Astrodynamics-informed kinodynamic motion planning for relative spacecraft motion

by

Taralicin Deka

B.Tech., Electrical Engg, National Institute of Technology Silchar, 2015

M.S., Aerospace Engg, University of Colorado Boulder, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Aerospace Engineering Sciences

2023

Committee Members:

Jay W. McMahon, Chair

Dr. Hanspeter Schaub

Dr. Issa Nesnas

Dr. Marcus Holzinger

Dr. Morteza Lahijanian

Deka, Taralicin (Ph.D., Aerospace Engineering Sciences)

Astrodynamics-informed kinodynamic motion planning for relative spacecraft motion

Thesis directed by Dr. Jay W. McMahon

The work presented in this dissertation takes inspiration from robotic sampling-based motion planning ideas and presents an astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm which is a single-step sampling-based kinodynamic approach to orbital motion planning problems (as opposed to existing two-step sampling-based motion planning approaches in literature). The AIKMP algorithm can quickly find solutions to spacecraft relative transfer problems in a very cluttered environment and it iteratively improves on its computed transfer solution and thus holds the potential of computing near-optimal transfers given a sufficient number of algorithm iterations – without needing an initial guess of the solution. This algorithm introduces an astrodynamics-informed pruning module that allows the motion planner to maintain and store a sparse set of nodes improving its overall computation efficiency by $\approx 98\%$ and storage efficiency by $\approx 80\%$. This work also presents a novel extension of the linearized Lambert solution (LLS) called the closed-loop linearized Lambert guidance solution that allows a spacecraft to apply correction burns during the transfer to improve the targeting accuracy in relative guidance problems in the presence of perturbations like Drag, J2, and Solar Radiation Pressure (SRP). This novel closed-loop guidance law is applied to Spacecraft Formation Flying (SFF) problem, and by presenting theoretical developments backed with multiple simulation results it is shown that the algorithm allows for stringent targeting accuracy with fuel efficiency in different SFF problems. It is also demonstrated that closed-loop LLS guidance can be used to enable a spacecraft to safely follow a reference trajectory generated by an orbital motion planner like the AIKMP algorithm at a cost of $\approx 3\%$ fuel increase as compared to open-loop guidance that is unable to provide safety guarantees in a cluttered and perturbed environment. Lastly, this work presents an extension of the popular E/I vector separation method derived for the case of drifting relative motion to infuse passive

collision avoidance capabilities in sampling-based orbital motion planners. The resulting motion planner performs tree extension in a more informed way such that collision avoidance constraints are satisfied by each node and edge in the sampling-based tree. This way, every transfer solution computed from the tree is guaranteed to be collision-free for the entire duration of the transfer in the presence of multiple static (static relative to a moving reference frame) and moving obstacles – a capability that is crucial for SFF applications. Overall, the AIKMP algorithm describes the sufficient number of steps that can be adapted from a sampling-based robotic motion planner with required modifications to achieve fuel-efficient and collision-free sampling-based motion planning in astrodynamics applications. A notable aspect of this proposed motion planning framework lies in its modularity, as individual modules of the algorithm can be adapted to any existing sampling-based motion planner (tree-based planners or graph-based planners) to improve their applicability to orbital motion planning.

Dedication

This thesis is dedicated to the unwavering love and unparalleled sacrifices of my beloved parents, Akhil Chandra Deka and Cinema Medhi Deka, as well as my brother, Debasish Deka. Their enduring encouragement has been my guiding light throughout this academic journey. Ma, Papa - Thank you for staying strong through thick and thin while I pursued my dreams thousands of miles away from home. Dada, thank you for turning my dreams of pursuing higher studies in the US into reality with your boundless support, both emotionally and financially. Thank you for never holding me back and always believing in me. I could not have done this without you.

Acknowledgements

Foremost, I extend my deepest gratitude to my advisor, Dr. Jay McMahon, whose unwavering support has been the cornerstone of my Ph.D. journey. He has not only been a mentor and guide but also a friend and guardian, offering encouragement and hope when I needed it most. My time at ORCCA Lab will forever hold a special place in my heart. Thank you for everything, Jay, and thank you, ORCCA. I would also like to express my appreciation to Dr. Dale Lawrence for introducing me to the captivating world of research and providing invaluable guidance and career advice during pivotal moments in my life. Heartfelt thanks to Dr. Hanspeter Schaub for imparting lessons of utmost professionalism and humility in work. The insights gained from you, both within and beyond the curriculum, are truly invaluable. My gratitude extends to all my Ph.D. committee members for their constructive feedback, significantly enhancing the quality of this thesis. A special acknowledgment is reserved for Dr. Saptarshi Bandyopadhyay, who played a crucial role in my Ph.D. committee during my comprehensive exam. To all my friends, your companionship during moments of laughter and tears has made this journey so much more meaningful. I found some of my best friends during my time at CU, and I am forever grateful for these relationships. To my fiancé, Ankit, thank you for the unwavering emotional support and for the constant reminders to share moments of fun and adventures throughout this time. A special thank you to the 'Assamese community of Colorado' for being a home away from home throughout these years. Lastly, thank you to all my well-wishers for supporting me in every way possible and for contributing to the person I am today.

A proud Orcca signing off!

Contents

Chapter

Tables

Table

Figures

Figure

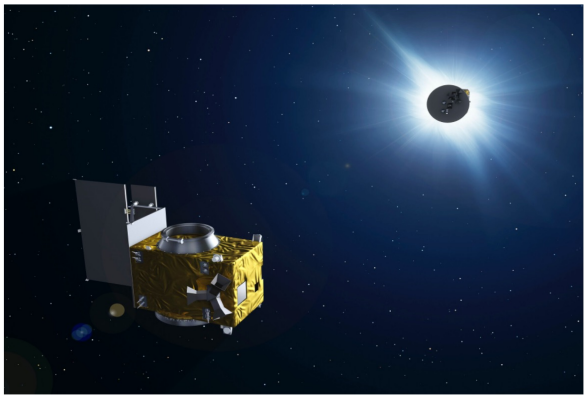
Chapter 1

Introduction

1.1 Background and Motivation

A space mission designed with multiple smaller spacecraft has many advantages compared to a mission comprising a single traditional monolithic spacecraft. Besides introducing a single point of failure to the entire mission, building, testing, and launching a monolithic spacecraft can be more expensive compared to modular designs as the cost of manufacturing and integrating all components into a single structure can be substantial. In fact, the complexity of assembling and testing an entire monolithic spacecraft can also lead to longer development and deployment cycles [20, 97, 91, 113]. As a result, there has been an increased interest in developing distributed spacecraft formation flying (DSFF) missions in the aerospace sector [35]. Ever since NASA's Gemini

(a) NASA's Gemini 6 and 7 (1965)



(b) ESA's Proba-3 mission (planned 2024)

Figure 1.1: Evolution of spacecraft formation flying missions

space program demonstrated the ability to perform coordinated spacecraft operations for the first time [64] (Figure 1.1a), distributed spacecraft systems have attracted the interest of several leading aerospace organizations. There are multiple NASA [126, 60, 11], DoD [112, 22, 40] and ESA missions [62, 49, 90] that use spacecraft formation flying technology and are either flying, under development or proposed. A few of such exciting missions include the Laser Interferometer Space Antenna (LISA) mission that aims to detect and accurately measure gravitational waves [38, 65], the Darwin mission that is a long-range interferometry mission to find Earth-like planets [62, 39], and the upcoming PROBA-3 mission which will demonstrate high precision formation flying technologies and techniques without relying on information from the ground [90, 125] (Figure 1.1b). Distributed spacecraft systems also have great applications in on-orbit servicing (OOS)[68, 59, 88, 95]. In fact, a recently released national strategy document by the United States White House details their interest in developing autonomous robotic servicing space missions to aid satellite servicing, refueling, in-space construction, and maintenance of large structures [1]. Such missions will heavily depend on multiple spacecraft working in coordination to achieve the mission goals.

While spacecraft formation flying is an enabling technology for many next-generation space missions, they are inherently very risky as multiple spacecraft need to coordinate and operate in close proximity to each other [5, 7]. Reconfiguration of spacecraft within the formation is considered one of the major challenges of such systems, which is why a great deal of research goes into the problem of spacecraft motion planning with collision avoidance [51, 108, 117, 24]. The stringent requirements of formation flying demand that the states of each agent in the formation are defined relative to the other agents; this new dynamical description renders conventional control and trajectory design approaches (used on monolithic spacecraft) ineffective. Hence, there is an immense need for fast and efficient algorithms to compute near-optimal, safe, and collision-free relative trajectories for such multi-spacecraft systems.

In the field of astrodynamics, traditional guidance algorithms like Lambert's solution can compute the transfer orbit for a spacecraft between two given position vectors [17, 32, 131]. However, such algorithms do not come with obstacle-avoidance capabilities. State-of-the-art methods

for spacecraft collision avoidance include model predictive control [101, 132, 136] and artificial potential functions [12, 28, 72, 135]. These methods work well in static uncluttered environments; however, they often fall short when optimization and time-varying constraints become key features of the problem. Moreover, gradient-based methods like artificial potential functions are also known for their possible convergence to local minimum solutions. Another well-known approach for spacecraft collision avoidance is conjunction assessment. This method handles collision avoidance by actively tracking the obstacles and performing escape maneuvers whenever the collision probability gets higher than a specified threshold [102, 25, 114]. However, such an approach is only valid for short-term encounters which is why they will be impractical for spacecraft formation flying applications. This is where inspiration can be taken from the vast literature on collision-free motion planning techniques, popular in robotics, that can be leveraged for collision-free motion planning applications in astrodynamics [79, 84, 123, 122, 57].

Building self-reliant autonomous systems has always been the ultimate goal in the field of robotics. While popular robotic planning methods like the Monte Carlo tree search (MCTS) exists, they are effective for planning in discrete action spaces and hence have great applications in designing game-playing bots or solving sequential decision problems [124]. As a result, in this work, ideas from another powerful robotic motion planning technique called sampling-based motion planning algorithms are leveraged due to their efficiency in exploring high-dimensional state spaces and handling complex, continuous environments making them more suitable for real-world applications [76, 63, 70, 73]. The efficacy of these methods has been already proven in several challenging real-world systems such as the 2007 DARPA Urban Challenge [21] and the 2021 DARPA Subterranean Challenge [105, 99]. In the 2007 DARPA Urban Challenge, vehicles were required to race autonomously in a simulated 60-mile urban setting. Several teams in the competition used sampling-based planning as their primary guidance technique, including MIT's 4th-place winning Talos car (Figure 1.2a) that used the Rapidly-exploring Random Trees (RRTs) [87, 81]. In the 2021 DARPA Subterranean Challenge, robots were required to conduct search and rescue operations in challenging underground tunnel environments with dangers like stairs, rough pathways,

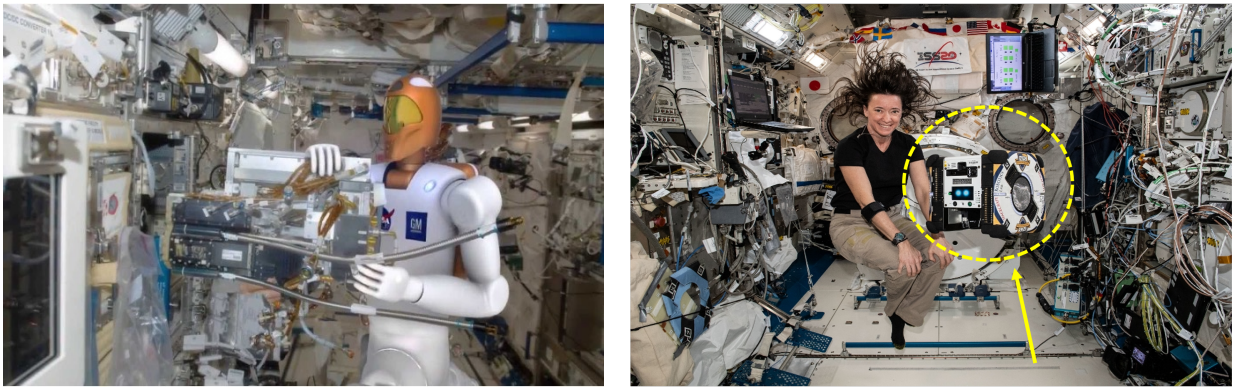
and dynamic obstacles like falling debris. CU Boulder's MARBLE robotic vehicle (Figure 1.2b) which used an RRT* planner to build the global map of the state space, won 3rd place [2, 98].

(a) DARPA Urban Challenge 2007 4th place winner: MIT's Talos car

(b) DARPA Subterranean Challenge 2021 3rd place winner: CU Boulder's MARBLE vehicle

Figure 1.2: Sampling-based planning in challenging terrestrial environments

Although not yet flown on any spacecraft hardware, sampling-based planning already has made its way to space systems. NASA's humanoid robot Robonaut [8, 75, 13] and NASA's au-



(a) NASA's humanoid robot: Robonaut

(b) Autonomously flying robot: Astrobee

Figure 1.3: Sampling-based planning in International Space Station (ISS)

tonomously free-flying manipulator robot Astrobee [118, 53, 3] are currently operational in the International Space Station (ISS) (Figure 1.3). Robonaut uses a variant of an RRT-based planner to plan its motion to assist astronauts with various tasks on board (Figure 1.3a). Similarly, As-

trobee which is the first microgravity testbed of its kind for free-flying manipulator dynamics, uses sampling-based motion planning to generate its initial reference trajectory to enable the robot to assist astronauts with routine tasks and perform experiments (Figure 1.3b).

Initial work in the field of robot motion planning treats the motion planning problem as two distinct parts: solving basic path planning and then finding a trajectory and controller that would track the path while satisfying the system dynamics [83]. Such algorithms consider only kinematics and entirely ignore system dynamics while building the trees. Gradually, robot motion planning algorithms were developed that were suitable for kinodynamic applications (problems with simultaneous kinematic and dynamic constraints) [85, 80, 54, 86]. These algorithms satisfy the probabilistic completeness property, i.e., they return a solution with a probability converging to one as the number of samples grows to infinity, if such a solution exists. More recently, however, the motive behind robot motion planning algorithms has shifted from providing feasible solutions to achieving high-quality solutions. Algorithms were developed that are asymptotically optimal for kinodynamic problems meaning that the solution approaches optimality as the number of samples goes to infinity [76, 10].

The framework of the sampling-based motion planning algorithms is sufficiently general that it applies to spacecraft and rovers just as it does to traditional robots [15, 121, 120, 61, 103, 74]. Existing work in the area of sampling-based orbital motion planning has looked into growing sampling-based random trees in the traditional position and velocity space to achieve collision-free motion planning of spacecraft: Frazzoli et. al. demonstrated that fast orbital motion planning can be achieved by developing an RRT in the spacecraft's relative position and velocity space [61]. In their proposed method, every node in the tree is involved in tree extension in every iteration of the algorithm, and the tree attempts to connect to the goal state every time ($Goal\ bias = 1$), adding to the density of the tree and hence computational inefficiency of the overall approach. Bandyopadhyay et. al devised a 2-step sampling-based planner to achieve fast motion planning for spacecraft swarms in a cluttered environment [15]. The first step of their method explores the 3D space to generate a feasible path from the start state to the target state by extending the tree

in the obstacle-free space using straight lines of adjustable step size (instead of using a steering law). In the second step, the algorithm uses a sequence of convex optimization to generate a fuel-optimal solution from the already generated feasible paths. Starek et. al. extended the application of sampling-based Fast Matching Trees (FMTs) to achieve fast, safe, and fuel-efficient spacecraft motion planning under Clohessy-Wiltshire-Hill Dynamics assumption [74, 121, 119]. This method also results in non-smooth trajectory solutions that demand additional post-processing (a second step) to compute the final fuel-efficient and smooth motion planning solution. Another work was published recently that uses an RRT-based path planner for closeup on-orbit inspection of large complex space structures using multiple small inspector spacecraft with applications to on-orbit inspection missions [58]. However, as explained before and as will be discussed in the details of this dissertation, RRT-based sampling-based planners can get computationally very expensive for onboard applications. As a result, improvements in terms of implementation and computational efficiency are desirable to make such algorithms suitable for fast and fuel-efficient sampling-based orbital motion planning. Figure 1.4 summarizes how the proposed research in this thesis compares with the discussed existing sampling-based orbital motion planners in the literature. It is to be noted here that, while none of the mentioned existing planners consider external perturbations, in this work although perturbations are not included in the motion planning directly, a closed-loop guidance law is developed that can be used to extend the application of any sampling-based orbital motion planners to perturbed environments.

1.2 Proposed Research

As mentioned before, several sampling-based motion planners exist in the robotics literature. Because of their theoretical advancement in the field and the potential of their online implementations, such methods have started gaining attention for space applications including in multi-agent spacecraft systems such as in spacecraft formation flying. However, extending the application of robotic sampling-based motion planners to achieve orbital motion planning requires additional considerations. Due to strict onboard memory limitations and run-time constraints, there is an

METHODS	Base-algorithm	Sampling-space	Single-step planning	Tree sparsity	Moving obstacles	External Perturbations
Frazolli (2003)	Rapidly exploring Random Trees (RRT)	Relative position-velocity	—	✗	✓	✗
Starek (2016)	Fast Marching Trees (FMT)	Relative position-velocity	✗	✗	✗	✗
Bandyopadhyay (2017)	Spherical Expansion-Sequential Convex Programming (SE-SCP)	Relative position-velocity	✗	✗	✓	✗
Faghihi (2021)	Rapidly exploring Random Trees (RRT)	Relative position-velocity	✗	✗	—	✗
Proposed method	Stable Sparse RRT (SST)	**Relative orbital element**	✓	✓	✓	✓

Figure 1.4: Comparison of proposed research to existing sampling-based orbital motion planers in the literature.

immense need for motion planning algorithms to be fast and efficient in computing near-optimal, safe, and collision-free relative transfer trajectories for multi-spacecraft systems. This is where this research comes in.

The thesis statement is as follows:

> *This research develops an astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm framework to enable fast computation of fuel-efficient, collision-free, and near-optimal trajectories for spacecraft re-configuration in cluttered and perturbed environments - without requiring any initial guess of the solution.*

Figure 1.5 shows the different fields that have been combined in this work.

The characteristics that make the proposed algorithm unique are:

(1) **Overall motion planning framework**: As opposed to growing the sampling-based random tree in the traditional position and velocity space to achieve collision-free motion planning of spacecraft, the AIKMP algorithm is developed in the relative orbital element space of the deputy spacecraft around a chief spacecraft. This provides very useful geometric insight into relative spacecraft motion.

The AIKMP algorithm also takes advantage of a spacecraft's natural motion (coasting arcs)

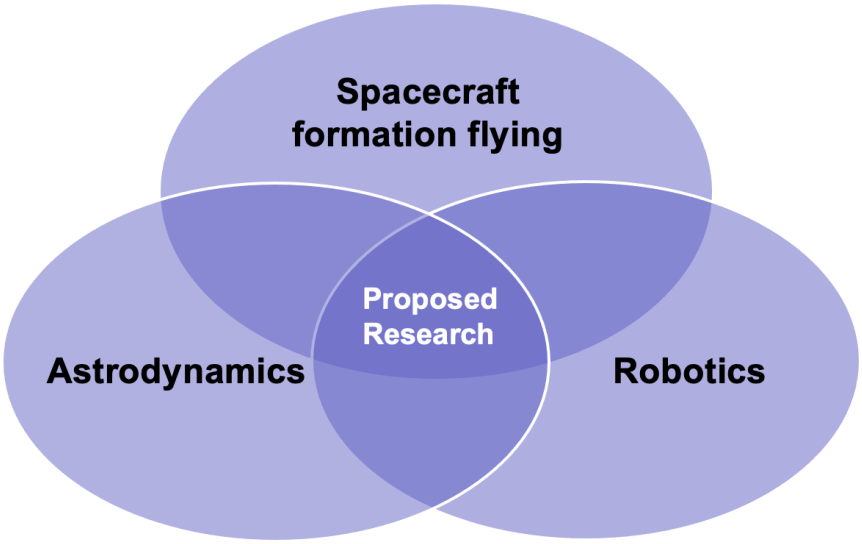


Figure 1.5: Venn diagram showing the different research fields that come together in the work presented in this thesis.

during tree extension, to compute a safe and fuel-efficient transfer trajectory from one node of the tree to another. This generates a single-step smooth final transfer solution without demanding additional post-processing of the solution, unlike the existing sampling-based orbital motion planning approaches as discussed in section 1.1.

(2) **Efficiency**: The AIKMP algorithm adapts tree pruning ideas from asymptotically near-optimal sampling-based robotic motion planners like the Stable sparse RRT (SST) algorithm and proposes an effective way of pruning a sampling-based random tree for astrodynamics applications. This helps the orbital motion planner remove nodes and edges from the tree that do not contribute to overall better solutions allowing the tree to maintain and store a sparse set of nodes. This not only makes the algorithm computationally fast but also significantly improves its storage efficiency.

(3) **Modularity**: This proposed sampling-based orbital motion planner is not specific to a single definition of spacecraft relative motion dynamics. Any relative motion dynamical model can be plugged in without any changes to the remaining modules of the algorithm.

Similarly, the steering law that drives the extension of the sampling-based random tree can also be updated without any changes to the rest of the algorithm. Moreover, the individual modules of the AIKMP algorithm can be adapted to any existing sampling-based motion planner (tree-based planners or graph-based planners) to improve their applicability to orbital relative motion planning.

The contributions of the proposed work to the literature are as follows:

(1) An astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm is proposed which is a single-step sampling-based kinodynamic approach to orbital motion planning problems as opposed to existing two-step sampling-based motion planning approaches in the literature [15, 121, 119, 74]. The algorithm introduces an astrodynamics-informed pruning module during the motion planning process to improve the overall computation efficiency and storage efficiency which is often an issue with such sampling-based orbital motion planners.

(2) With the pruning module excluded from the overall AIKMP algorithm, the same algorithm translates to an astrodynamics-informed modified version of the popular sampling-based Rapidly Exploring Random Trees (RRT). As opposed to a typical RRT-based orbital motion planner [61], this algorithm uses a cost function minimization for tree extension and also takes advantage of a spacecraft's natural motion (coasting arcs), to perform safe and fuel-efficient tree extension from one node of the tree to another. Moreover, the proposed algorithm also uses a *Goal bias* probability (< 1) to guide the tree extension towards the desired goal state. Such modifications make this astrodynamics-informed version of the RRT more suitable for fuel-efficient orbital motion planning applications.

(3) A novel extension of the linearized Lambert solution (LLS) [93]- closed-loop linearized Lambert guidance solution- is developed that allows a spacecraft to travel on a Lambert-like arc in the presence of perturbations such as Drag, J2, Solar Radiation Pressure (SRP) with

minimal targeting error. This extension of the LLS is applied to Spacecraft Formation Flying (SFF) problem, and it is shown that the resulting closed-loop LLS-guidance algorithm allows for stringent targeting accuracy in different SFF problems.

(4) An extension of the popular E/I vector separation method [56, 37] is derived for the case of drifting relative motion and is used to infuse passive collision avoidance capabilities in sampling-based orbital motion planners. The resulting motion planner performs tree extension in a more informed way such that collision avoidance constraints are satisfied by each node and edge in the sampling-based tree. This way, every transfer solution computed from the tree is guaranteed to be collision-free for the entire duration of the transfer in the presence of multiple static (static relative to a moving reference frame) and moving obstacles – a capability that is crucial for spacecraft formation flying applications.

1.3 Publications

The work done in this dissertation generated the following publications:

1.3.1 Journal Articles

(1) **T. Deka** and J. W. McMahon, Astrodynamics-Informed Kinodynamic Sampling-Based Motion Planning for Relative Spacecraft Motion, **Journal of Guidance, Control, and Dynamics**, Vol. 46, No. 12 (2023), pp. 2330-2345. The manuscript has been published.

(2) **T. Deka** and J. W. McMahon. Closed-loop Linearized Lambert Solution for Onboard Formation Control and Targeting. **Journal of Guidance, Control, and Dynamics**, 2023. The manuscript is under review.

(3) **T. Deka** and J. W. McMahon. Efficient astrodynamics-informed kinodynamic motion planning for relative spacecraft motion. **Advances in Space Research** (selected for publication in a special issue on Formation Flying at IWSCFF 2022 conference), 2022. The manuscript is under review.

(4) **T. Deka** and J. W. McMahon. Stochastic sampling-based motion planning for relative spacecraft motion with passive collision avoidance. **Journal of Guidance, Control, and Dynamics**, 2023. The manuscript is in preparation.

1.3.2 Conference Papers

(1) **T. Deka**, H. K. Sipowa, and J. W. McMahon. "Closed-loop linearized Lambert Solution (LLS) for on-board formation control and targeting." In **AAS/AIAA Astrodynamics Specialists Conference**, held virtually, Aug 9–12, 2020. Paper No. AAS 20-463.

(2) **T. Deka** and J. W. McMahon. "Astrodynamics-informed kinodynamic motion planning for relative spacecraft motion." In **AAS/AIAA Astrodynamics Specialists Conference**, held virtually, Aug 9–11, 2021. Paper No. AAS 21-593.

(3) **T. Deka** and J. W. McMahon. "Astrodynamics-informed sparse kinodynamic motion planning for relative spacecraft motion." In **International Workshop on Satellite Constellations & Formation Flying (IWSCFF)**, Milan, Italy, June 7–10, 2022.

(4) **T. Deka** and J. W. McMahon. "Efficient astrodynamics-informed kinodynamic motion planning for safe relative spacecraft motion." In **AAS/AIAA Astrodynamics Specialists Conference**, Charlotte, North Carolina, Aug 7–11, 2022. Paper No. AAS 22-664.

(5) **T. Deka** and J. W. McMahon. "Efficient astrodynamics-informed kinodynamic motion planning for safe relative spacecraft motion." In **45th Annual AAS Guidance, Navigation, and Control (GN&C) Conference**, Breckenridge, CO, Feb 2–8, 2023. Paper No. AAS 23-103.

1.3.3 Other Publications

These were the publications that were co-authored during the timeline of this dissertation. However, these did not add directly to the dissertation.

(1) H. K. Sipowa, J.W. McMahon, and **T. Deka**. "Distributed unscented information Kalman filter (uikf) for cooperative localization in spacecraft formation flying". In **AIAA SciTech Forum**, 2020. Paper No. AIAA-2020-1917.

(2) J.W. McMahon, N. Ahmed, M. Lahijanian, P. Amorese, **T. Deka**, et al., "Expert-Informed Autonomous Science Planning for In-situ Observations and Discoveries", In **IEEE Aerospace Conference**, 2022. Paper No. 2491 1.

(3) J.W. McMahon, N. Ahmed, M. Lahijanian, P. Amorese, **T. Deka**, et al., "REASON-RECOURSE Software for Science Operations of Autonomous Robotic Landers", In **IEEE Aerospace Conference**, 2023. Paper No. 2452 1.

Chapter 2

Stochastic sampling-based orbital motion planning

2.1 Review of sampling-based robotic motion planning

A few definitions as shown in Table 2.1, will be useful in understanding this section:

Parent node	:	A node that has other nodes (child nodes) growing from them.
Leaf node	:	A node that is Parent to no other node.
V_{active}	:	All the nodes of the tree that take part in tree extension.
$Cost(x)$	:	Cost to reach node x from the start node (x_0).
x_{rand}	:	Sampled random node from the obstacle-free state space X.
δ_{near}	:	A radius that defines a neighborhood around x_{rand} for nearest neighbor selection. This can be also called the 'Nearest neighbor radius'.
$x_{nearest}$	:	The nearest node to x_{rand} within δ_{near} in V_{active} that has the best path cost from the start node.
x_{new}	:	Final node reached after extending tree from $x_{nearest}$ towards x_{rand}.
Goal bias	:	It is a probability used to bias the random exploration towards the goal state ($0 \leq Goal\_bias \leq 1$). For example: if goal bias is 0.1, once every 10 iterations $x_{rand} = x_{goal}$, and the tree attempts to extend towards the goal state. The lower the goal bias, the higher the random exploration by the motion planner and vice versa.

Table 2.1: Important terminologies used in algorithm description

.

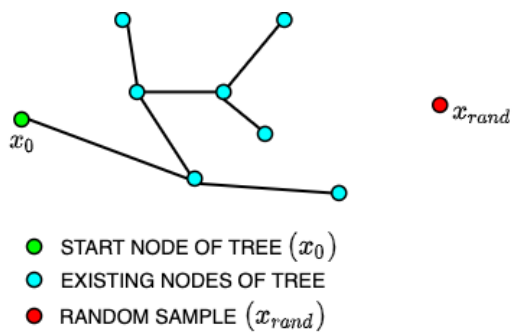
The basic idea behind most sampling-based planners can be described in 3 major steps as can be seen in Figure 2.1:

(1) Random sampling: A random collision-free state/node (x_{rand}) is sampled in the defined state space.

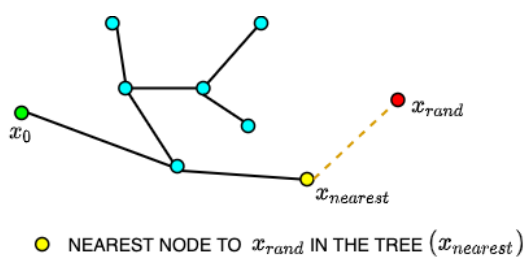
(2) Choosing the nearest node in the tree: The nearest node to x_{rand} in the existing tree ($x_{nearest}$) is chosen for extending the tree.

(3) Tree Extension: The existing tree is extended from $x_{nearest}$ towards x_{rand} (either by using a user-defined step size or by using a steering law) and the new node x_{new} and the new edge connecting nodes $x_{nearest}$ and x_{new} are added to the tree provided the node x_{new} is collision-free.

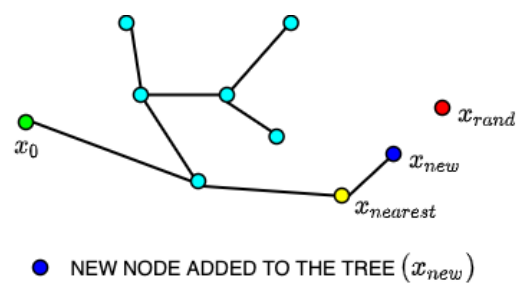
These steps are repeated till the goal state (x_{goal}) is reached. This way the planners connect nodes with intermediate trajectories building a tree of feasible trajectories. Most sampling-based plan-



(a) Step 1: Randomly sampling node x_{rand} in the collision-free state-space



(b) Step 2: Choosing the nearest node $x_{nearest}$ of the random sample x_{rand} in the existing tree



(c) Step 3: Extending the tree from $x_{nearest}$ towards x_{rand} and adding new node x_{new} to the tree

Figure 2.1: Basic concept behind sampling-based motion planners

ning algorithms differ from each other by slight changes in the way they implement these basic steps (shown in Figure 2.1), but these little changes contribute to significant differences in the overall performance of the algorithms. For example, Figure 2.1 explains the working of the popular

Rapidly Exploring Random Tree (RRT) algorithm [85]. The RRT was among the first probabilistically complete sampling-based planners developed that provides a fast implementation meaning that if such a solution exists, the algorithm returns a solution with a probability converging to one as the number of samples grows to infinity. There is a modified version of the RRT algorithm that is asymptotically complete and asymptotically optimal, called the RRT* algorithm. The RRT*, instead of directly picking the nearest node of x_{rand} as $x_{nearest}$ in Step 2, picks the nearest node (from multiple neighboring nodes within a certain radius) with the best cost from the starting node as $x_{nearest}$ [76]. To improve the cost of the overall computed solution, the RRT* also performs re-wiring of the edges of the tree once a new node is added. There is another popular probabilistically complete sampling-based algorithm (graph-based instead of being tree-based) called the *Probabilistic Road Maps* (PRMs) [77]. In this method, instead of sampling a single x_{rand} in Step 1, the algorithm samples multiple random nodes at once and connects each random node with all nodes close to it (within a certain radius) creating a graph structure. This graph structure can then be queried to find the path between any two nodes in the graph.

These above-mentioned sampling-based algorithms and their variations provide fast motion-planning solutions and are widely used in the field of robotics [10, 76, 92]. However, one common drawback of many basic sampling-based motion planners such as RRT and PRM is that they are computationally very inefficient due to storing too many nodes in the trees/graphs. To overcome this, Li et. al. developed an asymptotically near-optimal kinodynamic motion planning algorithm called the *Stable Sparse RRT* (SST) algorithm. SST is computationally very efficient due to its sparse nature as it performs additional steps for pruning the tree that help in removing sections (nodes and/or edges) of the tree that do not contribute to overall better solutions [89]. Another highlight of this algorithm is that it does not require a steering function for tree extension as solving for a steering law can be challenging particularly when dealing with complex dynamical systems. Instead, the algorithm picks a random control and a random propagation time to drive the $x_{nearest}$ state towards the x_{rand} state for building the tree. The SST algorithm is also computationally very efficient due to its sparse nature as it performs an additional step for pruning the tree that

helps in removing sections (nodes and/or edges) of the tree that do not contribute to overall better solutions.

