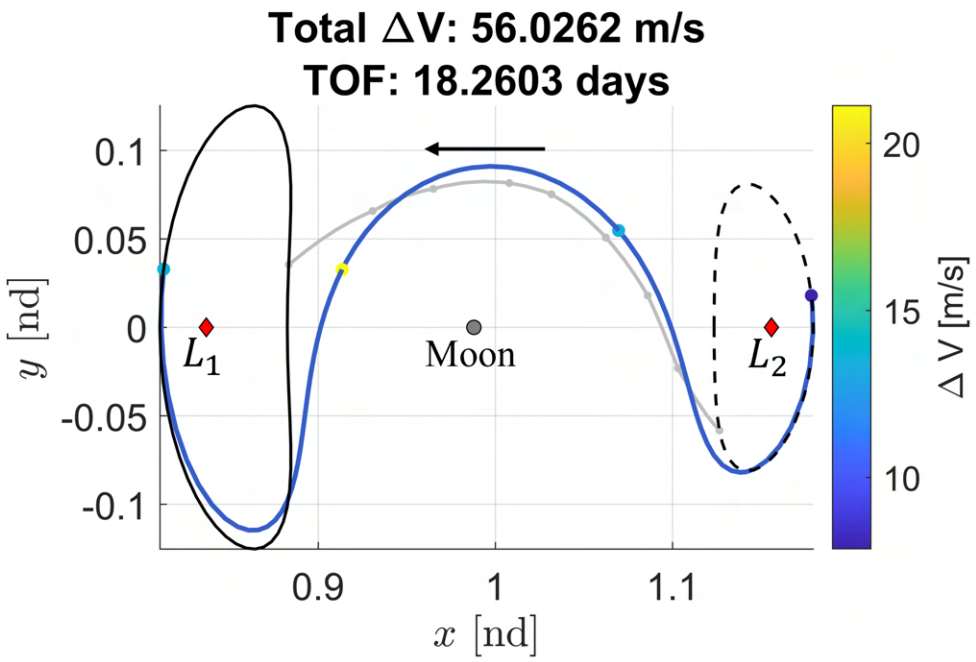


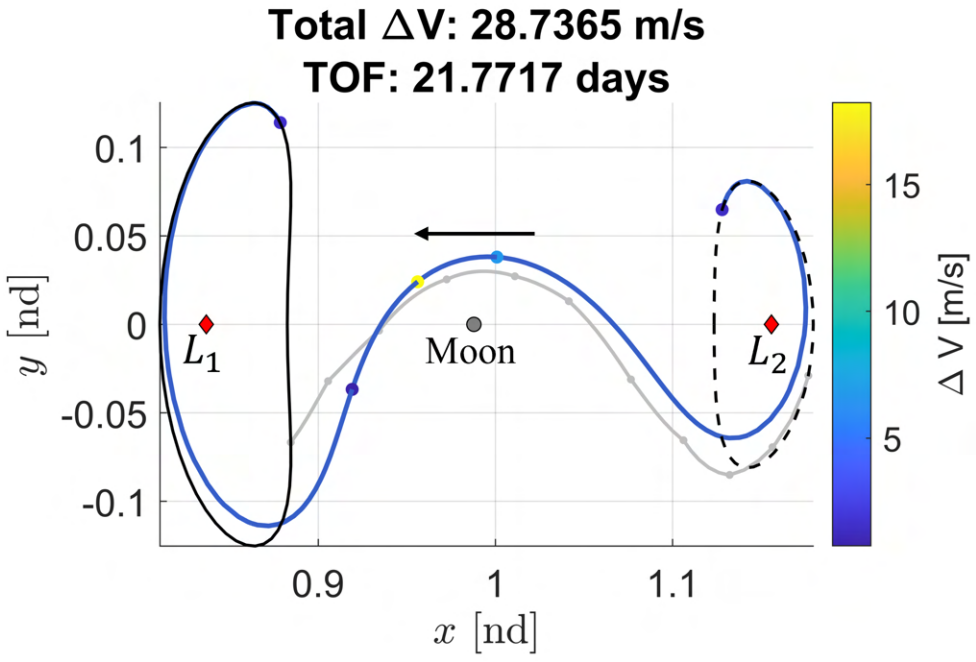
Figure 6.17: 2 unique initial guesses for transfers from an L_2 Lyapunov Orbit at $C_J = 3.155$ to an L_1 Lyapunov Orbit at $C_J = 3.125$.

56.0262 m/s and time of flight of 18.2603 days displayed in Figure 6.18a and the second initial guess selected is corrected to produce a continuous transfer with a total maneuver cost of 28.7365 m/s and time of flight of 21.7717 days displayed in Figure 6.18b.

Each corrected transfer closely resembles the respective initial guess with the selected number of maneuvers allowed along the path. While each initial guess has a relatively smooth departure off the L_2 Lyapunov orbit, the end of the transfer approaches the L_1 Lyapunov orbit at a more abrupt angle. The final segment and approach to the L_1 orbit is smoothed out during corrections, leading to a significant change in the arrival location on the orbit. Interestingly, the second initial guess resulted in a transfer that is not only sufficiently unique but also recovered a geometrically distinct transfer, despite being only the 9^{th} path found within the graph. This second initial guess appears much earlier in the shortest path list compared to the other initial guesses demonstrated in this section past the optimal initial guess. This shows the variability in the number of available



(a) The corrected solution using the optimal (blue) initial guess in Figure 6.17.



(b) [The corrected solution using the second optimal (orange) initial guess in Figure 6.17.

Figure 6.18: 2 continuous, maneuver-enabled transfers from an L_2 Lyapunov Orbit at $C_J = 3.155$ to an L_1 Lyapunov Orbit at $C_J = 3.125$.

paths for each new transfer design scenario; depending on the number of nodes and edges in the roadmap and the connections to the graph made from each initial and final orbit, Yen's algorithm may recover solution paths more rapidly in some scenarios.

6.3 Spatial Transfers Between Northern Halo Orbits at $C_J = 3.15$

To construct spatial transfers between northern halo orbits computed at $C_J = 3.15$, first a roadmap is constructed to summarize the solution space near the Moon. Using the sampling-based kinodynamic planning approach to trajectory design presented in the previous chapter, a graph is constructed using the parameters listed in Table 6.2.

