Extended Linear Quadratic Regulator Informed RRT*

Xavier Kipping
The University of Alabama
North Engineering Research Center 0042
251 Shelby Ln, Tuscaloosa, AL 35401
xskipping@crimson.ua.edu

Jordan D. Larson
The University of Alabama
201 7th Ave., Bevill 1103J
Tuscaloosa, AL 35487
jordan.larson@ua.edu

Ali Pakniyat
The University of Alabama
255 7th Ave, SERC 3010
Tuscaloosa, AL 35401
apakniyat@ua.edu

***Abstract*—** Autonomous guidance is critical for future space missions, particularly for cislunar satellite rendezvous operations, where communication delays and complex dynamics present significant challenges. This paper introduces the Extended Linear Quadratic Regulator-informed optimal Rapidly-exploring Random Tree (ELQR-RRT*) search, a novel sampling-based guidance algorithm designed for obstacle avoidance in nonlinear environments. The authors validate ELQR-RRT* by solving a fuel-optimal rendezvous problem in an Earth-Moon 9:2 L2 Southern Near Rectilinear Halo Orbit (NRHO). ELQR-RRT*'s performance is benchmarked against two established methods: a shooter-based RRT* (S-RRT*) and a direct optimization solution from the Astrodynamics Software and Science Enabling Toolkit (ASSET). This comparative analysis demonstrates that ELQR-RRT* generates a smooth, near-optimal trajectory, successfully navigating the complex multi-body dynamics of the cislunar regime while avoiding obstacles. The results affirm the algorithm's capability to deliver high-quality guidance solutions, highlighting its potential for real-time autonomous operations.

2026 IEEE Aerospace Conference | 979-8-3315-7360-7/26/$31.00 ©2026 IEEE | DOI: 10.1109/AERO66936.2026.11519892

TABLE OF CONTENTS

1. INTRODUCTION

Autonomous Rendezvous and Proximity Operations (RPO) are critical to future space mission concepts such as satellite servicing, Mars sample returns, active debris removal, and space station construction [1]. As NASA plans for long-term human exploration of the lunar south pole, a large number of spacecraft will be necessary to support NASA's lunar program, Artemis [2] [3]. The burden on ground operations will likely grow to support an increased number of RPOs and lead to missions competing for ground resources [4].

Spacecraft autonomy is proposed as a solution for scheduling conflicts, labor overhead, and human error associated with ground stations [5]. Spacecraft autonomy is also proposed to replace significant amounts of ground-station based guidance that will likely become prohibitively expensive [5]. State-of-the-art autonomous maneuvering techniques including sliding-mode controllers, model predictive control, and artificial potential functions are valuable in uncluttered settings, but fall short in scenarios that feature neighboring debris, other spacecraft, safety modes, and complex maneuvering [5].

Contribution

This paper introduces the Extended Linear Quadratic Regulator-informed RRT* (ELQR-RRT*), a novel sampling-based algorithm used to generate guidance laws for spacecraft rendezvous in obstacle-rich environments. The contribution of this work is the utilization of an LQR inspired RRT* algorithm for cislunar rendezvous that mitigates nonlinearization errors, common in typical iterative LQR methods, through LQR-smoothing, presented in [6]. Unaccounted nonlinearization errors in the envisioned cislunar environment can threaten mission success due to the sensitivity of the multi-body gravitational environment. This work manages LQR nonlinearity errors without a reference trajectory or additional algorithm, unlike the work seen in [7] and [8]. This study also expands the research provided in [9] by including a new algorithm and comparing the results achieved by the sampling-based algorithms of this work with the direct optimization solution achievable with the cutting-edge ASSET library. To the knowledge of the authors, no other notable LQR-RRT* inspired work exists for cislunar rendezvous and no other implementation of ELQR, from [6], to RRT searches exist.

Mission Scenario

The scenario envisioned for this analysis is a fuel-optimal rendezvous between two spacecraft in an L2 9:2 Southern Near Rectilinear Halo Orbit (NRHO). NASA's Gateway is anticipated to orbit the Moon in an L2 9:2 Southern NRHO and act as a staging point for both lunar and deep space exploration [10]. Spacecraft deployed from Gateway with small perturbations remain in the NRHO for several revolutions before departing [11]. Therefore, service missions to Gateway may require avoidance of nearby spacecraft following a similar NRHO. Overall, this scenario provides a trying environment to validate ELQR-RRT*. The performance of ELQR-RRT* is contrasted against a Shooter-RRT* (S-RRT*) process and a direct optimization solution from the Astrodynamics Software and Science Enabling Toolkit (ASSET) [12]. S-RRT* is a less complex sampling-based algorithm and can serve as a good baseline for the more involved ELQR-RRT* search. ASSET is used to demonstrate the solution obtained by a state-of-the-art direct optimization method to validate the results of the sampling-based methods. Results demonstrate that ELQR-RRT* produces a smooth, near-optimal trajectory, validating the proposed method and highlighting the distinct advantages of each guidance approach for this complex problem.

2. LITERATURE REVIEW

Significant research has investigated RPOs, as most mission architectures for exploration involve rendezvous and capture procedures [13]. This section provides rationale for the development of ELQR-RRT* and outlines other potential RPO guidance and control algorithms. This review is by no means comprehensive and relies on other rigorous literature reviews for support.

Guidance

Guidance, as defined by Isser, is the path from a vehicle's current state to a designated target state, as well as desired changes in its velocity, rotation, and acceleration for following that path [14]. Similarly, a guidance law can be defined as an algorithm that determines the commanded vehicle's required acceleration and rotation [15]. In this paper, path-planning algorithms work to solve optimal control problems and generate guidance solutions. For this reason, path-planning algorithms are analyzed and referred to as guidance algorithms within this work.

Path-Planning

Path-planning algorithms can be divided into two classes, sampling-based algorithms and exact algorithms. Sampling-based algorithms, such as Rapidly-exploring Random Tree (RRT) searches, randomly sample the solution space and use an external collision-detection routine to ensure a safe path [5]. External collision detection processes can be naturally replaced with sensor-based obstacle detection. If a collision-detection process is used, sampling-based algorithms ensure no collisions occur. In this work, moving obstacles are used, thus expanding the solution space to include the time domain for each obstacle. This ultimately means that obstacles must match the time and location of the vehicle to be considered unsafe.

Exact algorithms, such as artificial potential function algorithms, require an explicit representation of unsafe regions for collision avoidance [5]. One common approach to collision avoidance used by exact algorithms is the repulser model, as seen in [6]. The repulser model adds the negative exponential of the closest distance between the agent and the set of obstacles to the cost function as a means to push the trajectory away from the obstacles. Simple repulser models, such as in [6], do not guarantee safety without meeting safety definitions, such as those imposed by control barrier functions [16]. Implementing repulser models as artificial potential functions may provide safe trajectories, but can also push the solution into local minima [17]. Exact algorithms, such as the ASSET solution provided in this work, also require a complete re-optimization for stochastic changes to obstacles increasing the computational burden.

Rapidly-exploring Random Tree Searches

The Rapidly-exploring Random Tree (RRT) search algorithm is a method that iteratively samples a solution space to create nodes, creates paths between new and existing nodes, and adds the new node relationships after verifying the path is collision free. This work utilizes RRT*, a version of RRT that achieves *asymptotic optimality* [18, Definition 24]. The RRT* algorithm returns optimal solutions by *rewiring* the tree with each new sample. Existing nodes are rewired by comparing current paths with paths to a newly sampled node. If replacing any existing connections decreases the cost of the path, the node is replaced with the new node, permitted the path is safe. RRT algorithms can be computationally advantageous and can allow for sampling interruptions in real-time operations [5] [8]. RRT* algorithms can also return a sub-optimal trajectory if the computation time is constrained by the time-to-collision.

This work does not account for the uncertainty of obstacles; however, if uncertain obstacles are present in the environment, methods such as dynamic re-planning can be used to alter the vehicle's path using the existing tree rather than requiring re-optimization of the entire problem [19]. RRT-based algorithms efficiently explore the state space when the distance heuristic reflects the true cost-to-go for the system [20]. RRTs can be used where the path is constrained by linear or nonlinear dynamics by changing the mechanism for connecting the nodes. When operating in nonlinear dynamics, RRTs do not need initial guesses and are *probabilistically complete*, provided enough samples are taken [18, Theorem 16 & 23].

Optimal Control Algorithms

The optimal control problem is defined as finding a control and resulting state trajectory such that the cost function is minimized [21]. Optimal control problems with linear dynamics and quadratic costs can be solved as part of the family of Linear Quadratic Regulator (LQR) problems. The LQR is one of the most used control design methods in aerospace and has excellent performance, robustness, and capability to minimize control usage [22]. One advantage given by LQR methods is a guaranteed globally optimal solution, given linear dynamics and a quadratic cost function [21, Ch. 2.7 Theorem 1]. Although the guarantee is for linear systems, the path is anticipated to approach the true global optimal between nodes as the number of linearization points increases. Depending on the formulation, LQR problems can also have an analytic solution. Finally, LQR methods can easily be extended to produce command tracking controllers [22].

There are numerous alternatives to LQR based optimal control solutions including sliding mode, phase-plane, and direct optimization methods. Sliding mode and phase plane algorithms are not considered due to their lack of optimality [23]. Direct optimization methods discretize the optimal control problem into a finite-dimensional parameter optimization method, which is usually easier to solve than the original optimal control problem [24]. ASSET utilizes compressed Legendre-Gauss-Lobatto (LGL) collocation methods along with vectorization and a custom Parallel Sparse Interior-point Optimizer (PSIOPT) to solve optimal control problems very quickly [12]. Direct optimizers, like PSIOPT, guarantee a locally optimal solution for both nonlinear and linear problems.

Extended Linear Quadratic Regulator

The Extended Linear Quadratic Regulator (ELQR) algorithm was developed to improve the iterative LQR process (iLQR) by removing the need for a line search and introducing LQR-smoothing [6]. iLQR is identical to Differential Dynamic Programming (DDP), except iLQR neglects the second-order Taylor series term, thereby losing the quadratic convergence properties that DDP has [25]. The line search method used by iLQR ensures the new solution stays near the nominal trajectory.

A recent comparison study of ELQR and iLQR to random finite set-based swarms has shown that ELQR and iLQR have similar performance, but ELQR requires much less time to tune [26]. Unlike iLQR, ELQR converges about points only

feasible upon a convergent solution, removing the need for a line search [6]. Unlike LQR-RRT*, iLQR, or DDP, ELQR does not select linearization points arbitrarily, but instead finds the points with a minimum average of the cost-to-go and the cost-to-come values through forward and backward LQR [6].