Directly extending an asymptotically-optimal motion planner like the RRT* to astrodynamics applications will be computationally very expensive than using an efficient asymptotically near-optimal SST algorithm due to the intensive tree-rewiring in every step of the RRT* algorithm which essentially converts to computing multiple control solutions for the spacecraft motion at every step. As a result, this work takes inspiration from an efficient robotic sampling-based motion planner like the SST algorithm and presents an astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm. Although there is a compromise in terms of asymptotic optimality, the overall computation efficiency gained by adapting ideas from SST is significant as will be demonstrated in the upcoming chapters. The AIKMP algorithm describes the sufficient number of steps that can be adapted from a sampling-based robotic motion planner to achieve fuel-efficient and collision-free sampling-based motion planning in astrodynamics applications. This algorithm iteratively improves on its computed transfer solutions and thus holds the potential of computing near-optimal transfer solutions given a sufficient number of algorithm iterations – without needing an initial guess of the solution.

2.2 AIKMP algorithm description

As discussed before, the high-level structure of the AIKMP algorithm is inspired by the SST algorithm. However, to make such a sampling-based approach better suited for orbital motion planning applications, instead of growing a sampling-based random tree in traditional relative position and velocity space (as historically done in sampling-based orbital motion planning), the AIKMP algorithm is a kinodynamic approach and is developed in the relative orbital element space of the deputy spacecraft around a chief spacecraft. This way, each node in the tree is represented by a relative orbit that provides visual insight into the orbit geometry and hence into relative motion. Using relative orbital elements is also beneficial because, unlike the fast-varying cartesian relative state vectors within a relative orbit, there is only one relative orbit element (that depends on the

true anomaly difference) that needs to be solved to find the relative position of a spacecraft. The AIKMP algorithm also takes advantage of a spacecraft's natural motion (coasting arcs) during tree extension, to compute a safe and fuel-efficient transfer trajectory from one node of the tree to another. This ensures the smoothness of the final transfer solution for relative spacecraft motion.

Figure 2.2 demonstrates the flow of the AIKMP algorithm. All the colored blocks in the figure represent major sub-routines of the algorithm. The colored blocks on the left depict the basic three steps involved in any sampling-based motion planner (as discussed in Section 2.1). The colored blocks on the right correspond to the pruning module which helps in improving the computation efficiency of the motion planner. The various steps involved in the AIKMP algorithm are discussed

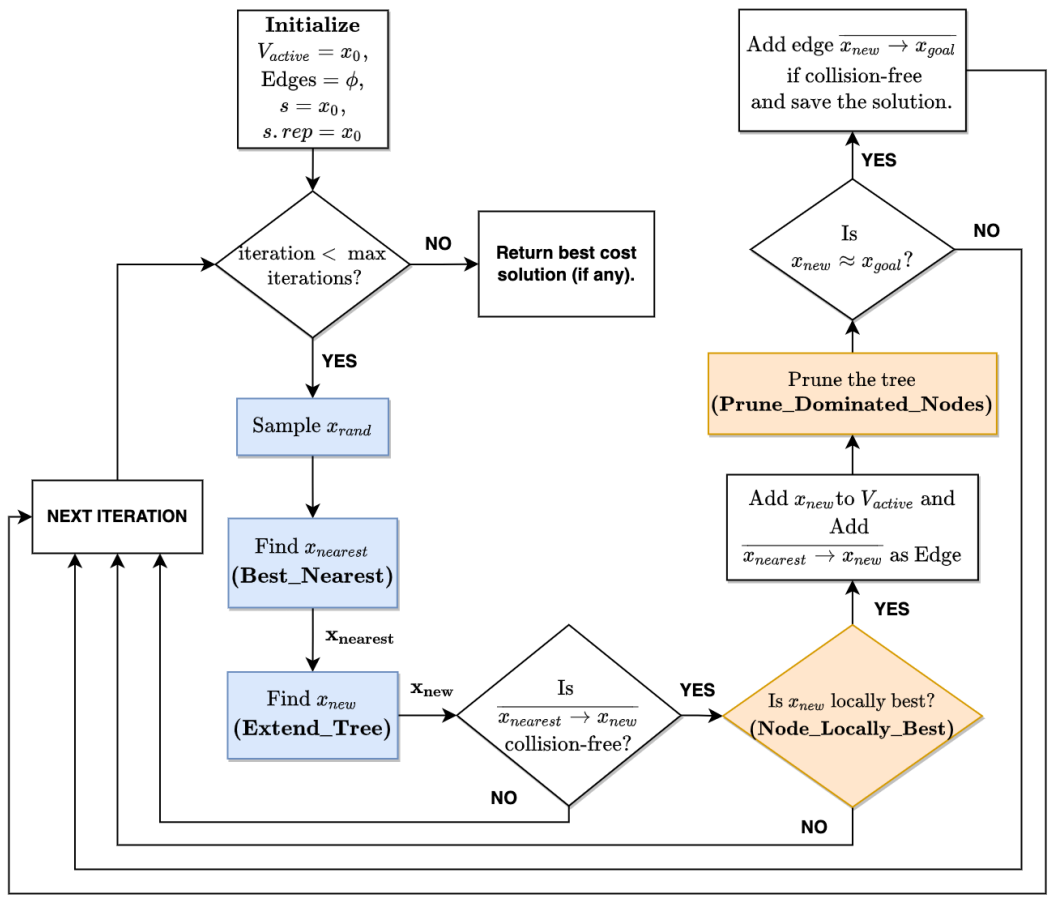


Figure 2.2: Flowchart showing the basic steps involved in the astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm (inspired by the SST algorithm).

here in detail:

2.2.1 Step 1 - Tree Initialization :

This step initializes the tree with the starting node x_0 and the edges of the tree are initialized as an empty set. At this stage, since x_0 is the only node in the tree, it is set as an active node (V_{active}) that takes part in tree extension.

2.2.2 Step 2 - Random Sampling (sampling $\mathbf{x_{rand}}$):

In this step, the algorithm samples a random node x_{rand} from the obstacle-free region in the given state space. Now as mentioned before, the AIKMP algorithm is developed in the relative orbital element space rather than in the relative position and velocity space. Thus, in this step, the AIKMP algorithm randomly samples a relative orbit around the chief with the relative orbital element set (δOE) described as:

$$\delta OE = OE_d - OE_c = (\delta a, \delta\theta, \delta i, \delta q_1, \delta q_2, \delta\Omega) \tag{2.1}$$

Here, OE_d and OE_c refers to deputy's and chief's orbital elements respectively, δa is the relative semi-major axis, $\delta\theta$ is the relative true latitude angle, δi is the relative orbit inclination angle, $\delta\Omega$ is the relative argument of the ascending node, and $\delta q_1 = \delta e_X$ and $\delta q_2 = \delta e_Y$ where relative eccentricity vector $\vec{\boldsymbol{\delta e}} = (\delta e_X, \delta e_Y)^T$ [111]. Details of the choice of spacecraft's relative orbit element description are given in Appendix A.

Now, although sampling-based motion planning algorithms are based on the random exploration of the state space, a very large state space for random exploration may involve the algorithm in too much randomness and this may hinder its convergence to the goal. Thus, given this along with the need for fuel efficiency for the orbital motion planning problem at hand, restricting the random search space to stable closed relative orbits around the chief spacecraft seems to be the most intuitive solution. Transferring to a closed relative orbit will allow the deputy spacecraft to spend time coasting around the chief, whenever necessary, without the fear of drifting in the random-search state space to compute the relative motion transfer solution.

In classical two-body orbital motion, the condition on two inertial orbits to have a stable

closed relative orbit is that their orbit energies must be equal which in terms of orbital elements means that the difference in their semi-major axis (a) should be zero [110].

$$\delta a = 0 \tag{2.2}$$

Thus, for the randomly explored relative orbits in the implementations in this chapter, the algorithm always explores stable closed relative orbits around the chief.

Along with randomly exploring the state space, the sampling-based tree also needs to connect to the goal state to find the transfer solution. As a result, to sample the relative orbital elements other than δa, the algorithm either bias the random sampling more towards the goal state (by using the *'Goal_bias'* probability that sets $x_{rand} = x_{goal}$) or in an informed way, they are sampled from suitable distributions such that the random exploration state space is bound to be "close" to the initial and final relative orbits to avoid introducing too much randomness to the algorithm. For implementation purposes, the δq_i are picked from a uniform distribution in the range $[-2\Delta q_i \quad 2\Delta q_i]$, where $\Delta q_i = \delta q_{i,\text{initial}} - \delta q_{i,\text{final}}$ of the deputy's desired relative transfer and $i = 1, 2$. Since inclination change maneuvers are expensive, the δi values for random exploration are assumed to be small and hence are picked from an exponential distribution in a suitable range as described in Table 2.4. The relative position and velocity bounds are picked independently such that the overall state space used for random exploration remains within linearization bounds of the linear relative motion dynamics (assumed to be within 5% of the chief's nominal state). These bounds on relative position and velocity are particularly useful to bind the intermediate transfer orbits/arcs used for transfers from one relative orbit to another that may not obey the relative OE bounds.

It is to be noted that such assumptions about limiting the random search space of the algorithm can be completely avoided. However, it is important to realize that more randomness may require more time for the convergence of the algorithm. Since the goal of using the orbital motion planner is to compute a feasible fuel-efficient and collision-free transfer, it is of utmost importance that the algorithm runs in a reasonable amount of time. Any description of relative orbital elements can be used with the AIKMP algorithm for random exploration and any necessary

constraints can be imposed on any element as required. Defining the state space for random exploration is critical and depends on the mission requirements.

2.2.3 Step 3 - Best_Nearest (finding $\mathbf{x_{nearest}}$):

In this step, the node with the best cost in the existing tree is picked as $x_{nearest}$ that is close to the newly picked random node x_{rand} and this node is used to extend the tree towards x_{rand}.

To compute the "closeness" between two relative orbits, the algorithm uses a non-dimensional weighted distance measure. The weighted distance (w) between two sets of relative orbital elements δOE_1 and δOE_2 is defined as:

$$w(\delta OE_1, \delta OE_2) = \sqrt{(\frac{\delta a_2 - \delta a_1}{a_{chief}})^2 + \alpha(\delta i_2 - \delta i_1)^2 + (\delta q_{1,2} - \delta q_{1,1})^2 + (\delta q_{2,2} - \delta q_{2,1})^2 + \beta(\delta\Omega_2 - \delta\Omega_1)^2} \tag{2.3}$$

where α and β are scaling factors that are assumed to be equal to 1 for demonstration purposes but can take any appropriate values.

The "*Cost*" of a node x here is defined as the total fuel required to reach node x from the starting node x_0 of the tree and is defined as follows:

$$Cost(x) = \sum_{X=x_0}^{x^-} \Delta v_X, \tag{2.4}$$

Here Δv_X refers to the impulsive Δv required to transfer from node/relative orbit X and $X = x^-$ refers to the node right before node x that is used to transfer to node x.

To compute $x_{nearest}$, after the random sampling step, the algorithm finds all the relative orbits within a radius of δ_{near} of x_{rand} in V_{active} for extending the tree. Let's call this set of relative orbits, x_{NEAR}. If no relative orbit is found in the δ_{near} radius neighborhood (i.e. if x_{NEAR} is empty), the algorithm selects the relative orbit closest to x_{rand} according to Eq. 2.3 as $x_{nearest}$. Otherwise, if a single relative orbit is found in x_{NEAR}, this is selected as the $x_{nearest}$. Else if multiple relative orbits are found in x_{NEAR}, the algorithm picks the relative orbit with the best path cost in x_{NEAR} (computed according to equation 2.4) and calls it $x_{nearest}$. This helps the algorithm, make sure that the nodes with the best cost take part in the tree extension.

For the purpose of this work, the δ_{near} value was carefully chosen such that it is less than the weighted distance (equation 2.3) between the initial and the final relative orbit.

$$\delta_{near} < w(x_{start}, x_{goal}) \tag{2.5}$$

This makes sure that whenever the goal relative orbit is targeted directly ($x_{rand} = x_{goal}$), the algorithm doesn't always pick x_{start} (corresponding to the initial relative orbit) as the nearest node (as $\text{Cost}(x_{start}) = 0$) and also give other randomly explored nodes a chance to connect to the goal. Equation 2.5 is a valid assumption because, in the first iteration, the AIKMP algorithm attempts to extend the tree from the starting node to the final goal node directly. Hence, this solution is already stored by the tree for comparison with other solutions computed by the algorithm.

2.2.4 Step 4 - Extend_Tree (computing $\mathbf{x_{new}}$):

Once x_{rand} and $x_{nearest}$ are picked, a random time of flight ($TOF_{nearest\_rand}$ which is less than the total time for the overall transfer TOF_{total}) is considered for this intermediate transfer. This routine attempts to extend the tree from $x_{nearest}$ towards x_{rand} for the picked time of flight by using a steering law. The node reached by this tree extension (x_{new}) is added as a new node to the tree provided it is collision-free. Note that the x_{rand} node itself is not directly added to the tree but the resulting node from tree extension from $x_{nearest}$ towards x_{rand} (x_{new}) is.

The AIKMP algorithm uses the Universal variable method of solving Lambert's problem [131] along with a particle-swarm-optimization-based [33, 133] Lambert burn strategy to find a Lambert transfer solution to extend the tree using minimum fuel. Any computed trajectory that passes through the obstacle space is discarded and the random sampling and tree extension steps are repeated to make sure that the intermediate trajectories constructed by the algorithm are collision-free.

Lambert's guidance solution as steering law for AIKMP tree extension: The SST algorithm suggests picking a random control and applying it for a random amount of time to extend the tree [89]. However, this does not work very well in astrodynamics applications (as will

be demonstrated later in Section 2.3). As a result, due to its flight heritage on the space shuttle missions [67], a traditional Lambert's guidance solution is used as the steering law for the AIKMP algorithm to demonstrate how this can be used to achieve orbital motion planning. However, it is to be noted that, the steering law for the AIKMP algorithm is not limited to using the Lambert solution as will be demonstrated in the following chapter.

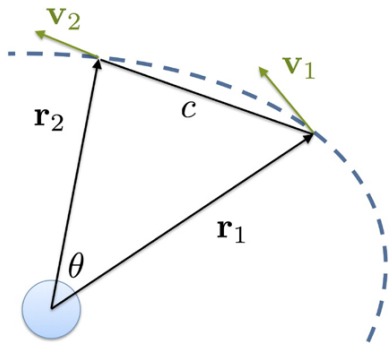


Figure 2.3: Lambert's Problem

Lambert's problem is concerned with determining the transfer orbit between two given inertial position vectors for a given time of flight under Keplerian dynamics (Figure 2.3). It is one of the most studied questions in the astrodynamics community. Many different approaches have been developed over the years to solve Lambert's problem [82, 17, 66, 32, 131]. In this work, the Universal variable method [131] is used along with a particle-swarm-based optimization strategy [33] to compute a minimum change in velocities required to achieve a desired transfer between two position vectors.

Particle swarm optimization-based Lambert tree extension strategy: The objective of using this strategy is to find a pair of true anomalies in the initial and final deputy orbits (f_1 and f_2 respectively) that require minimum fuel (total Δv) for the transfer between the two orbits. The Δv required for the transfer can be computed by summing the magnitude of change in velocity required by the spacecraft to be on the transfer orbit from its initial orbit (Δv_1) and the magnitude of change in velocity required by the spacecraft to be on the target orbit from its

transfer orbit (Δv_2), as shown by equations 2.6, 2.7 and 2.8.

$$\Delta v_1 = |\vec{v_{t1}} - \vec{v_1}| \tag{2.6}$$

$$\Delta v_2 = |\vec{v_2} - \vec{v_{t2}}| \tag{2.7}$$

$$\Delta v = \Delta v_1 + \Delta v_2 \tag{2.8}$$

Here, $\vec{v_1}$ is the velocity of the spacecraft in its initial orbit right before executing the initial impulsive transfer burn, $\vec{v_{t1}}$ is the velocity in the transfer orbit right after executing the initial burn, $\vec{v_2}$ is the velocity of the spacecraft in its final orbit after executing the final impulsive transfer burn, and $\vec{v_{t2}}$ is the velocity of the spacecraft in its transfer orbit right before executing the final burn.

A particle-swarm-based non-linear optimization technique [33, 133] is used to compute the pair of true anomalies (f_1 and f_2) in the initial and final orbits respectively, that will give a minimal Δv Lambert transfer solution between the two orbits. The values for the true anomalies f_1 and f_2 are bound between $[0, 2\pi)$. Specifically, f_1 is bounded between $[f_{snear}, f_{snear} + 2\pi - 0.1]$ radians and f_2 is bounded between $[f_{srand} - 2\pi + 0.1, f_{srand}]$ radians. Here f_{snear} is the state in the initial orbit that is already connected to the existing random tree and f_{srand} corresponds to a randomly picked state in the target orbit. To get around the problem of local minima in this optimization problem, 50 swarm particles are used and 1000 iterations are allowed for the swarm to converge to an optimal solution. This approach explores the entire solution space using the swarm particles and enables sharing of information among them. This way, each particle is always aware of the best solution explored by any particle in the swarm, and with every iteration, each particle moves towards the best solution explored until that iteration. This process continues until the entire swarm converges to a single solution or the maximum number of iterations is reached. The solution with the lowest cost is then picked as the final solution. It is important to note that a swarm-based method like particle swarm optimization (PSO) gets very computationally expensive depending on the number of optimization variables [33, 30, 128]. As a result, two different strategies of tree extension are tested here:

(1) Full PSO tree extension: In this strategy, the PSO routine optimizes for both f_1 and f_2 as

explained above.

(2) Single-orbit PSO tree extension: In this strategy, the tree extension uses an initial coasting arc, propagated for a randomly picked time of flight, to determine the f_1, and the swarm optimization is used to compute the optimal f_2 for the transfer.

The findings from using both these strategies will be elaborated later in the 2.3 section. One of the drawbacks of any Lambert algorithm is the difficulty of finding a solution for 180° transfers, as the transfer plane is not defined in this case. To get around this problem, coasting arcs are used (that require no additional fuel) and Lambert burns are performed only when $\theta \leq 170.9^o$ or $\theta \geq 180.1^o$, where θ is the angle of transfer.

It is also to be noted that, using higher numbers for the particle-swarm optimization parameters (swarms particles and/or a number of iterations) will only improve the computed solution. Since this chapter presents a proof of concept, the above-described values are chosen for the parameters in the interest of computation time. Also, in this work, both long and short-angle Lambert transfers are considered while picking the transfer with the least fuel requirement, and the intermediate transfers use single-revolution Lambert transfers. Although, the overall solutions can be multiple-revolution transfers.

In the work presented here, the spacecraft's relative motion planning problem is being solved by exploring the relative orbits of the deputy around the chief, and for that purpose, Lambert's solver has been used to aid the AIKMP algorithm in tree extension. Since the Lambert solver requires the deputy's initial and final inertial position vector to compute the required transfer, the tree extension step of the AIKMP algorithm here uses transformations from relative orbit elements to inertial states as required. Specifically, once x_{rand} is picked for tree extension from $x_{nearest}$, a random time of flight is considered for the transfer ($TOF_{nearest\_rand}$) and both these deputy relative orbits (x_{rand} and $x_{nearest}$) are used to compute the corresponding deputy's inertial orbits at the initial and final time instants. The swarm-based iterative-search Lambert burn strategy is then used to find the pair of true anomalies (f_1 and f_2) in the two deputy orbits respectively to extend

the tree using minimum fuel. The computed optimal f_2 defines the target state in the final orbit for the tree extension step. This state is used to compute the new node/relative orbit x_{new} in the tree that is reached during this iteration of tree extension.

Relative motion dynamics: To keep things simple initially for demonstration purposes, the linear model of orbital relative motion dynamics described by the Hill-Clohessy-Wiltshire (HCW) equations has been used. This model describes the relative motion of a deputy spacecraft near a chief that is in a circular orbit [32, 131]. These equations of motion are valid for circular chief orbits as long as $|\vec{\rho}|/|\vec{r_c}| << 1$ and show the characteristic decoupling of the motion in the radial-tangential direction (x-y plane) from the normal direction (z plane). The HCW equations are given as:

$$\begin{aligned} \ddot{x} - 2n\dot{y} - 3n^2 x &= 0 \\ \ddot{y} + 2n\dot{x} &= 0 \\ \ddot{z} + n^2 z &= 0 \\ n &= \sqrt{\frac{\mu}{a^3}} \end{aligned} \tag{2.9}$$

where, x, y, and z refer to the deputy states relative to the chief as expressed in the chief-centered LVLH frame and n is the chief's mean motion.

Many other collision-free spacecraft formation flying reconfiguration schemes have been developed that are specifically based on the Hill-Clohessy-Wiltshire (HCW) model [26, 52]. However, it is to be noted that the application of the AIKMP algorithm is not limited to this description of spacecraft relative motion and can be applied to any description of spacecraft relative motion as will be seen in the following chapters.

Adding edges to the tree by Extend_Tree: As explained before, the idea here is to find the transfer trajectory from point $A1$ in Orbit A to point $B1$ in Orbit B as shown in Figure 2.4. But by using the particle-swarm-based Lambert solver, suppose it was found that transferring from point $A2$ in the first orbit to point $B2$ in the second orbit using Lambert burns requires the least fuel. Thus the Extend_Tree routine will add three edges to the tree (provided these edges do

not intersect the obstacle space), i.e. edges $\overline{A1 \rightarrow A2}$, $\overline{A2 \rightarrow B2}$ and $\overline{B2 \rightarrow B1}$. Out of these edges, edges $\overline{A1 \rightarrow A2}$ and $\overline{B2 \rightarrow B1}$ are coasting arcs (shown in black markers in Figure 2.4) that do not require additional fuel for the transfers, thus, contributing to the minimum fuel requirement for the algorithm.

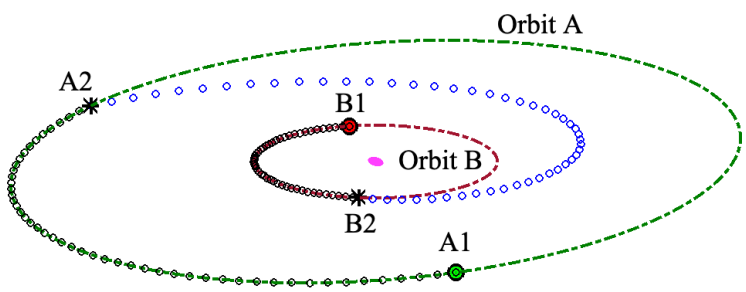


Figure 2.4: Collision-free edges added to the tree after every iteration of the AIKMP algorithm through the Extend_Tree routine

Once the final node (x_{new}) has been reached after the Lambert transfer and the edge/edges connecting $x_{nearest}$ to x_{new} have been found to be collision-free, all the computed edges are added to the existing tree and the orbit that x_{new} belongs to is added to V_{active}. This randomly explored orbit can now take part in the next iterations of tree extension.

Also, it is worth noting that, this strategy of tree extension along with the *Goal_bias* makes sure that the random-search-based tree is always able to find a collision-free transfer solution to the goal as the Lambert solution always attempts to connect the existing tree to the goal with probability = *Goal_bias*.

2.2.5 Steps 5 and 6 - Node_Locally_Best and Prune_Dominated_Nodes, (represented by the colored blocks on the right in Figure 2.2):

These two routines prune/delete the "bad nodes" from the tree which are nodes that do not contribute to a better-cost transfer solution. Active tree pruning is not a part of the work presented in this chapter meaning that every new node that is computed by the tree extension

step is considered the locally best node. The pruning modules will be considered in the following chapter (Chapter 3) where the focus will be more on improving the computational efficiency of this algorithm.

2.3 Implementation and simulation results

2.3.1 Collision-free spacecraft reconfiguration using AIKMP

Here a relative orbit transfer scenario is set up that comprises two spacecraft: the chief spacecraft and the deputy spacecraft. In this scenario, the deputy spacecraft attempts to safely reconfigure itself from a state in an initial relative orbit to a state in a final relative orbit around the chief as shown in Figure 2.5. The chief is assumed to be in a circular orbit. The deputy's transfer must be collision-free with the chief or any other obstacle present in the scenario (say another spacecraft in a formation or simply a "keep-out zone") and the deputy should also use less fuel during the transfer. It is to be noted that, since the initial and the final configuration of

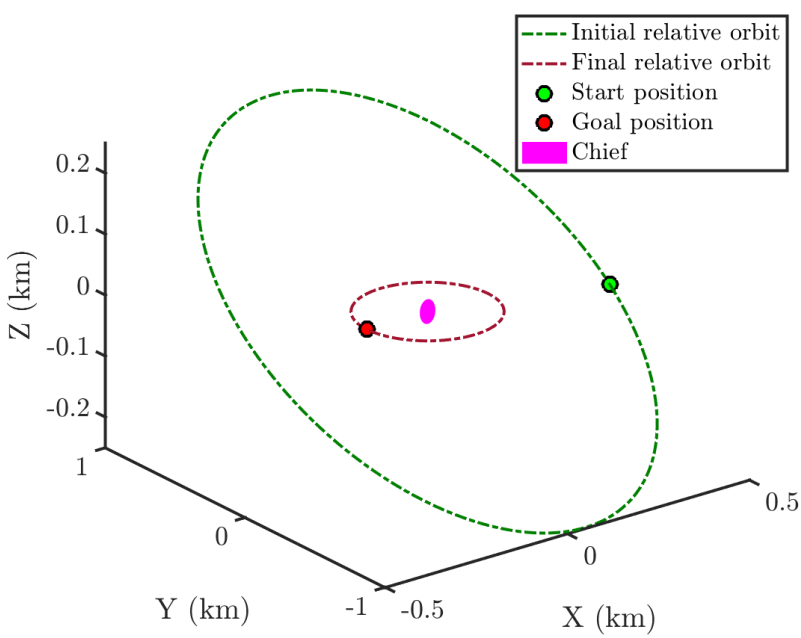


Figure 2.5: Relative orbit transfer scenario description.

the spacecraft are fixed in this implementation scenario, the total time of flight (TOF_{total}) for the

overall transfer will be constrained as shown in equation 2.10.

$$TOF_{total} = TOF_{start\_goal} + N * \text{Period\_chief} \tag{2.10}$$

Here, TOF_{start_goal} refers to the minimum time required to transfer from the defined initial relative state to the final relative state, N is a whole number constrained such that the TOF_{total} is within a maximum time-of-flight (TOF_{max}) allowed for the transfer.

Due to having a fixed initial and final spacecraft relative configuration in this scenario, the allowable times for the individual Lambert transfers ($TOF_{nearest\_rand}$, specifically TOF_{f1_f2}) for tree extension will also be constrained. Here (f_1, f_2) refers to the pair of true anomalies in the initial and final orbits respectively, chosen for the minimum fuel transfer between the two orbits. Whenever the goal relative orbit is targeted for tree extension, the $TOF_{nearest\_rand}$ which is now called $TOF_{nearest\_goal}$ will be constrained by the TOF_{total} as:

$$TOF_{nearest\_goal} = TOF_{total} - TOF_{start\_nearest} \tag{2.11}$$

where $TOF_{start\_nearest}$ is the total time taken by the tree to reach the $x_{nearest}$ node from the starting node of the tree. For all the other cases, when a random relative orbit is targeted for tree extension, the $TOF_{nearest\_rand}$ can be picked randomly such that TOF_{max} is not violated.

The classical orbital elements (COEs) of the chief and the initial and final COE differences of the deputy spacecraft (δCOE_i and δCOE_f respectively) are shown in Table 2.2, the initial and final relative states of the deputy spacecraft are shown in Table 2.3, and the state space bounds along with initialization of the various parameters used by both the SST algorithm and the AIKMP algorithm are described in Table 2.4.

To demonstrate the performance of the AIKMP algorithm, multiple test cases have been considered for the above-mentioned transfer scenario.

Case 1: Keplerian dynamics + Collision avoidance with chief:

In this case, there is no other obstacle in the scenario other than the chief spacecraft itself (as shown in Figure 2.5).

Chief COEs	Values	Deputy δCOE_i	Values	Deputy δCOE_f	Values
a	15000 km	δa	0 km	δa	0 km
e	0	δe	3.333×10^{-5}	δe	1.111×10^{-5}
i	50°	δi	$9.549 \times 10^{-4^o}$	δi	0°
Ω	10°	$\delta\Omega$	0°	$\delta\Omega$	0°
ω	10°	$\delta\omega$	0°	$\delta\omega$	0°

Table 2.2: Orbital elements of chief and deputy spacecraft.

Deputy initial relative state	Values	Deputy final relative state	Values
x (km)	4.999×10^{-1}	x (km)	-1.666×10^{-1}
y (km)	3.138×10^{-4}	y (km)	2.529×10^{-10}
z (km)	-4.333×10^{-2}	z (km)	-2.457×10^{-12}
$\dot{x}$ (m/s)	5.374×10^{-5}	$\dot{x}$ (m/s)	-3.724×10^{-11}
$\dot{y}$ (m/s)	-3.436×10^{-1}	$\dot{y}$ (m/s)	1.145×10^{-1}
$\dot{z}$ (m/s)	-8.461×10^{-2}	$\dot{z}$ (m/s)	-4.697×10^{-12}

Table 2.3: Initial and Final desired relative states of deputy spacecraft.

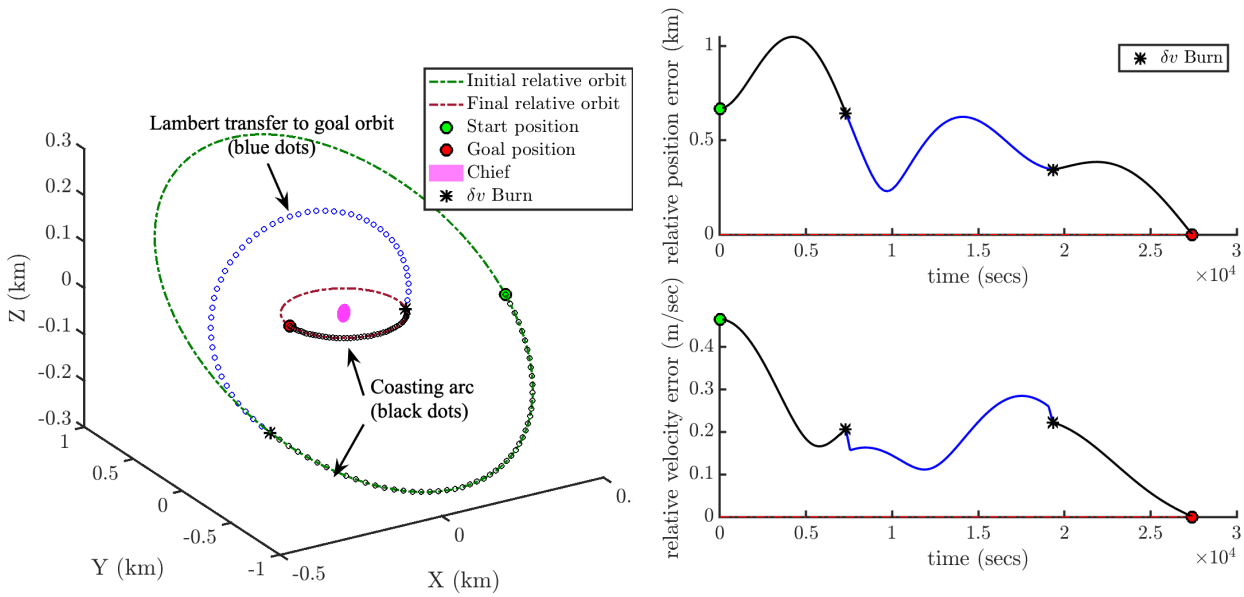
Relative position-velocity bounds	Values	Other parameters (in relative position-velocity space)	Values
x, z (km)	$[-0.5 \quad +0.5]$	δ_{near_SST}	0.005
y (km)	$[-1.5 \quad +1.5]$	δ_{best_SST}	0.0001
$\dot{x}$, $\dot{y}$, $\dot{z}$ (m/s)	$[-0.5 \quad +0.5]$		
Relative OE bounds (for random exploration)	**Values**	**Other parameters (in OE space)**	**Values**
δa (km)	0	δ_{near}	1.6×10^{-5}
δi (in rad)	$[10^{-8} \quad 10^{-3}]$	δ_{best}	0
δq_1	$[-2\Delta q_1 \quad 2\Delta q_1]$	Others	Values
δq_2	$[-2\Delta q_2 \quad 2\Delta q_2]$	*Goal bias*	0.1
$\delta\Omega$ (in rad)	0	Obstacle tolerance (sphere)	radius 20m

Table 2.4: Different state space bounds and algorithm parameters used in AIKMP algorithm and the basic SST algorithm implementations. Here $\Delta q_i = q_{i,\text{initial}} - q_{i,\text{final}}$

.