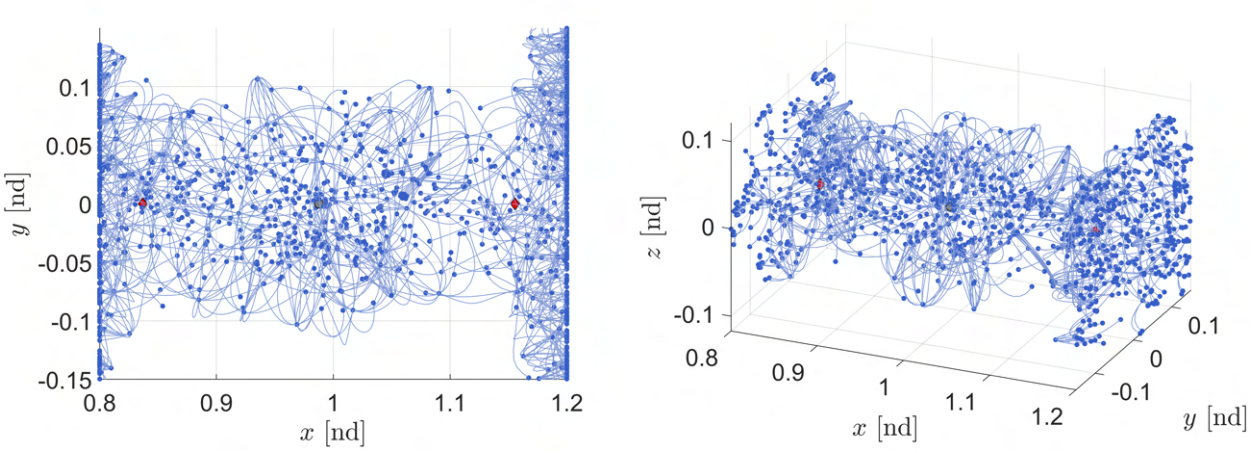
Table 6.2: Graph construction parameters selected to construct a spatial roadmap at $C_J = 3.15$.

Parameter	Value
ℓ	0.08
n_c	2
τ_{DT}	10^{-5}
n	3

The values of the parameters required to construct the spatial roadmap were selected by weighing the computational requirements required to construct and search the graph, as well as the memory requirements of the graph itself. Overall, larger, but fewer neighborhoods were required to sufficiently summarize the desired solution space with fewer overlapping neighborhoods required to cover the solution space. Similar to the planar roadmaps constructed, the number of desired velocity vectors computed for each set of 1-neighbors was selected at $n = 3$, the minimum recommended value in order to span the 3-dimensional velocity space. Since this roadmap is constructed at a single energy level, $n = 3$ was found to be a sufficient value in order for a neighborhood to explore its local region of the solution space.

To cover the solution space using the above selected graph parameters, 100 neighborhoods are constructed. The distribution of the nodes, depicted by the blue circles, and natural edges, depicted by the blue curves, within for neighborhood are displayed in Figure 6.19, shown both from the xy-plane in Figure 6.19a and a 3-dimensional representation in Figure 6.19b. While there seems to be a significant number of nodes at the boundaries of the desired solution space at $x = [0.8, 1.2]$, this is due to all edges constructed that depart past these boundaries being trimmed until the

entire edge is located within the desired region of the solution space; this results in nodes with an x-coordinate at the boundary.



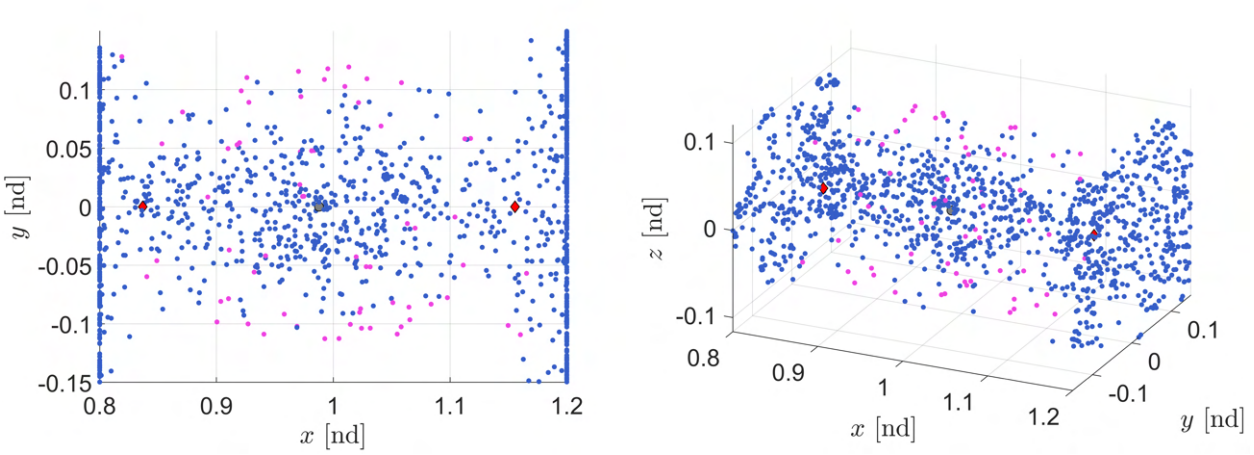
(a) xy-plane visualization.

(b) 3-dimensional visualization.

Figure 6.19: The neighborhoods containing nodes (blue circles) and natural edges (blue curves) for a spatial roadmap constructed at $C_J = 3.15$ using graph parameters $\ell = 0.08$ and $n_c = 2$.

Next, the dispersion is minimized in the configuration space. Due to the larger size of each neighborhood, there are larger regions in the solution space that do not containing any nodes. Furthermore, there is a denser number of nodes within the center near the Moon due to the neighborhoods constructed along the boundaries that propagate natural edges inward of the graph. Therefore, a tolerance of 10^{-5} was selected to construct additional nodes towards the boundaries of the feasible motion. The node distribution from the neighborhoods, depicted by the blue circles, and from the Delaunay triangulations, depicted by the pink circles, are displayed in Figure 6.20. Analyzing the distribution, it is noted that the triangulations constructed a majority of the additional nodes near the boundaries of the zero velocity surface. This is the desired outcome since minimal additional nodes were required near the Moon, but additional nodes were required to fill in the empty regions of the solution space. 64 additional unique position vectors were identified for the graph, resulting in 384 new nodes.

Finally, each edge gains a neighbor forward and backward in time until the graph is strongly



(a) xy-plane visualization.

(b) 3-dimensional visualization.

Figure 6.20: The distribution of nodes constructed from neighborhoods (blue) and Delaunay triangulations (pink).

connected. This results in 16689 time-varying edges constructed. The final completed roadmap is displayed in Figure 6.21. Now the roadmap can be used for various search queries by projecting different periodic orbits onto the graph as the set of initial and goal nodes. For this roadmap, transfers will be constructed between an L_1 and L_2 northern halo orbit constructed at $C_J = 3.15$. To set the initial and goal nodes within the graph, 10 states along each orbit are added to the roadmap. The number of states along each orbit is considerably smaller than the previous scenarios demonstrated to reduce the size of all potential start and goal node pairs allowed within Yen's algorithm. In doing so, the solution paths generated ideally have larger geometrical differences and a shorter computational requirement due to departing and arriving to the orbits at different locations.