*Linear Quadratic Regulator Informed RRT**

The Linear Quadratic Regulator informed RRT* (LQR-RRT*) algorithm was created to reduce the design effort required in sampling-based planners by using automatically derived heuristics for the Extend function, presented in the Methodology section [20]. The LQR-RRT* search uses the LQR to generate optimal paths to nodes and the RRT* search to sample the solution space and manage relationships. LQR-RRT* can generate optimal plans in domains with complex or underactuated dynamics without requiring domain-specific design choices [20]. Perez et al. utilized an infinite horizon LQR for the *extension method* to generate an optimal RPO solution [20].

LQR-RRT* in a nonlinear environment depends on locally linearized dynamics on selected operation points [20]. This means the locally linearized dynamics create inaccurate *distance* or cost metrics, forcing optimal solutions to be obtained only after the state space is more densely populated [20], which would require an additional linearization error mitigation process. LQR-RRT* is therefore not considered for the NRHO RPO problem because a poorly linearized solution could be returned, which is unusable due to the sensitivities of the nonlinear system. Although LQR-RRT* is not used in this work, ELQR-RRT* mirrors the implementation of LQR-RRT* for trajectory generation and tree management.

3. METHODOLOGY

The ELQR-RRT* algorithm uses an ELQR for path and control generation and an RRT* search for node sampling, node management, and collision avoidance. The ELQR-RRT* solution is compared against S-RRT* and ASSET's native compressed-LGL collocation and PSIOPT optimizer [12]. This section provides the necessary algorithms and processes for ELQR-RRT*, S-RRT*, and ASSET's solution.

Optimal Rapidly-exploring Random Tree Search Algorithm

This section outlines the steps for the optimal Rapidly-exploring Random Tree search (RRT*) algorithm and shows implementation-specific design choices. The RRT* algorithm used in this work is identical to that presented in [9]; however, this work includes the algorithms and functions for completeness. The RRT* algorithm differs from the prototypical RRT*, given by Karaman [18], by attempting to connect each new node to the end node, sampling the entire solution space, using an external collision bisection method to check for safe paths, and decreasing the number of iterations by ensuring the sampled node is reached [9]. Algorithm 1 presents the complete RRT* framework,

Algorithm 1: The RRT* Algorithm

1 $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset; i \leftarrow 0;$
2 **while** $i < N$ **do**
3 $\quad G \leftarrow (V, E);$
4 $\quad x_{rand} \leftarrow$ Sample(i);
5 $\quad i \leftarrow i + 1;$
6 $\quad (V, E) \leftarrow$Extend(G, x_{rand});
7 **end do**

where V is a list of the vertices in a graph, x_{init} is the initial state, E is a list of the edges of the graph, i is the iteration count, N is the user-defined number of iterations, and G is a graph. The iteration number is varied in this work, but could be based on a desired sample density or by a constraint on computation time. Algorithm 1 shows that the RRT* framework begins with an empty list of edges and only the initial state in the set of vertices. RRT* then iterates a set number of times, picking random states, x_{rand}, and attempting to connect with them. The trajectory is copied into the edge list, E, and the states are written into the vertex list, where the parent-child relation is saved to preserve the tree structure. Algorithm 1 also depends on the sample function given here.

Sample: The function Sample typically returns independent and identically-distributed samples from the collision-free space [27]. Samples are returned from the entire solution space, since the collision detection routine removes unsafe trajectories and this work investigates moving obstacles.

Algorithm 1 depends on the Extend procedure, which includes path generation, collision detection, connecting to the end, and rewiring. The multi-stepped Extend function is outlined in the pseudo-code provided in Algorithm 2, where x_i, z_i, u_i, and τ_i is the i^{th} state, trajectory, control list, and Time of Flight (TOF), respectively.

Algorithm 2: The Extend Procedure

1 $V' \leftarrow V; E' \leftarrow E;$
2 $(x_{nearest}, x_{new}, z_{new}, u_{new}, T_{new}) \leftarrow$ Nearest(G, x);
3 $V' \leftarrow V' \cup \{x_{new}\};$
4 $E' \leftarrow E' \cup \{(x_{nearest}, x_{new})\};$
5 $(z_N, u_N, T_N) \leftarrow$Steer$(x_{new}, x_{end})$
6 **if** ObstacleFree(z_N) **then**
7 $\quad V' \leftarrow V' \cup \{x_{end}\}$
8 $\quad E' \leftarrow E' \cup \{(x_{new}, x_{end})\}$
9 $V_{nearby} \leftarrow$ CompletePath$(G, x_{new}, |V|)$;
10 **for** all $x_{near} \in V_{nearby}$ **do**
11 $\quad (z_{near}, u_{near}, T_{near}) \leftarrow$ Steer(x_{new}, x_{near});
12 $\quad$ **if** $z_{near}(T_{near}) = x_{near}$ *and* ObstacleFree(z_{near}) *and* Cost(x_{near}) > Cost(x_{new}) + $J(z_{near})$ **then**
13 $\quad\quad x_{parent} \leftarrow$ Parent(x_{near});
14 $\quad\quad E \leftarrow$ E' $\setminus \{(x_{parent}, x_{near})\}$;
15 $\quad\quad E \leftarrow$ E' $\cup \{(x_{new}, x_{near})\}$;
16 $\quad$ **end if**
17 **end do**
18 **return** $G' = (V', E')$

Algorithm 2 uses $J()$ as the cost function and depends on the following functions:

CompletePath: The CompletePath function replaces the Near-by Vertices process found in [27]. Given a graph $G = (V, E)$, the CompletePath function returns all vertices in V for each trajectory that successfully completed the Steer to the end node, seen in line 5 of Algorithm 2, thereby ending at the terminal target state. The purpose of the CompletePath function is merely to provide the stored vertices, trajectories, and cost in preparation for the *rewire* step implemented in lines 10 to 15 of Algorithm 2. This function ensures that the computationally expensive *rewire* step is used on a select number of vertices to increase the effectiveness of the *rewire* and decrease the computational burden.

Algorithm 2 attempts to steer each node to the terminal target state after it has been added to the tree. The optimal path is

assumed to coast to the target the majority of the time with minor adjustments to avoid obstacles. Assuming only minor adjustments justifies the authors' efforts to connect each newly sampled node to the end node to increase the feasible solution space and provide a safe trajectory as quickly as possible.

Steer: Given two states, x_{first} and x_{second}, the Steer function returns the path starting at x_{first} and ending at x_{second}. Therefore, the steer function returns the trajectory, the inputs required to steer the system to the final state, and the time required for the maneuver.

ObstacleFree: The ObstacleFree procedure returns true if and only if the trajectory provided to it lies entirely in the obstacle-free space [27]. This function is also known as the collision-checking procedure. To ensure that the RRT* returns a collision-free path, each node undergoes an independent collision test. The collision test assumes complete knowledge of the obstacles' states. The test then simulates the vehicle's and obstacles' trajectories and iteratively solves for the point where the vehicle intercepts the obstacles' keep-out sphere.

The collisions are computed using a bisection search and cubic spline interpolation native to the University of Alabama's Astrodynamics Software and Science Enabling Toolkit (ASSET) [12]. The cubic spline allows for interpolation of any point in the trajectory and the bisection search looks for where the spacecraft violates the keep-out radius constraint. If the search finds no keep-out violations, then the path is considered safe.

Cost: The Cost function returns the cost of the trajectory from the root vertex to a given vertex [27].

Parent: Given a state, x, the Parent procedure returns the parent node or closest node to the state in the tree.

The final function definition for the algorithm is that of the Nearest procedure. As Algorithm 2 and Algorithm 3 show, the Nearest function is called to find the TOF, trajectory, and control effort necessary to travel from the newly sampled state to the *nearest* node of the tree. The *nearest* node is defined as the node that can be reached with the least cost as seen in Algorithm 3, where c_i is the i^{th} cost.

Algorithm 3: The Nearest Procedure

```
1  c_min ← ∞;
2  for all x_i ∈ G do
3      (z_i, u_i, τ_i) ← Steer(x_i, x);
4      x_new,i ← z_i(τ_i);
5      c_i ← Cost(x_i);
6      if ObstacleFree(z_i) and c_i < c_min and x_new,i = x
       then
7          x_nearest ← x_i;
8          x_new ← x_new,i;
9          c_new ← c_i;
10         z_new ← z_i;
11         u_new ← u_i;
12         T_new ← τ_i;
13 return N = (x_nearest, x_new, z_new, u_new, T_new)
```

Algorithm 3 does not require iterating through nearby points after the initial steer, unlike the RRT* in [27]. One less iteration can be made by ensuring that the final point of the trajectory, $x_{new,i}$, is equal to the sampled point, x, as seen in Algorithm 3, line 6. If a growth inhibiting heuristic is added to the Nearest procedure, it would change the input to the steer function, x, instead of adding another collision check proposed in [27].

Circular Restricted Three Body Problem

Although the ELQR-RRT* algorithm alone is novel, this work aims to apply the use of ELQR-RRT* to cislunar RPOs. To simulate the cislunar environment with reasonable accuracy, the Circular Restricted Three-Body Problem (CR3BP) is used as the main framework for the dynamics. The Earth-Moon CR3BP accounts for the gravitational influences of the Moon and can be used as a stepping stone for cislunar mission planning with an ephemeris model [28]. Directly utilizing the ephemeris model is avoided in this work for simplicity and because careful consideration is required for choosing the mission epoch in an ephemeris model. Future work can include an extension of ELQR-RRT* to ephemeris models or other complex nonlinear systems.

The three-body problem is formed by assuming there are three bodies acted upon solely by gravity and represented as point-masses. This problem becomes the Restricted Three Body Problem (R3BP) by assuming that the third body, the body of interest, has a negligible gravitational effect on the other two bodies. By further assuming the primaries are in circular orbits about the barycenter, where the reference frame is set, and allowing the frame to rotate with the second body, the R3BP becomes the CR3BP. The CR3BP regime can be seen in Figure 1, where m_i is the mass of the i^{th} primary.

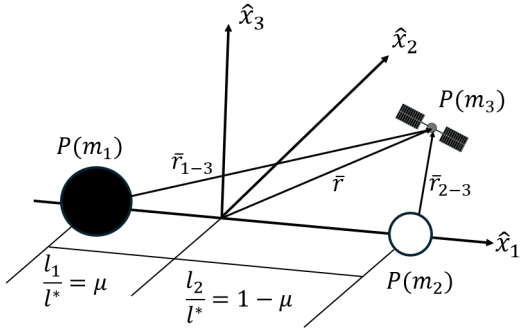


Figure 1. CR3BP reference frame.