Here, first, a 2-impulse Lambert's solution has been implemented to achieve this transfer. The result of this implementation is shown in Figure 2.6. After that, the AIKMP algorithm is

implemented in the same scenario using both the tree extension strategies as discussed in section 2.2.4. The results obtained are as shown in Figures 2.7 and 2.8. In these figures, the black color refers to coasting arcs, the cyan color refers to Lambert transfer to a randomly explored orbit and the blue color refers to the Lambert transfer to the goal orbit.



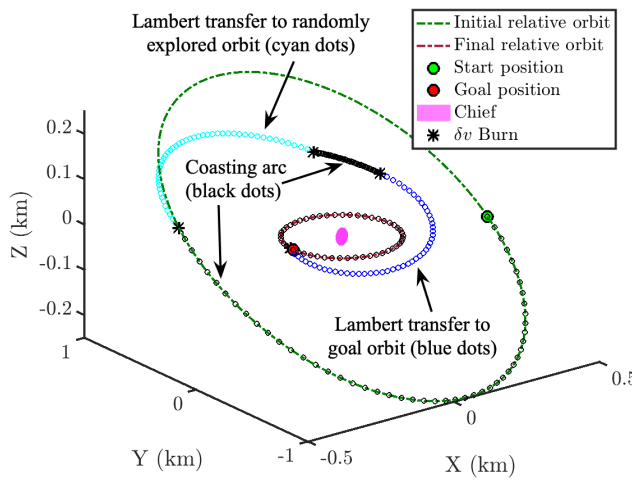
(a) 2-burn Lambert transfer solution

(b) Relative state error of solution trajectory with respect to the goal state.

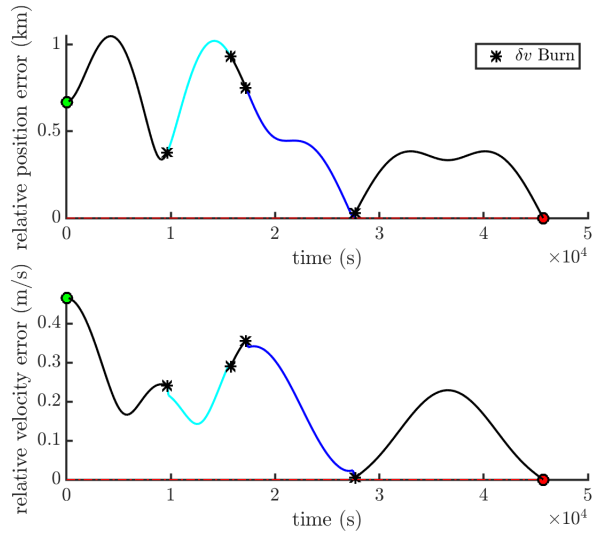
Figure 2.6: Lambert solution to transfer deputy from Start state to Goal state in Case 1. The total Δv required for the transfer is 15.05 cm/s.

From Figures 2.7 and 2.8, it is observed that the AIKMP transfer solution computed by the single-orbit PSO tree extension strategy requires lesser fuel (total Δv= 13.24 cm/s after 1500 iterations) as compared to the AIKMP solution computed using the full-PSO strategy (total Δv= 13.57 cm/s after 1000 iterations). This highlights an important property of the AIKMP solution that even though using the full-PSO gives better cost intermediate transfer arcs, the cost of the overall transfer solution depends on the random exploration nature of the algorithm and hence the number of iterations for which the algorithm was allowed to run.

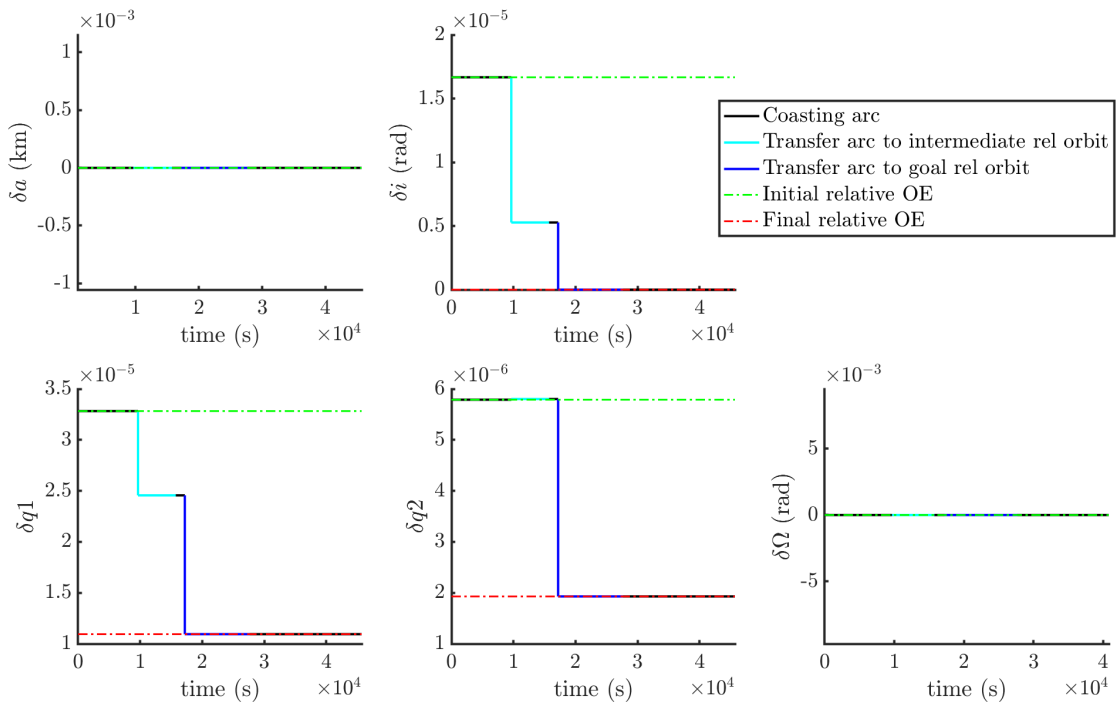
Comparing results from Figures 2.6 and 2.8, it is also observed that the AIKMP algorithm, in fact, was able to compute a transfer trajectory that required lesser fuel (total Δv= 13.24 cm/s) than



(a) Transfer solution in 3D relative position space.

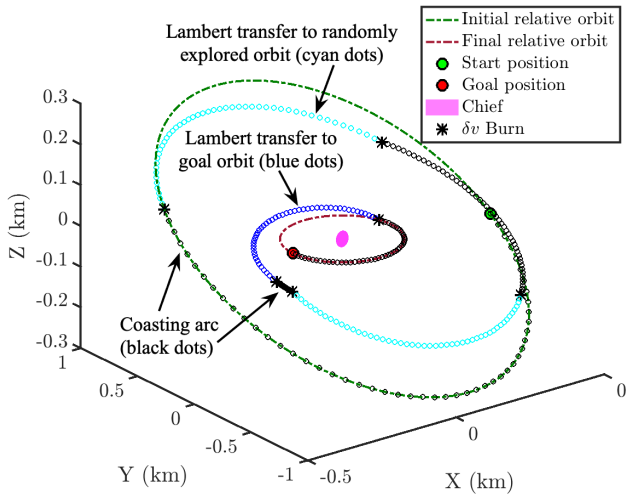


(b) Relative state error of solution trajectory with respect to the goal state.

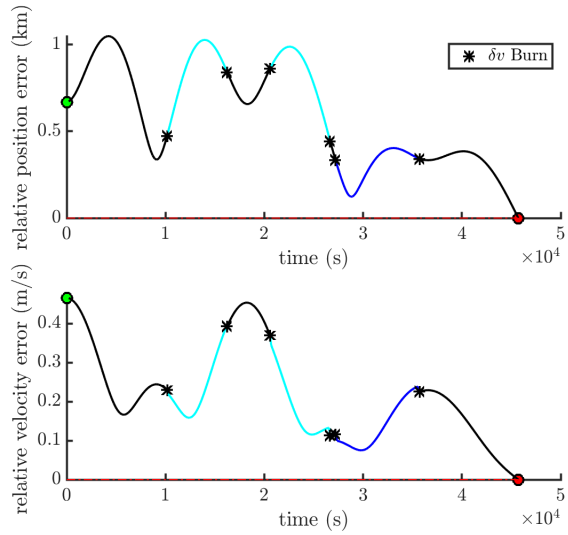


(c) Relative orbital elements vs time of the final AIKMP solution

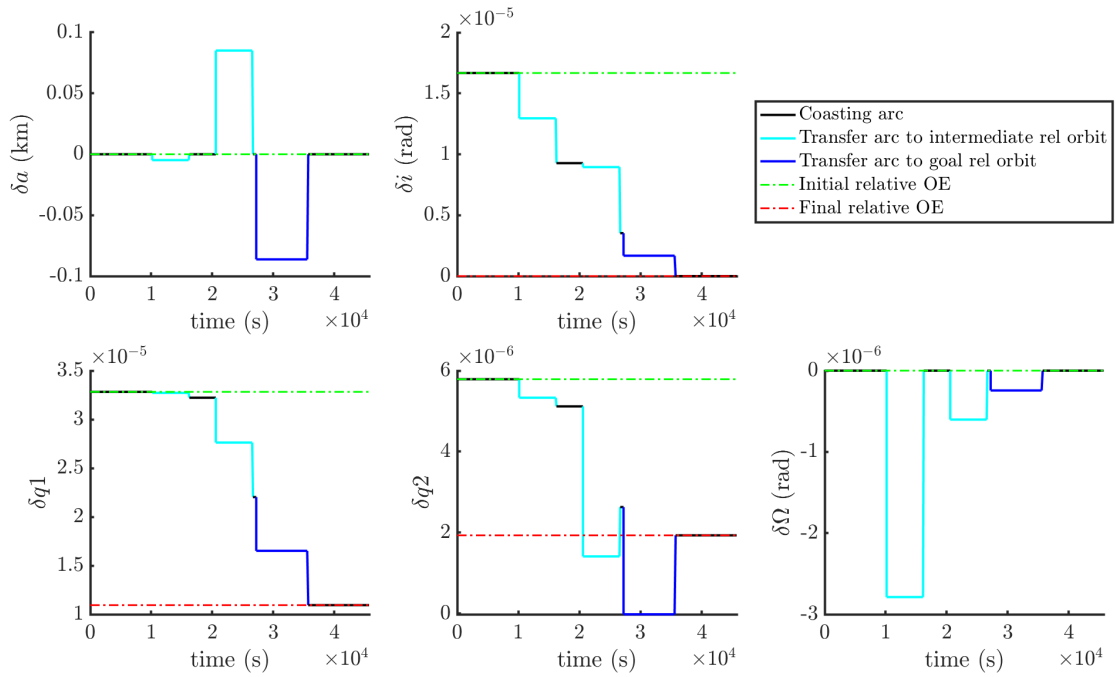
Figure 2.7: Transfer trajectory as computed by the AIKMP algorithm (ran for 1000 algorithm iterations) for Case 1 using the full PSO tree extension strategy. The total Δv required for the transfer is 13.57 cm/s.



(a) Transfer solution in 3D relative position space.



(b) Relative state error of solution trajectory with respect to the goal state.



(c) Relative orbital elements vs time of the final AIKMP solution

Figure 2.8: Transfer trajectory as computed by the AIKMP algorithm (ran for 1500 algorithm iterations) for Case 1 using the single-orbit PSO tree extension strategy. The total Δv required for the transfer is 13.24 cm/s.

the transfer solution computed by the full-PSO based Lambert solver ($\Delta v_{lambert}$= 15.05 cm/s). It is to be noted here that the total time taken by the solution computed by the AIKMP algorithm is slightly higher than the time taken by the Lambert transfer solution. This is because the total time of flight for the transfer trajectory is not considered a constraint in this problem description. The goal of this implementation is to achieve fuel-efficient transfer with collision avoidance.

For the remaining test cases in this chapter, the single-orbit PSO tree extension strategy has been used and the AIKMP algorithm to allowed to run for 1500 iterations.

Case 2: Case 1 + obstacle avoidance with a static obstacle with respect to chief

Although the Lambert solution trajectory shown in Figure 2.6 is not in collision with the chief in this case, collision-free transfer solutions cannot be guaranteed using traditional guidance methods like Lambert's solution because such methods have no collision-detection capabilities built in them. This is where the AIKMP algorithm can be most beneficial.

To demonstrate the obstacle-avoidance capability of the proposed algorithm, the initial scenario (Figure 2.5) has been modified by introducing another static obstacle with respect to the chief (a keep-out zone for the motion planner). This obstacle is placed strategically such that the original Lambert solution trajectory (as shown in Figure 2.6a) actually collides with it as shown in Figure 2.9. This is to make sure that the problem at hand cannot be solved directly by using a traditional guidance algorithm like Lambert's solution.

Both the SST algorithm (uses a random control for a random amount of time to propagate the tree) and the AIKMP algorithm are implemented with the single-orbit PSO-based tree extension strategy in this obstacle-avoidance scenario. The results of the random exploration of both algorithms are shown in Figure 2.10.

In this implementation, it is to be noted that the basic SST algorithm was not able to find a solution trajectory connecting the start state to the goal state in the allowed 1500 iteration. However, the AIKMP algorithm did find multiple solutions, and the final best cost solution trajectory (Δv= 13.57 cm/s) computed by the algorithm is shown in Figure 2.11.

Case 3: Case 2 + obstacle avoidance with a static obstacle that obstructs the deputy's

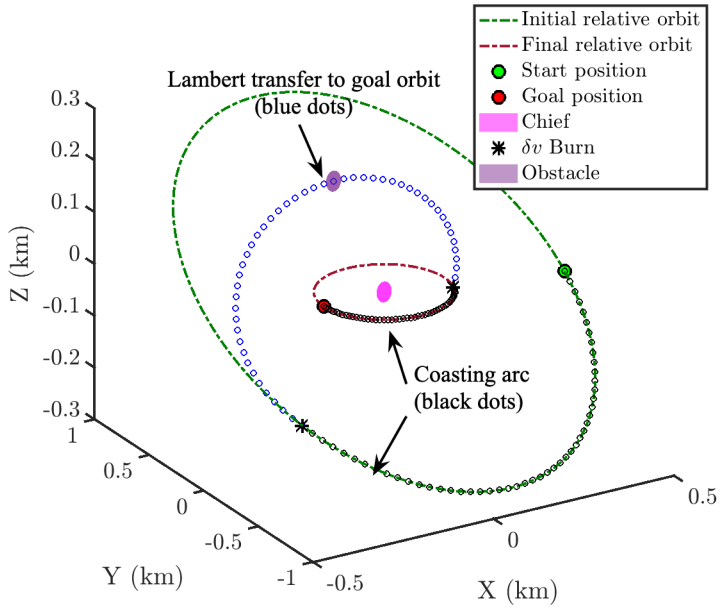
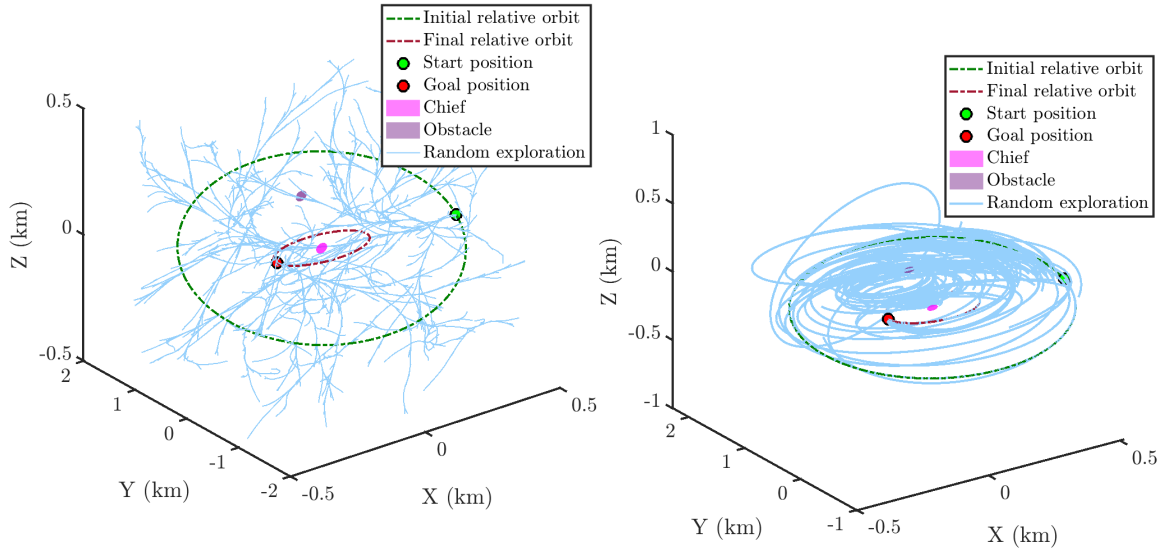


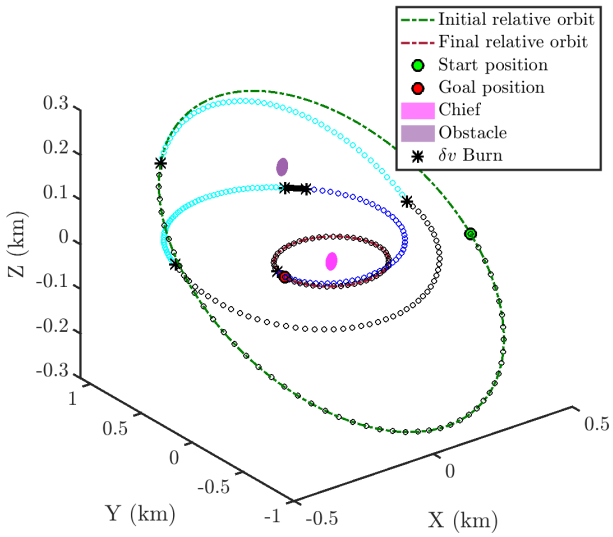
Figure 2.9: Modified implementation scenario with a static obstacle/keep-out zone in the chief-centered relative frame such that the original Lambert solution is in collision



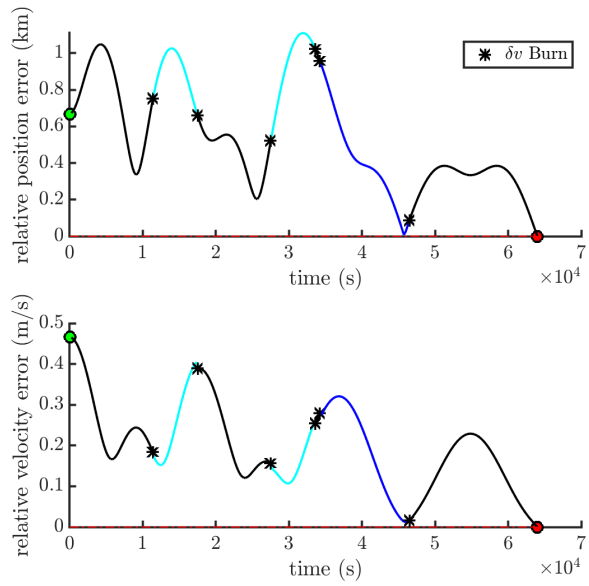
(a) Random exploration by SST motion planning algorithm in relative position and velocity state-space

(b) Random exploration by the AIKMP algorithm in relative OE state-space

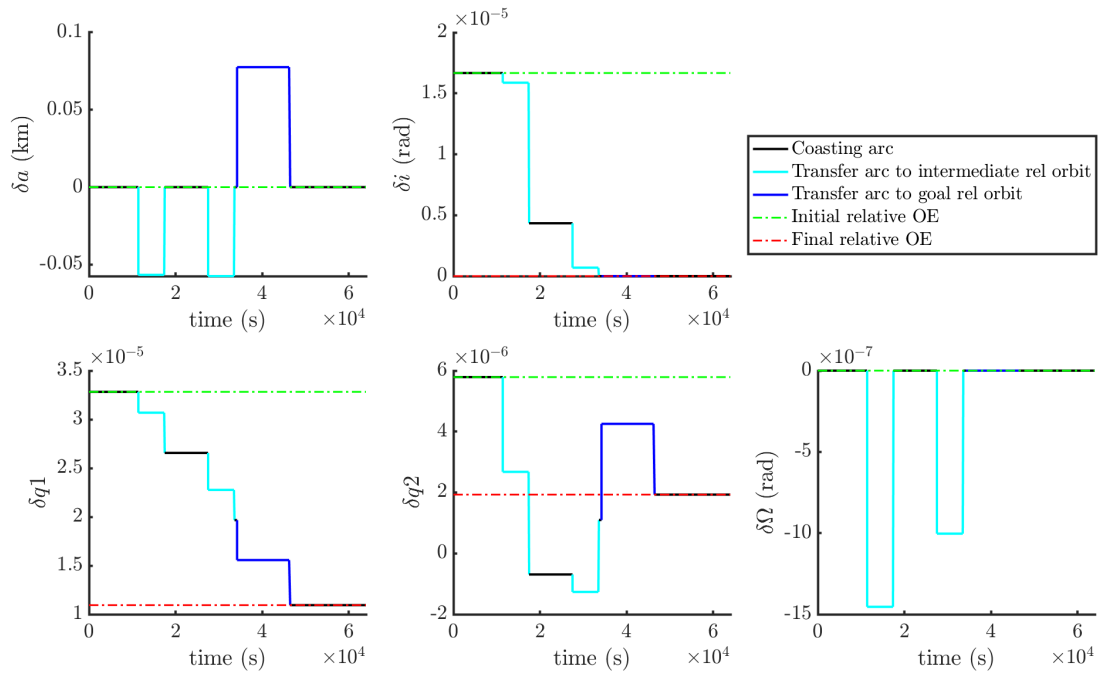
Figure 2.10: Random exploration by both SST and the AIKMP algorithm for Case 2



(a) Transfer solution in 3D relative position space.



(b) Relative state error of solution trajectory with respect to the goal state.



(c) Relative orbital elements of the final AIKMP solution versus time

Figure 2.11: Transfer trajectory as given by the AIKMP algorithm for Case 2. The total Δv required for the transfer is 13.57 cm/s.

initial relative orbit.

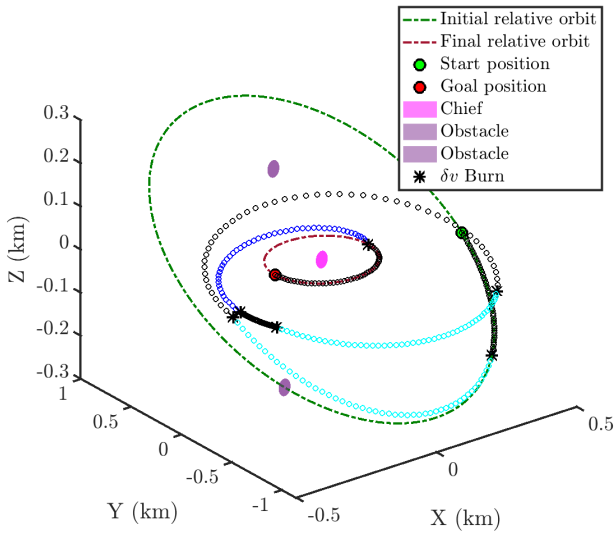
In Case 1 and Case 2, it is observed that the transfer solution computed by the AIKMP algorithm performs an initial coast (the initial black colored section of the plots in figure 2.8 and 2.11) before transferring to a random orbit (the cyan colored arcs in the same figures). Thus, to test the obstacle-avoidance capability of the algorithm further, in this implementation, a keep-out zone (with respect to the chief) is considered on the initial trajectory of the deputy spacecraft to test how the algorithm computes a new collision-free transfer solution. The results of this implementation are shown in Figure 2.12.

The results for Case 3 validate the reason why the solution trajectory in Case 1 and Case 2 comprised the long initial coasting arcs. This is because, by using the initial coasting arc, the overall fuel requirement of the transfer trajectory was found to be lesser: For Case 1: $\Delta v = 13.24$ cm/s, Case 2: $\Delta v = 13.57$ cm/s, For Case 3: $\Delta v = 15.74$ cm/s.

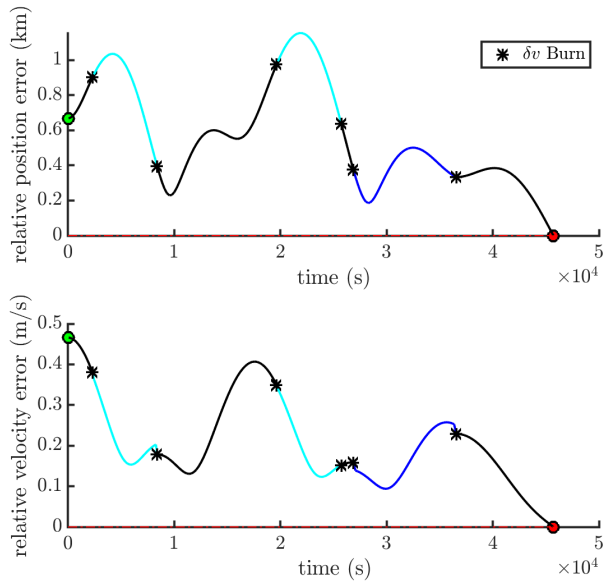
Using 5 different sequences of random numbers for the random exploration for both Case 2 and Case 3, as demonstrated in Figure 2.13, it is observed that the cost (total required Δv) of the AIKMP solution improves with more iterations. This conforms with the expected asymptotically near-optimal nature of the sampling-based SST algorithm, which approaches a near-optimal solution as the number of iterations approaches infinity [89]. Also, it is to be noted here that the minimum Δv Lambert transfer solution, in this case, is in collision and hence has a very high cost ($\Delta v_{lambert} = \infty$).

Case 4: Obstacle avoidance with multiple large obstacles with respect to the chief:

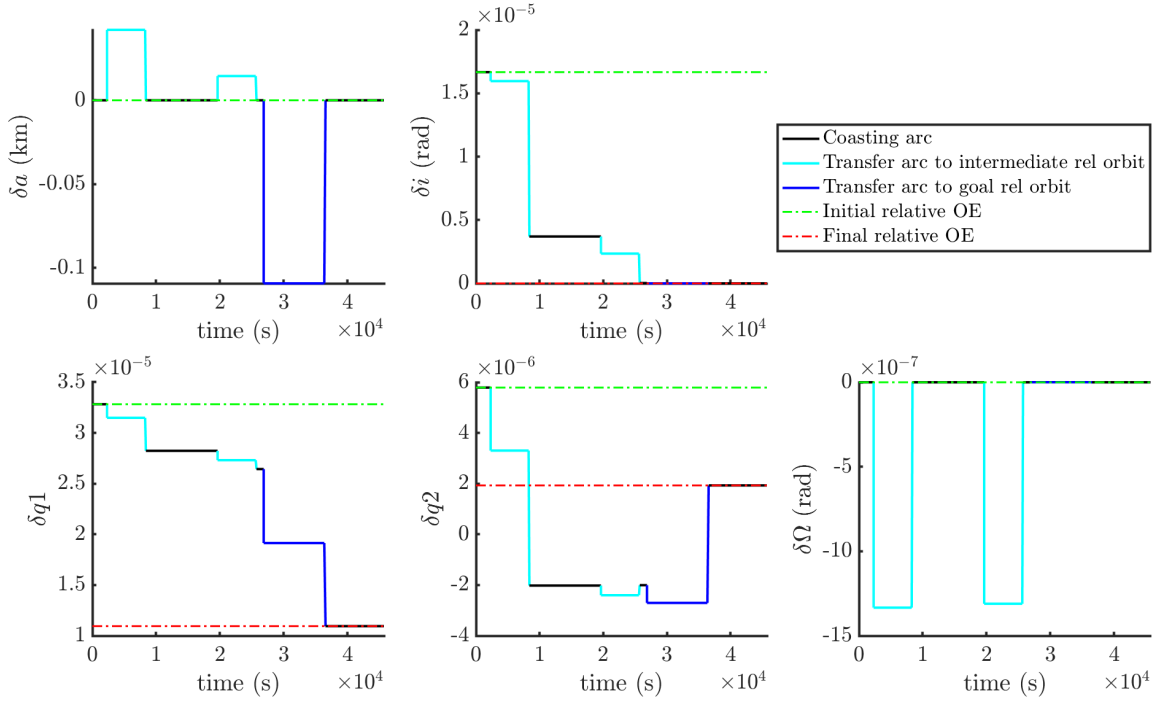
To demonstrate the obstacle avoidance capability of the AIKMP algorithm in a very cluttered environment, multiple large obstacles (spheres of radius = 100m) are added in the previous scenario and the results computed are as shown in Figures 2.14 and 2.15. The results clearly show the capability of the AIKMP algorithm to detect an obstacle and then perform maneuvers to avoid the collision while using less fuel.



(a) Transfer solution in 3D relative position space.

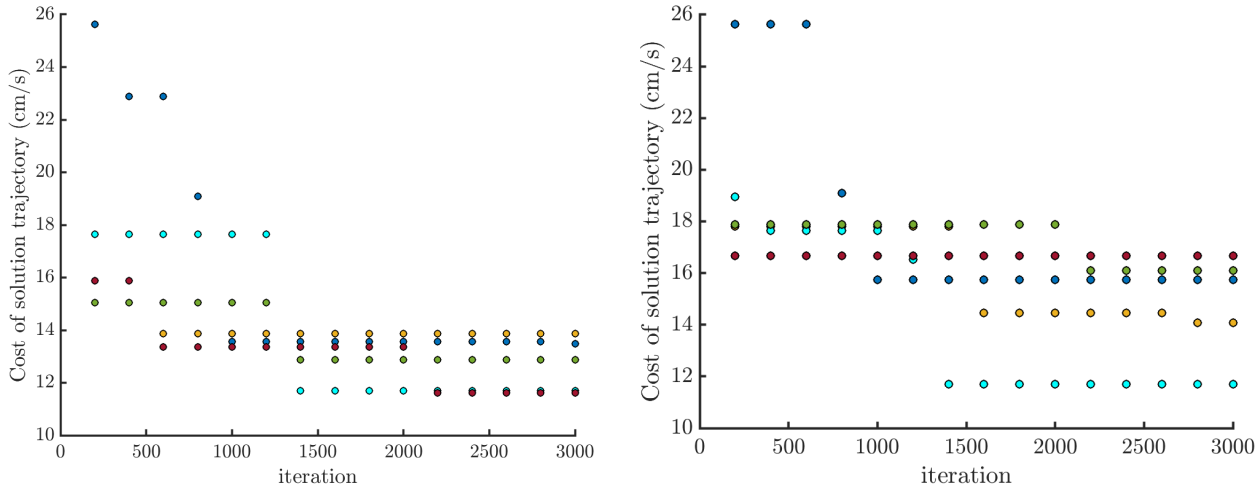


(b) Relative state error of solution trajectory with respect to the goal state.



(c) Relative orbital elements of the final AIKMP solution for Case 3 versus time

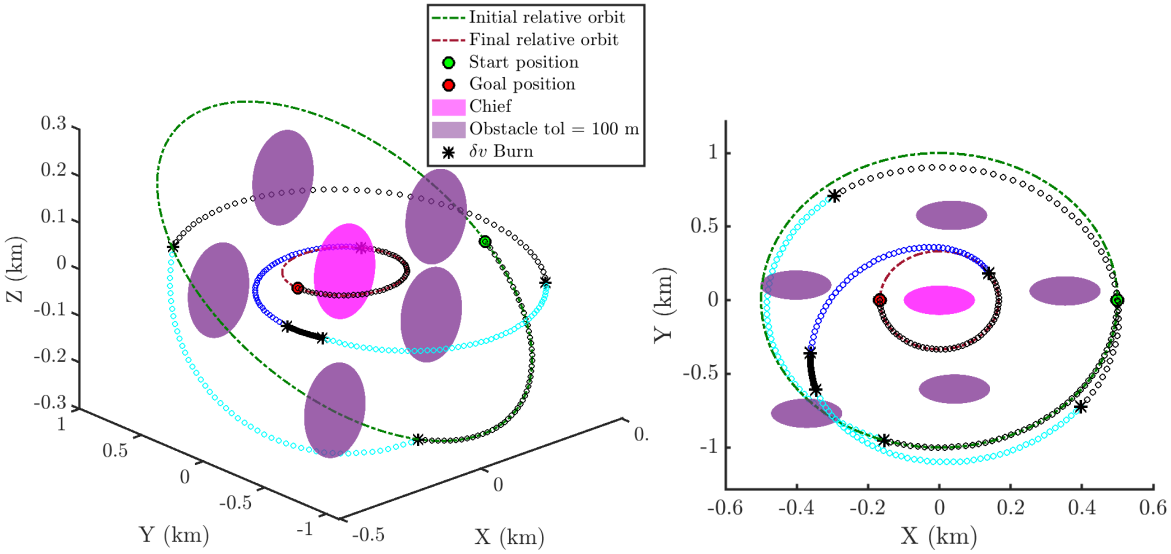
Figure 2.12: Transfer trajectory as computed by the AIKMP algorithm for Case 3. The total Δv required for the transfer is 15.74 cm/s.