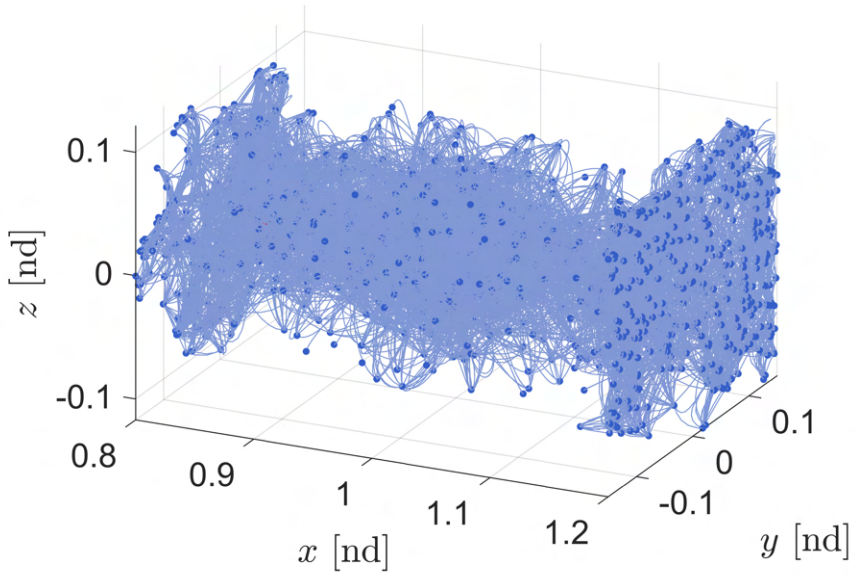
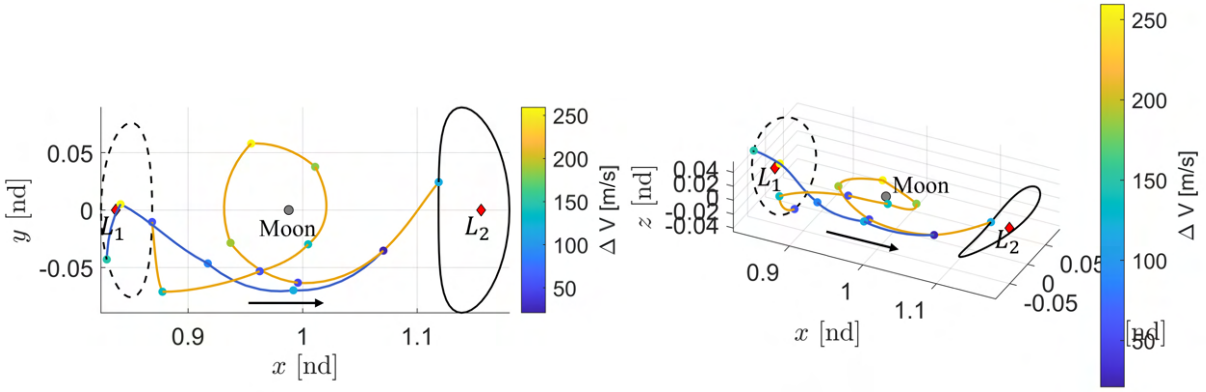


Figure 6.21: A spatial roadmap constructed for $C_J = 3.15$ to summarize the solution space near the Moon.

6.3.1 Spatial Transfers from an L_1 Northern Halo Orbit to an L_2 Northern Halo Orbit at $C_J = 3.15$

Using Dijkstra's and Yen's algorithm, the first two unique solution paths recovered are used to construct two initial guesses for transfers from the L_1 northern halo orbit to the L_2 northern halo orbit. The optimal initial guess, depicted by the blue curves, and second unique sub-optimal initial guess, depicted by the orange curves, are displayed in Figure 6.22 where the maneuvers at each node are colored by their magnitudes. The initial and final halo orbits are plotted with dashed and solid black lines, respectively. The first unique initial guess, also the optimal solution path for the graph, possesses a total $\Delta V = 755.9347\ m/s$, while the second unique initial guess, the 31^{st} k best path, possesses a total $\Delta V = 1205.201\ m/s$.

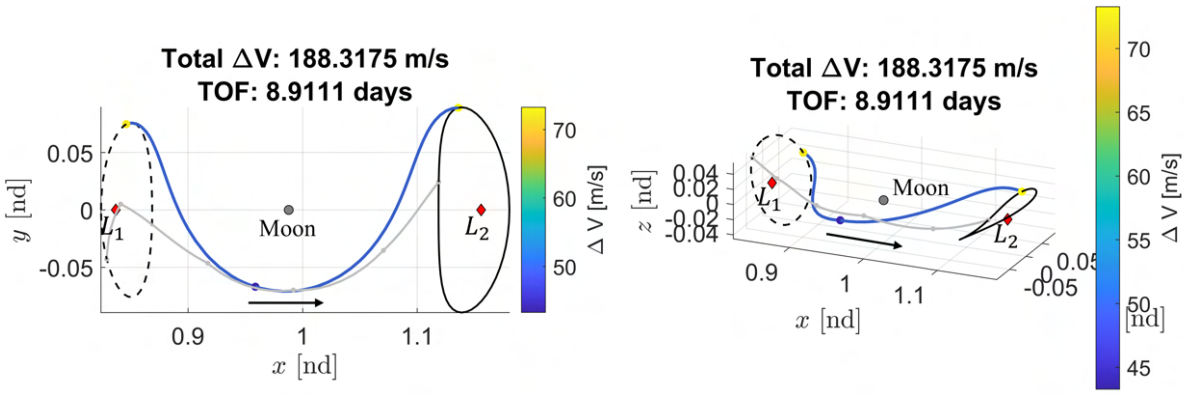
Each initial guess in then input into a collocation corrections algorithm and optimized using at least one maneuver to depart the initial L_1 orbit and one maneuver to arrive onto the L_2 orbit. The first initial guess is corrected using 1 maneuver along the initial guess at the location of the



(a) xy-plane visualization. (b) 3-dimensional visualization.

Figure 6.22: The optimal initial guess (blue) and second unique initial guess (orange) for a transfer from an L_1 to an L_2 northern halo orbit at $C_J = 3.15$.

node with a similar x-coordinate as the Moon. The optimized transfer has a total maneuver cost of 188.3175 m/s and time of flight of 8.9111 days, and is depicted in Figure 6.23. The second unique

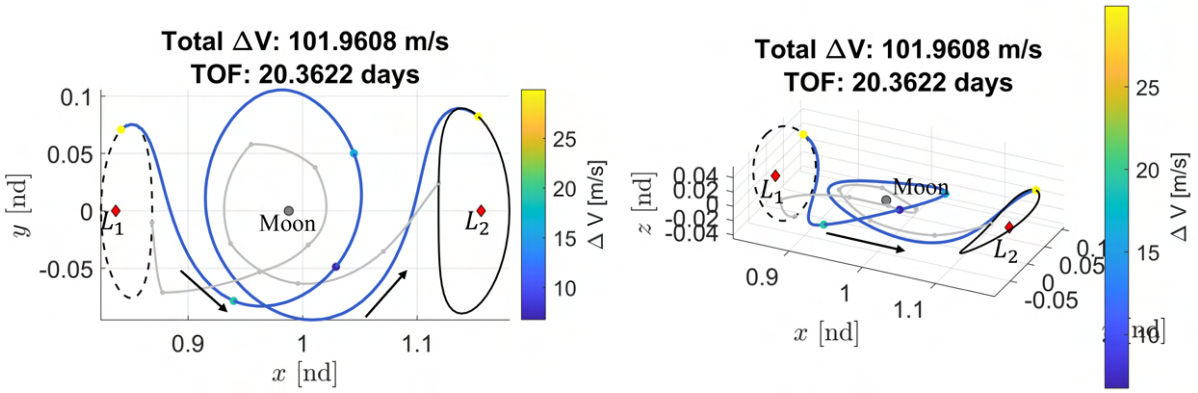


(a) xy-plane visualization. (b) 3-dimensional visualization.