The CR3BP equations of motion are nondimensionalized to benefit the computational accuracy using the characteristic length (l^*), mass (m^*), and time (t^*), given in (1) - (3).

$$m^* = m_1 + m_2 \tag{1}$$

$$l^* = l_1 + l_2 \tag{2}$$

$$t^* = \sqrt{\frac{l^{*^3}}{Gm^*}} \tag{3}$$

The term μ is the nondimensionalized mass of the second primary given by (4).

$$\mu = \frac{m_2}{m^*} \tag{4}$$

The variables from Figure 1 are assigned to the state, x, in (5)

to demonstrate the nonlinearity of the system

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{5}$$

where x refers to the state of the third body, the vehicle of interest.

The resulting derivative of the state, $\dot{x}$, is given in (6) - (8)

$$\dot{x} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_1 + 2x_5 - \frac{(1-\mu)(x_1+\mu)}{C_1(x)} - \frac{\mu(x_1-1+\mu)}{C_2(x)} + u_1 \\ x_2 - 2x_4 - \frac{(1-\mu)x_2}{C_1(x)} - \frac{\mu x_2}{C_2(x)} + u_2 \\ -\frac{(1-\mu)x_3}{C_1(x)} - \frac{\mu x_3}{C_2(x)} + u_3 \end{bmatrix} \tag{6}$$

$$C_1(x) = ((x_1+\mu)^2 + x_2^2 + x_3^3)^{3/2} \tag{7}$$

$$C_2(x) = ((x_1+\mu-1)^2 + x_2^2 + x_3^3)^{3/2} \tag{8}$$

where $\dot{x} = f(x, u)$. It is important to note that the equation of motion models continuous thrust, whereas both S-RRT* and ELQR-RRT* will assume impulsive ΔV control, as seen in the appendix.

The integration of the CR3BP equations of motion and the determination of the state transition matrix, defined in the appendix, is completed in ASSET to utilize the computational speed of its vector function modeling language [12].

Near Rectilinear Halo Orbit

Key to NASA's exploration of the Moon's South Pole is the development of Gateway. As provided by [29] the intended orbit for Gateway is a southern L_2 Near Rectilinear Halo Orbit (NRHO), which offers constant communication with Earth, and extended coverage over the Moon's south pole.

The particular NRHO of interest for the Gateway mission is the 9:2 L2 Southern NRHO, achieving 9 orbits for every two synodic periods [29]. The 9:2 NRHO used in this work has a period of approximately 6.562 days and is found with a single-shooter method using the NRHO's symmetry given in [9], a typical solution for periodic orbit determination in the CR3BP [28] [30] [31] . The operation time and location are bounded by the mean anomaly values of $80°$ and $280°$ for the NRHO, to prevent dangerous lunar flybys and to utilize the central manifolds present in this region [32]. Although this work implements the mean anomaly bounds presented by Franzini [32], a large lunar obstacle can be included to prevent close lunar flybys and remove the mean anomaly bounds. However, it is important to note that an increase in the size of the solution space requires more samples and computational resources to provide the same sample density.

Extended Linear Quadratic Regulator

The Extended Linear Quadratic Regulator (ELQR) is derived from the iLQR, which uses a second order approximation of the Bellman equation to find the optimal control sequence. Unlike iLQR, ELQR performs backward and forward recursions with additional terms to account for linearization errors and to form LQR variants that depend on the cost function. This work implements the same foundational ELQR algorithm proposed in [6] with simplifications relative to the application provided in this section.

Native ELQR uses discrete time dynamics where the future state can be found using (9)

$$x_{t+1} = g_t(x_t, u_t) \tag{9}$$

where $x_t \in \mathbb{X}$, $u_t \in \mathbb{U}$, $g_t \in \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ for the state space $\mathbb{X}$ and the input space $\mathbb{U}$ [6]. The cost function to be minimized, J, is given in (10)

$$J = c_0(x_0) + c_L(x_L) + \sum_{t=1}^{L-1} c_t(x_t, u_t) \tag{10}$$

where $c_L(x_L) \in \mathbb{X} \to \mathbb{R}$ is the endpoint cost and $c_t(x_t, u_t) \in \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ is the local cost function. ELQR uses a backward and forward pass to compute the control policy, $\pi_t \in \mathbb{X} \to \mathbb{U}$, and minimize the cost for all time steps [6].

For the application of spacecraft rendezvous in the CR3BP, the cost function needs to penalize terminal states that are not in the position of the nonzero state of the target spacecraft, x_{targ} as seen in (11)

$$c_L(x_L) = \frac{1}{2}(x_L - x_{targ})^T Q_L (x_L - x_{targ}) \tag{11}$$

where n is the dimension of the state variable and $Q_L \in \mathbb{R}^{n \times n}$. Likewise, the cost function needs to penalize trajectories that do not start at the initial state of the chaser spacecraft, x_{chas}, as seen in (12)

$$c_0(x_0) = \frac{1}{2}(x_0 - x_{chas})^T Q_0 (x_0 - x_{chas}) \tag{12}$$

where $Q_0 \in \mathbb{R}^{n \times n}$. The quadratic running cost, c_t, is based solely on the control effort for the ELQR that feeds the RRT* search as given by (13)

$$c_t(x_t, u_t) = \frac{1}{2} u_t^T R u_t \tag{13}$$

where m is the dimension of the input and $R \in \mathbb{R}^{m \times m}$. Equation (13) shows the L_2 norm of the input for the cost functional, which is the quadratization of the typical fuel-optimal L_1 norm. The total cost accrued from the initial state to the final state can be found as the sum of the cost-to-come and cost-to-go functions as seen in (14)

$$\begin{aligned} \hat{s}(x_{t,k}) &= s_{t,k}(x_{t,k}) + \zeta_{t,k}(x_{t,k}) \\ &= \frac{1}{2} x_{t,k}^T (S_{t,k} + Z_{t,k}) x_{t,k} + x_{t,k}^T (s_{t,k} + \zeta_{t,k}) \end{aligned} \tag{14}$$

where $s_{t,k}$, defined in (15), and $\zeta_{t,k}$, defined in (35), are functions of $x_{t,k}$ and represent the scalar cost-to-go and cost-to-come, respectively. The other variables (14) uses are: the cost-to-go matrix, $S_t \in \mathbb{R}^{n \times n} > 0$, defined in (23), the cost-to-come matrix, $Z_t \in \mathbb{R}^{n \times n} \geq 0$, defined in (43), and k is the iteration index for ELQR. The cost function is minimized while accounting for the nonlinear dynamics by using the forward and backward passes of the ELQR. The backward and forward passes alternate starting with the backward pass.

Backward Pass—The backward pass uses the cost-to-go function with the forward pass' cost-to-come function to find the optimal states for linearization [6]. The cost-to-go function can be found in (15)

$$s_{t,k}(x_{t,k}) = \frac{1}{2}x_{t,k}^T S_{t,k} x_{t,k} + x_{t,k}^T s_{t,k} \tag{15}$$

where $S_{t,k}$ and $s_{t,k}$ are found through backward recursion [6]. The backward pass starts with cost-to-go variables at the terminal stage, $t = L$ defined in (16)

$$S_L = Q_L, \qquad s_L = q_L = \frac{\partial c_L}{\partial x_L}(\hat{x}_L) - Q_L \hat{x}_L \tag{16}$$

where $\hat{x}_L$ is the terminal linearization/quadratization point from the forward pass, or the final linearization point dictated by the designer [6]. The only variable in (16) dependent on the iteration number, k, for this work is $\hat{x}_L$, since Q_L and q_L are constants. The recursion for $S_{t,k}$ and $s_{t,k}$ values starts at $t = L - 1$ and continues until $t = 0$. Before the cost-to-go at stage t can be found, the best estimate of the state at stage $t + 1$ is calculated in (17).

$$\hat{x}_{t+1,k} = -(S_{t+1,k} + Z_{t+1,k-1})^{-1}(s_{t+1,k} + \zeta_{t+1,k-1}) \tag{17}$$

The state and input at stage t can be found using the inverse dynamics, $g_t^{-1}(x_{t+1}, u_t)$, and the inverse control policy, π^{-1}, as seen in (18) and (19).

$$\hat{u}_{t,k} = \pi_{t,k-1}^{-1}(\hat{x}_{t+1,k}) \tag{18}$$

$$\hat{x}_t, k = g_t^{-1}(\hat{x}_{t+1,k}, \hat{u}_{t,k}) \tag{19}$$

Equation (19) can be simplified for spacecraft rendezvous by integrating the future state backwards to a previous state and then subtracting the ΔV vector from the previous state.

The variables for the linearized forward dynamics can then be solved for using (20) - (22), where κ_t is a linearization error tracking term and $\Phi_{t,k}$ and $B_{t,k}$ are the state transition and discrete input matrices, respectively, at time t for the k^{th} iteration.

$$\Phi_{t,k} = \frac{\partial g_t}{\partial x_t}(\hat{x}_{t,k}, \hat{u}_{t,k}) \tag{20}$$

$$B_{t,k} = \frac{\partial g_t}{\partial u_t}(\hat{x}_{t,k}, \hat{u}_{t,k}) \tag{21}$$

$$\kappa_{t,k} = \hat{x}_{t+1,k} - \Phi_{t,k}\hat{x}_{t,k} - B_{t,k}\hat{u}_{t,k} \tag{22}$$

The algebraic equations for the cost-to-go variables are presented in (23) - (34)

$$S_{t,k} = D_{t,k} - C_{t,k}^T E_{t,k}^{-1} C_{t,k} \tag{23}$$

$$s_{t,k} = d_{t,k} - C_{t,k}^T E_{t,k}^{-1} e_{t,k} \tag{24}$$

$$C_{t,k} = P_{t,k} + B_{t,k} S_{t+1,k} \Phi_{t,k} \tag{25}$$

$$D_{t,k} = Q_{t,k} + \Phi_{t,k}^T S_{t+1,k} \Phi_{t,k} \tag{26}$$

$$E_{t,k} = R_{t,k} + B_{t,k}^T S_{t+1,k} B_{t,k} \tag{27}$$

$$d_{t,k} = q_{t,k} + \Phi_{t,k}^T s_{t+1,k} + \Phi_{t,k}^T S_{t+1,k} \kappa_{t,k} \tag{28}$$

$$e_{t,k} = r_{t,k} + B_{t,k}^T s_{t+1,k} + B_{t,k}^T S_{t+1,k} \kappa_{t,k} \tag{29}$$

$$\pi_{t,k}(x_{t,k}) = L_{t,k} x_{t,k} + l_{t,k} \tag{30}$$

$$L_{t,k} = -E_{t,k}^{-1} C_{t,k} \tag{31}$$

$$l_{t,k} = -E_{t,k}^{-1} e_{t,k} \tag{32}$$

$$q_{t,k} = \frac{\partial c_t}{\partial x_t}(\hat{x}_{t,k}, \hat{u}_{t,k}) - Q_{t,k}\hat{x}_{t,k} - P_{t,k}^T \hat{u}_{t,k} \tag{33}$$

$$r_{t,k} = \frac{\partial c_t}{\partial u_t}(\hat{x}_{t,k}, \hat{u}_{t,k}) - P_{t,k}\hat{x}_{t,k} - R_{t,k}\hat{u}_{t,k} \tag{34}$$

where $\pi_{t,k}$ is the control policy. This implementation sets P_t, the cross-cost matrix, equal to the zero matrix and $r_t = q_t = 0$. Careful consideration of this work's cost function also reveals $Q_{t,k}$ to be the zero matrix at all points but the initial and final points and $R_{t,k}$ to be the constant R for all intermediate states.