(a) Cost of AIKMP solution versus the number of iterations for Case 2

(b) Cost of AIKMP solution versus the number of iterations for Case 3

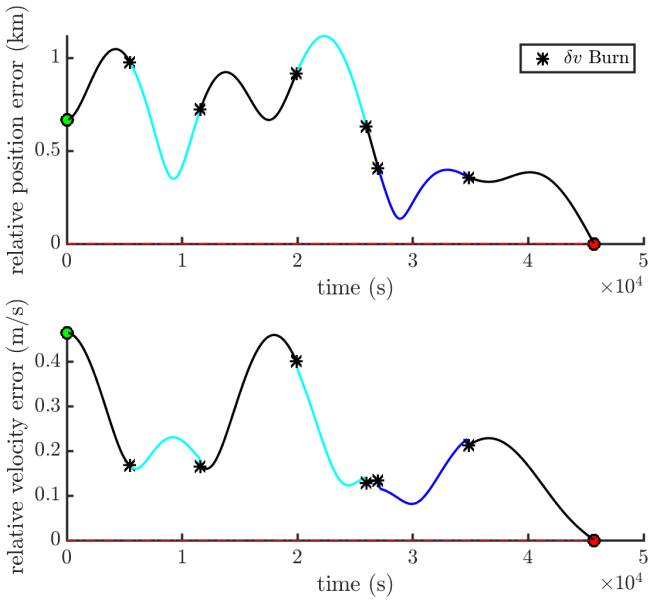
Figure 2.13: Cost of AIKMP solution versus the number of iterations using 5 different sequences of random numbers (denoted by the different colors) for the random exploration.



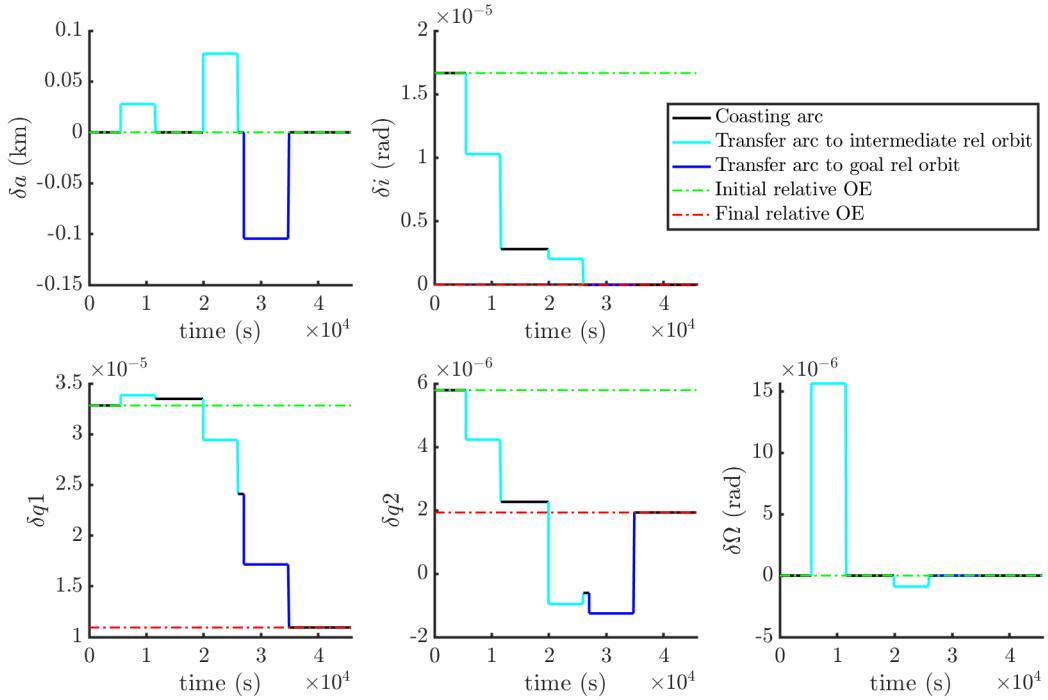
(a) Transfer solution in 3D relative position space.

(b) In-plane motion of the AIKMP transfer solutions in relative position space

Figure 2.14: Transfer trajectory as computed by the AIKMP algorithm with larger and more obstacles (Case 4). The total Δv required for the transfer is 21.24 cm/s.



(a) Relative state error of solution trajectory with respect to the goal state.



(b) Relative orbital elements of the final AIKMP solution versus time

Figure 2.15: Transfer trajectory as computed by the AIKMP algorithm with larger and more obstacles (Case 4). The total Δv required for the transfer is 21.24 cm/s.

2.3.2 Collision-free leader-follower formation using AIKMP

Here, a target leader-follower formation scenario is defined for the chief and deputy spacecraft systems. In this scenario, the deputy starts at an initial relative orbit and re-configures itself to the defined final leader-follower formation. The difference in this implementation as compared to the previous cases is that, in the final transfer arc to the goal relative state, the deputy has to target one particular relative state depending on where the chief is at that time. In implementation, this just removes the requirement of needing the PSO loop for this final transfer arc to the goal.

The classical orbital elements (COEs) of the chief and the initial and final COE differences of the deputy spacecraft (δCOE_i and δCOE_f respectively) are re-defined as shown in Table 2.5.

Chief COEs	**Values**	**Deputy** δCOE_i	**Values**	**Deputy** δCOE_f	**Values**
a	15000 km	δa	0 km	δa	0 km
e	0	δe	3.333×10^{-5}	δe	0
i	30°	δi	$9.549 \times 10^{-4^o}$	δi	0°
Ω	5°	$\delta\Omega$	$1 \times 10^{-5^o}$	$\delta\Omega$	0°
ω	10°	$\delta\omega$	0°	$\delta\omega$	0°

Table 2.5: Orbital elements of chief and deputy spacecraft for the Leader-Follower formation scenario.

In the targeted formation, the deputy is desired to stay $\approx$ 13 km ahead of the chief spacecraft which translates to a true anomaly difference of 0.05^o with the chief spacecraft.

Case 5: Leader-Follower scenario + Collision avoidance with chief + Obstacle avoidance with static obstacle with respect to chief

Same as Case 1 and Case 2 as described above, the re-configuration solution is initially computed using the swarm-based Lambert transfer and considers a static obstacle on this computed Lambert solution. This ensures that the current scenario cannot be solved by a straightforward Δv-optimal Lambert solver, as shown in Figure 2.16. The results of the AIKMP algorithm in this scenario (Δv= 63.51 cm/s) are as shown below in Figure 2.17. The obstacles considered in this scenario

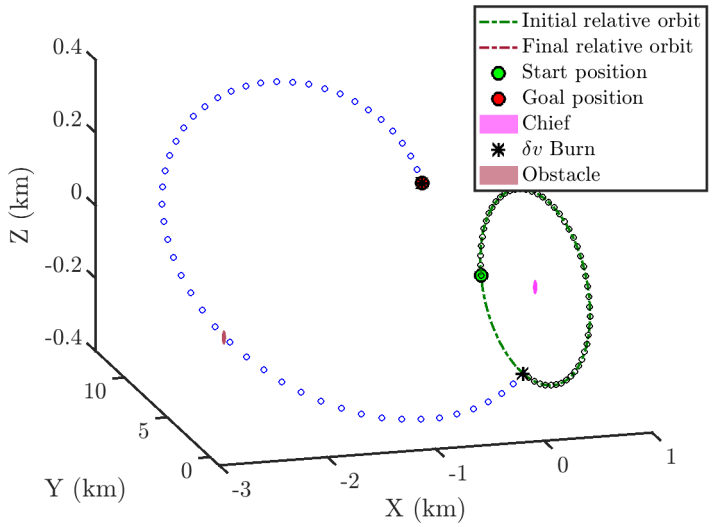


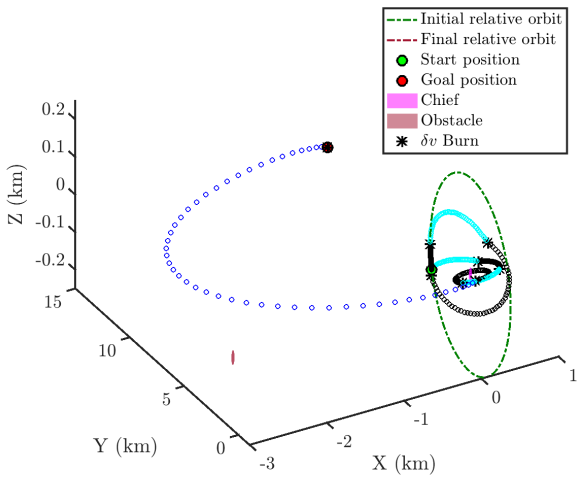
Figure 2.16: Minimum Δ_v requiring Lambert solution colliding with the obstacles in the leader-follower scenario

are solid spheres with a defined radius or obstacle tolerance. In the results shown in Figure 2.17, the obstacle tolerance = 20 meters.

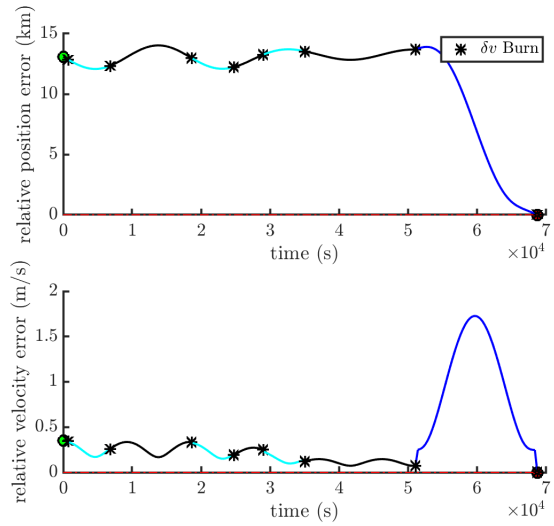
Case 6: Case 5 + Obstacle avoidance with another static obstacle with respect to chief

In this case, another obstacle (static in the chief-centered relative frame) is introduced to the scenario of Case 4, and implement the AIKMP algorithm for two very different obstacle tolerances/radii: 20 m and 200 m, as identified by the differently colored spheres in Figure 2.18. The goal is to demonstrate how the AIKMP algorithm recomputes a transfer solution for active obstacle avoidance with any defined obstacle space.

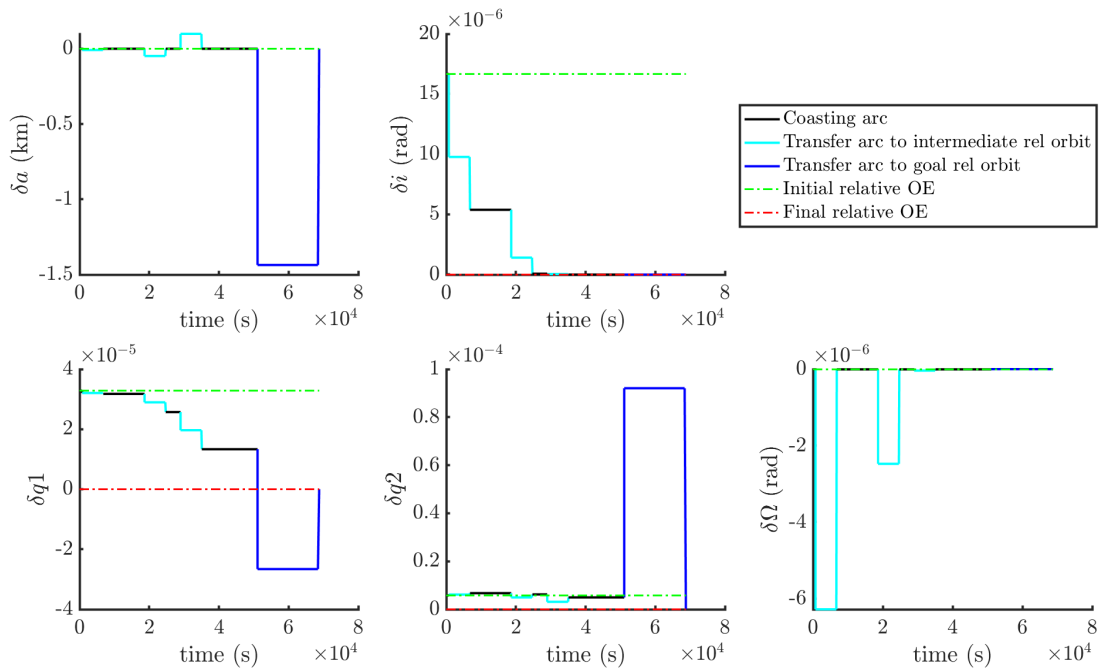
Figure 2.18 compares the AIKMP solutions for the two different obstacle tolerances. As can be seen, the AIKMP solution for an obstacle tolerance of 20 m (as shown by the plot in black color) originally passes through the obstacle space with a tolerance of 200 m (as shown by the spherical regions in yellow color). But with a prescribed larger obstacle space (10 times larger radius than the former) the algorithm was able to compute a collision-free solution transfer solution. It is also observed that in this case, the total fuel required by the solution with a larger obstacle tolerance



(a) Transfer solution in 3D relative position space.



(b) Relative state error of solution trajectory with respect to the goal state.



(c) Relative orbital elements of the final AIKMP solution for Case 3 versus time

Figure 2.17: Transfer trajectory as computed by the AIKMP algorithm for Case 4. The total Δv required for the transfer is 63.51 cm/s.

is lesser than the solution with a smaller obstacle tolerance. A possible reason for this is that the

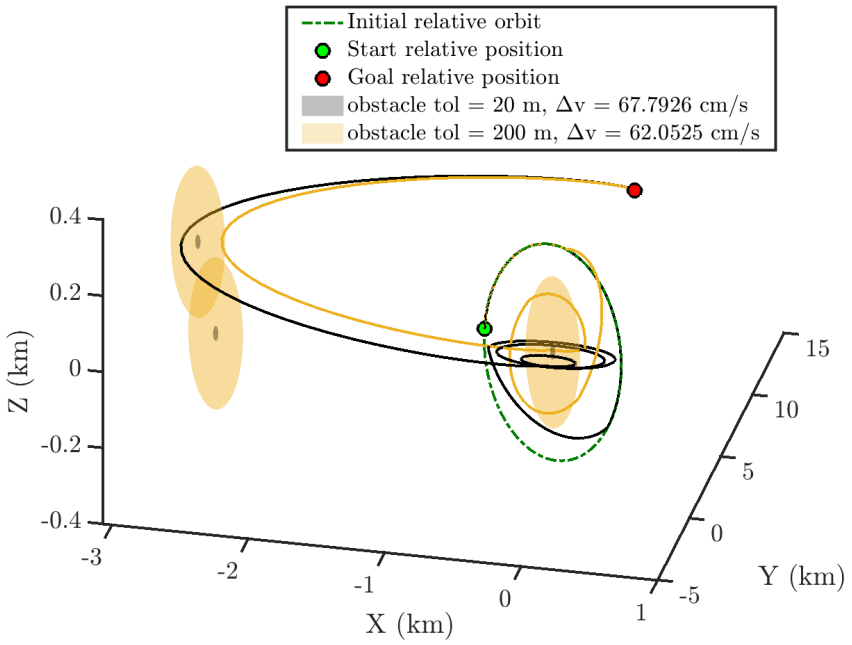


Figure 2.18: Comparison of AIKMP solutions for the leader-follower formation scenario with multiple obstacles (static in chief-centered relative frame) and with two different obstacle tolerances/radii

number of iterations allowed for computing the transfer (1000 iterations) wasn't enough for the algorithm to fully explore the state space to compute better solutions. As a result, the tree grown in the case with smaller obstacle tolerance couldn't capture the presence of other better solutions. This demands improved computation efficiency of the algorithm that will make it feasible to run such a random exploration-based motion planning algorithm for a longer number of iterations in search of better transfer solutions. The results of this case, however, do clearly demonstrate the capability of the AIKMP algorithm to detect an obstacle that is in the way of a spacecraft and then perform maneuvers to avoid the collision while using less fuel.

2.4 Summary

By augmenting sampling-based motion planning ideas from the field of robotics with knowledge of astrodynamics, fuel-efficient and collision-free motion planning can be achieved in astro-

dynamics applications. In this chapter, an astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm is presented that can regularly find solutions to spacecraft relative transfer problems – without requiring an initial guess of the solution. Instead of growing a sampling-based tree in traditional relative position and velocity space (as historically done in sampling-based orbital motion planning), the AIKMP algorithm is developed in the relative orbital element space of the deputy spacecraft around a chief spacecraft. This provides very useful geometric insight into relative spacecraft motion, as well as helps in generating an overall smooth final transfer trajectory. Through several examples, it was demonstrated that the AIKMP algorithm can efficiently detect an obstacle and compute a safe and fuel-efficient transfer trajectory for spacecraft around obstacles. It was also shown that this algorithm iteratively improves on its computed solutions given that the maximum allowed time of flight for the transfer is flexible although bounded (the final solution cannot take an infinitely long amount of time of flight) – a very reasonable assumption to consider in the context of relative motion planning with obstacle-avoidance capabilities. The version of the AIKMP algorithm presented in this chapter (without the pruning modules), in fact, translates to an astrodynamics-informed modified version of the popular sampling-based Rapidly Exploring Random Trees (RRT). This algorithm uses cost function minimization for tree extension as opposed to extending every node in the tree in every step. Also, this algorithm uses a *Goal bias* probability < 1 to guide the tree extension towards the desired goal state instead of attempting to connect to the goal state in every iteration of the algorithm [61]. Such modifications make this algorithm more suitable for fuel-efficient orbital motion planning applications. The use of the goal bias along with the Lambert steering law biases the tree extension step toward the goal state and ensures that the tree connects to the goal state provided that at least a single goal bias run is executed. This contributes to probabilistically complete nature of the AIKMP algorithm meaning that as the number of iterations of the algorithm tends to infinity, the planner will be able to find a feasible transfer solution with probability $= 1$, if a solution exists.

Chapter 3

Efficient Astrodynamics informed sampling-based planning

In the previous chapter, the algorithm used Lambert's solver along with particle swarm-based optimization to extend the tree from one node to another while using minimum fuel for the intermediate transfer. However, it was shown that irrespective of how good the intermediate transfers are, the overall solution is driven by the random exploration and hence the number of algorithm iterations allowed to run. Thus it is necessary that the algorithm is fast and efficient so that it can be run for more iterations and has possible onboard applications. Two major aspects of the previous version of the AIKMP algorithm that can be improved upon for computational efficiency are:

(1) Tree extension using Lambert's steering law: A common drawback of the solutions to Lambert's problem is the requirement of an iterative root-finding method to calculate the time of flight that can quickly become computationally expensive and hence are undesirable for onboard applications.

(2) Storing too many nodes in the tree: This will directly affect the performance of several modules in the AIKMP algorithm, particularly the routines that search over the existing tree nodes to achieve their goal, such as the Nearest neighbor selection routine.

Thus in this work, the focus is on improving the computation efficiency of the previously demonstrated AIKMP algorithm.

A few new definitions on top of definitions in Table 2.1, will be useful in understanding this section

as shown in Table 3.1:

$V_{inactive}$	:	The nodes that remain in the tree (because they are a parent to some active node) but they do not participate in tree expansion themselves.
Redefined $Cost(x)$	:	Sum of cost to reach node x from the start node (x_0) and an estimate of the cost to reach the target node from node x.
Witness node (s)	:	Every witness node defines a local neighborhood around themselves in the state space. These neighborhoods need not be mutually exclusive, however, all these neighborhoods together define the entire state space for the planning problem.
Representative node ($s.rep$)	:	Each witness node s maintains one *representative node* ($s.rep$) within their neighborhoods that describes the path with the best cost from the start node to that particular region.
δ_{best} (Pruning radius)	:	A radius that defines a neighborhood around witness node s to check for better cost solutions in that region. If better solutions exist, nodes in this neighborhood and their parent nodes undergo pruning.

Table 3.1: Important notations related to tree pruning used in the algorithm description

.

3.1 Modifications to steps of AIKMP:

With the goal of improving computation efficiency in mind, some major modifications have been introduced to the basic steps of the previous version of the AIKMP algorithm that has been detailed here.

3.1.1 Modified Step 1 (Tree initialization):

Same as before, this step initializes the tree with the starting node, x_0. As this is the only node in the tree, in this stage, it is set as an active node (V_{active}) that takes part in tree extension. The witness node (s) (which defines local neighborhoods in the given state space), as well as the witness representative node ($s.rep$) (which represents the node with the best cost in the region defined by s), are also set as x_0, to begin with. The definitions of the witness node and the witness representative node can be found in Table 3.1

3.1.2 Modified Step 3 (Best_Nearest - finding $x_{nearest}$):

This routine uses the same weighted distance measure as shown in equation 2.3 to compute the nodes that lay within a neighborhood of radius δ_{near} of the sampled node x_{rand}. Same as before, if no relative orbit is found in the δ_{near} radius neighborhood, the algorithm selects the relative orbit closest to x_{rand} according to Eq. 2.3 as $x_{nearest}$. Otherwise, it picks the relative orbit with the best path cost in that neighborhood as $x_{nearest}$.

The "*Cost*" of a relative orbit, for this work has been updated to the sum of total fuel required to reach that relative orbit from the starting relative orbit (cost-to-come) and an estimate of the total fuel that will be required to reach the goal orbit from the current node (cost-to-go). This is done because, fundamentally, arriving at an intermediate node explored by the tree using less fuel doesn't guarantee that the overall fuel required to transfer to the goal state will be also reduced. As a result, the new cost of a node x is formally defined as:

$$Cost(x) = \sum_{X=x_0}^{x^-} \Delta v_X + \Delta v_{est}, \tag{3.1}$$

where Δv_X refers to the impulsive Δv applied to transfer from node/relative orbit X and $(X = x^-)$ refers to the node right before node x that is used to transfer to node x. Δv_{est} refers to the estimated cost-to-go, that is the estimated Δv required to directly transfer from the current node to the goal node.

The details of computing the estimated cost-to-go (Δv_{est}) are described in Appendix B.

Same as in the previous chapter, for the purpose of this work, the δ_{near} value is chosen such that it is less than the weighted distance (equation 2.3) between the initial and the final relative orbit.

$$\delta_{near} < w(x_{start}, x_{goal}) \tag{3.2}$$

This makes sure that whenever the goal relative orbit is targeted directly ($x_{rand} = x_{goal}$), the algorithm doesn't always pick x_{start} (corresponding to the initial relative orbit) as the nearest node and also give randomly explored nodes a chance to connect to the goal. From equation 3.1,

$\text{Cost}(x_{start}) = \Delta v_{est}$ as the first term ($\sum_{X=x_0}^{x^-} \Delta v_X$) $= 0$ (since this is the starting node and $x^- = x_0$ here). Thus, it is possible that x_{start} may have a lower cost than other nodes in the tree which is why the tree may always try to extend the starting node to the goal node. Equation 3.2 is a valid assumption because anyway, the first tree extension that the AIKMP algorithm attempts to do is a direct extension from the starting node to the final goal node. Hence, this solution is already stored by the tree for comparison with other solutions computed by the algorithm.

3.1.3 Modified Step 4 (Extend_Tree - computing $\mathbf{x_{new}}$):

The previous chapter showed how a traditional guidance algorithm like Lambert's solution along with a swarm-based optimization technique can be used as the steering function for solving the spacecraft's relative motion planning by exploring the relative orbits of the deputy around the chief [43]. One common drawback of the solutions to Lambert's problem is the requirement of an iterative root-finding method to calculate the time of flight. Such iterative processes can quickly become computationally expensive and hence are undesirable for onboard applications. Moreover, since the Lambert solver requires the deputy's initial and final inertial position vector to compute the required transfer, the previous tree extension step often used transformations from relative orbit elements to inertial states and back as required – adding to its computational inefficiency. As a result, to overcome these shortcomings, the entire tree extension step is translated to the relative state space by replacing Lambert's solution with a linearized Lambert solution (LLS) [93], which is a linearization of the Prussings and Conway's Lambert solution [32] [Appendix C]. The chief's trajectory is used as the nominal trajectory for the LLS used in this work to compute the deputy's transfer.

Linearized Lambert Steering function: The LLS states that if the transfer solution to a nominal trajectory is known, the solution to its nearby trajectories can be quickly computed as a simple linear update to the nominal trajectory solutions without requiring the iterative root-solving process. Lambert's solution gives the terminal velocities ($\boldsymbol{v_1}$ and $\boldsymbol{v_2}$) required to transfer between any two position vectors ($\boldsymbol{r_1}$ and $\boldsymbol{r_2}$) in a given amount of time of flight. The LLS gives the

additional velocities ($\boldsymbol{\delta v_1}$ and $\boldsymbol{\delta v_2}$) required to achieve a neighboring transfer ($\boldsymbol{\delta r_1}$ and $\boldsymbol{\delta r_2}$) with an allowed difference in time of flight (δt), shown in orange in Figure 3.1b)) to a nominal transfer. A visual depiction of Lambert's solution and the LLS is shown in Figure 3.1. Defining a nominal

(a) Lambert's solution (b) Linearized Lambert Solution (LLS)

Figure 3.1: Lambert's solution as compared to the LLS. The LLS gives the additional velocities (δv_1 and δv_2) required to achieve a neighboring transfer (δr_1 and δr_2) with an allowed difference in time of flight (δt) to a nominal transfer.

transfer solution as functions of the position vectors $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and the transfer arc semi-major axis a we have:

$$\Delta t = g(\boldsymbol{r_1}, \boldsymbol{r_2}, a) \tag{3.3}$$

$$v_i = f_i(\boldsymbol{r_1}, \boldsymbol{r_2}, a), \text{ for } i = \{1, 2\} \tag{3.4}$$

The complete definitions of functions g and f are described in Appendix C. Here, Δt refers to the transfer time, and $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ refer to the initial and final velocity vectors respectively. The LLS says that variations in position vectors ($\boldsymbol{\delta r_1}$ and $\boldsymbol{\delta r_2}$) and time of flight (δt) can be introduced and corrections to the terminal velocities ($\boldsymbol{\delta v_1}$ and $\boldsymbol{\delta v_2}$) and the transfer arc semi-major axis (δa) can be computed as a simple linear update to the nominal trajectory solutions without requiring the iterative root solving process [93]:

$$\delta a = \left[\frac{\partial g}{\partial a}\right]^{-1} \left(\delta t - \frac{\partial g}{\partial \boldsymbol{r_1}}\boldsymbol{\delta r_1} - \frac{\partial g}{\partial \boldsymbol{r_2}}\boldsymbol{\delta r_2}\right) \tag{3.5}$$

$$\boldsymbol{\delta v_i} = \Gamma_i \delta t + \left[\frac{\partial f_i}{\partial \boldsymbol{r_1}} - \Gamma_i \frac{\partial g}{\partial \boldsymbol{r_1}}\right] \partial \boldsymbol{r_1} + \left[\frac{\partial f_i}{\partial \boldsymbol{r_2}} - \Gamma_i \frac{\partial g}{\partial \boldsymbol{r_2}}\right] \partial \boldsymbol{r_2} \tag{3.6}$$

$$\text{where, } \Gamma_i = \frac{\partial f_i}{\partial a}\left[\frac{\partial g}{\partial a}\right]^{-1} \tag{3.7}$$

The details of the derivation of the equations 3.5, 3.6 and 3.7 can be found in [93]. One thing to note here is that the $\boldsymbol{\delta v_i}$s computed by equation 3.6 refers to the additional velocity on top of the nominal transfer trajectory. The net Δv required by the deputy for the transfer can be computed as:

$$\Delta v_1 = |(\boldsymbol{\delta v_1} + \boldsymbol{v_1}) - \boldsymbol{v_{1\_d}}| \tag{3.8}$$

$$\Delta v_2 = |\boldsymbol{v_{2\_d}} - (\boldsymbol{\delta v_2} + \boldsymbol{v_2})| \tag{3.9}$$

$$\Delta v = \Delta v_1 + \Delta v_2 \tag{3.10}$$

This Δv is used to compute the "cost-to-come" (Δv_X) in the new definition of the cost as shown in Equation 3.1. Here, $\boldsymbol{v_{1\_d}}$ is the velocity of the deputy spacecraft in its initial orbit right before executing the initial impulsive transfer burn, $\boldsymbol{v_{2\_d}}$ is the velocity of the deputy spacecraft in its final orbit after executing the final impulsive transfer burn, $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are velocities of the nominal transfer at the instant of deputy's initial burn and final burn respectively as shown in Figure 3.1a. It is to be noted that for implementation purposes in this work, the LLS implementation assumes $\delta t = 0$. Assuming $\delta t = 0$ for the intermediate LLS-based tree extensions doesn't affect the potential of the AIKMP algorithm in finding near-optimal transfers. This is because, in every iteration of the AIKMP algorithm, a random time of flight is picked for the intermediate transfers which is equivalent to exploring different values of δt for that particular transfer. When the algorithm is allowed to run for a large number of iterations, the results will be the same as exploring different values for δt for the intermediate transfers but indirectly. However, the AIKMP algorithm framework is not limited to this assumption.

Particle swarm optimization-based LLS tree-extension strategy: Same as the single-orbit PSO strategy used before (section 2.2.4), here the transfer uses an initial coasting arc (propagated for a randomly picked coasting time) before executing the transfer to the target relative state. However, it is important to note here that different states in a relative orbit do not correspond to simply varying the relative true latitude ($\delta\theta$) values from 0 to 2π. Varying the $\delta\theta$ means considering different phasing between the deputy and the chief that results in relative states

that belong to different relative orbits around the chief [Appendix A]. Thus, to pick the target relative states in the target relative orbit (x_{rand}) for fuel-efficient tree extension, the PSO variable used by the AIKMP algorithm is 'propagation time' ranging from $[0, \text{Period}]$ instead of the true latitude angle.

The PSO approach explores the entire solution space using the swarm particles which in this case will be different relative states in the target relative orbit x_{rand}. The particles share information among themselves in every iteration of the algorithm and as a result, each particle is always aware of the best solution explored by any particle in the swarm. Every iteration, each particle moves towards the best solution explored till that iteration and this process continues until the entire swarm converges to a single solution or the maximum number of iterations is reached. The solution with the lowest cost is then picked as the final transfer solution and the final relative orbit corresponding to this solution is called the x_{new} node for this particular tree extension step. To get around the problem of local minima in this optimization problem, we use 50 swarm particles and allow a maximum of 1000 iterations for the swarm to converge to a near-optimal solution.

Relative motion dynamics: Once the required Δv impulses for the transfer are computed using the PSO-base LLS, the transfer arc (edge of the tree) is computed using the full non-linear dynamical model of orbital relative motion [111]. These equations are provided below:

$$\begin{aligned}
\ddot{x} &= -\frac{\mu}{r_d^3}(r_c + x) + \frac{\mu}{r_c^2} + x\dot{f}^2 + 2\dot{f}\left(\dot{y} - y\frac{\dot{r_c}}{r_c}\right) \\
\ddot{y} &= -\frac{\mu}{r_d^3}(y) + y\dot{f}^2 - 2\dot{f}\left(\dot{x} - x\frac{\dot{r_c}}{r_c}\right) \\
\ddot{z} &= -\frac{\mu}{r_d^3}(z)
\end{aligned} \tag{3.11}$$

where, x, y and z refer to the deputy states relative to the chief as expressed in the chief-centered LVLH frame, $\dot{f}$ refers to the rate of change of the chief's true anomaly, r_c refers to chief's orbit radius and $r_d = \sqrt{(r_c + x)^2 + y^2 + z^2}$.