Figure 6.23: The optimized solution for a transfer from an L_1 to an L_2 northern halo orbit at $C_J = 3.15$ using the optimal initial guess.

initial guess is corrected using 3 maneuvers along the initial guess to preserve geometry of the revolution. The optimized transfer has a total maneuver cost of 101.9608 m/s and time of flight of 20.3622 days, and is depicted in Figure 6.24. The optimized transfer does possess a larger revolution around the Moon, as well as a change in orientation of the first half of the transfer departing the L_1

orbit, compared to the initial guess. This continuous transfer more closely resembles the geometry of trajectories along the available unstable and stable hyperbolic invariant manifolds that depart and arrive from the selected northern halo orbits.



(a) xy-plane visualization.

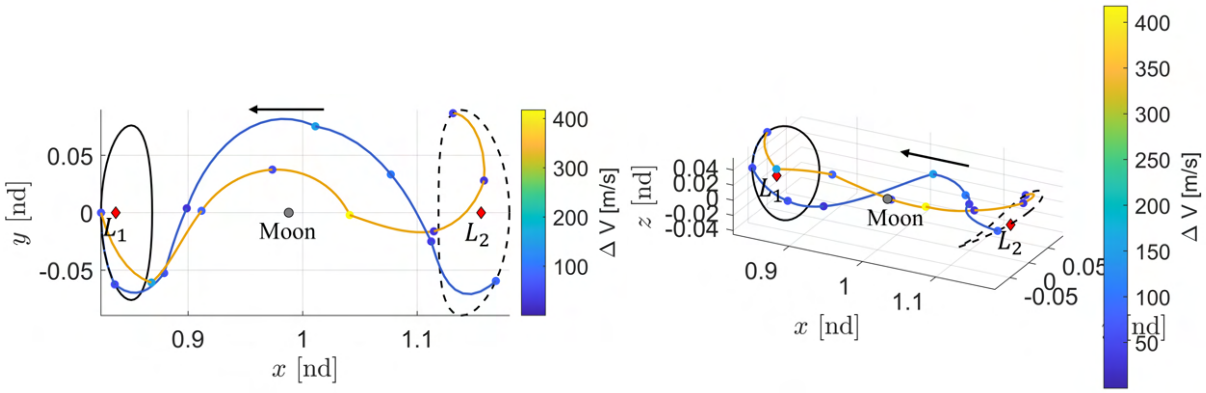
(b) 3-dimensional visualization.

Figure 6.24: The optimized solution for a transfer from an L_1 to an L_2 northern halo orbit at $C_J = 3.15$ using the second unique initial guess.

The two optimized transfers vary slightly from the geometry of their respective initial guess, most notably at the departure location from the L_1 orbit. This abrupt change in velocity at the first edge for both initial guesses is smoothed out during corrections, resulting in a different departure location and initial geometry of both transfers. Due to the increase in size of the graph required to sufficiently summarize spatial motion for this roadmap, the search algorithm process takes significantly longer than the previous scenarios demonstrated. For this search query, the second unique solution path provided an initial guess that possesses a substantial difference in geometry compared to the first optimal initial guess, which also required a relatively short computational time to generate these two paths. To recover additional transfers with new geometries, such as additional revolutions about the Moon, the computational time required to search the graph sufficiently is anticipated to be quite large compared to the planar search queries.

6.3.2 Spatial Transfers from an L_2 Northern Halo Orbit to an L_1 Northern Halo Orbit at $C_J = 3.15$

By switching the set of initial and goal nodes, the optimal solution path can be recovered to construct an initial guess for a transfer from the L_2 northern halo orbit to the L_1 northern halo orbit with Dijkstra's algorithm. The optimal initial guess, depicted by the blue curves, and second unique initial guess, depicted by the orange curves, are displayed in Figure 6.25 where the maneuvers at each node are colored by their magnitudes. The first unique initial guess possesses a total $\Delta V = 531.301\ m/s$, while the second unique initial guess, the 915^{th} k best path, possesses a total $\Delta V = 885.2348\ m/s$.

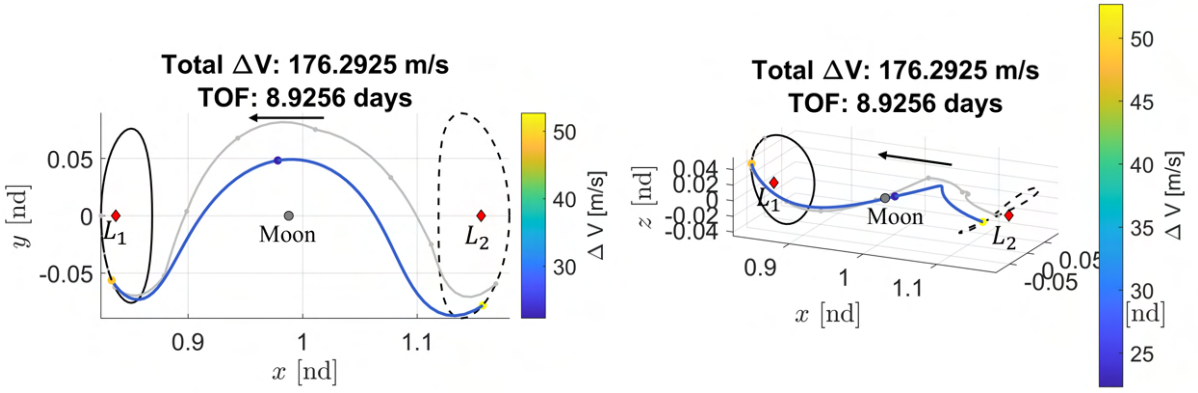


(a) xy-plane visualization.

(b) 3-dimensional visualization.

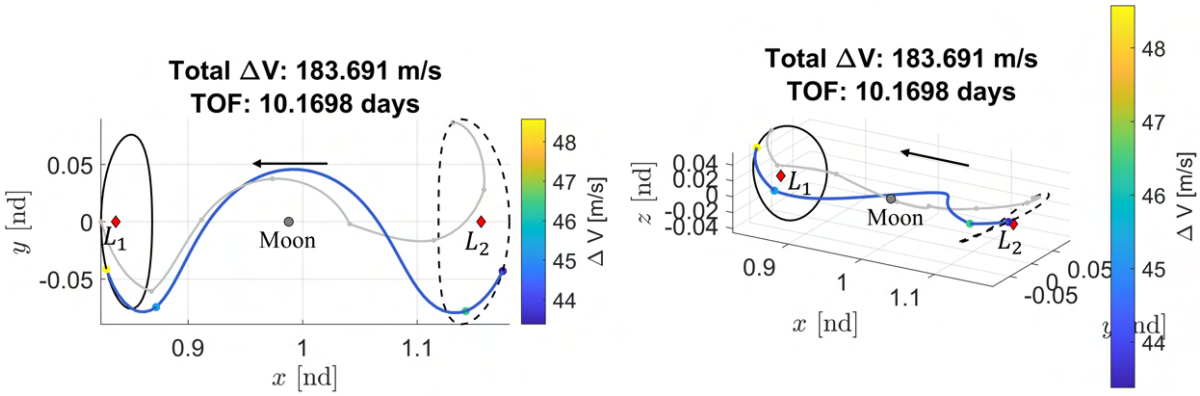
Figure 6.25: The optimal initial guess (blue) and second unique initial guess (orange) for a transfer from an L_2 to an L_1 northern halo orbit at $C_J = 3.15$.