Forward Pass—The cost-to-come function is used to calculate the past cost accrued between stage 0 and stage t and is given in (35) [6].

$$\zeta_{t,k}(x_{t,k}) = \frac{1}{2}x_{t,k}^T Z_{t,k} x_{t,k} + x_{t,k}^T \zeta_{t,k} \tag{35}$$

Unlike the cost-to-go function, the cost-to-come function is highly dependent on the linearized inverse dynamics, given in (36)

$$x_t = \Omega_t x_{t+1} + \Upsilon_t u_t + \sigma_t \tag{36}$$

where $\Omega_t = \Phi_t^{-1}$, $\Upsilon_t = -\Phi_t^{-1} B_t$, and $\sigma = -\Phi^{-1}\kappa_t$. Like the *Backward Pass*, the *Forward Pass* computes an estimate of the state at the next stage and then linearizes about that point. To compute the state at stage $t + 1$, first the state at stage t is found using (37).

$$\hat{x}_{t,k} = -(S_{t,k} + Z_{t,k})^{-1}(s_{t,k} + \zeta_{t,k}) \tag{37}$$

The state at stage $t+1$ and input at stage t can be found using (38) and (39)

$$\hat{u}_{t,k} = \pi_{t,k}(\hat{x}_{t,k}) \tag{38}$$

where π_t is the control policy. It is important to note that $\hat{u}_{t,k}$ and $\hat{x}_{t,k}$ are new values defined from (37) and (38) and not values from the backward pass.

$$\hat{x}_{t+1,k} = g_t(\hat{x}_{t,k}, \hat{u}_{t,k}) \tag{39}$$

For the simplified satellite dynamics, (39) is the integration of the state $x_{t,k}$ summed with the ΔV vector. The inverse dynamics variables can be found by linearizing about the state $\hat{x}_{t+1,k}$ and input $\hat{u}_{t,k}$ as seen in (40) - (42).

$$\Omega_{t,k} = \frac{\partial g^{-1}}{\partial x_{t+1}}(\hat{x}_{t+1,k}, \hat{u}_{t,k}) \tag{40}$$

$$\Upsilon_t = \frac{\partial g^{-1}}{\partial u_t}(\hat{x}_{t+1,k}, \hat{u}_{t,k}) \tag{41}$$

$$\sigma_{t,k} = \hat{x}_{t,k} - \Omega_{t,k}\hat{x}_{t+1,k} - \Upsilon_{t,k}\hat{u}_{t,k} \tag{42}$$

The linearized inverse dynamics model can then be used to find the cost-to-come variables and the control policy using forward recursion as seen in (43) - (52).

$$Z_{t+1,k} = H_{t,k} - I_{t,k}^T J_{t,k}^{-1} I_{t,k} \tag{43}$$

$$\zeta_{t+1,k} = h_{t,k} - I_{t,k}^T J_{t,k}^{-1} m_{t,k} \tag{44}$$

$$H_{t,k} = \Omega_{t,k}^T (Z_{t,k} + Q_{t,k})\Omega_{t,k} \tag{45}$$

$$I_{t,k} = \Upsilon_{t,k}^T (Z_{t,k} + Q_{t,k})\Omega_{t,k} + P_{t,k}\Omega_{t,k} \tag{46}$$

$$J_{t,k} = \Upsilon_{t,k}^T (Z_{t,k} + Q_{t,k})\Upsilon_{t,k} + R_{t,k} + P_{t,k}\Upsilon_{t,k} + \Upsilon_{t,k}^T P_{t,k}^T \tag{47}$$

$$h_{t,k} = \Omega_{t,k}^T (\zeta_{t,k} + q_{t,k}) + \Omega_{t,k}^T (Z_{t,k} + Q_{t,k})\sigma_{t,k} \tag{48}$$

$$m_{t,k} = r_{t,k} + P_{t,k}\sigma_{t,k} + \Upsilon_{t,k}^T (\zeta_{t,k} + q_{t,k}) + \Upsilon_{t,k}^T (Z_{t,k} + Q_{t,k})\sigma_{t,k} \tag{49}$$

$$\pi_{t,k}^{-1}(x_{t+1,k}) = N_{t,k} x_{t+1,k} + n_{t,k} \tag{50}$$

$$N_{t,k} = -J^{-1} I_{t,k} \tag{51}$$

$$n_{t,k} = -J_{t,k}^{-1} m_{t,k} \tag{52}$$

where $\pi_{t,k}^{-1}$ is the inverse control policy. The forward pass starts with cost-to-come variables $Z_{0,k}$ set as the zero matrix and $\zeta_{0,k} = 0$. The forward pass then iterates from $t = 0$ until the near-terminal stage $t = L - 1$. ELQR ends when the state estimates from the forward pass and the backward pass converge or when the iteration count reaches a maximum.

Linearization of the CR3BP

The CR3BP equations of motion are linearized and then discretized as presented in the appendix for each state estimate in the ELQR process. This linearization introduces error, which is accounted for by ensuring the discrete time steps of the ELQR process are small. It is important to note that ELQR outlines a process that discretizes using integration and then linearizes. The implementation used in this paper linearizes first to form the variational vector equation, seen in (64), and then integrates the State Transition Matrix (STM) for the duration of the discrete time step. This method is preferred to finite differencing, as finite differencing becomes inaccurate as the discrete time step becomes smaller.

RRT Sampling Region*

All samples for the RRT* algorithms come from a safety region tube defined in [9]. While the sampling occurs in the safety region tube, the trajectories are allowed to depart the area, permitted all intermediate points and rendezvous points lie within the tube. The safety region tube, with a radius of 3884 kilometers at apoapsis, is used to decrease computational burden and to maintain the RPO safety region. The resulting tube is shown in Figure 2.

*Extended Linear Quadratic Regulator Informed RRT**

The ELQR-RRT* algorithm is an embedding of the ELQR algorithm within the RRT* algorithm. The Steer and Cost functions used in Algorithm 2 and Algorithm 3 are computed using the ELQR algorithm. An example step in ELQR-RRT* is as follows: sample the solution space, find the nearest node using the cost function from ELQR, check for collisions, if no collisions, add the node and trajectory from converged ELQR from nearest node to new node, test whether the path can reach the end target state of ELQR-RRT* using ELQR, and then rewire the completed paths.

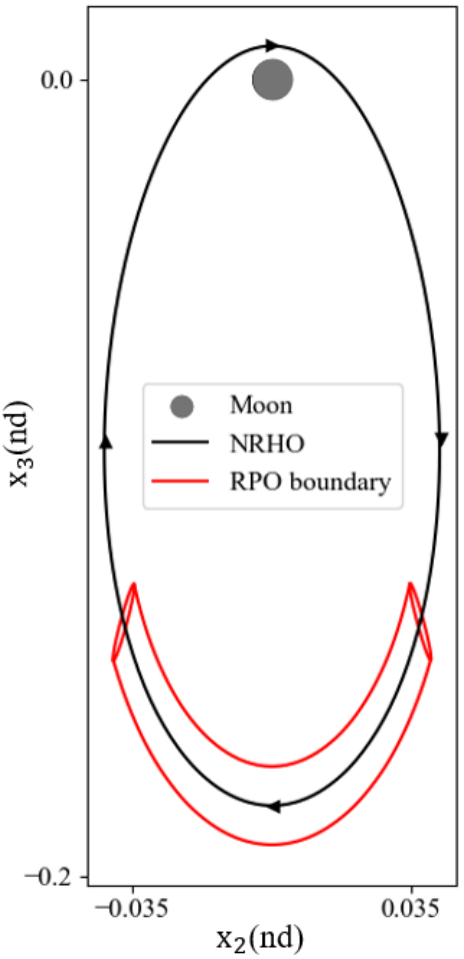


Figure 2. Safety region tube in CR3BP reference frame.

The ELQR-RRT* determines the intermediate points by randomly sampling the solution space. The node's position is sampled from the preset spatial region shown in Figure 2. The node's velocity is determined by finding the spacecraft's velocity at the same x_2 value along the NRHO and then varying the velocity by the arbitrarily selected ± 10.245, ± 10.245, and ± 20.490 meters per second, for x_4, x_5, and x_6, respectively. A velocity vector is chosen for the intermediate nodes to ensure that the terminal state cost matrix is positive definite, as prescribed in [6].