3.1.4 Introducing Step 5 (Node_Locally_Best - adding best cost nodes to the tree):

After the new node (x_{new}) is computed, the algorithm evaluates if x_{new} has locally the best path cost in the tree. To do so, the routine uses the weighted distance measure (equation 2.3) to check if x_{new} lies in the neighborhood described by a pruning radius (δ_{best}) of an existing node defining that area (also called the witness node of s_{new}). Figure 3.2 will be helpful here to visualize the working of the Pruning modules (both steps 5 and 6.) The new node of the tree x_{new} is

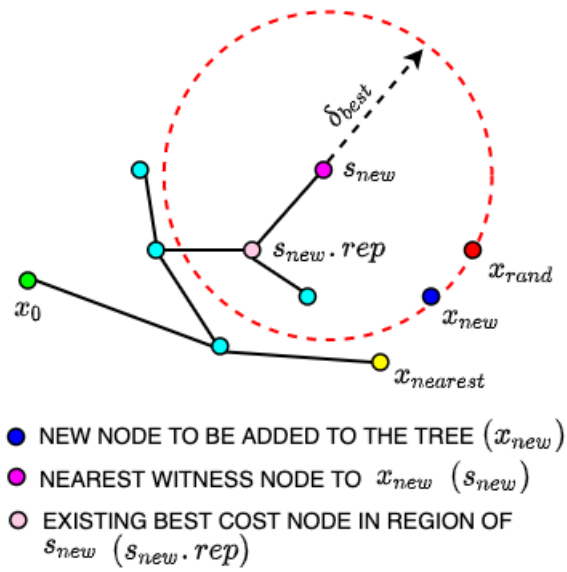


Figure 3.2: Visualisation of the different nodes involved in Pruning (applicable to AIKMP Steps 5 and 6)

considered the best node if either of the following conditions holds true:

(1) Using equation 2.3, if $w(x_{new} - s_{new}) \leq \delta_{best}$, and $Cost(x_{new}) \leq Cost(s_{new}.rep)$ (given by equation 3.1), where $s_{new}.rep$ is called the representative node denoting the best cost path in the area defined by s_{new}.

(2) If $w(x_{new} - s_{new}) \geq \delta_{best}$. In this case, x_{new} is added as another witness node to the tree.

x_{new} is added to the existing tree only if it is determined as the locally best node in the tree according to either of the above-mentioned conditions.

Tree pruning can be very beneficial for stochastic sampling-based orbital motion planners, and defining the radius parameters for the nearest neighbor selection (δ_{near}) and for pruning (δ_{best}) is an open and critical problem that determines the effectiveness of tree pruning in motion planning. For example: if the pruning radius is very large, that means that the algorithm will be very selective in adding new nodes in the tree and once a new node is added – more nodes will be pruned/deleted from its neighborhood, limiting the randomly exploring nature of the tree. On the other hand, if the pruning radius is very small, it will be equivalent to not using any pruning ($\delta_{best} = 0$ for no pruning case).

For the purpose of this work, the pruning radius (δ_{best}) is picked as smaller than the nearest neighbor radius (δ_{near}).

$$\delta_{best} < \delta_{near} \tag{3.12}$$

Otherwise, it is possible that most of the time a new node (x_{new}) may not be added to the tree because $x_{nearest}$ is already within δ_{near} of x_{rand} and the first term in the cost function ($\sum_{X=x_0}^{x^-} \Delta v_X$ in equation 3.1) for $x_{nearest}$ is lesser than x_{rand}, as the tree will require some Δv to extend to x_{rand} from $x_{nearest}$. Thus if Δv_{est} for $x_{nearest} < \Delta v_{est}$ for x_{new}, x_{new} will never be added to the tree over the $x_{nearest}$. Thus, to maximize the robustness and computational performance of the AIKMP algorithm, it is important to smartly define these parameters for a given implementation scenario.

3.1.5 Introducing Step 6 (Prune_Dominated_Nodes - deleting nodes and edges from the tree):

This routine is executed only if x_{new} was found as the best node by Node_Locally_Best routine according to condition 1 mentioned in Step 5 and if the time taken to reach node x_{new} from the initial node is lesser than the total time allowed for the overall solution trajectory. It computes the nearest witness node s_{new} to x_{new} and its representative node ($s_{new}.rep$). It then moves the old representative node from V_{active} to $V_{inactive}$ and sets $s_{new}.rep = x_{new}$ since x_{new} qualified as the new node with the best cost path in that region. If the moved node (old $s_{new}.rep$) is a leaf node, then the routine deletes the node completely from the tree and erases the edge it formed with its

parent node. The routine then repeatedly goes on deleting all the subsequent parent nodes erasing a section of the tree provided they belong to $V_{inactive}$ and are leaf nodes themselves. This helps the tree in maintaining a sparse set of nodes that in turn helps in improving the storage requirement as well as the computational efficiency of the overall algorithm, as will be demonstrated in the upcoming sections.

3.2 Implementation and simulation results

To demonstrate the performance of the AIKMP algorithm with the mentioned modifications the Case 2 scenario is picked as described in section 2.3 that has the chief spacecraft along with another keep-out zone for the orbital motion planner to avoid. Three different versions of the AIKMP algorithm are compared to solve the transfer solution to the scenario and compare their performances:

(1) AIKMP version 1: This version uses the PSO-based Lambert solution as the steering law to extend the tree forward (as previously demonstrated in results of Case 2 in section 2.3).

(2) AIKMP version 2: This version of the AIKMP algorithm uses the PSO-based LLS steering law (explained in section 3.1.3) along with the pruning routine (explained in sections 3.1.4 and 3.1.5).

(3) AIKMP version 3: In this version of the AIKMP algorithm, the PSO strategy is replaced with an orbit discretization approach. Same as the PSO approach, here the transfer uses an initial coasting arc (propagated for a randomly picked coasting time) before executing the transfer to the target relative orbit. The only difference is that instead of using a swarm-based optimization, this approach discretizes the targeted closed relative orbit states in the time interval of $[0, Period]$, and computes the best possible state that minimizes the cost of that particular transfer as given by equation 3.1.

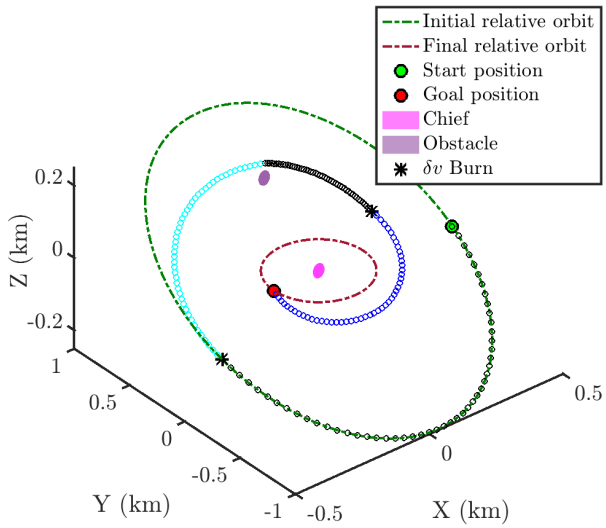
Performance comparison and analysis:

The solution for version 1 that uses the PSO-based Lambert solution as the steering law to extend

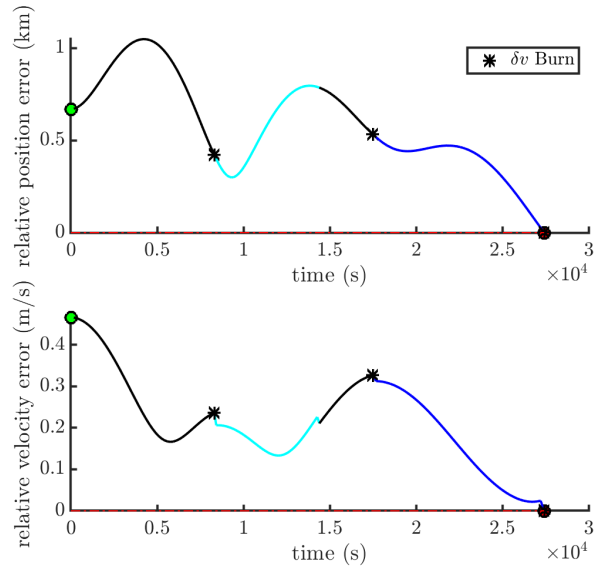
the tree is already demonstrated in Figure 2.11. The results of implementing versions 2 and 3 are shown in Figures 3.3 and 3.4 respectively. The cost of the transfer solution computed by versions 1, 2, and 3 are 13.57 cm/s, 14.98 cm/s, and 15.16 cm/sec respectively which are very similar to the order of magnitude. However, a significant difference in the performance as per computation requirements was observed between the different versions. The computation time required by the three versions of the AIKMP algorithm is compared and the results are shown in Figure 3.5.

As can be seen from Figure 3.5a, the computation time per iteration required by the AIKMP algorithm can be improved by $\approx 80\%$ by using the PSO-based LLS steering law along with pruning instead of using the PSO-based Lambert's steering law. Figure 3.5b shows the comparison between the cumulative time taken by the complete AIKMP algorithm vs the total number of iterations. This cumulative time is a combination of the time taken by the tree extension step as well as other steps of the algorithm that depend on the total number of edges in the tree (for example the *Best_Nearest* routine). It is observed that, for a total of 3000 iterations, using LLS along with pruning instead of Lambert's solution as the steering law for the motion planner improves the overall computation efficiency of the algorithm by $\approx 78\%$. This efficiency is further improved to $\approx 96\%$ by replacing the PSO routine with the orbit discretization strategy as discussed before. It was observed that discretizing the target relative orbit evenly at every 1-degree interval (360 equally spaced discretization points) yielded $\approx 96\%$ computation efficiency improvement for a cost of $\approx 12\%$ fuel increase as compared to using a PSO-based Lambert's steering law. The performance of version 3 of the algorithm is comparable to using a standard nonlinear optimization routine such as sequential convex programming (SQP)-based optimal LLS transfer, as seen in Figure 3.5. This frequency of discretization is a parameter that can be picked by the user. However, as proven and pointed out before (section 2.3.1), it is important to remember that irrespective of how much each individual transfer to the intermediate waypoints is improved, the cost of the overall transfer solution highly depends on the random exploration nature of the algorithm and hence the number of iterations for which the algorithm was allowed to run.

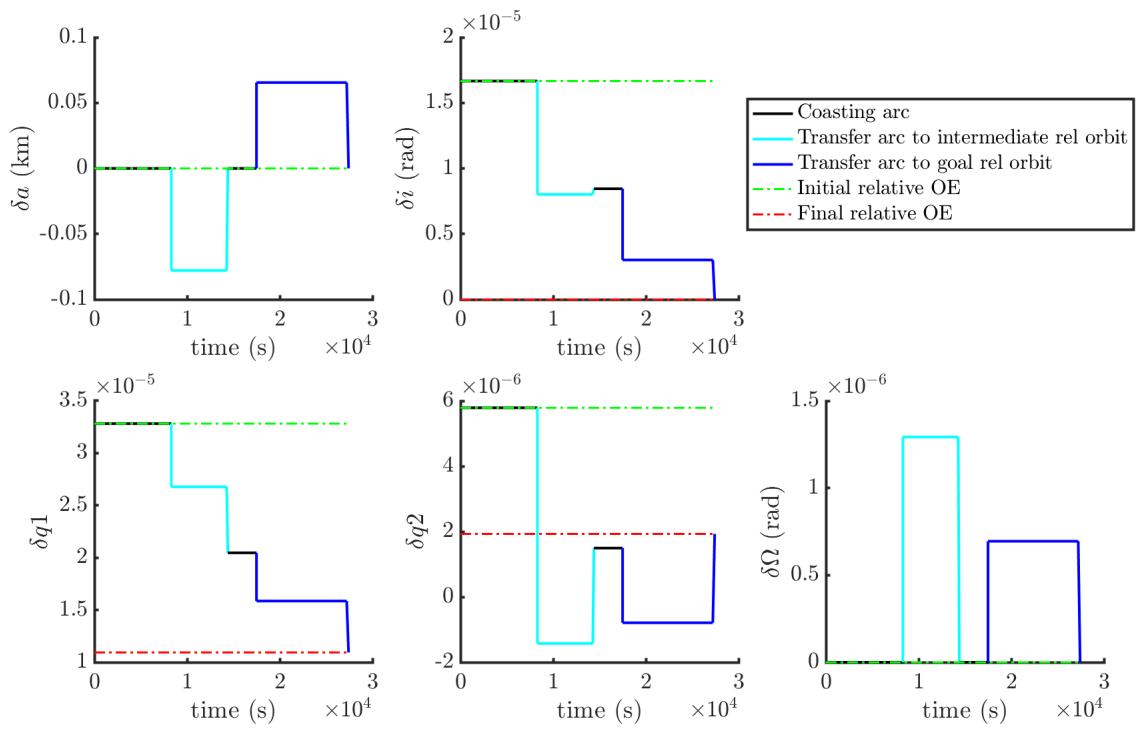
Figure 3.6 shows a comparison between the computation times required by AIKMP to com-



(a) Transfer solution in 3D relative position space.

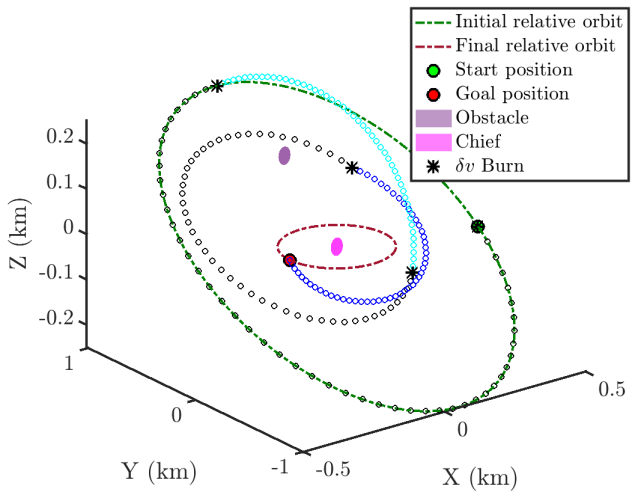


(b) Relative state error of solution trajectory with respect to the goal state.

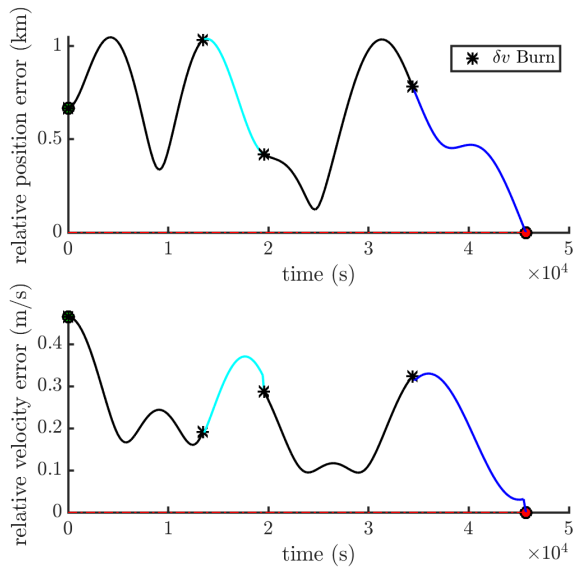


(c) Relative orbital elements of the final AIKMP solution for Case 2 versus time

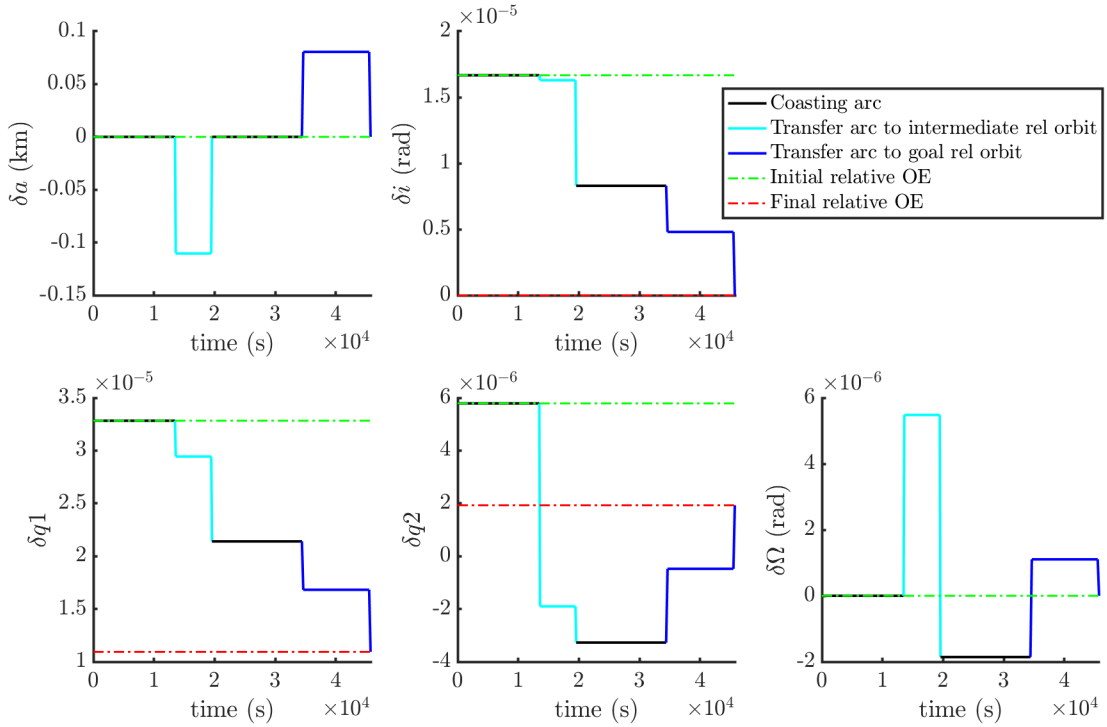
Figure 3.3: Transfer trajectory as computed by the AIKMP algorithm (Version 2) for Case 2. The total Δv required for the transfer is 14.98 cm/s.



(a) Transfer solution in 3D relative position space.

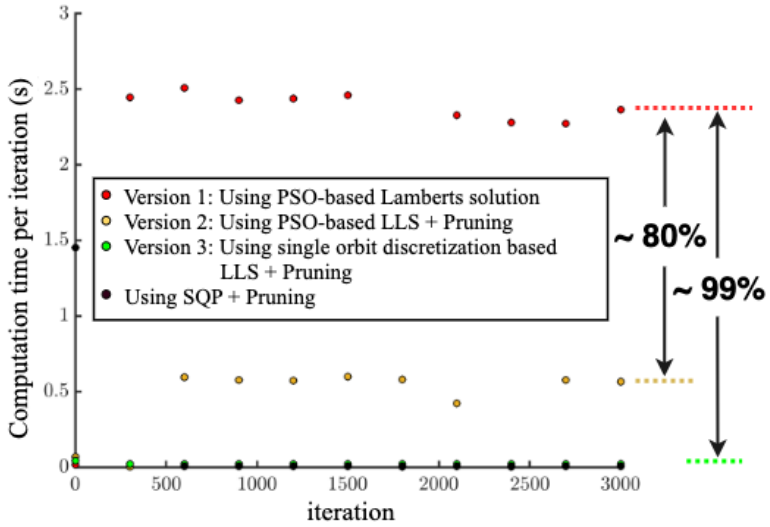


(b) Relative state error of solution trajectory with respect to the goal state.

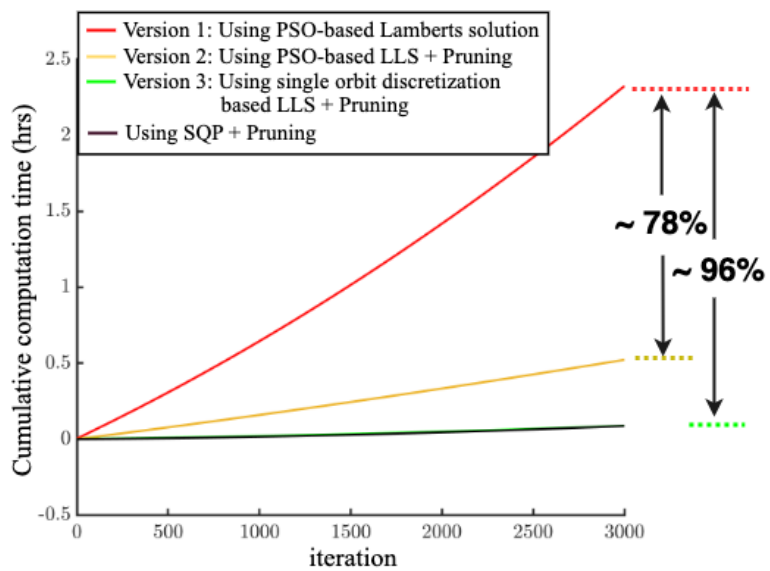


(c) Relative orbital elements of the final AIKMP solution for Case 2 versus time

Figure 3.4: Transfer trajectory as computed by the AIKMP algorithm (Version 3) for Case 2. The total Δv required for the transfer is 15.16 cm/s.



(a) Comparison of computation time per iteration required for the Tree extension step

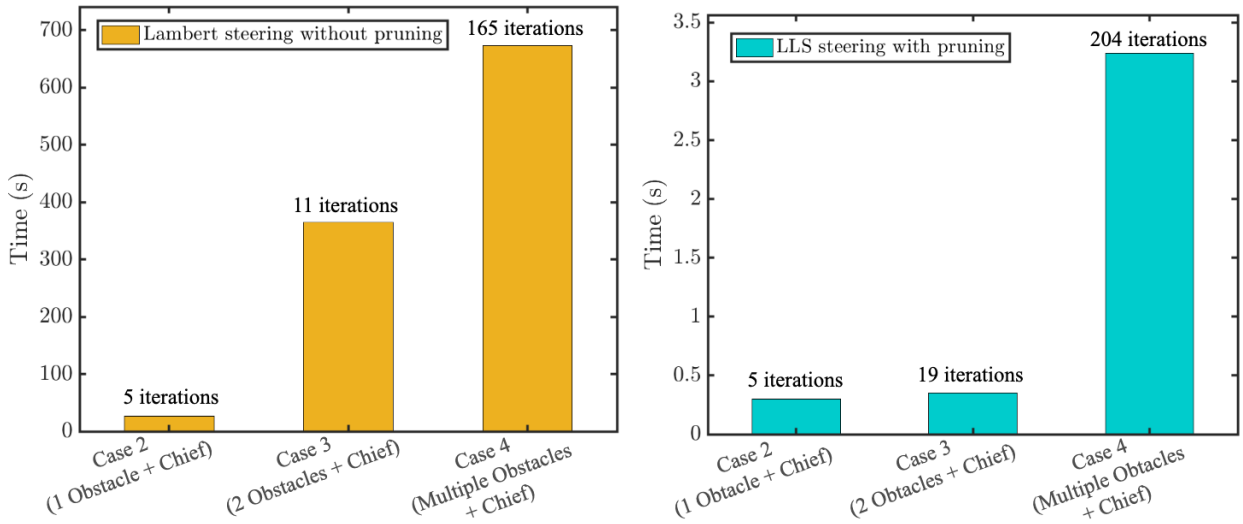


(b) Comparison of cumulative computation time required by the full AIKMP algorithm

Figure 3.5: Computation time requirement by the different versions of the AIKMP implementation vs number of algorithm iterations

pute the first transfer solution in the different test scenarios (Case 2, Case 3, and Case 4 with obstacles) described in Chapter 2 using Lambert steering law and LLS steering law with pruning. As evident from the figure, the efficient AIKMP algorithm is able to quickly find possible transfer

solutions to scenarios with a very cluttered environment (within ≈4 seconds in this case).



(a) Using Lambert steering law (Chapter 2)

(b) Using LLS steering law with pruning

Figure 3.6: Computation time required by AIKMP to compute the first transfer solution in the different test scenarios (Case 2, Case 3, and Case 4 with obstacles) using Lambert steering law and LLS steering law with pruning.

AIKMP steering law	**No. of nodes in final tree**	**No. of edges in final tree**
Lambert's solution OR Linearized Lambert's solution	≈ 2400	≈ 5200
Linearized Lambert's solution + Pruning	≈ 490 (≈80% less nodes)	≈ 1500 (≈70% less edges)

Table 3.2: No. of nodes and edges retained by the tree built by AIKMP algorithm with and without pruning for tree extension.

It was also observed that the total number of nodes and edges in the tree at the end of 3000 iterations without tree pruning was found to be ≈ 2400 nodes and ≈ 5200 edges. However, with pruning, the number of nodes stored by the random tree was reduced by ≈ 80%, and the number

of stored edges was reduced by $\approx 70\%$. The number of active nodes stored in the pruned tree was 419 nodes and inactive nodes were 64, which gives a total of 483 nodes and a total of 1496 edges were stored. This has been summarized in Table 3.2. Thus, using LLS steering law along with a pruning routine not only improves the computation time required by the AIKMP algorithm, it also significantly reduces the storage requirement by the algorithm.

3.3 Summary

The AIKMP algorithm has great potential to be used for efficient planning of collision-free and fuel-efficient transfer trajectories for astrodynamics applications and this algorithm does not require an initial guess of the solution trajectory. The algorithm can quickly compute feasible transfer solutions for relative motion planning problems in very cluttered environments and the algorithm is probabilistically complete meaning that as the number of iterations of the algorithm tends to infinity, the motion planner will be able to find a feasible transfer solution with probability $= 1$, if a solution exists. By using linearized Lambert's solution (LLS) instead of Lambert's solution as the steering law for the motion planner, it was shown that the overall computation efficiency of the algorithm can be improved by $\approx 78\%$. This efficiency can be further improved by augmenting the algorithm with a pruning module that improves the overall computation efficiency of the algorithm by $\approx 96\%$. It was also shown that the modifications made to the AIKMP algorithm also significantly reduced the storage requirement by the algorithm. With pruning, the number of nodes stored by the random tree was reduced by $\approx 80\%$, and the number of stored edges was reduced by $\approx 70\%$. The presented sparse and efficient version of the AIKMP algorithm is a single-step sampling-based kinodynamic approach to orbital motion planning problems (as opposed to existing two-step sampling-based orbital motion planning approaches in literature) and can quickly find solutions to spacecraft relative transfer problems – without requiring an initial guess of the solution.

Chapter 4

Closed-loop Linearized Lambert Solution for Onboard Formation Control and Targeting

Lambert's Problem is one of the most extensively studied problems in astrodynamics. The problem is concerned with determining the Keplerian transfer orbit between two position vectors for a given time of flight, as shown in Figure 3.1a. Many different approaches have been developed over the years to solve Lambert's Problem [82, 17, 66, 32, 131]. However, one common drawback of the solutions to this problem is the requirement of an iterative root-finding method involved in calculating the time of flight. Such iterative processes can quickly become computationally expensive and hence are undesirable for onboard applications. To address this problem, McMahon and Scheeres developed the linearized Lambert solution (LLS) algorithm, which can determine high-accuracy solutions to neighboring transfers to a wide range of nominal transfer trajectories [93]. They show that, if a nominal Lambert arc is known, a neighboring solution can be obtained through a linear update to the required nominal velocities, as illustrated in Figure 3.1b. In his work, McMahon derived this linearization solution using 2-body dynamics and proved that the algorithm gives roughly 99.9% computational savings as compared to the Lambert solution at the cost of less than 1% errors in accuracy.

The proposed study leverages the subtle properties of the LLS algorithm [93] and extends its application to a perturbed environment. A closed-loop LLS re-targeting scheme is implemented that segments the transfer arc into sub-arcs and applies multiple LLS correction burns to correct deviations due to linearization errors and non-Keplerian perturbations. An optimization problem is

formulated and it is demonstrated that the algorithm can be extended to satisfy mission constraints like maximum relative separation constraints with fuel-efficient transfers. This novel extension of the LLS is applied to Spacecraft Formation Flying (SFF) problem, and it is shown that the resulting closed-loop LLS-guidance algorithm allows for stringent targeting accuracy in different SFF problems. It is to be noted that the closed-loop LLS retargeting can be applied to any linearized Lambert's solution formulation [9, 116, 107] and is not restricted to first-order linearization solutions such as the one used in this paper.

4.1 Closed-Loop LLS as a re-targeting scheme in Spacecraft Formation Flying

The linearized Lambert solution (LLS) developed by McMahon and Scheeres states that if the transfer solution to a nominal trajectory is known, the solution to its nearby trajectories can be quickly computed as a simple linear update to the nominal trajectory solutions without requiring the iterative root-solving process [93]. The linearization of the Prussings and Conway's Lambert solution [32] that they use are shown in section 3.1.3. The closed-loop LSS solution segments a linearized Lambert transfer arc into sub-arcs and applies multiple correction burns to correct for deviations due to linearization errors and non-Keplerian perturbations during the transfer.

4.1.1 Scenario description:

To demonstrate the working of the closed-loop LLS algorithm a simple spacecraft reconfiguration example scenario is considered. The scenario comprises a Chief and a Deputy spacecraft where the Chief maneuvers from an initial state (true anomaly of 180°) to a target state (true anomaly of 20°) in its original GEO orbit in a quarter of a day (6 hours). The orbital elements of the Chief's initial trajectory and the transfer trajectory are shown in Table 4.1. The Deputy aims to maneuver around the Chief in order to maintain an initial and final desired relative separation $\boldsymbol{\delta r_1} = \boldsymbol{\delta r_2} = [5, 0, 0]\ km$ as seen by the Chief centered Radial, In-track and Cross-track (RIC) frame. The $\boldsymbol{\delta r_1}$ and $\boldsymbol{\delta r_2}$ need not necessarily be equal here and can take any values around the

reference trajectory provided the linearization is valid.

Orbital elements	Values for initial chief trajectory	Values for the transfer orbit
Semi-major axis (a)	41798 km	87593.54 km
Eccentricity (e)	0.75	0.88
Inclination (i)	3.219^o	176.78^o
RAAN (Ω)	65°	245°
Argument of Periapsis (ω)	5°	140.93^o
Initial true anomaly (ν_0)	180°	-146.02^o
Targeted true anomaly (ν)	20°	14.10^o

Table 4.1: Orbital elements of Chief's initial trajectory and transfer trajectory in defined reference scenario for demonstrating the concept of the closed-loop LLS (GEO case)

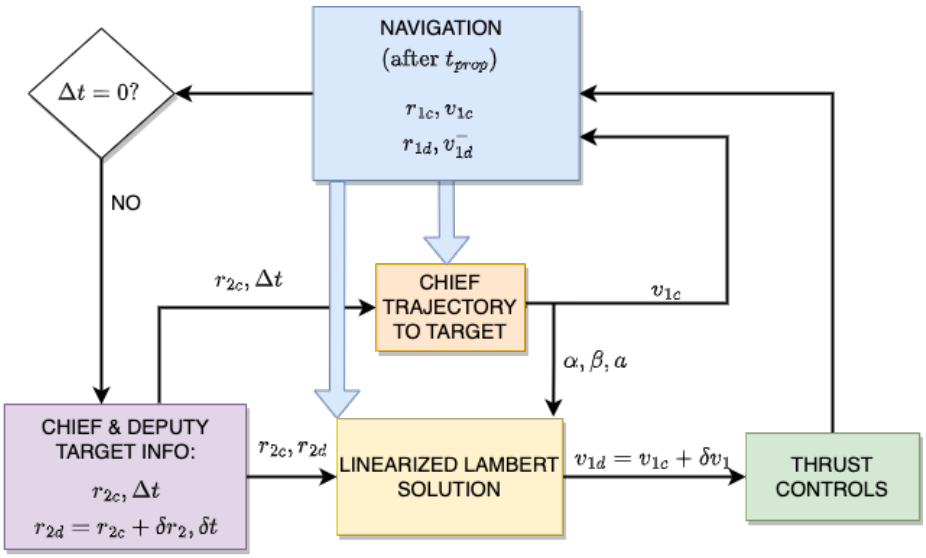


Figure 4.1: Working of the Deputy's closed-loop LLS in conjunction with Chief's Lambert solution in the spacecraft formation re-targeting scenario.

Figure 4.1 provides a flowchart of the proposed closed-loop LLS guidance of the Deputy spacecraft. Here, $(\boldsymbol{r_{1c}}, \boldsymbol{v_{1c}})$ represents the Chief's initial inertial state, $(\boldsymbol{r_{2c}}, \boldsymbol{v_{2c}})$ represents the Chief's final desired inertial state, $(\boldsymbol{r_{1d}}, \boldsymbol{v_{1d}})$ represents the Deputy's initial inertial state, $(\boldsymbol{r_{2d}}, \boldsymbol{v_{2d}})$ represents the Deputy's final desired inertial state, and $\boldsymbol{v_{1d}^-}$ refer to the velocity of the Deputy right before applying the LLS correction burn. Thus the initial and final desired relative separations of

Deputy wrt the Chief are $\boldsymbol{\delta r_1}$ and $\boldsymbol{\delta r_2}$ respectively, where

$$\boldsymbol{\delta r_1} = \boldsymbol{r_{1d}} - \boldsymbol{r_{1c}} \tag{4.1}$$

$$\boldsymbol{\delta r_2} = \boldsymbol{r_{2d}} - \boldsymbol{r_{2c}} \tag{4.2}$$

The total time of flight here is represented by TOF, propagation time in iteration L of closed-loop LLS is depicted by t_{prop_L}, and the remaining time of flight at the beginning of iteration L is given by Δt_L. The relation between the three is:

$$\Delta t_L = TOF - \sum_{l=1}^{L-1} t_{prop\_l} \tag{4.3}$$

Here, given the Chief's current state, target state, and Δt, the Chief performs a Lambert transfer to reach its goal and this is considered as the Chief's transfer trajectory to the target state as can be seen in Figure 4.1. With the knowledge of both the Chief and Deputy's current states, target states, time of flight (δt represents the difference in the time of flight of the chief and the deputy), parameters α, β and the Chief transfer trajectory semi-major axis a, the LLS is computed for the deputy spacecraft to maneuver it from its start state to goal state. This gives the linear update to the nominal velocities of the Chief trajectory that will be required by the Deputy ($\boldsymbol{\delta v_1}$) to reach its desired goal state from the start state. After this initial burn is applied, the Deputy is allowed to drift as per its dynamics for $t_{prop} <$ TOF. At the end of this t_{prop}, given the current state of the Chief spacecraft, the LLS solution for the Deputy is recomputed and the resulting velocity correction is applied as *correction burn* for re-targeting to its goal state again from the current state. This process of recomputing the LLS and applying correction burns is repeated until $\Delta t = 0$. The number of correction burns to be applied is completely based on the scenario/mission requirement. For example, if the mission requirement is to achieve better targeting accuracy, one might choose to implement multiple correction burns whereas if the mission requirement is only to minimize the overall fuel consumption, one might prefer lesser or no correction burns.