Each initial guess in then input into a collocation corrections algorithm and optimized using one maneuver to depart the initial L_2 orbit, one maneuver to arrive onto the L_1 orbit, and two maneuvers along the initial guess at locations of larger edge weights. The first unique initial guess is optimized to recover a transfer that has a total maneuver cost of 176.2925 m/s and time of flight of 8.9256 days, and is depicted in Figure 6.26. The second unique initial guess is corrected to recover an optimized transfer that has a total maneuver cost of 183.691 m/s and time of flight of 10.1698 days, is depicted in Figure 6.27.



(a) xy-plane visualization. (b) 3-dimensional visualization.

Figure 6.26: The optimized solution for a transfer from an L_2 to an L_1 northern halo orbit at $C_J = 3.15$ using the optimal initial guess.



(a) xy-plane visualization. (b) 3-dimensional visualization.

Figure 6.27: The optimized solution for a transfer from an L_2 to an L_1 northern halo orbit at $C_J = 3.15$ using the second unique initial guess.

While the first two unique initial guesses recovered from the graph are sufficiently dissimilar based on the selected dissimilarity tolerance, each initial guess is corrected and optimized to recover two transfers that are very geometrically similar and possess similar maneuver and time of flight requirements. Both transfers resemble the geometry of the optimal initial guess, rather than the second unique initial guess changing geometry considerably. This may indicate the second unique initial guess is not a sufficient initial guess for this geometry for this transfer and additional unique initial guesses should be recovered from the graph if additional transfers are desired to be explored.

However, due to the computational requirements for this search query, only initial guesses that resemble the two unique solution paths depicted in this section were able to be recovered. Additionally, while the search query departing from the L_1 northern halo orbit arriving to the L_2 northern halo orbit recovered more unique geometries, such as the 1-revolution transfer, this search query did not. This may indicate the dynamics were more sufficiently summarized in the L_1 to L_2 direction of motion compared to the L_2 to L_1 direction of motion.

6.4 Analysis of Roadmap Construction and Search Algorithm Efficiency for Trajectory Design Scenarios

The trajectory design framework presented in Chapter 5 is formulated such that it may be adaptable to other solution spaces modeled by any multi-body system with minimal input from a trajectory designer. However, the methods presented for graph construction do not guarantee an optimal roadmap. The selection of graph construction parameters and the size of the desired solution space the roadmap is constructed for both directly influence the size of the graph, increasing the number of nodes and edges. Furthermore, as the size of the graph increases, the computational time of the search algorithms increases as well. While additional unique initial guesses lie within the roadmap, Yen's algorithm requires an extensive amount of computational time to recover unique paths. This limitation results in fewer unique initial guesses constructed for each transfer design scenario presented in this section, specifically as demonstrated in the transfer scenario between Lyapunov orbits at distinct energy levels and the spatial transfer design scenario between halo orbits. In each of these two examples, the dimensionality of the problem increases and it becomes increasingly more difficult to efficiently summarize the solution space using the roadmap construction methods presented in this work. Other potential node sampling or edge construction methods may offer a better discrete graph representation to reduce the size of the graph and sufficiently summarize the solution space. Additional computational time or a more efficient search algorithm than Yen's algorithm should be implemented to generate additional unique paths that may be used to recover more geometrically distinct transfers.

Chapter 7

Concluding Remarks

A new method for designing initial guesses for trajectories in multi-body systems using sampling-based kinodynamic planning was presented in this dissertation. This chapter details the summary and main contributions of this work, as well as recommendations for future research.

7.1 Summary

Designing trajectories in chaotic multi-body systems can be a challenging process that relies heavily on the intuition of a human trajectory designer. Using state of the art trajectory design methods, constructing a feasible initial guess that produces a continuous trajectory that satisfies all spacecraft constraints and mission parameters is a time-consuming process. Exploring the complex design space for all feasible solutions is often an intractable problem for a human-in-the-loop. Furthermore, if any constraints or parameters for the trajectory are altered, redesigning the initial guess is a non-intuitive process.

Path planning techniques, commonly used within linear systems and fields such as robotics or control systems, offer solutions to similar challenges for automatically constructing a valid path that adheres to a given set of obstacles and parameters. Specifically, kinodynamic motion planning techniques, a category of path planning, explore a given environment by constructing a discrete graph representation that sufficiently summarizes the set of possible configurations and motion allowed within the environment. All information contained within the graph is then subject to a set of kinematic and dynamic constraints, as well as any static obstacles known within the

environment.

By leveraging these motion planning techniques, a framework for constructing trajectories modeled in the CR3BP is presented by transforming the trajectory design problem into a discrete path planning problem. First, kinodynamic motion planning techniques are used to summarize the chaotic solution space modeled by a multi-body system by constructing a discrete graph using random sampling that reduces the burden on a human-in-the-loop. Then, initial guesses for trajectories are recovered automatically by searching the graph with graph search algorithms. These initial guesses then seed a corrections and optimization algorithm using collocation to recover a geometrically distinct set of continuous trajectories between user-defined initial and final states.

Using techniques from sampling-based kinodynamic planning, specifically probabilistic roadmap generation, a graph is constructed to summarize a portion of the solution space modeled by the Earth-Moon CR3BP. This graph is constructed such that it contains nodes and edges; nodes represent valid spacecraft states and edges represent either impulsive maneuvers or small natural trajectory segments that connect neighboring nodes. Nodes and bundles of edges that represent natural trajectory motion are first constructed to represent local neighborhoods, which are small graphs that cover a spherical region of the configuration space. Neighborhoods are constructed until the desired solution space is covered. Then, edges are globally constructed until the graph is strongly connected, ensuring a spacecraft located at a state associated with any node in the graph can traverse to any other node within the graph. By employing an iterative approach to both the node sampling and edge construction, an adaptive number of nodes and edges are constructed for the graph. To construct a roadmap, three parameters are required to be selected prior to graph construction. The selection of these parameters, as well as the influence of their selection on the graph, are presented. This reduces the number of predefined variables required to select a priori in order to construct an efficient representation of the solution space. The procedure of constructing a graph is demonstrated by summarizing the solution space near the Moon.