*Shooter-Informed RRT**

The Shooter-informed RRT* (S-RRT*) algorithm is an extension of RRT* that uses a multidimensional Newton-Raphson shooter as the steer function in the RRT* algorithms [9]. Unlike ELQR, S-RRT*'s cost function is the L_1 norm of the input as shown in (53).

$$J = \sum_{t=0}^{L} |u_t| \tag{53}$$

This work utilizes the simple impulsive guidance law generated by S-RRT* as a benchmark solution for ELQR-RRT* and ASSET. S-RRT*'s trajectory design is composed of two subroutines: the single-shooter and the pseudo-multiple-shooter. The single-shooter is a straightforward extension of the Newton-Raphson method to trajectory design. The single-shooter is comprised of five steps as follows:

1. *Integrate:* The initial state, x_0, is integrated using the nonlinear dynamics. The initial state for the NRHO rendezvous problem has six variables, as given in (5).
2. *Solve for the Constraint:* This method is a multi-dimensional root finder; therefore, the trajectory of interest has been determined when the constraint returns the zero-vector. The constraint is a spatial constraint requiring that the final position matches the target position as given in (54)

$$F(x) = \begin{bmatrix} x_{f,1} - x_{target,1} \\ x_{f,2} - x_{target,2} \\ x_{f,3} - x_{target,3} \end{bmatrix} \tag{54}$$

where $x_{target,1}$, $x_{target,2}$, $x_{target,3}$ and $x_{f,1}$, $x_{f,2}$, $x_{f,3}$ are the positional coordinates of the target state and final state of the trajectory respectively.
3. *Determine the Derivative of the Constraint:* In order to guide the initial conditions to the target state, the derivative of the constraint with respect to initial state variables is necessary. The state variables that are varied are the velocities of the initial state and the time of flight to simulate a ΔV impulse.
This derivative can be solved for analytically or numerically through finite differencing. An analytical method is used for computational speed and is presented in (55)

$$DF(x) = \frac{\partial F(x)}{\partial x_{vary}} = \begin{bmatrix} \Phi_{14} & \Phi_{15} & \Phi_{16} & x_f, 4 \\ \Phi_{24} & \Phi_{25} & \Phi_{26} & x_f, 5 \\ \Phi_{34} & \Phi_{35} & \Phi_{36} & x_f, 6 \end{bmatrix} \tag{55}$$

where $x_{0,4}$, $x_{0,5}$, $x_{0,6}$ and $x_{f,4}$, $x_{f,5}$, $x_{f,6}$ are the velocity components of the chaser at the initial and final time, respectively, $x_{vary} = \begin{bmatrix} x_{0,4} & x_{0,5} & x_{0,6} & \tau \end{bmatrix}^T$, τ is time of flight, and Φ_{ij} is the component of the state transition matrix from the i^{th} row and the j^{th} column solved for in (64).
4. *Change the Initial State:* Once the constraint, derivative of the constraint, initial state, and final state are found, the initial state's velocity vector and time of flight can be adjusted to arrive at the target node using (56) [31].

$$x_{vary}^{i+1} = x_{vary}^{i} - [DF(x^i)]^{-1} F(x^i) \tag{56}$$

After x_{vary} for the i^{th} iteration is changed, using (56), the initial state x_0 is altered, and the process repeats.

Multiple-shooters are typically used to discretize a trajectory with sensitive dynamics into multiple segments and then correct each segment [31]. If all of the points lie on the same trajectory, then the constraint function is to ensure that the trajectory is continuous. A continuous constraint is guaranteed by ensuring that the final state of the first point, x_{first}, is the same as the initial state of the second point, x_{second}, and so on.

S-RRT* utilizes a pseudo-multiple-shooter to generate a trajectory from the current sample in the RRT* algorithm to the chaser spacecraft's state. A pseudo-multiple-shooter allows the constraint to account for the moving position of the target spacecraft. The constraint for the rendezvous trajectory is to match the position of the target spacecraft at the end of the safety region and is in a similar form to (54). This trajectory is assumed to be the most cost-effective by allowing the longest coast time to the target. The shooter trajectory does not account for the velocity matching the target spacecraft. Therefore, S-RRT* includes the cost of a final approach burn by adding the norm of the difference between the final velocity vector of the target and chaser spacecraft.

ASSET Optimization

Optimization through ASSET discretizes the optimal control problem into a nonlinear program that can be solved using direct optimization. For this work, ASSET discretizes the optimal control problem using cubic Legendre-Gauss-Lobatto (LGL) collocation and optimizes the resulting nonlinear program with it's native Parallel Sparse Interior-point Optimizer (PSIOPT) [12]. A solution of this type requires an initial guess, which is generated using the pseudo-multiple-shooter from the S-RRT* algorithm.

ASSET's direct optimization can utilize a non-quadratic cost function and can natively handle constraints outside of the cost function. ASSET is therefore able to handle the L_1 norm cost function, shown in (53) to provide the fuel-optimal solution. Instead of weighting the cost function to ensure the spacecraft starts and ends at the required states, these constraints can be formulated as equality constraints for the system in (57)-(58).

$$h_1(x_{t=0}) = x_t - x_0 \tag{57}$$

$$h_2(x_{t=L}) = x_t - x_{targ} \tag{58}$$

where x_{targ} references (11). To prevent collisions between the vehicle and obstacles, a constraint is set in (59), where the state of each obstacle is determined using cubic interpolation of the obstacle's states. Equation (59) uses $x_{ob,i,j}$ as the state of the i^{th} object at time j, r_i as the radius of the i^{th} object, $\bar{0}$ as the zero vector, and ψ as the number of obstacles.

$$g_1(x_t) = \begin{bmatrix} |x_t - x_{ob,1,t}| - r_1 \\ |x_t - x_{ob,2,t}| - r_2 \\ \cdots \\ |x_t - x_{ob,\psi,t} - r_\psi| \end{bmatrix} \leq \bar{0} \tag{59}$$

The final inequality constraint aids in the convergence of the optimizer by assuming the control rates cannot be greater than an arbitrary value. For this study the control rates cannot have a value larger than 10 nondimensional units of jerk as seen in (60)

$$g_2(u_t) = \frac{1}{\Delta t} \begin{bmatrix} ||u_{t,1} - u_{t-1,1}|| \\ ||u_{t,2} - u_{t-1,2}|| \\ ||u_{t,3} - u_{t-1,3}|| \end{bmatrix} - \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix} \leq \bar{0} \tag{60}$$

where u_t, i is the i^{th} control at time t and Δt is the time between $t-1$ and t. ASSET's optimizer is set to ensure the convergence tolerance for the Karush-Kuhn-Tucker, equality, inequality, and barrier infeasibility is less than $1e^{-6}$, so the optimizer returns a locally optimal trajectory. It is important to note that the ASSET solution directly solves (6) and does not assume impulsive control. The ΔV cost for the ASSET solution can be found by numerically integrating the control effort, providing a direct comparison to ELQR-RRT* and S-RRT*.

4. RESULTS

The Results section is composed of the methods undergone to tune the ELQR-RRT* and S-RRT* algorithms as well as the comparative analysis of ELQR-RRT*, S-RRT*, and ASSET's cubic LGL collocation solution. The guidance solutions are compared by considering a rendezvous for two satellites in a string of pearl formation. Three passive satellites will also be considered for this study and will act as moving obstacles during the RPO. The satellites will have the same configuration as [9], as shown in Figure 3.

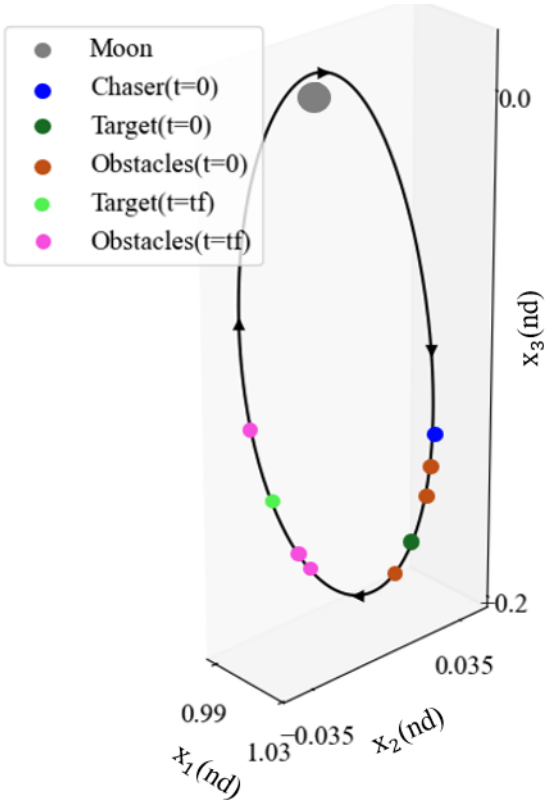


Figure 3. Target and Obstacles Initial and Final States

The obstacles shown in Figure 3 start with mean anomalies of $92.5°$, $105°$, and $150°$. The chaser and target spacecraft have initial mean anomalies of $130°$ and $80°$, respectively. The mean anomaly of the NRHO is defined like that of the mean anomaly for Keplerian orbits, as seen in (61),

$$M = 2\pi \frac{t_p}{T} \tag{61}$$

where t_p is the time since periapsis and T is the period of the orbit [34].

As discussed in [9], when the chaser spacecraft enters the safe rendezvous region, the target spacecraft has been in the safety region for around 21.875 hours. Therefore, the time of flight (TOF) is about 65.625 hours, since the satellites are in the safety region for approximately 87.50 hours on their NRHO. The RRT* algorithm is not bounded by ensuring every node is within a set *connection distance*, allowing for an entirely uniform random sampling of the solution space [9].

The keep-out radius for the three passive satellites is arbitrarily set to 50 kilometers to limit the solution space. The chaser's approach angle, with respect to the target spacecraft, is not considered. This section reveals that ELQR-RRT* generates safe trajectories that approach the ASSET solution for continuous dynamics. As the cost function for ELQR-RRT* can be seen as the quadratization of the L_1 norm, the L_1 norm of each trajectory is compared. It is important to note that for the same algorithm using a L_1 or L_2 norm can result in different costs; therefore, this work cannot rely solely on a quantitative analysis. Currently, there is work to bridge the gap between L_1 and L_2 optimization, such as the algorithm proposed in [33]. S-RRT* and ASSET are shown to require less control effort than ELQR-RRT*, but ELQR-RRT* still provides a smooth, low-cost trajectory.

ELQR-RRT Tuning*

Before the ELQR-RRT* and S-RRT* results can be compared, the ELQR-RRT* is tuned to ensure the lowest cost trajectory is generated for the RPO. The cost functions presented in (12) - (13) are driven by the tracking error weight matrix at the initial stage and terminal stage, Q_0 and Q_L, respectively, and the input weight matrix, R.

Manual tuning of the ELQR-RRT* weight matrices is required to obtain an acceptable accuracy for a cislunar transfer. The weight matrices for this work are set as $Q_0 = Q_L = (10^5)\mathrm{I}^{6\times6}$ and $R = (0.05)\mathrm{I}^{3\times3}$, where $\mathrm{I}^{i\times j}$ is a $i \times j$ identity matrix. With the given weight matrices, the ELQR-RRT* process is able to arrive within 7.6788 kilometers of the target spacecraft's final position, with a step size of 1.312 minutes. The accuracy of the ELQR-RRT* algorithm does not permit the spacecraft to arrive within 0.1 kilometers of the target spacecraft, the requirement to switch from a rendezvous to a docking procedure outlined in [23], but is well within the 100 kilometer approach required for far cislunar rendezvous, provided in [35].

The accuracy of the ELQR's convergence depends linearly on the integration step size for step sizes of 1.75 - 26.25 minutes. The target miss distance was found to range from approximately 11.00 - 166.65 kilometers for step sizes of 1.75 - 26.25 minutes for the ELQR-RRT* guidance solution of the NRHO rendezvous. With step sizes smaller than 1.75 minutes, the accuracy produces diminishing returns for increased computational effort. The integration step size is therefore set to 1.75 minutes to ensure a computationally efficient and accurate solution.