4.1.2 Performance of the closed-loop LLS:

For ease of understanding the performance of the closed-loop LLS algorithm, the implementation is broken down into two parts. First, the algorithm is implemented in the described scenario using Keplerian dynamics only. For demonstration purposes, the total number of burns is limited to 3 that are arbitrarily placed during the transfer, and no restriction is assumed in terms of fuel usage. The first correction burn is applied once the Deputy has propagated for an angle $= \theta/2$, where θ is the *total transfer angle* of the transfer arc from the start state to the goal state. The second correction burn is applied when the total propagation angle $= 3\theta/4$.

Here,

$$n \text{ total number of burns} \implies 1 \text{ LLS burn} + (n-1) \text{ correction burns.} \tag{4.4}$$

It is to be noted here that in this example, equation 4.4 considers only the initial LLS burn because the goal for the deputy here is to reach the target relative position δr_2 irrespective of the arrival velocity.

Let $\boldsymbol{r'_{2d}}$ and $\boldsymbol{r'_{2c}}$ be the final actual positions of the Deputy and the Chief respectively each wrt the inertial frame. Let $\boldsymbol{\delta r'_2}$ be the Deputy's actual final position relative to the Chief's actual final position, defined as

$$\boldsymbol{\delta r'_2} = \boldsymbol{r'_{2d}} - \boldsymbol{r'_{2c}} \tag{4.5}$$

The targeting error that is considered here for evaluation is e_{RIC} which is defined as the targeting error as seen by the Chief-centered RIC frame.

$$e_{RIC} = \|\boldsymbol{\delta r_2} - \boldsymbol{\delta r'_2}\|_2 \tag{4.6}$$

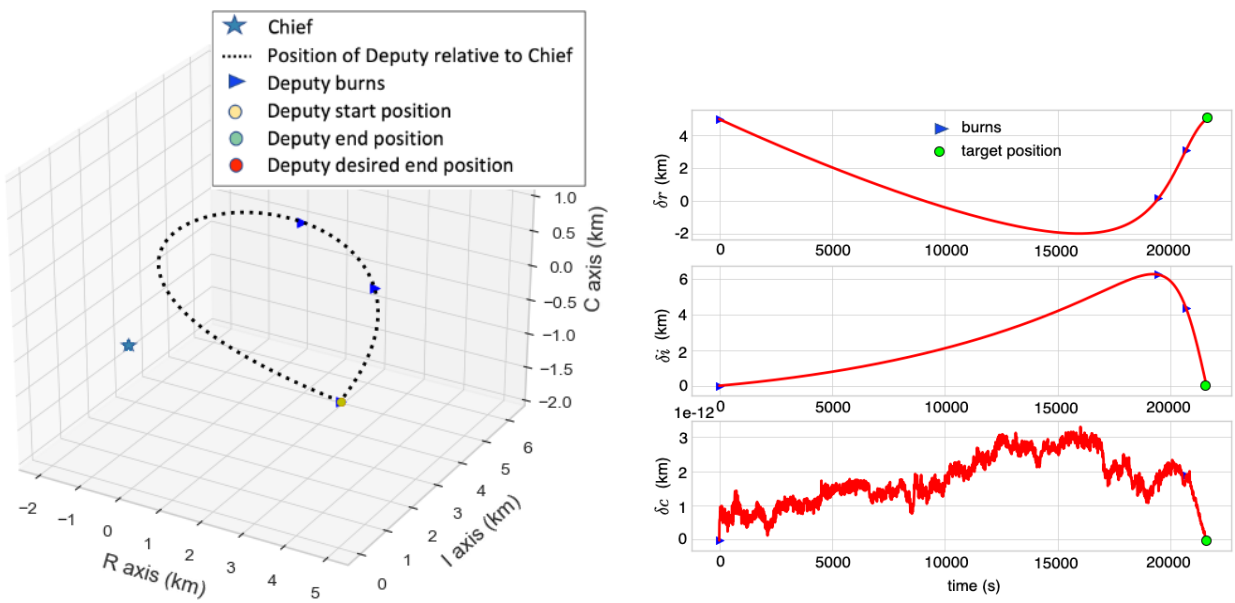
4.1.2.1 Performance with Keplerian dynamics:

Since external perturbations are not considered in this case, the result of this implementation will demonstrate the effect of the closed-loop LLS in overcoming the linearization errors of the single burn LLS.

For the case when only a single LLS burn is applied to transfer the deputy from its initial state to its

desired final state, the e_{RIC} is found to be 1.5 meters (Table 4.2), which is due to the linearization error of the LLS.

Now, to compare its performance with the closed loop LLS and to evaluate the targeting accuracy, as described earlier, after propagating the Deputy for $\theta/2$ angle, the LLS to target the final desired position is recomputed and a single correction burn is applied. In another case, a second correct burn is computed using LLS after the Deputy has propagated for a total angle of $3\theta/4$. Figure 4.2 shows the resulting transfer of the deputy spacecraft in the Chief-centered RIC frame after the application of the closed-loop LLS burns. As can be seen from Table 4.2, with just a single correction burn, the e_{RIC} is reduced by $\approx 27\%$, whereas with two correction burns it is reduced by $\approx 67\%$.



(a) 3D representation

(b) Component-wise representation

Figure 4.2: Deputy relative position wrt Chief as seen by Chief's RIC frame (with two closed-loop LLS correction burns).

No. of total burns	Deputy's required Δv (in km/s)	e_{RIC} (in m)	% reduction in targeting error
1	Initial burn: 2.52	1.5	–
2	Initial burn: 2.52 First correction burn: 6.21×10^{-7}	1.1	$\approx 27\%$
3	Initial burn: 2.52 First correction burn: 6.21×10^{-7} Second correction burn: 1.48×10^{-7}	0.5	$\approx 67\%$

Table 4.2: Fuel required by the deputy for each closed-loop LLS burn and deputy's final position error with Keplerian dynamics in the re-targeting scenario

4.1.2.2 Performance with non-Keplerian dynamics:

Here external perturbations are introduced to the previous implementation scenario namely: solar radiation pressure (SRP), drag, and J2 perturbations which are defined as follows:

Cannonball model for computing SRP acceleration:

$$\boldsymbol{a_{srp}} = -\frac{P_{srp} C_r A}{m} \hat{\boldsymbol{\delta r}} \tag{4.7}$$

Here $P_{srp} = P_0(R_0/R_{sun})^2$ is the SRP acting on an object at a distance R_{sun} from the Sun, where $P_0 = 4.57 \times 10^{-6}$ Pa is the SRP at $R_0 = 1$ AU which is the distance from the Earth to the Sun. C_r is the spacecraft coefficient of reflectivity, A is the cross-sectional area of the spacecraft facing the sun, m is the mass of the spacecraft, $\boldsymbol{\delta r} = \boldsymbol{r_{sun}} - \boldsymbol{r_s}$ is the relative distance between the spacecraft and the Sun, $\boldsymbol{r_{sun}}$ is the position vector of Sun and $\boldsymbol{r_s}$ is the position vector of the spacecraft, both relative to the ECI frame.

The differential SRP acceleration expressed in the chief centered LVLH frame components is defined as:

$$^{O}\boldsymbol{\Delta a_{srp}} = [ON](\boldsymbol{a_{srp,d}} - \boldsymbol{a_{srp,c}}) \tag{4.8}$$

Here, $\boldsymbol{a_{srp,d}}$ and $\boldsymbol{a_{srp,c}}$ refer to the SRP acceleration acting on the deputy and the chief spacecraft respectively. [ON] is the Direction Cosine Matrix (DCM) that maps vectors from the inertial frame

to the chief-centered LVLH frame or the O frame.

Drag acceleration:

$$\boldsymbol{a_{drag}} = -\frac{\rho C_d A |\boldsymbol{V_{rel}}| \boldsymbol{V_{rel}}}{2m} \tag{4.9}$$

Here, C_d is the spacecraft drag coefficient and A is the cross-sectional area of the spacecraft perpendicular to its direction of motion. $\boldsymbol{V_{rel}} = \boldsymbol{V}_s - \boldsymbol{V_{atm}}$ is the spacecraft's velocity relative to the velocity of the surrounding atmosphere $\boldsymbol{V_{atm}}$. Here the air particles in the atmosphere are assumed to be rotating at the same spin rate as the Earth. This gives $\boldsymbol{V_{atm}} = \boldsymbol{\omega_{atm}} \times \boldsymbol{r_s}$, where inertial angular velocity of the atmosphere $\boldsymbol{\omega_{atm}} = [0 \quad 0 \quad \frac{2\pi}{86400}]^T$ rad/s.

The atmospheric density ρ is computed by using the Exponential decay model of atmospheric density as shown in equation 4.10, where $R_e = 6.4 \times 10^6$ m is the radius of the Earth, $H = 8.5 \times 10^3$ m is the atmospheric scale height and $\rho_0 = 1.3$ kg/m^3 is the atmospheric density at the Earth's surface ($|\boldsymbol{r_s}| = R_e$).

$$\rho = \rho_0 \exp\left[\frac{-(|\boldsymbol{r_s}| - R_e)}{H}\right] \tag{4.10}$$

The differential drag acceleration expressed in the chief centered LVLH frame components is defined as:

$$^{O}\boldsymbol{\Delta a_{drag}} = [ON](\boldsymbol{a_{drag,d}} - \boldsymbol{a_{drag,c}}) \tag{4.11}$$

$\boldsymbol{a_{drag,d}}$ and $\boldsymbol{a_{drag,c}}$ refer to the drag acceleration acting on the deputy and the chief spacecraft respectively.

J2 acceleration:

$$\boldsymbol{a_{J2}} = \frac{3J\mu {R_e}^2}{2|\boldsymbol{r_s}|^5} \begin{bmatrix} (5\frac{Z^2}{|\boldsymbol{r_s}|^2} - 1)X \\ (5\frac{Z^2}{|\boldsymbol{r_s}|^2} - 1)Y \\ (5\frac{Z^2}{|\boldsymbol{r_s}|^2} - 3)Z \end{bmatrix} \tag{4.12}$$

Here, $J = 0.001082$ is the constant coefficient reflecting the gravitational effect of Earth's oblateness due to its rotation, X, Y, and Z are the x,y, and z-components of the position vector

$\boldsymbol{r_s}$ of the spacecraft relative to the ECI frame.

The differential J2 acceleration expressed in the chief centered LVLH frame components is defined as:

$$^{O}\boldsymbol{\Delta a_{J2}} = [ON](\boldsymbol{a_{J2,d}} - \boldsymbol{a_{J2,c}}) \tag{4.13}$$

$\boldsymbol{a_{J2,d}}$ and $\boldsymbol{a_{J2,c}}$ refer to the $J2$ acceleration acting on the deputy and the chief spacecraft respectively.

In the presence of these perturbations, the exact 3 cases as described in section 4.1.2.1 are run, and the e_{RIC} and burn magnitude for the 3 re-targeting cases with different allowed total numbers of burns are depicted in Table 4.3.

No. of total burns	**Deputy's required Δv (in km/s)**	e_{RIC} **(in m)**	**% reduction in targeting error**
1	Initial burn: 2.52	5.4	–
2	Initial burn: 2.52 First correction burn: 2.61×10^{-3}	4.1	$\approx 24\%$
3	Initial burn: 2.52 First correction burn: 2.61×10^{-3} Second correction burn: 3×10^{-3}	1.2	$\approx 78\%$

Table 4.3: Fuel required by the deputy for each closed-loop LLS burn and deputy's final position error with non-Keplerian dynamics in the re-targeting scenario

As can be seen, the closed-loop is able to reduce the e_{RIC} by $\approx 78\%$ after the application of the two correction burns. The main takeaway from this implementation is that, with a set of rightly spaced correction burns over the transfer trajectory, the closed-loop LLS has the potential to improve spacecraft targeting accuracy significantly even with the presence of external perturbations.

Now, it is important to note here that although the demonstrated example case considers the reference trajectory of a chief spacecraft, for the LLS or the closed-loop LLS algorithm to work, the Chief need not even be a real spacecraft doing maneuvers and can just be a nominal reference

trajectory. Moreover, the Chief can also have different dynamics as compared to the Deputy, if necessary.

4.2 Closed-loop LLS for spacecraft re-targeting with maximum relative separation constraints

4.2.1 Scenario description:

Here, the closed-loop LLS is used to ensure a maximum separation distance requirement between a reference trajectory (a "virtual chief") and the deputy spacecraft with minimum fuel usage. A scenario is defined with a single deputy spacecraft and a nominal GEO reference trajectory as the virtual chief. The scenario begins with the Deputy spacecraft starting in an eccentric orbit around the central body it is orbiting, staying close to the virtual chief. As expected, with the passage of propagation time, the Deputy will deviate from its original trajectory due to the effect of external perturbations. However, around the virtual chief, a sphere of radius 10 km is defined and the Deputy must perform maneuvers to stay within this sphere.

To achieve this, a *Planning Algorithm* is developed using closed-loop LLS to prevent the Deputy from violating the maximum relative separation constraint. This Planning Algorithm comprises two main parts:

(1) A Constraint checker: This is responsible for monitoring the relative separation between the virtual chief and the Deputy's current position. If the relative separation is below the specified threshold, no correction burn will be required; therefore the Deputy will be allowed to drift as per its natural dynamics.

(2) A Targeting scheme: If/once the constraint is violated (i.e. the relative separation has grown too large), a targeting scheme will be activated. This scheme will be responsible for determining a new target that will abide by the relative proximity constraint and a time of flight to allow the LLS to compute the required correction burns. The new target is selected to be on a naturally closed relative orbit (or quasi-periodic orbit due to the

perturbations) whose size is less than the maximum allowable relative separation. This will allow the Deputy to spend as much time coasting without violating the relative separation constraint and ultimately reducing the total delta-v requirement.

The complete scenario can be visualized in Figure 4.3.

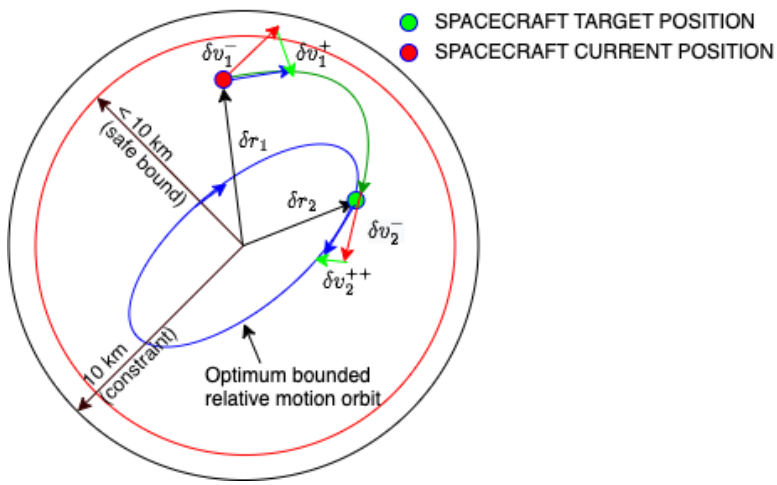


Figure 4.3: Spacecraft re-targeting scheme using closed-loop LLS when relative separation bounds are close to being violated.

Here, $\boldsymbol{\delta v_1^-}$ represents the initial relative velocity of the Deputy at the point of exiting the constraint sphere, $\boldsymbol{\delta v_1^+}$ is the initial correction velocity given by LLS, $\boldsymbol{\delta v_2^-}$ is the velocity with which the Deputy arrives at the target position and $\boldsymbol{\delta v_2^{++}}$ is the final velocity correction required by the Deputy to be in the desired relative orbit. Figure 4.4 presents a concise block diagram describing the inner workings of this proposed LLS guidance with the Planning algorithm for a single spacecraft.

4.2.2 Planning Algorithm Development to satisfy maximum relative separation constraint

4.2.2.1 Spacecraft retargeting to closed relative orbit:

To reduce the required velocity change throughout the length of the mission, the idea is to utilize the natural motion of the spacecraft as much as possible. Hence, targeting an orbit

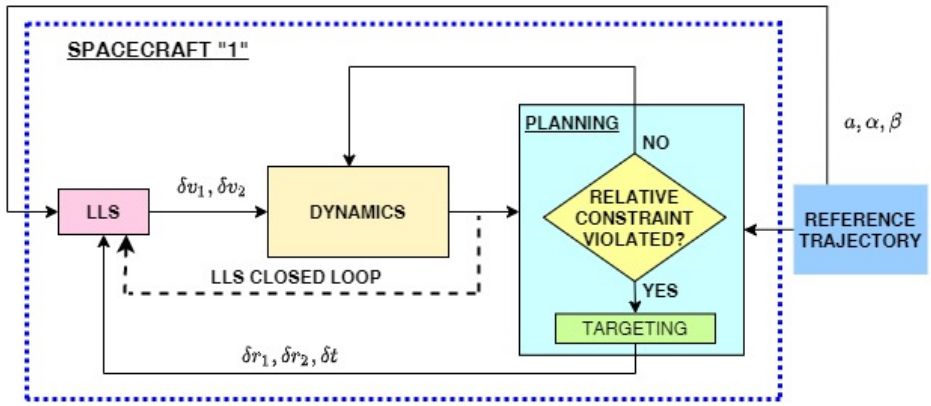


Figure 4.4: Work-flow of the closed-loop LLS for the single spacecraft re-targeting case with maximum relative separation constraints.

that would naturally fulfill the maximum separation requirement for a longer time, seemed like an excellent strategy to achieve this. Stable closed relative orbits are the most intuitive of such natural orbits. The Tschauner-Hempel (T-H) equations have been used to describe the motion of the Deputy around the eccentric chief. The solutions to a deputy's relative motion around a chief according to the T-H equations are given by equations B.6 that are reproduced here for reference:

$$\bar{x}(f) = K_1\rho sin(f) + K_2\rho cos(f) + K_3(2 - 3e\rho sin(f)\bar{k}(\tau - \tau_0))$$

$$\bar{y}(f) = K_1(\rho + 1)cos(f) - K_2(\rho + 1)sin(f) - 3K_3\rho^2\bar{k}(\tau - \tau_0) + K_4$$

$$\bar{z}(f) = K_5 sin(f) + K_6 cos(f)$$

where the parameters K_1, K_2, K_3, K_4 are constant of the motion, $\tau = nt$, n is the Chief's mean motion, and $\bar{k} = \sqrt{\mu}[a(1 - e^2)]^{-3/2}$.

It can be seen that $\bar{k}(\tau - \tau_0)$ grows unbounded as time evolves. To ensure bounded relative motion which is centered around the Chief's position (a desirable behavior in our analysis), it is imperative to set $K_3 = K_4 = 0$, K_4 being the y-offset. These constants of the motion can be expressed as a function of the initial conditions [14] as follows:

$$K_3 = (3\rho(f_0) + e^2 - 1)\bar{x}(f_0) + es(f_0)\bar{x_0}'(f_0) + \rho(f_0)^2\bar{y_0}'(f_0) = 0 \tag{4.14}$$

$$K_4 = -3e\left(\frac{s(f_0)}{\rho(f_0)}\right)\left(1 + \frac{1}{\rho(f_0)}\right)\bar{x_0} + (1 - e^2)\bar{y_0} + (ec(f_0) - 2)\bar{x_0}' - es(f_0)\left(1 + \frac{1}{\rho}\right)\bar{y_0}' = 0 \tag{4.15}$$

Solving the above equations we get,

$$\bar{x_0}' = -\frac{e\sin(f_0)}{e\cos(f_0)+1}\,\bar{x_0} - \left(\frac{1}{e\cos(f_0)+2} - 1\right)\bar{y_0} \tag{4.16}$$

$$\bar{y_0}' = -\left(\frac{1}{e\cos(f_0)+1} + 1\right)\bar{x_0} - \frac{e\sin(f_0)}{e\cos(f_0)+2}\,\bar{y_0} \tag{4.17}$$

$$\text{Where,}\quad \rho(f) = 1 + e\cos(f), \tag{4.18}$$

$$c(f) = \rho(f)\cos(f), \tag{4.19}$$

$$s(f) = \rho(f)\sin(f), \tag{4.20}$$

$$(\bar{x}, \bar{y}, \bar{z}) = \frac{(x, y, z)}{R}, \tag{4.21}$$

$$R = \frac{1}{1 + e\cos(f)} \tag{4.22}$$

Also, here prime notation $()' = \frac{d()}{df}$.

Equations 4.16 and 4.17 can be used to obtain the closed relative orbit under Keplerian motion.

4.2.2.2 Optimal targeting angle:

A parameter minimization problem is derived to determine the optimal transfer angle (θ) defining the target $\boldsymbol{\delta r_2}$ in the chief-centered LVLH frame that will minimize the fuel requirement of the overall mission.

$$\boldsymbol{\delta r_2} = \rho\left[\cos(\theta), \sin(\theta), 0\right]^T \tag{4.23}$$

Here, scalar ρ is a user-prescribed design parameter that will be heuristically chosen given the level of perturbations. Given the scenario and close relative motion around the virtual Chief, this definition of the transfer angle (θ) is a valid assumption in this case that was assumed for ease of calculation.

The minimization problem is defined as:

$$\text{Cost function } (J) = \min_{\theta} \left(\frac{1}{2}\Delta v_1^T \Delta v_1 + \frac{1}{2}\Delta v_2^T \Delta v_2 \right), \text{ where } \theta \text{ is the transfer angle} \tag{4.24}$$

$$\text{s.t: } \Delta v_1 = |\boldsymbol{\delta v_1^-} - \boldsymbol{\delta v_1^+}(\theta)| \tag{4.25}$$

$$\Delta v_2 = \Delta v_2(\theta) = |\boldsymbol{\delta v_2^+}(\theta) - \boldsymbol{\delta v_2^{++}(\theta)}| \tag{4.26}$$

$$0 \leq \theta < 2\pi \tag{4.27}$$

Where $\boldsymbol{\delta v_1^-}$ is the initial relative velocity of the Deputy, $\boldsymbol{\delta v_1^+(\theta)} = A_1\boldsymbol{\delta r_1} + B_1\boldsymbol{\delta r_2}$ is the initial velocity correction given by the LLS, $\boldsymbol{\delta v_2^+(\theta)} = A_2\boldsymbol{\delta r_1} + B_2\boldsymbol{\delta r_2}$ is the final velocity correction given by LLS [93] (assuming $\delta t = 0$), and $\boldsymbol{\delta v_2^{++}(\theta)} = C\boldsymbol{\delta r_2}$ is the velocity required for closed relative orbit (given as dimensionalized components in the LVLH), i.e,

$$\mathbf{C} = \dot{f} \begin{bmatrix} 0 & 1 - \frac{1}{e\cos(f)+2} & 0 \\ -\left(\frac{1}{e\cos(f)+1} + 1\right) & \frac{e\sin(f)}{2+3e\cos(f)+(e\cos(f))^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.28}$$

The radii $\boldsymbol{\delta r_1}$ and $\boldsymbol{\delta r_2}$ are the initial relative radius at the beginning of the transfer and the targeted final radius, respectively.

Taking consecutive derivatives of the cost (equation 4.24) wrt the parameter θ yields,

$$\frac{\partial}{\partial\theta}(J) = -\left(\boldsymbol{\delta v_1^-} - \boldsymbol{\delta v_1^+}\right)^T \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_1^+}\right) + \left(\boldsymbol{\delta v_2^+} - \boldsymbol{\delta v_2^{++}}\right)^T \left(\frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^+}\right) + \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^{++}}\right)\right) \tag{4.29}$$

$$\begin{aligned} \frac{\partial^2}{\partial\theta^2}(J) = {} & \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_1^+}\right)^T \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_1^+}\right) + \left(\boldsymbol{\delta v_1^-} - \boldsymbol{\delta v_1^+}\right)^T \frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_1^+}\right) \\ & + \left(\frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^+}\right) + \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^{++}}\right)\right)^T \left(\frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^+}\right) + \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^{++}}\right)\right) \\ & + \left(\boldsymbol{\delta v_2^+} - \boldsymbol{\delta v_2^{++}}\right)^T \left(\frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_2^+}\right) + \frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_2^{++}}\right)\right) \end{aligned} \tag{4.30}$$

$$\text{Where, } \frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_1^+}\right) = \mathbf{B}_1\frac{\partial}{\partial\theta}(\boldsymbol{\delta r_2}), \quad \frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_1^+}\right) = \mathbf{B}_1\frac{\partial^2}{\partial\theta^2}(\boldsymbol{\delta r_2}) \tag{4.31}$$

$$\frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^+}\right) = \mathbf{B}_2\frac{\partial}{\partial\theta}(\boldsymbol{\delta r_2}), \quad \frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_2^+}\right) = \mathbf{B}_2\frac{\partial^2}{\partial\theta^2}(\boldsymbol{\delta r_2}) \tag{4.32}$$

$$\frac{\partial}{\partial\theta}\left(\boldsymbol{\delta v_2^{++}}\right) = \mathbf{C}\frac{\partial}{\partial\theta}(\boldsymbol{\delta r_2}), \quad \frac{\partial^2}{\partial\theta^2}\left(\boldsymbol{\delta v_2^{++}}\right) = \mathbf{C}\frac{\partial^2}{\partial\theta^2}(\boldsymbol{\delta r_2}) \tag{4.33}$$

$$\frac{\partial}{\partial\theta}(\boldsymbol{\delta r_2}) = \rho\left[-\sin(\theta), \cos(\theta), 0\right]^T, \quad \frac{\partial^2}{\partial\theta^2}(\boldsymbol{\delta r_2}) = -\rho\left[\cos(\theta), \sin(\theta), 0\right]^T \tag{4.34}$$

Substituting these values into equation 4.29 and 4.30 gives the derivatives of the cost function J as a function of θ. Mathematically, equating $\frac{\partial}{\partial\theta}(J) = 0$ and evaluating the $\frac{\partial^2}{\partial\theta^2}(J)$ at its solution values gives us the global extremum values (minimum or maximum) of the cost function J.

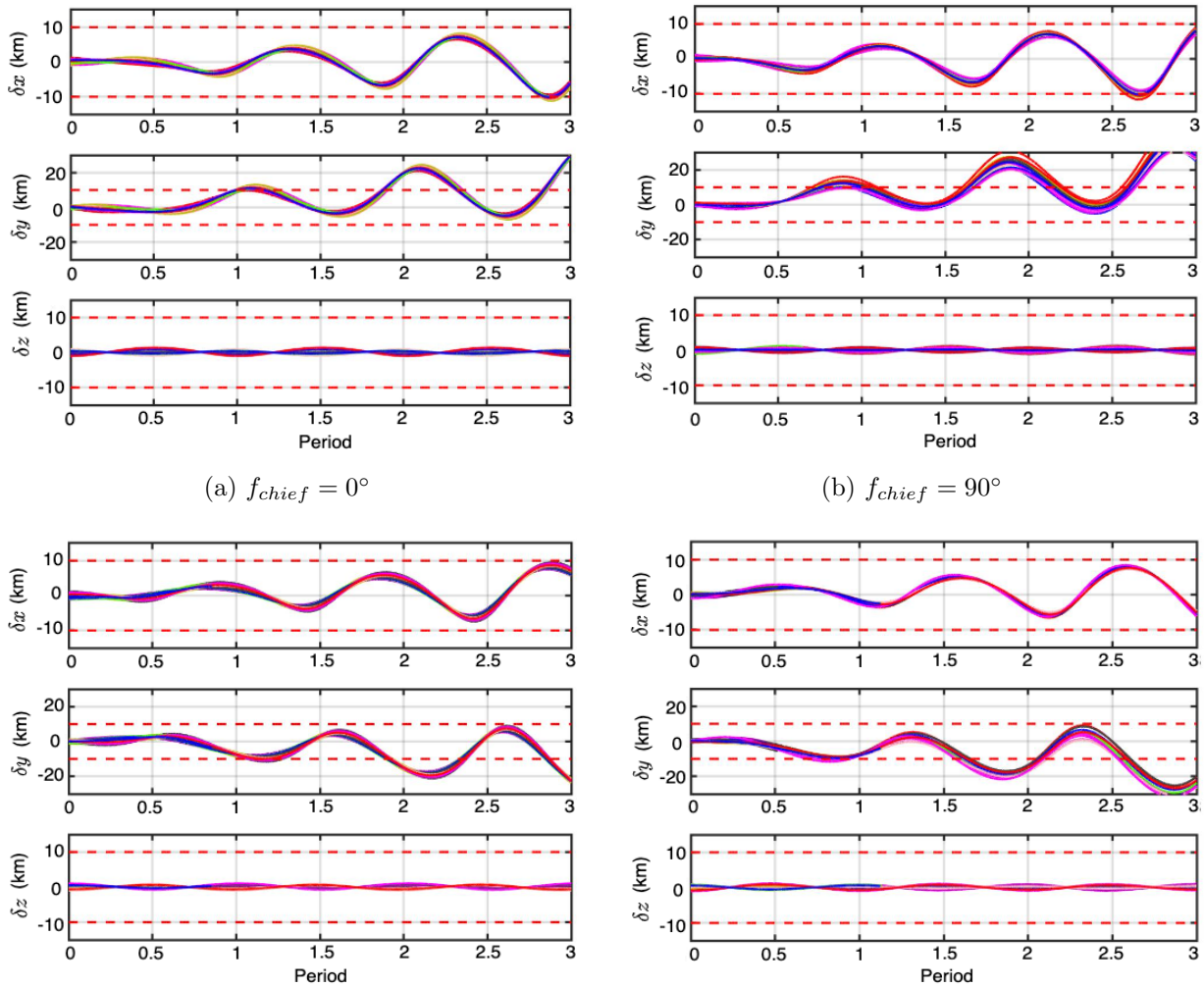
Equations 4.16 and 4.17 can be leveraged to generate closed relative motion of the deputy around the chief spacecraft. With the addition of SRP and J2 perturbations, it is well known that the relative motion will no longer stay closed and the two spacecraft will eventually drift apart. However, analysis shows that for a Deputy starting ≈ 1.5 km away from the Chief, the drift between the two spacecraft is rather slow as can be seen from the Monte Carlo simulation results in Figure 4.5. Thus for the current example scenario, setting our target condition in a closed (Keplerian) relative orbit for a deputy starting 1.5 km away from the Chief ($\rho = 1.5$) will allow us to sufficiently delay any correction burn and hence save on fuel requirements of the mission.

4.2.3 Implementation and results

The deputy is arbitrarily initialized in a closed relative orbit around the virtual chief such that it is within the 10 km relative separation bound. The total time of flight used for the simulation is 2 days. During this period, when the constraint checker detects a violation of the maximum relative separation bound, the optimal targeting angle is computed to define the target point for deputy reconfiguration. To compute this optimal targeting angle, the values for matrices A_1, A_2, B_1, B_2, and C are computed from the simulation at a point when the Deputy is exiting the constraints sphere, and the cost function is investigated for different values of θ using equation 4.24. Figure 4.6a shows that a global minimum of the cost function exists and Figure 4.6b confirms the same as $\frac{\partial}{\partial\theta}(J) = 0$ and $\frac{\partial^2}{\partial\theta^2}(J) > 0$ at the same point.

Conceptually, this optimal target angle (θ) provides a point at which the Deputy velocity will be tangent to the closed relative orbit around the virtual chief, allowing for the $\boldsymbol{\delta v_2^{++}}$ velocity change to be minimum. This idea of tangential velocity can be better understood by contrasting results in Figure 4.7 and Figure 4.8.

In Figure 4.7, a random targeting angle θ is used to define the target state for reconfiguration.