Using the graph previously constructed, initial guesses for trajectories are then automatically recovered by searching the graph for solution paths between user-defined start and goal nodes with

discrete graph search algorithms. Specifically Dijkstra's algorithm and Yen's algorithm with a dissimilarity measure are used to recover a set of unique solution paths by repeatedly searching various subgraphs. All solution paths recovered are a sequence of nodes and edges that minimize the sum of the desired search heuristic across the path. In this work, all solution paths minimize the sum of scaled impulsive maneuvers, where each impulsive maneuver is normalized by the minimum speed of the states associated with the beginning and ending node. These solution paths are then used as initial guesses for trajectories. From the set of recovered unique initial guesses, selected paths are then corrected and optimized for the sum of maneuvers squared to recover a set of continuous, geometrically distinct trajectories. Using the graph constructed to summarize the desired solution space near the Moon, the process is analyzed by constructing multiple geometrically distinct trajectories between an L_1 and L_2 Lyapunov orbit.

This trajectory design process using sampling-based kinodynamic planning is demonstrated for constructing geometrically distinct transfers between periodic orbits in three scenarios: 1) planar transfers between an L_1 and L_2 Lyapunov orbit at a single energy level, 2) planar transfers between L_1 and L_2 Lyapunov orbits at distinct energy levels, and 3) spatial transfers between an L_1 and L_2 northern halo orbit at a single energy level. For each of these examples analyzed, a single roadmap was generated and repeatedly searched to recover the set of unique initial guesses a human trajectory designer can explore, aiding in the rapid and automatic exploration of the design space. Furthermore, as demonstrated in the multi-energy level scenario, the desired initial and final orbits may be modified without the need to reconstruct the entire roadmap. By changing the boundary conditions for the search queries, new unique initial guesses are constructed rapidly. These initial guesses then seed the corrections algorithm to recover a set of geometrically distinct transfers between the desired periodic orbits selected by a user.

In this work, a new method for trajectory design is presented using techniques from motion planning, graph theory, and graph searches. Using this combination of algorithms, the burden on a human trajectory designer is reduced by limiting the parameters required to be selected a priori. The discrete graph is constructed to sufficiently summarize a portion of the solution space

using a combination of randomly sampled spacecraft states and constructing impulsive maneuvers and small trajectory segments between states. By searching the graph repeatedly, unique initial guesses are recovered that geometrically resemble feasible trajectories within the solution space. These initial guesses are then corrected and optimized for the total sum of maneuver magnitudes squared to produce a set of geometrically distinct trajectories. The scenarios presented in this work demonstrate the applicability of using sampling-based kinodynamic planning across foundational trajectory design scenarios in the Earth-Moon CR3BP.

7.2 Recommendations for Future Work

Using the sampling-based kinodynamic planning framework presented in this dissertation, many phases within the trajectory design process may be improved upon for future research to increase the efficiency of the process as well as the computational burden. The recommendations for items that warrant further investigation include:

- During roadmap construction, the graph is constructed to summarize the solution space. Additional research may improve the efficiency of the graph by investigating any optimality criteria for both graph construction and the quality of solution paths. If the number of nodes and edges required to efficiently summarize a solution space and sufficiently capture the dynamics of the system was precomputed for each new environment, then the graph construction may be more computationally efficient and produce higher quality solution paths that may be more optimal than the paths demonstrated in this work. Additionally, optimal solution paths may offer a guarantee of potentially desirable trajectories for a given search query based on a more efficient roadmap.

- The roadmap requires three parameters to be selected prior to graph construction. While the influence of these parameters on roadmap constructed was explored within this work, additional investigations into these parameters may further reduce the burden on a human-in-the-loop and improve the quality of the results. Investigations to either improve the

automatic selection of these parameters based on environment parameters, or new methods that eliminate the need for these parameters, would be useful avenues for future work.

- While Yen's algorithm was selected for this work to demonstrate the applicability of graph search algorithms, it is very computationally inefficient while requiring a large space complexity. Additionally, the run time and the characteristic of being a sequential algorithm is a limiting factor of the search results presented in this work. Further investigations for more computationally efficient search algorithms, such as Lawler's algorithm or bi-objective Dijkstra's algorithm, could improve the robustness of the search process and led to additional geometries for various trajectory design scenarios.

- The roadmap constructed for the scenarios demonstrated in this work serves as a proof of concept for trajectory design using sampling-based kinodynamic planning. The graphs are constructed to be adaptable and easily modifiable in order to construct unique initial guesses for various trajectory design scenarios. Further investigations on the adaptable steps of the roadmap, relative to unique search queries, can increase the variety of the solution paths recovered, directly influencing the exploration of the design space to rapidly construct new sets of geometrically distinct trajectories. Suggestions include the use of different periodic orbits or spacecraft states as the boundary conditions on the search results, the selection of the objective function to construct the edge weights within the graph, or the computational time Yen's algorithm is provided to run.

- Motion planning techniques and corrections algorithms for trajectory design both support the use of various constraints in order to support end mission goals. An investigation on translating trajectory design constraints to static constraints or obstacles, such as cone angle constraints for communication or fly by requirements for a primary body, can be imposed onto a roadmap after graph construction to increase the adaptability of a roadmap for a given trajectory design scenario without requiring a new graph to be constructed. Additionally, during the graph searches, solution path constraints, such as total time of flight

or propellant requirements, can be incorporated in order to find all solution paths within the set of trajectory constraints. The exploration of the addition of any desired constraints both during graph construction and the search queries can increase the applicability of constructing complex trajectories in multi-body systems subject to various constraints within the solution space that are feasible for a wide variety of missions.

- The application of using sampling-based kinodynamic planning for the trajectory design process in a wider variety of complex scenarios and multi-body systems may be an avenue of future exploration. The graph construction process presented in this work may be adapted to a larger portion of the solution space and does not have any necessary parameters that depend on the selected CR3BP. Therefore, one suggestion would be to construct a graph for other multi-body systems and construct trajectories in regions of the solution space that are not as well studied as the Earth-Moon system to determine if the research presented here is system independent and capable of constructing feasible trajectories.

After further investigation of the above recommendations for future research, the trajectory design process using sampling-based kinodynamic planning could be an efficient and autonomous method for constructing complex trajectories in various solution spaces. These suggested improvements, along with the work presented in this dissertation, could also expand the applicability of using motion planning techniques for astrodynamics problems, such as constructing trajectory correction maneuvers during an onboard search process from a preloaded graph or designing a set of collision-free orbits for satellites in cislunar space.

Bibliography

[1] Introduction Descriptions of Experimental Investigations and Instruments for the ISEE Spacecraft. IEEE Transactions on Geoscience Electronics, 16(3):151–153, 1978. DOI: https://doi.org/10.1109/TGE.1978.294535.

[2] E. Belbruno. Lunar Capture Orbits, a Method of Constructing Earth Moon Trajectories and the Lunar GAS Mission. 19th International Electric Propulsion Conference, 1987. DOI: https://doi.org/10.2514/6.1987-1054.

[3] E. Belbruno and J. Carrico. Calculation of Weak Stability Boundary Ballistic Lunar Transfer Trajectories. DOI: https://doi.org/10.2514/6.2000-4142.