To ensure a near-optimal solution for the ELQR-RRT*, the TOF is optimized for transfers to intermediate nodes. Temporal optimization is possible for future implementations of ELQR-RRT*; however, for this work, the TOF is optimized manually. To explore the effect that the TOF has on the ΔV cost, the ELQR-RRT* is iterated with 50 samples for an intermediate TOF of approximately 5.2110 to 56.2785 hours with a time step of 2.6250 minutes. The variable design is presented in Figure 4.

Figure 4 reveals a jagged ΔV response to time-of-flight, which is expected considering the optimal sample varies as the TOF increases. The spacecraft rendezvous problem has a cost that is tightly coupled to the rendezvous time; therefore, it is safe to assume that as the TOF increases, the optimal intermediate node will change as seen in Figure 4. There are several alternatives to this TOF analysis, such as expanding the solution space to include the TOF as a random variable or restricting the distance of the nodes and TOF to generally optimal times for the node distance.

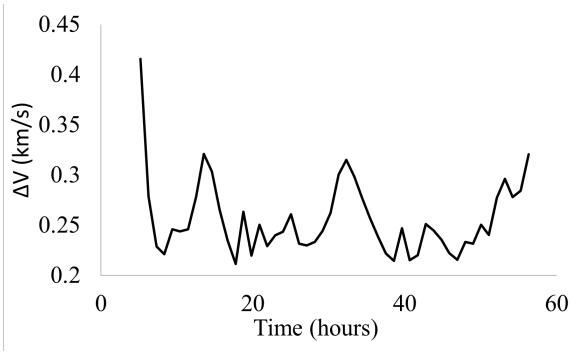


Figure 4. ELQR-RRT* Time of Flight Optimization

S-RRT Tuning*

The S-RRT* search is tuned in [9], resulting in an optimal intermediate time of flight (TOF) of about 0.1 nondimensional units (10.42 hours). Although the TOF is a free variable in the shooting algorithm, the tuning of S-RRT* allows a comparative analysis without concern of time-optimality skewing results. The intermediate TOF for ELQR-RRT* and S-RRT* are set at 17.72 and 10.42 hours, respectively.

Comparative Analysis

This section provides a comparison of ELQR-RRT* and S-RRT* after tuning the TOF to be optimal for the samples provided. This section also compares both solutions to the ASSET solution. Fifty samples of the solution space for ELQR-RRT* and S-RRT* were used in this analysis, resulting in a total ΔV or L_1 norm cost of approximately 0.2139, 0.1942, and 0.1220 kilometers per second for the guidance laws generated by ELQR-RRT *, ASSET and S-RRT *, respectively. The resulting trajectories for the guidance laws can be seen in the relative reference frame given in Figure 5.

Figure 5 reveals that ASSET, S-RRT*, and ELQR-RRT* follow similar paths and validate the ability of S-RRT* and ELQR-RRT* to approach the optimal solution quickly with only 50 samples. To further analyze the results of ELQR-RRT*, S-RRT*, and ASSET's solutions, the control effort for each algorithm is provided in Figures 6 - 8.

Figures 6 - 8, along with the total cost of the trajectories, show that the more strictly the guidance laws can implement impulse and coast trajectories, the more cost effective they are. Further analysis of Figure 7 and Figure 8 reveal that ASSET reaches the max control rate constraint and generates a control response with a fixed slope at the beginning and end of the trajectory. The ELQR-RRT* response shown in Figures 6 - 8 presents a quasi-continuous impulsive control effort with discontinuities due to the abrupt switching of targets.

To compare the fuel optimality of ELQR-RRT*, S-RRT*, and ASSET, the L_1 norm cost for all solutions is shown in Figure 9. Figure 9 proves that ELQR-RRT* is able to achieve a similar magnitude L_1 norm cost with regards to that of ASSET and S-RRT*, despite requiring impulsive burns every 1.75 minutes. The comparison provided by Figure 9 also shows how similar the ASSET and S-RRT* solution is and how the the L_2 norm and additional cost function terms change the behavior of the fuel-optimality of ELQR-RRT*. Further work with ELQR that allows for a coasting arc, more complex engine constraints, and techniques that close the gap in L_1 and L_2 norm optimality can allow for a more optimal trajectory usable for specific mission design.

It is anticipated that as ELQR-RRT* increases the sample size, the trajectory will become more optimal, allowing for an *asymptotically optimal* solution to the constraints imposed by ELQR-RRT*. An analysis of the total ΔV cost of ELQR-RRT* with respect to sample size can be seen in Table 1, where the random seed numbers are those utilized by Python's random library.

Table 1. **ELQR-RRT* Cost Given Sample Size**

Sample Size	Random Seed	ΔV
10	20	0.44383 km/s
25	20	0.37817 km/s
50	20	0.25813 km/s
300	20	0.22534 km/s
10	30	0.32071 km/s
25	30	0.30968 km/s
50	30	0.21387 km/s
300	30	0.21387 km/s
10	40	0.48579 km/s
25	40	0.27028 km/s
50	40	0.24473 km/s
300	40	0.22778 km/s
10	50	0.22655 km/s
25	50	0.22655 km/s
50	50	0.22655 km/s
300	50	0.19820 km/s

Table 1 shows a general trend of *asymptotic optimality* with several outliers, where a near-optimal point is found earlier in the sample group. The outlier group for random seed 30 was temporally optimized at 50 samples, thereby providing a fuel-optimal trajectory earlier than 300 samples. These results support asymptotic optimality of the ELQR-RRT* algorithm and present that after 50 samples ELQR-RRT* can reliably present near-optimal guidance solutions for the NRHO rendezvous. Table 1 also presents that with the correct node sampled, ELQR-RRT* can come within 4 meters per second of the cost of the ASSET trajectory, proving that ELQR-RRT* can approximate the locally optimal trajectory shown with ASSET's cubic LGL collocation scheme.

5. CONCLUSION

This paper presents the Extended Linear Quadratic Regulator-informed Rapidly-exploring Random Tree (ELQR-RRT*) search as a method to generate near-optimal guidance laws for nonlinear dynamics and obstacles. This work also demonstrates how to linearize the CR3BP for an impulsive state-space model and reveals the tuning process for the ELQR-RRT* algorithm. The comparative analysis completed in this work verifies the RRT* guidance solutions. S-RRT* finds the locally optimal solution for a 9:2 L2 Southern NRHO rendezvous with only 50 samples and ELQR-RRT* is shown to produce near-optimal low-cost trajectories with 50 to 300 samples.