(a) $f_{chief} = 0°$

(b) $f_{chief} = 90°$

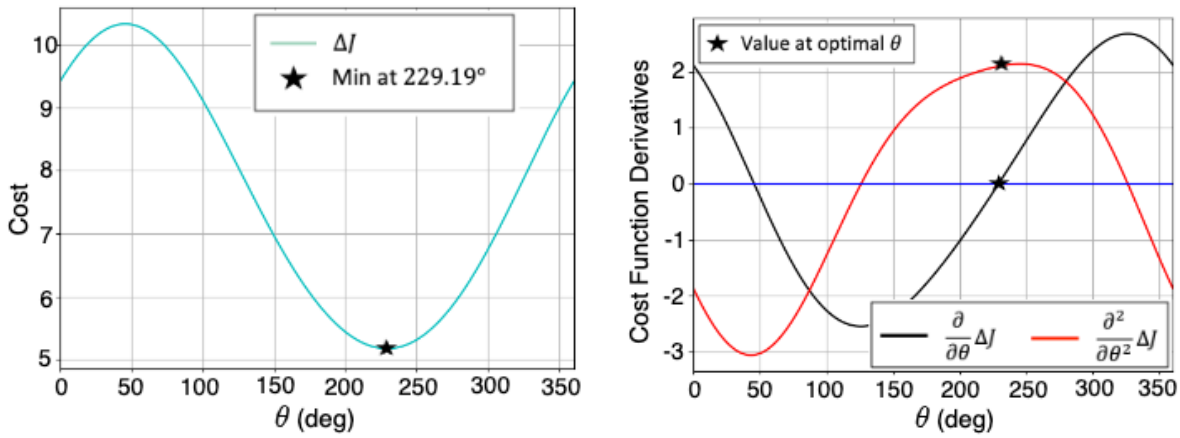
(c) $f_{chief} = 180°$

(d) $f_{chief} = 270°$

Figure 4.5: Slowly drifting perturbed relative position of a Deputy starting 1.5 km away from the Chief spacecraft for different Chief's true anomaly (f_{chief}).

As can be seen, the velocity needed to be in the closed relative orbit, in this case, is not aligned with the Deputy velocity. As a result, the spacecraft has to drastically change its direction of motion. However, in Figure 4.8, the optimal targeting angle is calculated and hence the deputy spacecraft transitions smoothly from the transfer orbit to the closed relative orbit. The use of optimal targeting angle, in this case, allowed $\delta v_{save} \approx 0.6 m/sec$, where δv_{save} is the delta-v saved during the transfer.

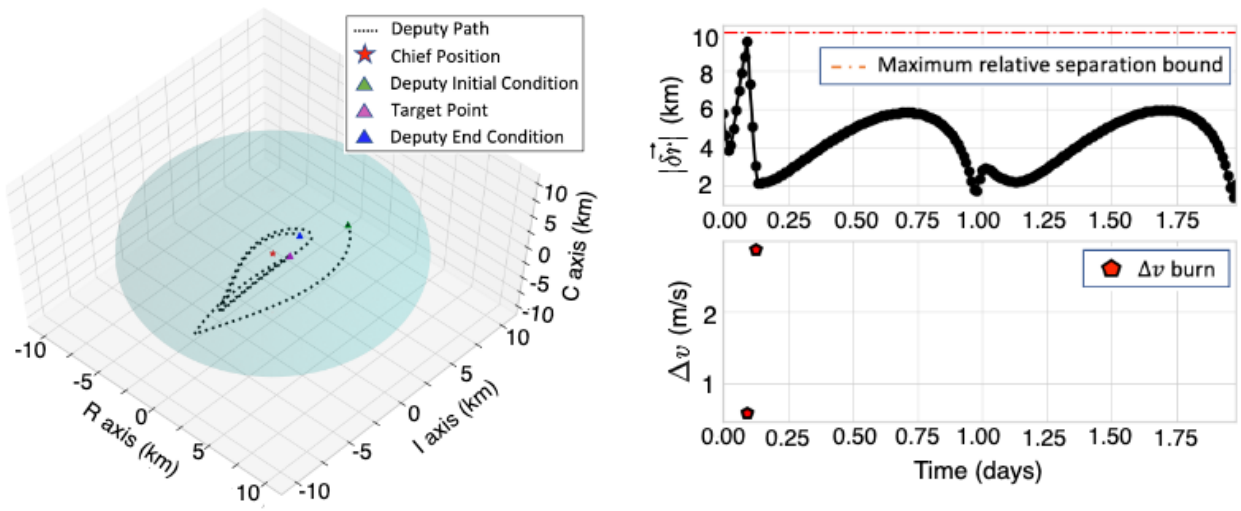
The performance of this algorithm was tested for a longer period of 50 days and the results



(a) Cost Function vs θ

(b) Cost Function Derivatives vs θ

Figure 4.6: Cost Function minimization for calculation of optimal angle for fuel-efficient closed-loop LLS transfer.



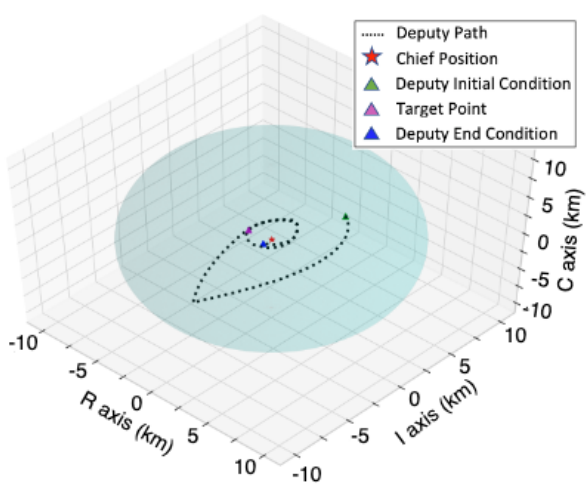
(a) 3-D Relative trajectory

(b) Deputy's relative position over time and computed Δv burns

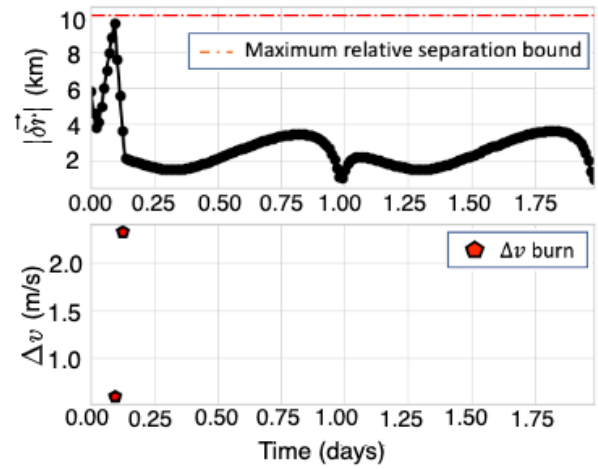
Figure 4.7: Non-optimal angle spacecraft re-targeting using closed-loop LLS for maintaining maximum relative separation constraint for 2 days. **Total Δv** of the transfer = 3.5 m/s.

are shown in Figure 4.9.

In this case, the developed targeting algorithm allowed the deputy to stay within the maximum separation bounds with $\delta v_{save} \approx 13 m/sec$.

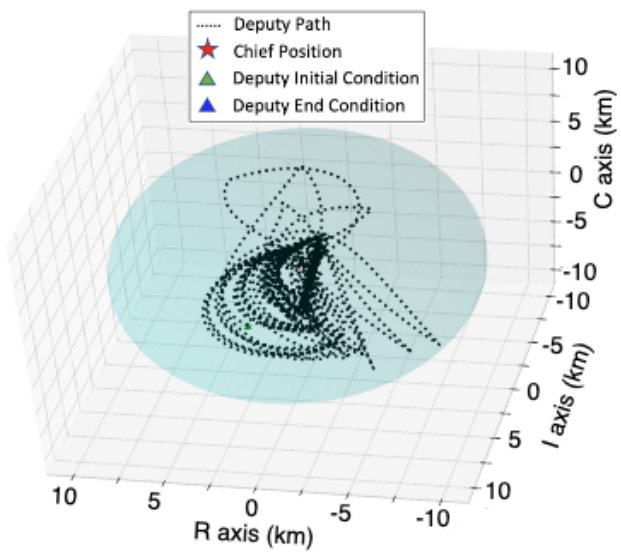


(a) 3-D Relative trajectory

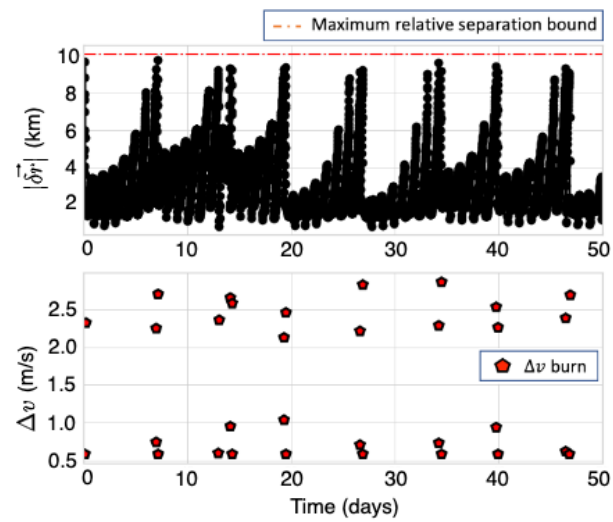


(b) Deputy's relative position over time and computed Δv burns

Figure 4.8: Optimal angle spacecraft re-targeting using closed-loop LLS for maintaining maximum relative separation constraint for 2 days. **Total Δv** for the transfer = 2.9 m/s.



(a) 3-D Relative trajectory



(b) Deputy's relative position over time and computed Δv burns

Figure 4.9: Maintaining maximum relative separation constraint for 50 days using optimal angle spacecraft re-targeting using closed-loop LLS. **Total Δv** for the transfer = 50.6 m/s.

4.3 Using closed-loop LLS for safe spacecraft reconfiguration in a cluttered environment

4.3.1 Scenario description

The re-targeting strategy discussed in section 4.2.2 for a single spacecraft can be extended to when the environment is cluttered with obstacles/keep-out zones (can be multiple spacecraft in the scenario).

For demonstration purposes, an example scenario similar to Case 3 as described in section 2.3 is used where a deputy spacecraft attempts to safely reconfigure itself from an initial relative orbit to a final relative orbit around a chief spacecraft. The transfer must be collision-free with the chief and any other known obstacles present in the scenario (defined keep-out zones) and the deputy should also use less fuel during the transfer. This scenario has been reproduced here in Figure 4.10 for reference. As seen in Figure 4.10, this particular example scenario has three obstacles/keep-out

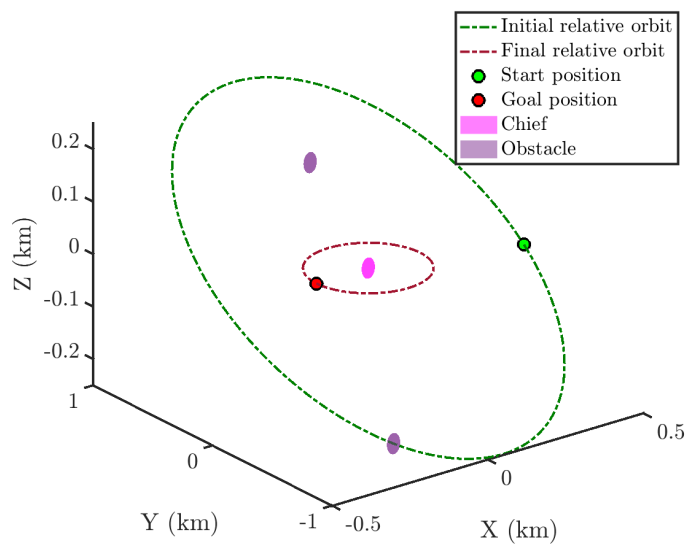


Figure 4.10: Relative orbit transfer scenario description.

zones for the deputy spacecraft to avoid (including the chief). The classical orbital elements (COEs) of the chief and the initial and final relative orbital elements of the deputy spacecraft with respect to the chief (δCOE_i and δCOE_f respectively) and the initial and final relative states of the deputy

spacecraft are shown in Tables 2.2 and Table 2.3 respectively.

To demonstrate the performance of the closed-loop LLS in this case, a two-step approach is considered. In the first step, a collision-free and fuel-efficient transfer solution (reference trajectory) is computed for spacecraft reconfiguration considering Keplerian dynamics using the efficient AIKMP algorithm. And then in the next step, the closed-loop LLS is used as the guidance law to allow the spacecraft to closely follow the planned path in the presence of external perturbations.

4.3.2 Implementation and results

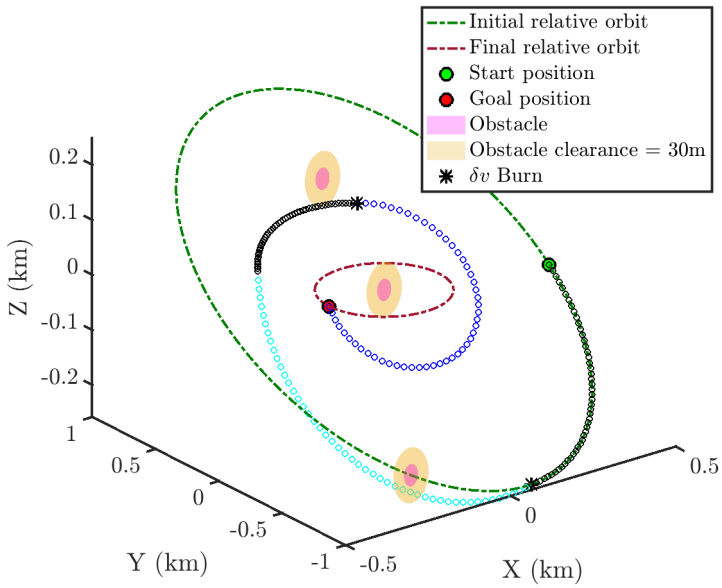
Step 1: Generating a motion plan in a cluttered environment considering Keplerian dynamics:

In this step of the implementation, the AIKMP algorithm is used to generate the motion plan (reference trajectory) using Keplerian dynamics [47, 46]. For generating the motion plan, an inflated keep-out zone is considered for the AIKMP algorithm. Specifically, for the known spherical keep-out zones of radius = 20 m (as considered before), an additional obstacle clearance region of 30 m is considered, which essentially means that the generated motion planning solution is guaranteed to have a clearance of 30 m from the given keep-out zones/obstacles.

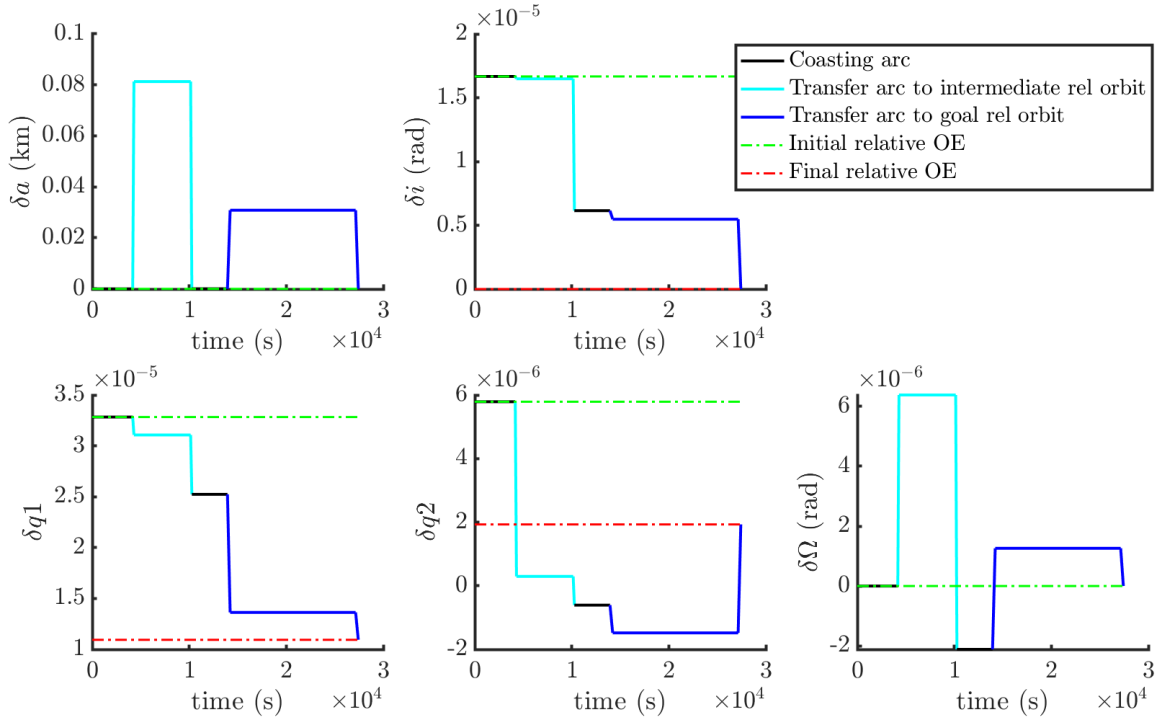
The collision-free and fuel-efficient motion plan computed by the AIKMP algorithm is shown in Figure 4.11. In these figures, the black color refers to coasting arcs, the cyan color refers to Lambert transfer to a randomly explored orbit and the blue color refers to the Lambert transfer to the goal orbit. Figure 4.11b shows how the relative orbit elements of the deputy change over the length of the final transfer solution.

Step 2: Closed-loop LLS-based guidance:

Closed-loop LLS-based guidance is developed to follow the motion plan generated in Step 1 closely in the presence of SRP, and drag perturbations. Also, it is to be noted that the AIKMP transfer solution is a combination of multiple sub-arcs (as described in Figure 4.11) resulting from its stochastic tree-based framework [44]. In this step of the implementation, the LLS solution is used to compute the transfer in small sections defined by these sub-arcs. This means that an LLS



(a) Transfer solution in 3D relative position space.



(b) Relative orbital elements of the reference trajectory vs time of flight

Figure 4.11: Reference trajectory generated by the AIKMP algorithm considering Keplerian dynamics in Step 1. The **total Δv** required for the transfer = 19.22 cm/sec

.

solution is computed to transfer the deputy from the beginning of a sub-arc to the end of the same sub-arc and this is repeated until all the sub-arcs are covered.

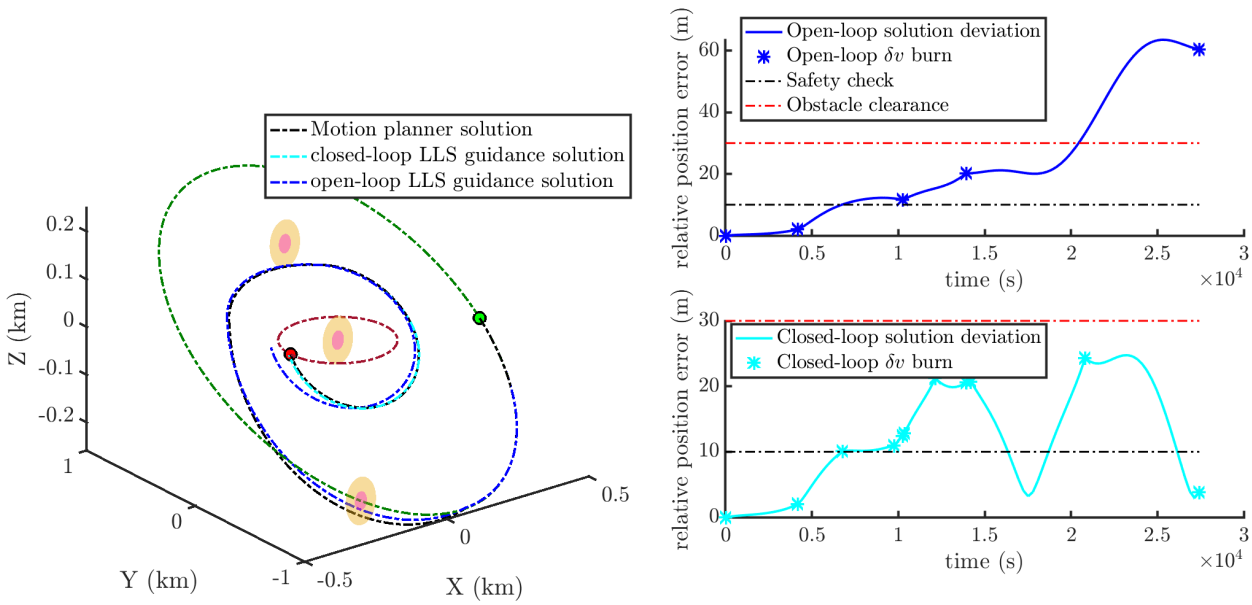
In Step 1, the generated reference trajectory (considering Keplerian dynamics) has a clearance of 30 m from the defined keep-out zones. Thus, when following the generated motion plan in the presence of external perturbations, there is an allowable tolerance of 30 m around the planned transfer trajectory to still guarantee collision avoidance by the final transfer solution. For safety guarantees of the closed-loop LLS implementation, a 'safety check' value is picked such that:

$$\text{Safety check tolerance} < \text{clearance of the trajectory} \tag{4.35}$$

In this case, a safety check tolerance of 10 m ($<$ 30 m trajectory clearance) is considered to activate the correction burns by the guidance algorithm. The closed-loop guidance algorithm may use different strategies to pick a 'refresh rate' to check if the safety check tolerance has been violated (happens when the deputy trajectory has deviated from the motion plan beyond the safety check tolerance). In our implementation, each sub-arc in the generated motion plan is divided into 50 equal time intervals and this is used as the refresh rate for each sub-arc. This strategy is solely used for ease of implementation because the AIKMP motion planner in Step 1 was set up to store 50 nodes corresponding to each sub-arc. Alternatively, a fixed time interval can also be used across all sub-arcs as the refresh rate. If the safety check tolerance is violated, the closed-loop LLS algorithm computes an LLS correction burn from the present state of the deputy to a target state in the generated motion plan. This can be any target state in the reference trajectory that is within the obstacle clearance region. By doing so, every time the deputy is close to violating the collision-avoidance constraints, an LLS correction burn is applied to bring the deputy back to the collision-safe zone. It is to be noted that, in implementing closed-loop LLS for collision avoidance, picking a suitable refresh rate (frequency of constraint violation checking) is very important. If the refresh rate is too spread out/large, this may lead to constraint violation for a longer time. If the refresh rate is too small, this will lead to using too many correction burns because it takes some time for an implemented burn to move the trajectory such that the violation no longer occurs.

Using a smaller refresh rate will not only increase the total cost of the transfer solution but will also lead to higher computation time required by the routine. Picking the target states for the LLS correction burns is also crucial to guarantee successful collision avoidance. The choice of the refresh rate and the target state for LLS correction burn depends on the mission requirements and will vary from problem to problem.

To demonstrate the advantages of using a closed-loop LLS over an open-loop LLS, the latter is also implemented for comparison of results. In the open loop implementation, regular LLS guidance is used to compute the transfer solution from one node to another node in the final desired transfer trajectory without any feedback on the violation of collision avoidance constraints. The results of the implementation are shown in Figure 4.12.



(a) Transfer trajectories in 3D relative position space.

(b) Deviation of open-loop and closed-loop LLS solution trajectories from the reference trajectory.

Figure 4.12: Comparison of open-loop and closed-loop LLS guidance performance in tracking the reference trajectory in the presence of external perturbations.

As can be seen in Figure 4.12b, the LLS computes the transfer solution following each of the sub-arcs in the AIKMP solution, one sub-arc at a time. Since the open-loop LLS guidance has no feedback, no correction burn is applied to prevent the deputy from violating the collision-avoidance

constraints. As a result, the open-loop solution not only results in a final targeting error of ≈ 60 m to the goal state (as seen in figures 4.12a and 4.12b) but this transfer solution also has the possibility of collision with an obstacle. However, for the closed-loop LLS solution, the guidance law receives feedback whenever the collision-avoidance constraints are violated and a correction burn is applied to bring the deputy back to the collision-safe zone hence guaranteeing collision safety of the transfer. In the demonstrated example, the final targeting error with the closed-loop LLS ≈ 4 m. Also, the cost of the open-loop solution = 20.78 cm/sec, whereas the cost of the closed-loop solution = 24.18 cm/sec.

Implementation with multiple large obstacles in the scenario:

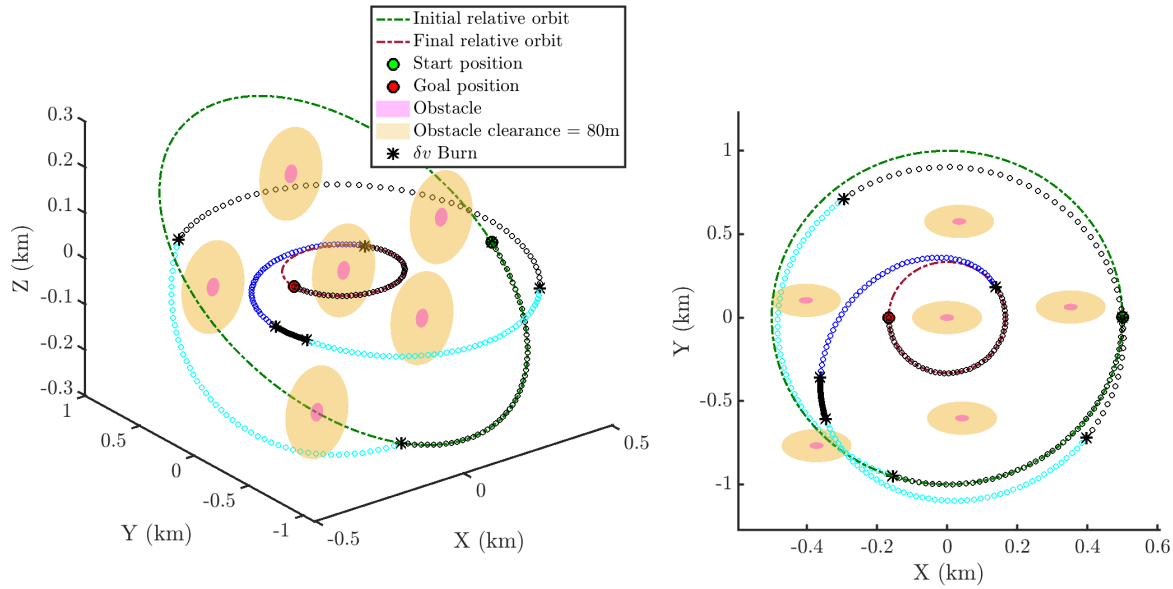
To demonstrate the performance of the closed-loop LLS guidance in a very cluttered environment, multiple obstacles (spheres of radius = 20 m) are introduced in the previous scenario and an obstacle clearance region of 80 m around the obstacles is imposed. The AIKMP algorithm is then used to compute a safe and fuel-efficient transfer trajectory considering Keplerian dynamics. The resulting motion plan is shown in Figures 4.13 and 4.14.

The developed closed-loop LLS guidance is then used with a safety check tolerance of 50 m to implement the generated motion plan with the same perturbations as before. A comparison of results using a closed-loop LLS guidance over an open-loop LLS is shown in Figure 4.15.

Similar observations are made from Figure 4.15 that an open-loop LLS guidance is unable to provide safety guarantees and in this case, the final implemented trajectory is violating the defined relative-separation constraints. However, a closed loop around the LLS guidance solution is able to provide timely input to the system and initiate correction burns so that relative separation constraints are satisfied. In this case, the cost of the open-loop solution = 36.02 cm/sec, and the cost of the closed-loop solution = 36.85 cm/sec.

4.4 Summary

In this work, the application of the LLS algorithm has been extended to non-Keplerian environments. With a GEO reference orbit example case, it was demonstrated that a closed-loop



(a) Transfer solution in 3D relative position space.

(b) In-plane motion in relative position space

Figure 4.13: Motion plan (reference trajectory) as computed by the AIKMP algorithm with larger and more obstacles. The **total Δv** required for the transfer = 21.24 cm/s.

system with the LLS can significantly improve the targeting accuracy in relative guidance problems in the presence of non-Keplerian perturbations. After the application of two correction burns to the LLS solution in this case, it was observed an improvement of targeting accuracy by $\approx 86\%$ in the presence of perturbations like SRP, drag, and J2. A planning algorithm was developed to find the optimal relative target state in order to satisfy the maximum relative separation constraints of a deputy spacecraft with respect to a reference trajectory. By presenting theoretical developments backed with multiple simulation results, it was shown that using the closed-loop LLS in conjunction with the planning algorithm can improve the fuel requirement of re-targeting maneuvers. In fact, it was shown that using the planning algorithm with the closed-loop LLS can help save $\approx 26\%$ fuel over a duration of 50 days for a maximum relative-separation constrained re-targeting scenario. With the help of more example scenarios, it was shown that closed-loop LLS can also be used in conjunction with complicated orbital motion planners - in particular AIKMP - providing improved targeting accuracy of $\approx 93\%$ at a cost of $\approx 16\%$ fuel increase as compared to open-loop guidance.

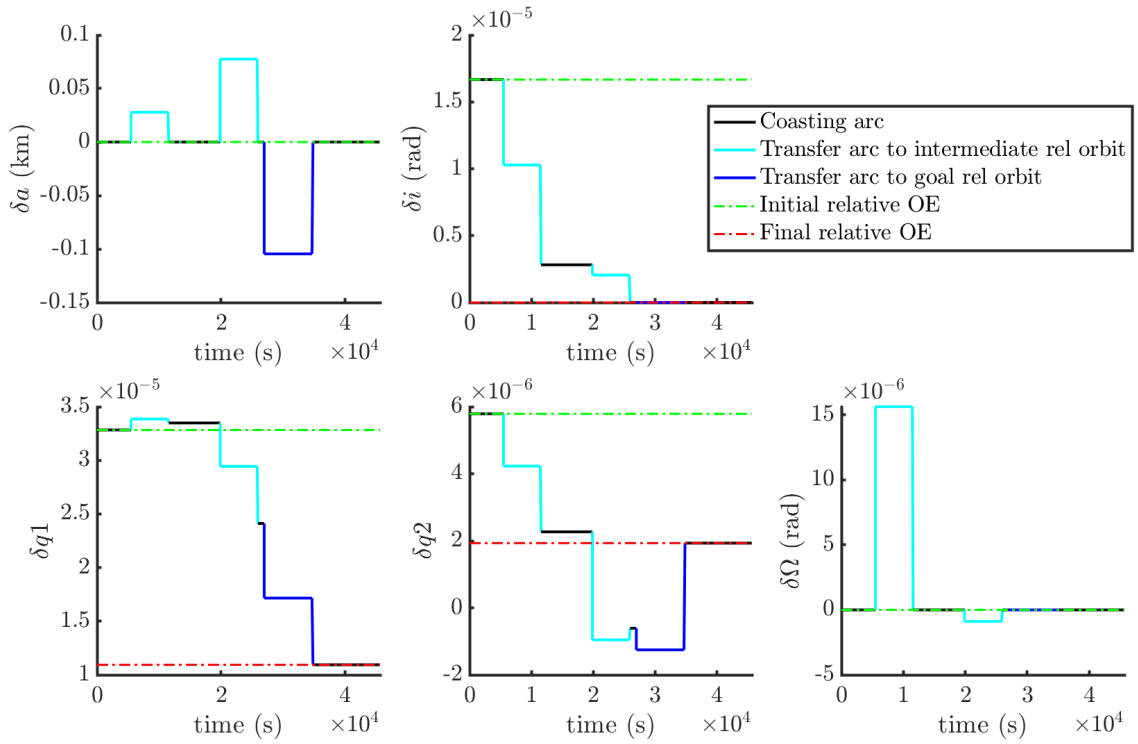
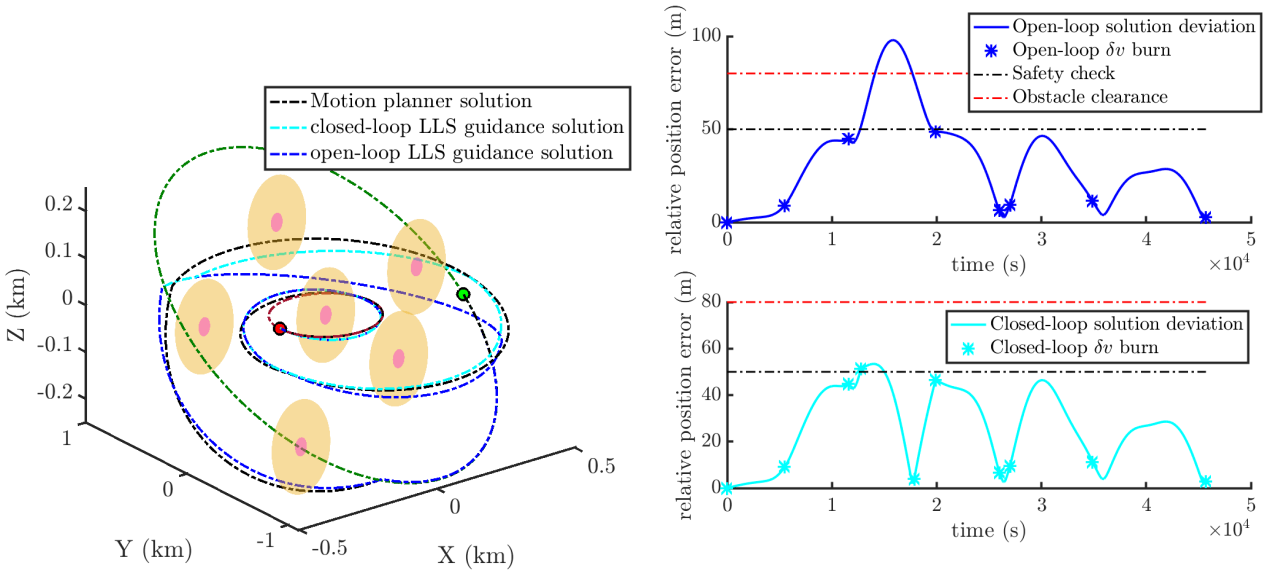


Figure 4.14: Relative orbital elements of the motion plan (reference trajectory) generated by the AIKMP algorithm versus time for the scenario with multiple large obstacles.