[4] S. Bique, J. Hershberger, V. Polishchuk, B. Speckii, S. Suri, T. Talvitie, K. Verbeek, and H. Yıldız. Geometric k Shortest Paths. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 2015:1616–1625, 10 2015. DOI: https://doi.org/10.1137/1.9781611973730.107.

[5] B. Blumenthal and R. Sood. Application of Local Lyapunov Exponents for NASA's Artemis-1 Trajectory Design and Maneuver Planning. Austin, TX, January 2023. AAS/AIAA Space Flight Mechanics Meeting.

[6] N. Bosanac. Leveraging Natural Dynamical Structures to Explore Multi-Body Systems. PhD thesis, Purdue University, IN, 2016.

[7] A. Bowyer. Computing Dirichlet Tessellations*. The Computer Journal, 24(2):162–166, 01 1981. DOI: https://doi.org/10.1093/comjnl/24.2.162.

[8] J. V. Breakwell and J. V. Brown. The 'Halo' Family of 3-Dimensional Periodic Orbits in the Earth-Moon Restricted 3-Body Problem. Celestial Mechanics, 20:389–404, 1979. DOI: https://doi.org/10.1007/BF01230405.

[9] K. L. Bruchko and N. Bosanac. Designing Spatial Transfers in Multi-Body Systems Using Roadmap Generation. Charlotte, NC, August 2022. AAS/AIAA Astrodynamics Specialist Conference.

[10] K. L. Bruchko and N. Bosanac. Adaptive Roadmap Generation for Trajectory Design in the Earth-Moon System. Austin, TX, January 2023. AAS/AIAA Space Flight Mechanics Meeting.

[11] J. Burt and B. Smith. Deep Space Climate Observatory: The DSCOVR Mission. In 2012 IEEE Aerospace Conference, pages 1–13. IEEE, 2012.

[12] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming. Mathematical Programming, 89(1):149–185, 2000. DOI: https://doi.org/10.1007/PL00011391.

[13] R. H. Byrd, M. E. Hribar, and J. Nocedal. An Interior Point Algorithm for Large-Scale Nonlinear Programming. SIAM Journal on Optimization, 9(4):877–900, 1999. DOI: https://doi.org/10.1137/S1052623497325107.

[14] B. A. Conway. Spacecraft Trajectory Optimization. Cambridge University Press, New York, USA, 2010. DOI: https://doi.org/10.1017/CBO9780511778025.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. MIT Press and McGraw-Hill, 2 edition, 2001.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms, Third Edition. The MIT Press, 3rd edition, 2009.

[17] A. Das-Stuart, K. C. Howell, and D. Folta. A Rapid Trajectory Design Strategy for Complex Environments Leveraging Attainable Regions and Low-Thrust Capabilities. Adelaide, Australia, September 2017. 68th International Astronautical Congress.

[18] T. Deka and J. Mcmahon. Efficient Astrodynamics-Informed Kinodynamic Motion Planning for Safe Relative Spacecraft Motion. Charlotte, NC, August 2022. AAS/AIAA Astrodynamics Specialist Conference.

[19] R. Diestel. Graph Theory, volume 173. Springer-Verlag, Heidelberg, 5 edition, 2016.

[20] A. Dobson and K. E. Bekris. Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning. The International Journal of Robotics Research, 33(1):18–47, 2014. DOI: https://doi.org/10.1177/0278364913498292.

[21] V. Domingo, B. Fleck, and A. I. Poland. The SOHO Mission: An Overview. Solar Physics, 162:1–37, 1995.

[22] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic Motion Planning. Journal of the ACM (JACM), 40(5):1048–1066, 1993.

[23] T. Ely. Stable Constellations of Frozen Elliptical Inclined Lunar Orbits. Journal of the Astronautical Sciences, 53:301–316, 07 2005. DOI: https://doi.org/10.1007/BF03546355.

[24] R. W. Farquhar. The Control and Use of Libration-Point Satellites. PhD thesis, 1969.

[25] D. C. Folta, N. Bosanac, D. Guzzetti, and K. C. Howell. An Earth-Moon System Trajectory Design Reference Catalog. Acta Astronautica, 110:341–353, 2015. DOI: https://doi.org/10.1016/j.actaastro.2014.07.037.

[26] S. Fuller, E. Lehnhardt, C. Zaid, and K. Halloran. Gateway Program Status and Overview. Journal of Space Safety Engineering, 9(4):625–628, 2022. DOI: https://doi.org/10.1016/j.jsse.2022.07.008.

[27] D. J. Grebow. Generating Periodic Orbits in the Circular Restricted Three-Body Problem with Applications to Lunar South Pole Coverage. Master's thesis, Purdue University, West Lafayette, IN, 2006.

[28] D. J. Grebow and T. A. Pavlak. MCOLL: MONTE Collocation Trajectory Design Tool. Stevenson, WA, August 2017. AAS/AIAA Astrodynamics Specialist Conference.

[29] M. Greenhouse. The James Webb Space Telescope: Mission Overview and Status. In 2019 IEEE Aerospace Conference, pages 1–13, 2019. DOI: https://doi.org/10.1109/AERO.2019.8742209.

[30] J. J. Guzman, D. S. Cooley, K. C. Howell, , and D. C. Folta. Trajectory Design Stragegies that Incorporate Invariant Manifolds and Swingby. AAS/GSFC 13th International Symposium on Space Flight Dynamics, 1998.

[31] A. F. Haapala. Trajectory Design in the Spatial Circular Restricted Three-Body Problem Exploiting Higher-Dimensional Poincaré Maps. PhD dissertation, Purdue University, West Lafayette, IN, 2014.

[32] A. L. Herman. Improved Collocation Methods with Application to Direct Trajectory Optimization. PhD dissertation, University of Illinois, 1995.

[33] A. L. Herman and B. A. Conway. Direct Optimization Using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules. Journal of Guidance, Control, and Dynamics, 19(3):592–599, 1996. DOI: https://doi.org/10.2514/3.21662.

[34] K. C. Howell, B. T. Barden, and M. W. Lo. Application of Dynamical Systems Theory to Trajectory Design for a Libration Point Mission. The Journal of the Astronautical Sciences, 45:161–178, 1997. DOI: https://doi.org/10.1007/BF03546374.

[35] Y. Huang and K. Gupta. A Delaunay Triangulation Based Node Connection Strategy for Probabilistic Roadmap Planners. New Orleans, LA, April 2004. IEEE International Conference on Robotics and Automation.

[36] G. E. Jan, C. Sun, W. C. Tsai, and T. Lin. An O(nlogn) Shortest Path Algorithm Based on Delaunay Triangulation. IEEE/ASME Transactions on Mechatronics, 19(2), April 2014. DOI: https://doi.org/10.1109/TMECH.2013.2252076.

[37] M. Kallmann and M. Kapadia. Discrete Search Algorithms, pages 13–29. Springer International Publishing, Cham, 2016. DOI: https://doi.org/10.1007/978-3-031-02588-4_2.

[38] L. E. Kavraki and S. M. LaValle. Motion Planning. Springer Handbook of Robotics. Springer, Berlin, Heidelberg, 2008. DOI: https://doi.org/10.1007/978-3-540-30301-5_6.