The ELQR-RRT* is shown to have accuracy errors due to

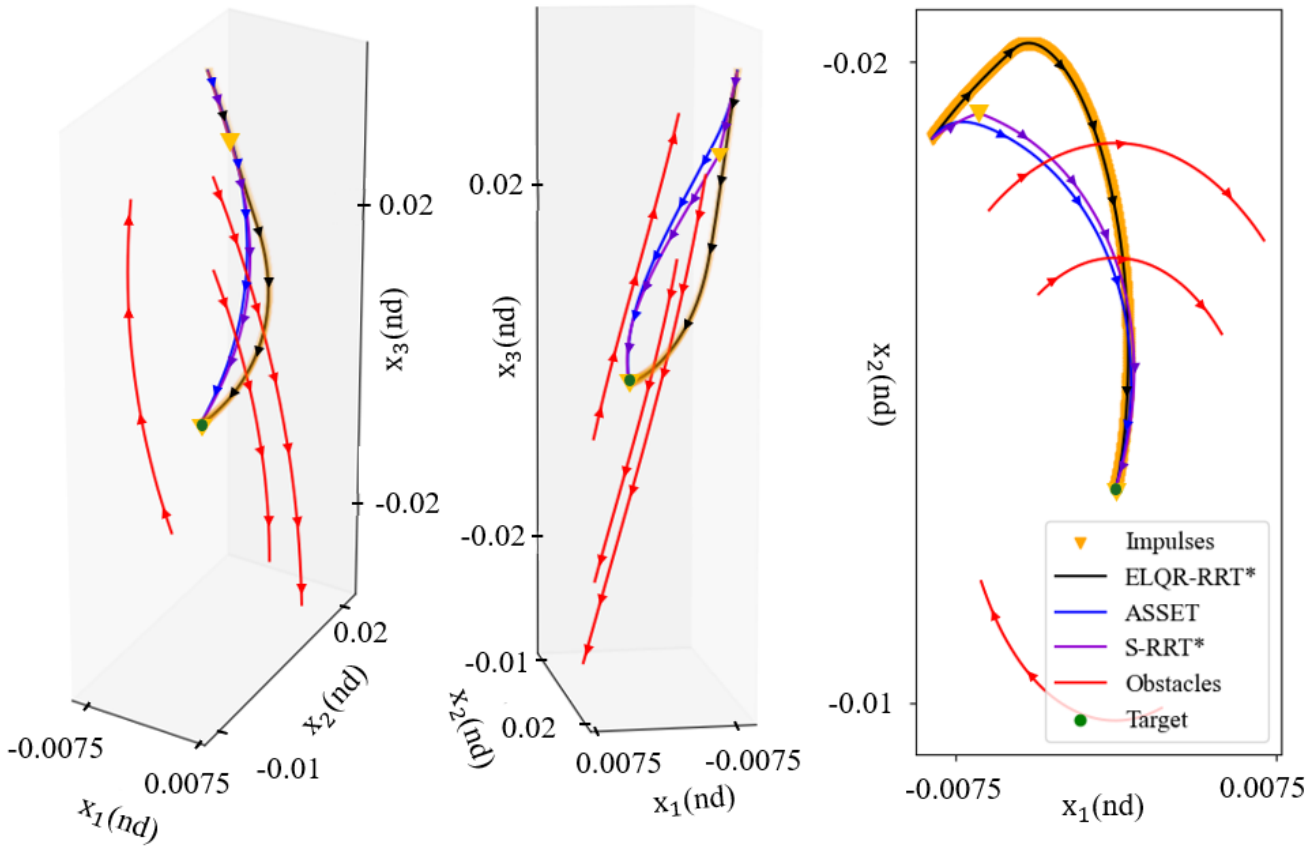


Figure 5. ELQR-RRT* vs. S-RRT* In Reference Frame Relative to Target Spacecraft

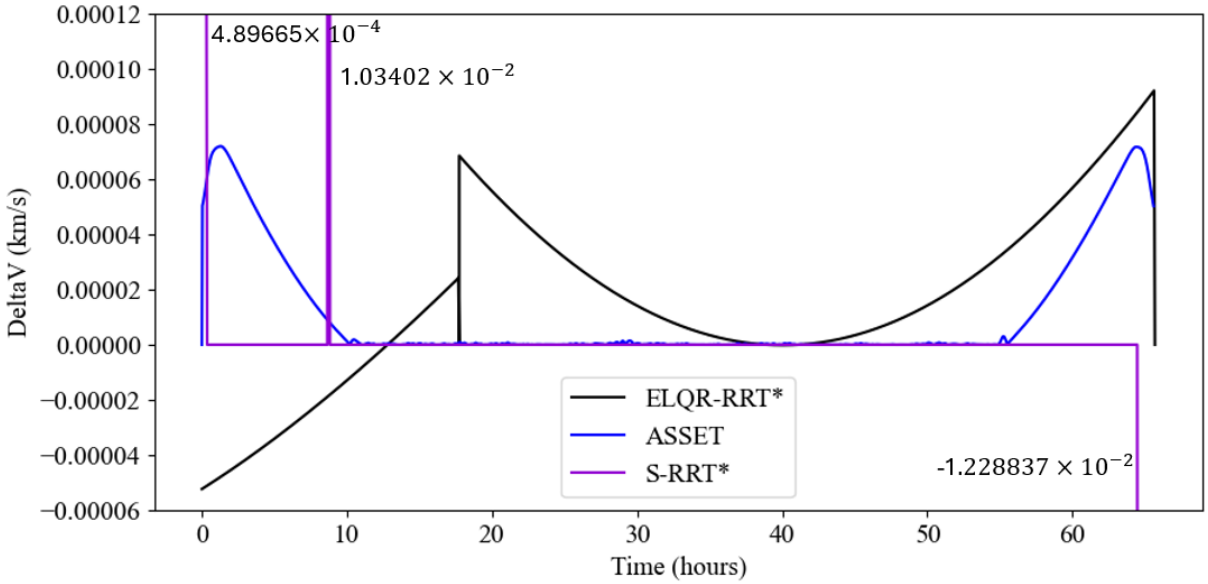


Figure 6. Control Effort u_1

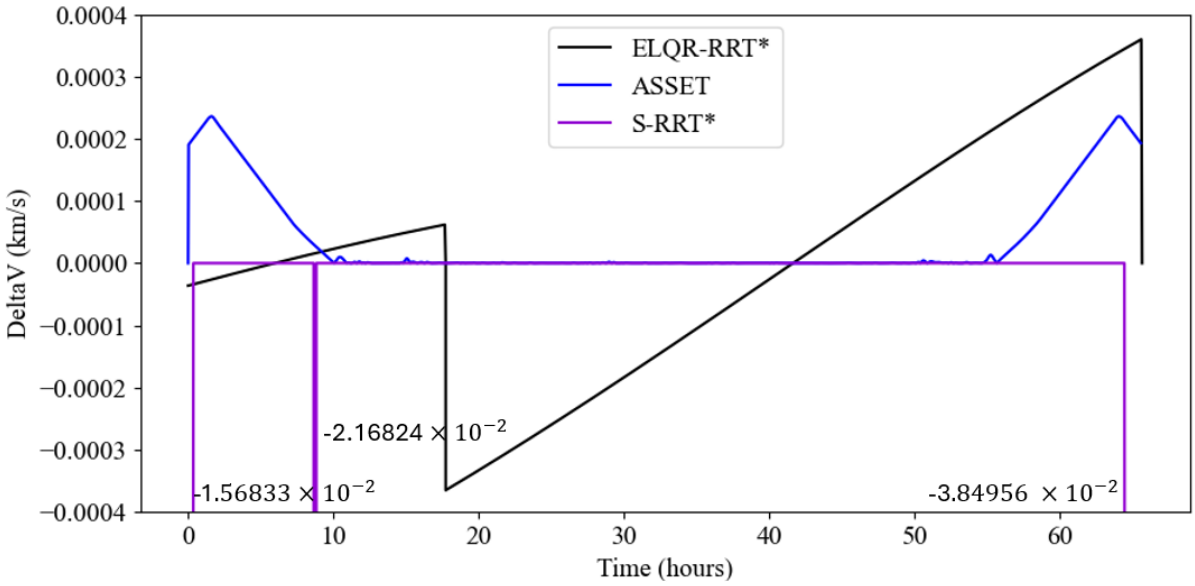


Figure 7. Control Effort u_2

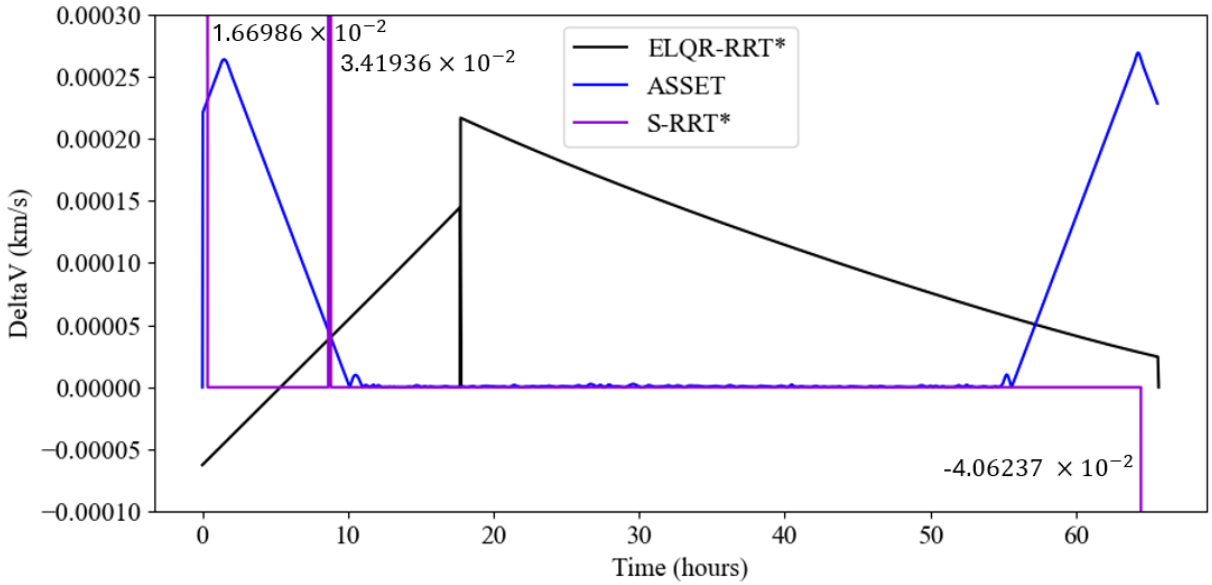


Figure 8. Control Effort u_3

the use of tracking terms in the cost function. ELQR-RRT* also optimizes the L_2 norm of the input with a quasi-continuous control response, making the control solution less fuel-optimal than the strictly impulsive guidance solution presented by the Shooter-based RRT* (S-RRT*). Despite the drawbacks of ELQR-RRT*, it is able to approach the locally optimal solution of ASSET's LGL collocation and can be adjusted for preliminary, mission-specific guidance solutions.

Future work is anticipated to include a numerical optimization of the ELQR-RRT* algorithm and speed comparisons with ASSET's and S-RRT*'s solutions. Further studies are also recommended to investigate the addition of temporal optimization, as well as engine constraints, coast phases, and L_1 and L_2 norm bridging methods to the ELQR-RRT* guidance algorithm. Further work is anticipated to increase accuracy and fuel-optimality. This study may also be extended to ephemeris models or other complex nonlinear systems.

APPENDIX

This section contains the linearization and discretization of the CR3BP for use by the ELQR. The CR3BP can be linearized by using a Taylor series expansion and then neglecting the higher-order terms [30]. The linearized matrix for the

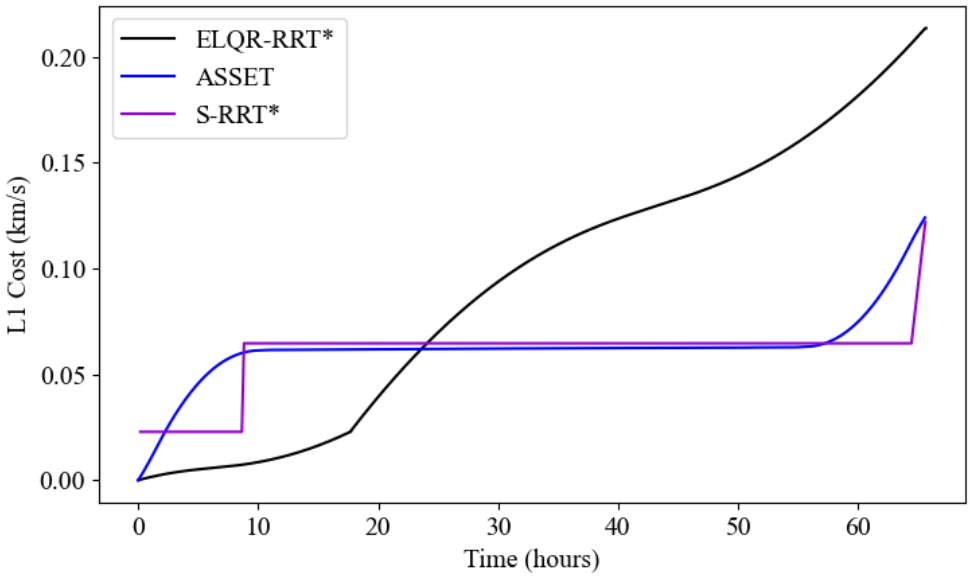


Figure 9. L_1 Norm or Total ΔV

CR3BP, A, is given in (62)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ U^*_{xx} & U^*_{xy} & U^*_{xz} & 0 & 2 & 0 \\ U^*_{xy} & U^*_{yy} & U^*_{yz} & -2 & 0 & 0 \\ U^*_{xz} & U^*_{yz} & U^*_{zz} & 0 & 0 & 0 \end{bmatrix} \tag{62}$$

where U^*_{ij} is the partial derivative of the pseudo-potential function U^* given in (63) with respect to i and j.

$$U^* = \frac{(1-\mu)}{d} + \frac{\mu}{r} + \frac{1}{2}(x^2 + y^2) \tag{63}$$

The linearized A matrix is time and state dependent [30]. The A matrix is calculated at every linearization point and then forms the state transition matrix through the variational vector equation seen in (64).

$$\dot{\Phi}(t, t_0) = A\Phi(t, t_0) \tag{64}$$

By setting t_0 to be the time of the current state and setting $t = t_0 + \Delta t$, where Δt is the discrete time step for the simulation, the CR3BP can be linearized and discretized. The input for this problem assumes impulsive ΔV burns, meaning an instantaneous change in velocity is performed at each timestep allowing the input matrix to take the form seen in (65) [23].

$$B(t, t_0) = \Phi(t, t_0) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{65}$$

The discrete time dynamics for the CR3BP equations of motion can be constructed as given in (66)

$$x_{t+1} = \Phi(t+1, t)x_t + B(t+1, t)u_t \tag{66}$$

where x_{t+1} and x_t are the future and present states of the third primary, or spacecraft of interest.

ACKNOWLEDGEMENTS

The First Author (F.A.) would like to thank Carrie Sandel, Hunjoon Daniel Lee, and Aiden Moran for their reviews of this work. The F.A. also acknowledges support from the Science, Math, and Research for Transformation Scholarship (SMART). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the SMART program. Generative artificial intelligence, Overleaf's Wrifefull, was used to aid in grammar and sentence structuring.

REFERENCES

[1] National Research Council, "NASA space technology roadmaps and priorities: restoring NASA's technological edge and paving the way for a new era in space," *The National Academies Press*, 2012, pp. 151, doi:10.17226/13354.

[2] A. Bowman, Oct. 2025, "Gateway space station," NASA. [Online]. Available: https://www.nasa.gov/reference/gateway-about/

[3] E. Anzalone et al., "Lunar navigation beacon network using global navigation satellite system receivers," presented at the Int. Astronautical Congr., Washington, DC, USA, Oct. 21-25, 2019.

[4] A. Fear, E. G. Lightsey, "Autonomous rendezvous and docking implementation for small satellites using model

predictive control, " *J. Guid. Control Dyn.*, vol. 47, no. 3, pp. 539-547, Mar. 2024.

[5] J. Starek, B. Acikmese, I. Nesnas, M. Pavone, "Spacecraft autonomy challenges for next-generation space missions," *Lecture Notes in Control System Technology for Aerospace Applications*, vol. 460, pp. 1-48, 2016.

[6] J. Berg, "Extended LQR: locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost," in *Robotics Research*, STAR, vol. 114, Cham, Switzerland: Springer, 2016, doi:10.1007/978-3-319-28872-3.

[7] X. Zhong, Z. Wei, T. Chen, "Motion Planning and Pose Control for Flexible Spacecraft Using Enhanced LQR-RRT*," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 6, pp. 8743-8751, Dec. 2023, doi:10.1109/TAES.2023.3309612.

[8] B. Doerr, R. Linares, "Motion planning and control for on-orbit assembly using LQR-RRT* and nonlinear MPC," 2020, *arXiv:2008.02846v1*.

[9] X. Kipping, J. Larson, R. Sood, "An application of rapidly expanding random trees to spacecraft rendezvous," in *AIAA AVIATION AND ASCEND*, Las Vegas, Nevada, 2025, doi:10.2514/6.2025-4051.

[10] S. Fuller, E. Lehnhardt, C. Zaid, K. Halloran,"Gateway program status and overview," *J. Space Safety Engineering*, vol. 9, no. 4, pp. 625-628, Dec. 2022.

[11] D. C. Davis, E. M. Zimovan-Spreen, R. J. Power, K. C. Howell, "Cubesat deployment from a near rectilinear halo orbit," in *AIAA SCITECH 2022 FORUM*, San Diego, CA, 2022, pp. 2022-1765, doi:10.2514/6.2022-1777.

[12] J. Pezent et al., "ASSET: astrodynamics software and science enabling toolkit," in *AIAA SCITECH 2022 FORUM*, San Diego, CA, 2022, pp. 2022-1131, doi:10.2514/6.2022-1131.

[13] Z. Douglas, "Autonomous rendezvous, capture and in-space assembly: past, present and future", in *1st Space Exploration Conf.: Continuing Voyage Discovery*, Jan. 30–Feb. 1 2005, [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2005-2523

[14] A. Isser, "Introduction to guidance, navigation, and control (GNC)", QinetiQ, Inc., Lorton, VA, DSAIC-BCO-2021-172, Jul. 2021, Accessed: Oct. 2025. [Online]. Available:https://dsiac.dtic.mil/wp-content/uploads/2022/10/AD1182620.pdf

[15] R. Yanushevsky, "Basics of missile guidance," in *Modern missile guidance*, 2nd Ed., Boca Raton, FL, USA: Taylor and Francis, 2019 pp. 2-4.

[16] A. Singletary et al., "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Prague, Czech Republic, Sep. 27 - Oct. 1, 2021, doi:10.1109/IROS51168.2021.9636670.

[17] J. Hou et al., "Large-scale vehicle platooning: advances and challenges in scheduling and planning techniques," *Engineering*, vol. 28, pp. 26-48, Sep. 2023, doi:10.1016/j.eng.2023.01.012.

[18] S. Karaman, E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The Int. J. of Robotics Research*, vol. 30, no. 7, pp. 846-894, Jun. 2011, doi:10.1177/0278364911406761.

[19] D. Connel, H. M. La, "Dynamic path planning and replanning for mobile robots using RRT*," in *IEEE Int. Conf. on SMC*, Banff, Canada, Oct. 5-8, 2017, doi:10.1109/SMC.2017.8122814.

[20] A. Perez, R. Platt Jr., G. Konidaris, L. Kaelbling, T. Lozano, "LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics," in *IEEE Internat. Conf. on Robotics and Automation*, Saint Paul, MN, May 14-18, 2012, pp. 2537-2542.

[21] H. P. Geering, *Optimal control with engineering applications*, London, UK: Springer, 2007.

[22] E. Lavretsky, K. A. Wise, "Robust and adaptive control with aerospace applications," in *Advanced Textbooks in Control and Signal Processing*, M. J. Grimble, M. A. Johnson, Eds., London, UK: Springer, 2013.

[23] C. M. Jewison, "Guidance and control for multi-stage rendezvous and docking operations in the presence of uncertainty," Ph.D. dissertation, Dept. Aero. and Astro., MIT, Cambridge, MA. USA, 2017.

[24] J. Pezent, "On the implementation and applications of a high performance trajectory optimization toolkit," Ph.D. dissertation, Dept. Aero Eng. and Mech, Univ. of Alabama, Tuscaloosa, AL. USA, 2024.

[25] T. Tassa, T. Erez, E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in IEEE/RSJ Int. Conf. on Intel. Robots and Sys., Vilamoura, Algarve, Portugal, Oct. 7-12, 2012, pp. 4906-4913.

[26] R. W. Thomas, J. D. Larson, "Receding horizon extended linear quadratic regulator for RFS-based swarms with target planning and automatic cost function," *IEEE Transactions Control Network Systems*, vol. 8, no. 2, Jun. 2021, doi: 10.1109/TCNS.2021.3050133.

[27] S. Karaman, E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," *IEEE Conf. on Decision and Control*, Atlanta, GA, Dec 15-17, 2010, doi: 10.1109/CDC.2010.5717430.

[28] J. Sikes, "On the applications of signed distance fields to spacecraft trajectory," Ph.D. dissertation, Dept. Aero Eng. and Mech, Univ. of Alabama, Tuscaloosa, AL. USA, 2024.

[29] S. T. Scheuerle, K. C. Howell, "Characteristics and analysis of families of low-Energy ballistic lunar transfers," *AAS/AIAA Astro. Specialist Conf.*, Big Sky, MT, USA, 2021.

[30] R. Sood, "Significance of specific force models in two applications: solar sails to Sun-Earth L_4/L_5 and GRAIL data analysis suggesting lava tubes and buried craters on the Moon," Ph.D. dissertation, Dept. of Aero. and Astro., Purdue, West Lafayette, IN. USA, 2016.

[31] D. J. Grebow, "Trajectory design in the Earth-Moon system and lunar south pole coverage," Ph.D. Dissertation, Dept. of Aero. and Astro., Purdue, West Lafayette, IN., USA, 2010.

[32] G. Franzini, M. Innocenti, "Relative motion dynamics in the restricted three-body problem," *J. Spacecraft and Rockets*, vol. 56, no. 5, 2019, pp. 1322-1337.

[33] S. C. del Valle, P. Solano-Lopez, H. Urrutxua, "A Dual-Based Linear Programming Formulation of Optimal Control: Fuel-Optimal Rendezvous Guidance and Boresight Pointing Applications," *Aerospace Europe Conference*, Switzerland, Jul. 9-13, 2023, doi:10.13009/EUCASS2023-057

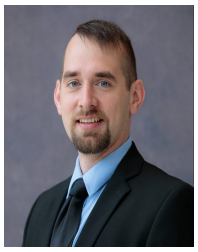
[34] H. D. Curtis, "Orbital position as a function of time," in

Orbital mechanics for engineering students, revised 4th ed., Cambridge, MA, USA: Elsevier, 2021, pp. 141-180.

[35] T. J. Fay, A.P. Wilmer, R.A. Bettinger, "Investigation of near-rectilinear halo orbit search and rescue using staging L1/L2 Lyapunov and distant retrograde orbit families," in *J. of Space Safety Engineering*, vol. 11, no. 2, 2024, pp. 165-173.

BIOGRAPHY

Xavier Kipping received his B.S. in Mechanical Engineering with an Aerospace Emphasis from Utah State University. Xavier is currently pursuing a Ph.D. in Aerosapce Engineering and Mechanics from the University of Alabama with a focus on Guidance, Navigation and Control.



Dr. Jordan D. Larson is an Assistant Professor at the University of Alabama in the Department of Aerospace Engineering and Mechanics where he directs the Laboratory for Autonomy, GNC, and Estimation Research (LAGER) and leads the Platforms team for the Remote Sensing Center (RSC) at the University of Alabama (UA). Dr. Larson received his Ph.D. from the Department of Aerospace Engineering and Mechanics at the University of Minnesota-Twin Cities. His research interests include autonomous multi-vehicle systems, guidance, navigation control (GNC), multi-target tracking (MTT), multi-sensor data fusion, and information risk analysis.

Dr. Ali Pakniyat is an Assistant Professor in the department of Mechanical Engineering at the University of Alabama. He received the B.Sc. degree in Mechanical Engineering from Shiraz University, the M.Sc. degree in Mechanical Engineering from Sharif University of Technology, and the Ph.D. degree in Electrical Engineering from McGill University. After holding a Lecturer position at the Electrical and Computer Engineering department of McGill University, and two postdoctoral positions in the department of Mechanical Engineering at the University of Michigan and the Institute for Robotics and Intelligent Machines at Georgia Tech, he joined the University of Alabama in 2021 where he is now an Assistant Professor in the department of Mechanical Engineering. His research interests include deterministic and stochastic optimal control, nonlinear and hybrid systems, multi-agent systems and mean field games, with applications in automotive industry, robotics and mathematical biology.