[39] B. Keinert, M. Innmann, M. Sänger, and M. Stamminger. Spherical Fibonacci Mapping. ACM Trans. Graph., 34(6), 2015.

[40] H. B. Keller. Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems. Applications of Bifurcation Theory, 1977.

[41] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross. Dynamical Systems, The Three-Body Problem and Space Mission Design. Marsden Books, 3rd edition, 2006.

[42] D. A. Kopriva. Implementing Spectral Methods for Partial Differential Equations. Springer Dordrecht, 2009. DOI: https://doi.org/10.1007/978-90-481-2261-5.

[43] A. Kume, T. Maki, T. Sakamaki, and T. Ura. Path Re-Planning Method for an AUV to Image Rough Terrain by On-Site Quality Evaluation. In 2012 Oceans - Yeosu, pages 1–9, 2012. DOI: https://doi.org/10.1109/OCEANS-Yeosu.2012.6263618.

[44] D. Landau. Efficient Maneuver Placement for Automated Trajectory Design. Journal of Guidance, Control, and Dynamics, 41(7):1531–1541, 2018. DOI: https://doi.org/10.2514/1.G003172.

[45] S. M. LaValle. Planning Algorithms. Cambridge University Press, Cambridge, UK, 1st edition, 2006.

[46] H. Liua, C. Jin, B. Yang, and A. Zhou. Finding Top-k Shortest Paths with Diversity. IEEE Transactions on Knowledge and Data Engineering, 30(3), March 2018. DOI: https://doi.org/10.1109/TKDE.2017.2773492.

[47] K. Lynch, G. Kantor, H. Choset, L. Kavraki, S. A. Hutchinson, W. Burgard, and S. Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge, MA, 1st edition, 2005.

[48] J. D. Marble and K. E. Bekris. Asymptotically Near-Optimal Planning with Probabilistic Roadmap Spanners. IEEE Transactions on Robotics, 29(2), April 2013. DOI: https://doi.org/10.1109/TRO.2012.2234312.

[49] Matlab. Optimization Toolbox Version 9.3 (R2022a). The MathWorks Inc., Natick, Massachusetts.

[50] M. Morales, S. Thomas, and N. M. Amato. Incremental Map Generation. Advanced Robotics, January 2006.

[51] M. Müller. Dynamic Time Warping, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. DOI: https://doi.org/10.1007/978-3-540-74048-3_4.

[52] NASA. General Mission Analysis Tool (GMAT) [Online]. Available: https://software.nasa.gov/software/GSC-17177-1.

[53] NASA JPL The Navigation and Ancillary Information Facility. Spice Toolkit [Online]. Available: https://naif.jpl.nasa.gov/naif/toolkit.html.

[54] M. R. Osborne. On Shooting Methods for Boundary Value Problems. Journal of Mathematical Analysis and Applications, 27(2):417–433, 1969. DOI: https://doi.org/10.1016/0022-247X(69)90059-6.

[55] B. Park and W. K. Chung. Adaptive Node Sampling Method for Probabilistic Roadmap Planners. St. Louis, USA, October 2009. IEEE/RSJ International Conference on Intelligent Robots and Systems.

[56] J. Park and T. Yoon. Maximizing the Coverage of Roadmap Graph for Optimal Motion Planning. Hindawi, 2018, November 2018. DOI: https://doi.org/10.1155/2018/9104720.

[57] N. L. Parrish. A* Pathfinding Continuous-Thrust Trajectory Optimization. In AAS 37th Annual Guidance & Control Conference, Breckenridge, CO, January 2014.

[58] L. Perko. Differential Equations and Dynamical Systems. Springer New York, NY, 3rd edition, 2001. DOI: https://doi.org/10.1007/978-1-4613-0003-8.

[59] H. Poincaré. Les Méthodes Nouvelles de la Mécanique Céleste, volume 1-3. Gauthier-Villars, 1899.

[60] R. Pritchett. Strategies for Low-Thrust Transfer Design Based on Direct Collocation Techniques. PhD dissertation, Purdue University, IN, 2020.

[61] A. H. Qureshi, Z. Tahir, G. Tariq, and Y. Ayaz. Re-Planning Using Delaunay Triangulation for Real Time Motion Planning in Complex Dynamic Environments. In 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pages 905–911, 2018. DOI: https://doi.org/10.1109/AIM.2018.8452685.

[62] H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1):43–49, 1978.

[63] R. Shome and L. E. Kavraki. Asymptotically Optimal Kinodynamic Planning Using Bundles of Edges. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 9988–9994, 2021. DOI: https://doi.org/10.1109/ICRA48506.2021.9560836.

[64] T. R. Smith. Using Motion Primitives to Rapidly Design Trajectories in Multi-Body Systems. PhD dissertation, University of Colorado Boulder, Boulder, CO, 2023.

[65] Z. Sodnik, C. Heese, P. D. Arapoglou, K. J. Schulz, I. Zayer, R. Daddato, and S. Kraft. Deep-Space Optical Communication System (DOCS) for ESA's Space Weather Mission to Lagrange Orbit L5. In 2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS), pages 28–33. IEEE, 2017.

[66] S. Sridharan and R. Balakrishnan. Discrete Mathematics: Graph Algorithms, Algebraic Structures, Coding Theory, and Cryptography. Chapman and Hall/CRC, New York, 1 edition, 2019. DOI: https://doi-org/10.1201/9780429486326.

[67] J. A. Starek, E. Schmerling, G. D. Maher, B. W. Barbee, and M. Pavone. Real-Time, Propellant-Optimized Spacecraft Motion Planning under Clohessy-Whilshire-Hill Dynamics. Big Sky, MT, March 2016. IEEE Aerospace Conference.

[68] P. Svestka and M. H. Overmars. Robot Motion Planning and Control. J.P. Laumond, Springer, Cerlin 1998.

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

[70] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad. A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms. IEEE Access, 9:119310–119342, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3108177.

[71] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 1999.

[72] F. Topputo, D. A. Tos, K. V. Mani, S. Ceccherini, C. Giordano, V. Franzese, and Y. Wang. Trajectory Design in High-Fidelity Models. In 7th International Conference on Astrodynamics Tools and Techniques (ICATT), pages 1–9, 2018.

[73] E. Trumbauer and B. Villac. Heuristic Search-Based Framework for Onboard Trajectory Redesign. Journal of Guidance, Control and Dynamics, 37(1), January-February 2014. DOI: https://doi.org/10.2514/1.61236.

[74] G. A. Tsirogiannis. A Graph Based Methodology for Mission Design. Celestial Mechanics and Dynamical Astronomy, 114:353–363, 2012. DOI: https://doi.org/10.1007/s10569-012-9444-9.

[75] J. Y. Yen. Finding the K Shortest Loopless Paths in a Network. Management Science, 17, July 1971.

[76] C. Zhou, B. Huang, and P. Fränti. A Review of Motion Planning Algorithms for Intelligent Robotics. Journal of Intelligent Manufacturing, 33:387–424, November 2021. DOI: https://doi.org/10.1007/s10845-021-01867-z.