It was also demonstrated that closed-loop LLS guidance can enable a spacecraft to closely follow a reference trajectory to ensure safety within targeted keep-out zones at the cost of $\approx 3\%$ fuel increase as compared to open-loop guidance in a cluttered and perturbed environment. This targeting accuracy and cost of fuel increase can be adjusted depending on mission requirements by the choice of closed-loop LLS guidance parameters like the refresh rate and the target state for the LLS correction burn.



(a) Transfer trajectories in 3D relative position space.

(b) Deviation of open-loop and closed-loop LLS solution trajectories from the reference trajectory.

Figure 4.15: Comparison of open-loop and closed-loop LLS guidance performance in tracking the reference trajectory in the presence of multiple obstacles and perturbations.

Chapter 5

AIKMP with passive collision avoidance

In the previous chapters, the framework of the AIKMP algorithm was fully developed, and it was demonstrated that the algorithm can quickly and very efficiently compute feasible transfer solutions for relative motion planning problems in very cluttered environments (in the presence of multiple obstacles/keep-out zones that are static in the chief-centered relative frame). Spacecraft relative motion planning in the presence of static obstacles has important applications in satellite servicing missions and in spacecraft proximity operations where the chief spacecraft/ primary body may have several keep-out zones that the deputy needs to avoid. However, in more dynamic multi-spacecraft systems, such as in spacecraft formation flying, multiple spacecraft are maneuvering and operating in close proximity to each other. In such scenarios, the static obstacle assumption in the chief-centered relative frame is very limiting and does not always hold true. Moreover, the AIKMP algorithm implementation results (section 2.3 and section 3.2) had shown that most of the transfer arcs used by the algorithm comprise of non-zero semi-major axis differences ($\delta a \neq 0$) with respect to the chief, indicating that the relative motion of the deputy will not remain bounded in those transfer arcs. Depending on the time of flight and the magnitude of the δa of these transfer arcs, the deputy's relative motion will drift away from the chief and this can pose a possible collision threat with other spacecraft/moving obstacles in the scenario. Thus, it is very important to augment the AIKMP algorithm with collision avoidance capabilities not just with static obstacles but also with moving obstacles.

Different collision avoidance strategies exist that include reactive methods [109], proactive

methods [121], and passive collision avoidance methods [71, 69, 37]. In general, "reactive" implies that spacecraft must recalculate trajectories in real-time to avert collisions following a contingency. On the other hand, "proactive" can be seen as a refinement of reactive, ensuring the availability of a controlled escape trajectory or maneuver at all times, even in the face of a partial degradation, not a complete loss, of control capabilities. Lastly, "passive" denotes that controlled trajectories have been pre-arranged to ensure safe separation for collision avoidance guarantees. Examples of reactive and proactive collision avoidance strategies in the literature are relatively recent [109, 121]. Whereas the first examples of passive-safe strategies date back to the 60s with the co-elliptic rendezvouses of the Apollo missions [137]. In this chapter, the theory behind E-I vector separation [56, 37] is leveraged to derive a passive collision avoidance strategy that can be infused with the tree extension step of the AIKMP algorithm. With the help of simulation results, it is demonstrated that the resulting AIKMP algorithm picks the intermediate random nodes for tree extension in a more informed way (as compared to the previous versions described in Chapter 2 and Chapter 3), such that obstacle tolerance constraints are satisfied for the entire duration of the transfer in the presence of multiple moving obstacles – a capability that is crucial for spacecraft formation flying applications.

5.1 E/I vector separation theory

The concept of eccentricity and inclination (E/I) vector separation was originally developed to achieve safe collocation of geostationary satellites [56] which was later generalized and extended to low-Earth orbit (LEO) formations [37]. This concept is based on the consideration that uncertainty in predicting the along-track separation between two spacecraft is much higher than the radial and cross-track components. Small uncertainties in orbit determination errors and maneuver execution errors can lead to secularly growing along-track errors. Therefore, for collision avoidance in the presence of in-track position uncertainties, the two spacecraft must be properly separated in the radial and normal directions. Several works in the past have looked into using this strategy of collision avoidance assuming bounded relative motion between two spacecraft ($\delta a = 0$) and have

concluded that a parallel (or anti-parallel) alignment of relative eccentricity and inclination vectors is sufficient to guarantee a relative separation between the two spacecraft in the radial-normal plane. Relative eccentricity vector $\vec{\delta \boldsymbol{e}}$ and relative inclination vector $\vec{\delta \boldsymbol{i}}$ are defined as,

$$\vec{\delta \boldsymbol{e}} = \begin{pmatrix} \delta e_x \\ \delta e_y \end{pmatrix} = \delta e \begin{pmatrix} cos\phi \\ sin\phi \end{pmatrix} \tag{5.1}$$

$$\vec{\delta \boldsymbol{i}} = \begin{pmatrix} \delta i_x \\ \delta i_y \end{pmatrix} = \delta i \begin{pmatrix} cos\theta \\ sin\theta \end{pmatrix} \tag{5.2}$$

where the relative angular location of the perigee (ϕ) and the ascending node of the relative orbit (θ) are defined as:

$$\phi = \text{atan}(\frac{\delta e_y}{\delta e_x}) \tag{5.3}$$

$$\theta = \text{atan}(\frac{\delta i_y}{\delta i_x}) \tag{5.4}$$

The angle $\theta - \phi$ is the angle enclosed by $\vec{\delta \boldsymbol{e}}$ and $\vec{\delta \boldsymbol{i}}$.

In this work, an analytical solution will be derived to describe the relative separation between two drifting ($\delta a \neq 0$) spacecraft in the radial-normal plane in terms of relative orbital elements. Based on the derived solution, two approaches will be detailed that can be used to pick a target safe relative orbit that is guaranteed to avoid collision with moving obstacles present in the scenario for a desired amount of time.

5.2 Relative separation between two spacecraft with drifting relative motion ($\delta a \neq 0$)

Apart from the relative eccentricity vector ($\vec{\delta \boldsymbol{e}}$), relative inclination vector ($\vec{\delta \boldsymbol{i}}$) and relative semi-major axis (δa), the relative mean longitude ($\delta \lambda$) and relative mean argument of latitude (δu) also affect the relative motion of satellites in a formation. As a result, for this analysis, we define the relative orbital element set of the deputy with respect to the chief (δOE) as:

$$\delta OE = (\delta a, \delta \lambda, \delta i, \delta e_x, \delta e_y, \delta \Omega) \tag{5.5}$$

where,

$$\delta\lambda = \delta u + \delta\Omega cos(i) \tag{5.6}$$

$$u = \omega + M \tag{5.7}$$

ω being the argument of periapsis and M the mean anomaly.

Using the Hill-Clohessy-Wiltshire definition of relative motion, the relative motion between two spacecraft can be described in terms of the orbit element difference as [55, 34, 111]:

$$\begin{aligned} \delta r_R &= \delta a - a\delta e_x cos(u) - a\delta e_y sin(u) \\ &= \delta a - a\delta e cos(\phi) cos(u) - a\delta e_y sin(\phi) sin(u) \\ \implies & \boxed{\delta r_R = \delta a - a\delta e cos(u-\phi)} \end{aligned} \tag{5.8}$$

$$\boxed{\delta r_I = -2a sin(u-\phi)\delta e + a\delta\lambda_0 - \frac{3}{2}(u-u_0)\delta a} \tag{5.9}$$

$$\begin{aligned} \delta r_N &= -a\delta i sin(\theta) cos(u) + a\delta i cos(\theta) sin(u) \\ \implies & \boxed{\delta r_N = a\delta i sin(u-\theta)} \end{aligned} \tag{5.10}$$

where $\delta r_R, \delta r_I, \delta r_N$ are the hill frame position coordinates (radial, in-track, and normal respectively) expressed in terms of relative orbital element differences.

Using equations 5.8 and 5.10, the separation between two spacecraft as projected in the radial-normal plane can be computed as:

$$\begin{aligned} \delta r_{RN} &= \sqrt{\delta r_R^2 + \delta r_N^2} \\ &= \sqrt{(\delta a - a\delta e cos(u-\phi))^2 + (a\delta i sin(u-\theta))^2} \\ &= \sqrt{\delta a^2 - 2a\delta a\delta e cos(u-\phi) + a^2\delta e^2 cos^2(u-\phi) + a^2\delta i^2 sin^2(u-\theta)} \end{aligned} \tag{5.11}$$

Since $-1 \leq cos(u-\phi) \leq 1$, from equation 5.11, we have:

$$\begin{aligned} \delta r_{RN} &\geq \sqrt{\delta a^2 - 2a\delta a\delta e + a^2\delta e^2 cos^2(u-\phi) + a^2\delta i^2 sin^2(u-\theta)} \\ &\geq \sqrt{\frac{2\delta a^2 - 4a\delta a\delta e + a^2\delta e^2 + a^2\delta i^2 + a^2\delta e^2 cos2(u-\phi) - a^2\delta i^2 cos2(u-\theta)}{2}} \end{aligned} \tag{5.12}$$

Now,

$$
\begin{aligned}
&[a^2\delta e^2 cos2(u-\phi) - a^2\delta i^2 cos2(u-\theta)]^2 + [a^2\delta e^2 sin2(u-\phi) - a^2\delta i^2 sin2(u-\theta)]^2 \\
&= a^4\delta e^4 + a^4\delta i^4 - 2a^4\delta e^2\delta i^2 cos2(u-\phi)cos2(u-\theta) - 2a^4\delta e^2\delta i^2 sin2(u-\phi)sin2(u-\theta) \\
&= a^4\delta e^4 + a^4\delta i^4 - 2a^4\delta e^2\delta i^2 cos2(\theta-\phi)
\end{aligned}
\tag{5.13}
$$

Here, $(\theta - \phi)$ is the angle enclosed between $\vec{\boldsymbol{\delta e}}$ and $\vec{\boldsymbol{\delta i}}$.

Since $[a^2\delta e^2 sin2(u-\phi) - a^2\delta i^2 sin2(u-\theta)]^2 \geq 0$, equation 5.13 gives,

$$
\begin{aligned}
&|a^2\delta e^2 cos2(u-\phi) - a^2\delta i^2 cos2(u-\theta)| \leq \sqrt{a^4\delta e^4 + a^4\delta i^4 - 2a^4\delta e^2\delta i^2 cos2(\theta-\phi)} \\
\implies &a^2\delta e^2 cos2(u-\phi) - a^2\delta i^2 cos2(u-\theta) \geq -\sqrt{a^4\delta e^4 + a^4\delta i^4 - 2a^4\delta e^2\delta i^2 cos2(\theta-\phi)}
\end{aligned}
\tag{5.14}
$$

Substituting equation 5.14 in equation 5.12,

$$
\frac{\sqrt{2}\delta r_{RN}}{a} \geq \sqrt{2\Delta a^2 - 4\Delta a\delta e + \delta e^2 + \delta i^2 - \sqrt{\delta e^4 + \delta i^4 - 2\delta e^2\delta i^2 cos2(\theta-\phi)}}
\tag{5.15}
$$

where, $\Delta a = \frac{\delta a}{a}$.

Solving for $\sqrt{\delta e^4 + \delta i^4 - 2\delta e^2\delta i^2 cos2(\theta-\phi)}$, we have

$$
\begin{aligned}
\delta e^4 + \delta i^4 - 2\delta e^2\delta i^2 cos2(\theta-\phi) &= (\delta e^2)^2 + (\delta i^2)^2 - 2\delta e^2\delta i^2 cos2(\theta-\phi) \\
&= (\delta e^2)^2 + (\delta i^2)^2 - 2\delta e^2\delta i^2[2cos^2(\theta-\phi) - 1] \\
&= (\delta e^2)^2 + (\delta i^2)^2 + 2\delta e^2\delta i^2 - 4\delta e^2\delta i^2 cos^2(\theta-\phi) \\
&= (\delta e^2 + \delta i^2)^2 - (2\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2 \\
&= (\delta e^2 + \delta i^2 + 2\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})(\delta e^2 + \delta i^2 - 2\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}}) \\
&= |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}|^2 |\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|^2 \\
\implies \sqrt{\delta e^4 + \delta i^4 - 2\delta e^2\delta i^2 cos2(\theta-\phi)} &= |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|
\end{aligned}
\tag{5.16}
$$

Simplifying equation 5.15 using equation 5.16 gives,

$$
\begin{aligned}
\frac{\sqrt{2}\delta r_{RN}}{a} &\geq \sqrt{2\Delta a^2 - 4\Delta a\delta e + \delta e^2 + \delta i^2 - |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|} \\
\implies (\frac{\sqrt{2}\delta r_{RN}}{a})^2 &\geq 2\Delta a^2 - 4\Delta a\delta e + \delta e^2 + \delta i^2 - |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}| \\
&\geq 2\Delta a(\Delta a - 2\delta e) + \frac{(\delta e^2 + \delta i^2 - |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|)(\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|)}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|} \\
&\geq 2\Delta a(\Delta a - 2\delta e) + \frac{(\delta e^2 + \delta i^2)^2 - (|\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|)^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|} \\
&\geq 2\Delta a(\Delta a - 2\delta e) + \frac{4(\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|} \\
\implies \delta r_{RN} &\geq a\sqrt{\Delta a(\Delta a - 2\delta e) + \frac{2(\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|}}
\end{aligned}
\tag{5.17}
$$

This gives,

$$
\boxed{\delta r_{RN\_min} = a\sqrt{\Delta a(\Delta a - 2\delta e) + \frac{2(\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|}}} \tag{5.18}
$$

δr_{RN_min} is the minimum separation between the two spacecraft in terms of the relative orbital elements describing their relative motion assuming $\delta a \neq 0$.

If $\delta a = 0$, from equation 5.18 we get:

$$
\delta r_{RN\_min,\delta a=0} = \frac{\sqrt{2}a\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}}}{\sqrt{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|}} \tag{5.19}
$$

This is the exact expression derived in [55, 34, 115] which dictates that for δr_{RN_min} between two spacecraft to be maximum for bounded relative motion, a parallel (or anti-parallel) alignment of relative eccentricity and inclination vectors between them is desired. However, in this work, since we assume drifting relative motion between the spacecraft, we seek a good balance between δa, $\vec{\boldsymbol{\delta e}}$ and $\vec{\boldsymbol{\delta i}}$ such that equation 5.18 satisfies $\delta r_{RN\_min} > 0$.

5.3 Computing safe relative orbits for passive collision avoidance

Equation 5.18 can be leveraged to compute a transfer trajectory for a deputy spacecraft that will guarantee collision avoidance with a chief spacecraft. To achieve this, two different approaches

can be taken that are described below.

5.3.1 Approach 1

As described before, $\delta r_{RN\_min} > 0$ is necessary to guarantee safe relative motion between two spacecraft. For the two spacecraft to maintain a minimum separation of a given obstacle tolerance ('obs_tol'), the required condition is:

$$\delta r_{RN\_min} \geq \text{obs\_tol} \tag{5.20}$$

Thus, from equation 5.18 we get

$$a\sqrt{\Delta a(\Delta a - 2\delta e) + \frac{2(\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|}} \geq \text{obs\_tol} \tag{5.21}$$

Let,

$$\Delta a(\Delta a - 2\delta e) = A \tag{5.22}$$

$$\frac{2(\vec{\boldsymbol{\delta e}} \cdot \vec{\boldsymbol{\delta i}})^2}{\delta e^2 + \delta i^2 + |\vec{\boldsymbol{\delta e}} + \vec{\boldsymbol{\delta i}}||\vec{\boldsymbol{\delta e}} - \vec{\boldsymbol{\delta i}}|} = B \tag{5.23}$$

$$\text{obs\_tol} = \mathcal{O} \tag{5.24}$$

Therefore, equation 5.21 can be written as:

$$\begin{aligned} & a\sqrt{(A+B)} \geq \mathcal{O} \\ \implies & a^2(A+B) \geq \mathcal{O}^2 \\ \implies & A + B \geq \frac{\mathcal{O}^2}{a^2} \\ \implies & A - (\frac{\mathcal{O}^2}{a^2} - B) \geq 0 \\ \implies & \boxed{\Delta a^2 - 2\Delta a \delta e - (\frac{\mathcal{O}^2}{a^2} - B) \geq 0} \end{aligned} \tag{5.25}$$

Equation 5.25 is a quadratic equation that can be used to solve for Δa $(= \frac{\delta a}{a})$ and has the following

roots,

$$\begin{aligned} \Delta a &= \frac{2\delta e \pm \sqrt{4\delta e^2 + 4(\frac{\mathcal{O}^2}{a^2} - B)}}{2} \\ \implies \Delta a &= \delta e \pm \sqrt{\delta e^2 + (\frac{\mathcal{O}^2}{a^2} - B)} \\ \implies \Delta a &= \delta e \pm D \end{aligned} \tag{5.26}$$

where, $D = \sqrt{\delta e^2 + (\frac{\mathcal{O}^2}{a^2} - B)}$.

At these two roots, $\delta r_{RN\_min} = \text{obs\_tol}$. These two roots divide the solution space of Δa into three intervals namely $(-\inf,\ \delta e - D], (\delta e - D,\ \delta e + D), [\delta e + D,\ \inf)$. Values of Δa from these intervals that satisfy equation 5.25 are valid solutions to $\delta r_{RN\_min} \geq \text{obs\_tol}$.

It is to be noted that the Δa solution from equation 5.26 depends on $\vec{\boldsymbol{\delta e}}$ and $\vec{\boldsymbol{\delta i}}$ which needs to be picked first depending on user preferences.

This approach is very convenient when a single chief spacecraft is known and the goal is to design a relative orbit that will maintain a certain relative separation from this chief. However, when there are multiple spacecraft to avoid, solving for a single safe relative orbit that will avoid collision with every obstacle, using this approach, can get tricky. This is where this next approach comes in handy.

5.3.2 Approach 2

In this approach, a relative orbit is picked at random, and its minimum separation from each given obstacle is computed using the analytical solution as shown in equation 5.18. For computing this r_{RN_min} of the deputy with respect to each obstacle, each obstacle is individually treated as the chief in their respective turns, and the relative orbital elements of the randomly picked relative orbit with respect to this obstacle/ 'new chief' ($\delta OE_{\text{deputy|obs}}$) are determined as:

$$\delta OE_{\text{deputy|obs}} = \delta OE_{\text{deputy|chief}} - \delta OE_{\text{obs|chief}} \tag{5.27}$$

where $\delta OE_{\text{deputy|chief}}$ refers to the relative orbit elements of the randomly picked relative orbit with respect to the original chief and $\delta OE_{\text{obs|chief}}$ is the relative orbit elements of the obstacle with

respect to the original chief.

For all obstacles, if $\delta r_{RN\_min} >$ obs_tol, then the picked relative orbit is considered a valid collision-free solution. However, for any of the obstacles, if $r_{RN\_min} \leq$ obs_tol, there are two possible options:

(1) The randomly picked relative orbit can be discarded and a new relative orbit may be sampled. OR

(2) Further steps are required to check whether the obs_tol violation occurs in the time interval of interest.

In the presence of multiple moving obstacles in the scenario, the first option may be expensive in terms of the required number of random draws to find a safe relative orbit. As a result, here we explore the second option further.

Additional analysis to evaluate obs_tol violation in the time interval of interest:

If $r_{RN\_min} \leq$ obs_tol, the goal of this routine is to check if the overall relative separation between the two spacecraft (δr_{RIN}) violates the obs_tol in the time interval of interest, where δr_{RIN} is defined by using equations 5.8, 5.9, and 5.10 as:

$$\delta r_{RIN} = \sqrt{\delta r_R^2 + \delta r_I^2 + \delta r_N^2} \tag{5.28}$$

To achieve this, an optimization scheme is set up for the time interval of interest [t_0, t], where the minimum value of δr_{RIN} is computed according to equation 5.28. This δr_{RIN} depends on the optimization variable 'time' because the $\delta r_R, \delta r_I, \delta r_N$ (as shown in equations 5.8, 5.9 and 5.10) depend on the mean argument of latitude of the chief (u). Moreover, the in-track relative separation (δr_I) secularly grows with time due to its direct dependency on u. Thus, once the minimum δr_{RIN} is computed in the time interval of interest, it is compared with the allowable obstacle tolerance value. If $\delta r_{RIN} >$ obs_tol, then the picked relative orbit is considered a valid solution for the given time frame. Otherwise, the relative orbit is declared unsafe and a new relative orbit is picked and the entire analysis is repeated.

5.4 Implementation and simulation results

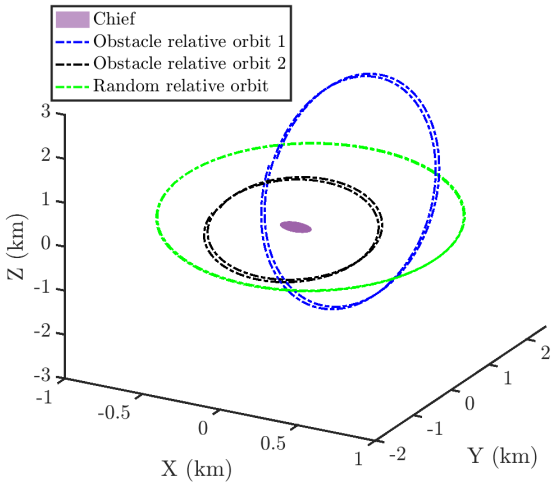
5.4.1 Demonstration of 'Approach 2' for computing safe relative orbits:

Figure 5.1 demonstrates an implementation of approach 2 as described in section 5.3.2. Here two spacecraft in the scenario are in their respective relative orbits with respect to a common chief spacecraft (orbital elements shown in Table 5.1). The goal is to compute a safe relative orbit that

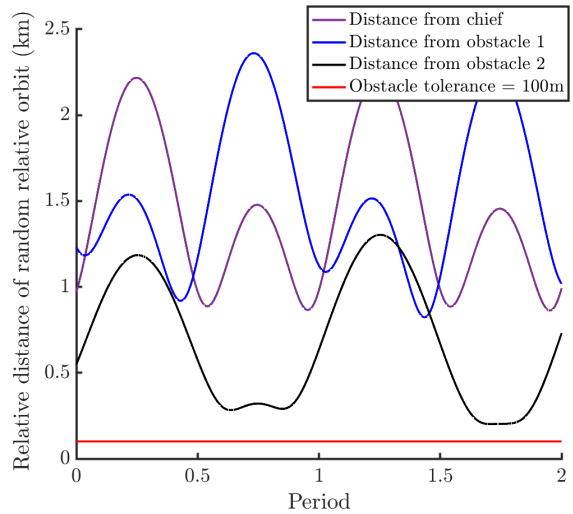
COEs	**Chief**	**δCOEs**	**Obstacle 1**	**Obstacle 2**	**Picked Random Deputy**
a	15000 km	δa	10 m	10 m	-2.6 m
e	0	δe	3.333×10^{-5}	3.333×10^{-5}	5.94×10^{-5}
i	50°	δi	$7.64 \times 10^{-3^o}$	$1.91 \times 10^{-3^o}$	$3.36 \times 10^{-5^o}$
Ω	10°	$\delta\Omega$	$5.73 \times 10^{-5^o}$	0°	$6.01 \times 10^{-8^o}$
ω	10°	$\delta\omega$	0°	0°	$1.08 \times 10^{-5^o}$
M	0°	δM	$5.73 \times 10^{-3^o}$	0°	$1.39 \times 10^{-5^o}$

Table 5.1: Orbital elements of the chief, obstacle spacecraft, and the deputy's safe solution trajectory.

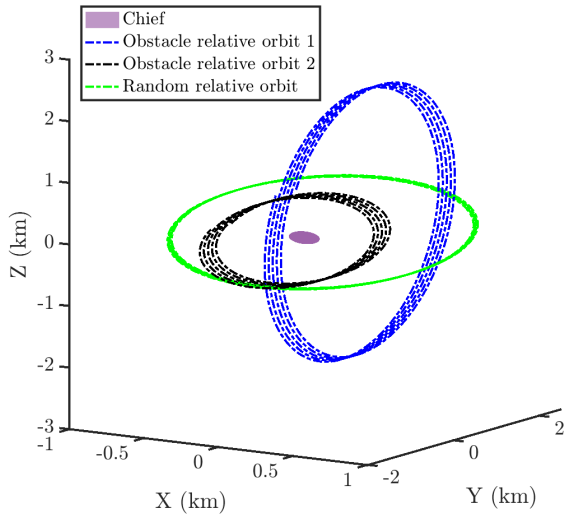
is guaranteed to avoid collision with the existing spacecraft/ moving obstacles in the scenario. As approach 2 dictates, a random relative orbit is picked (orbital elements shown in Table 5.1) and it is evaluated for its r_{RN_min} separation from the obstacles. In this particular implementation, $r_{RN\_min} <$ obs_tol. However, after performing the additional analysis as described in section 5.3.2, the randomly picked relative orbit (in green) was found to be safe from a collision for the initial $\approx$ 2 Period time of flight (as can be seen from Figures 5.1a and 5.1b). In contrast to this, the relative motion of the deputy was evaluated for a longer period of time (5 Period), and in this case, the additional analysis yielded $\delta r_{RIN} <$ obs_tol meaning that the relative trajectory is not safe anymore. This result can be visualized in Figures 5.1c and 5.1d. Thus, in this case, the picked relative orbit is a valid solution if collision avoidance is desired for the initial time of flight $\leq$ 2 Periods. Otherwise, a different safe relative orbit needs to be computed.



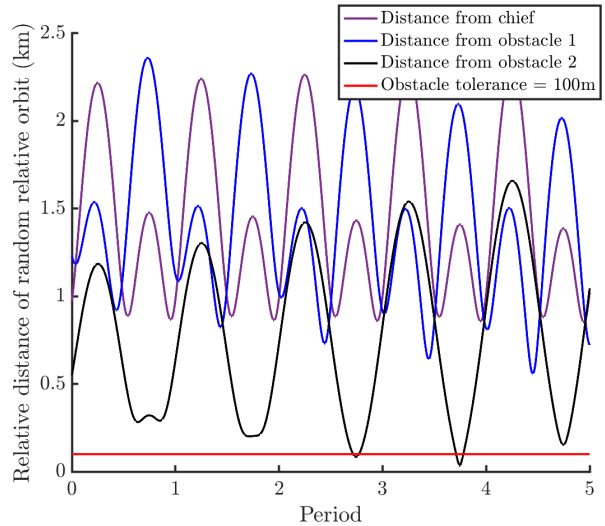
(a) Collision-free relative motion in picked random relative orbit for 2 orbit periods.



(b) Relative separation between randomly picked relative orbit and moving obstacles for 2 orbit periods.



(c) Picked random relative orbit in collision with an obstacle when propagated for 5 orbit periods.



(d) Relative separation between randomly picked relative orbit and moving obstacles for 5 orbit periods.

Figure 5.1: Demonstrating working of approach 2 in picking random relative orbit for collision-free spacecraft motion for given time of flight in the presence of multiple spacecraft/moving obstacles in the scenario.

5.4.2 Demonstration of AIKMP with passive collision avoidance:

In this implementation, the tree extension step of the AIKMP algorithm is modified such that the tree accepts a randomly picked node for tree extension only if the node is found to be safe (collision-free from all existing obstacles) using the analysis as described in the approach 2 (section 5.3.2). To demonstrate the working of this modified AIKMP algorithm, a transfer scenario similar to the scenario described in section 2.3 is defined but now with both static obstacles (static in the chief-centered relative frame) and moving obstacles.

In this scenario, the deputy spacecraft attempts to safely reconfigure itself from a state in an initial closed relative orbit to a state in a final closed relative orbit around the chief as described in Table 5.2. The chief is in a circular orbit and it is assumed to have two cameras attached to it with an angle of view of 20° making observations in the negative along-track direction (-Y) and the cross-track direction (Z) (both acting as static obstacles in the relative frame). The deputy spacecraft should not obstruct the field of view of these cameras during the reconfiguration. There are three other spacecraft in the scenario (moving obstacles) assumed to be in three different relative orbits around the chief (orbital elements shown in Table 5.3) whose relative motions with respect to the chief are drifting over time. The overall scenario is illustrated in Figure 5.2.

Relative states	**Deputy (Initial)**	**Deputy (Final)**
x (km)	9.999×10^{-1}	-1.249×10^{-1}
y (km)	4.113×10^{-1}	-2.664×10^{-1}
z (km)	1.308×10^{-1}	-1.915×10^{-1}
$\dot{x}$ (m/s)	1.278×10^{-7}	-7.441×10^{-5}
$\dot{y}$ (m/s)	-6.873×10^{-4}	8.586×10^{-5}
$\dot{z}$ (m/s)	-1.242×10^{-4}	5.521×10^{-5}

Table 5.2: Initial and Final desired relative states of deputy spacecraft.

The goal of the transfer is that the deputy's transfer must be collision-free with the chief and the obstacles present in the scenario and the deputy should also use less fuel for the transfer.

The transfer solution computed by the AIKMP algorithm (using LLS steering law and pruning) is shown in Figures 5.3 and 5.4.

COEs	Chief	δCOEs	Obstacle 1	Obstacle 2	Obstacle 3
a	15000 km	δa	10 m	10 m	10 m
e	0	δe	3.333×10^{-5}	3.333×10^{-5}	3.333×10^{-5}
i	50°	δi	$3.82 \times 10^{-3^o}$	$9.55 \times 10^{-3^o}$	$5.73 \times 10^{-4^o}$
Ω	10°	$\delta\Omega$	0°	$5.73 \times 10^{-5^o}$	0°
ω	10°	$\delta\omega$	0°	0°	0°
M	0°	δM	0°	$5.73 \times 10^{-3^o}$	$5.73 \times 10^{-4^o}$

Table 5.3: Orbital elements of chief and other spacecraft/moving obstacles in the scenario.

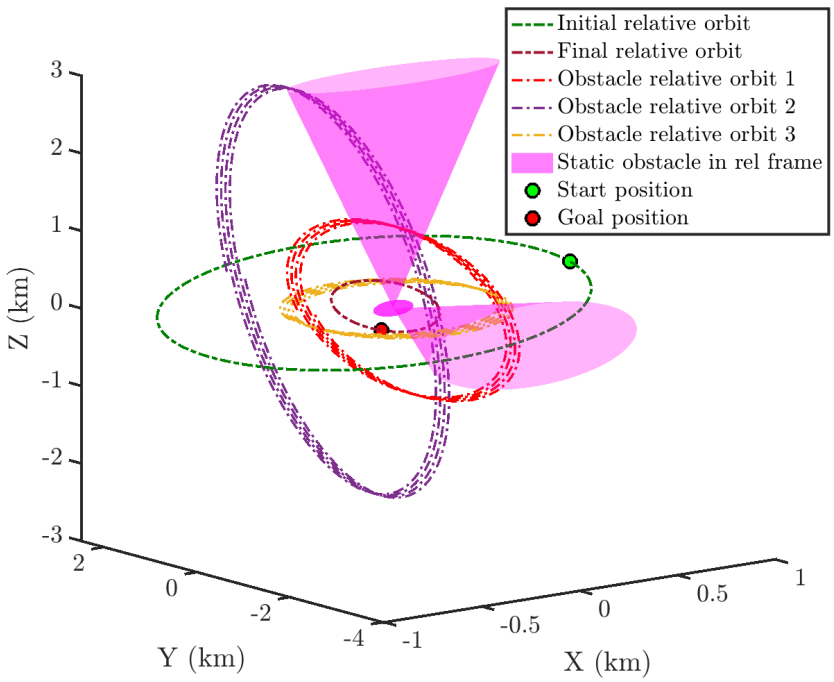
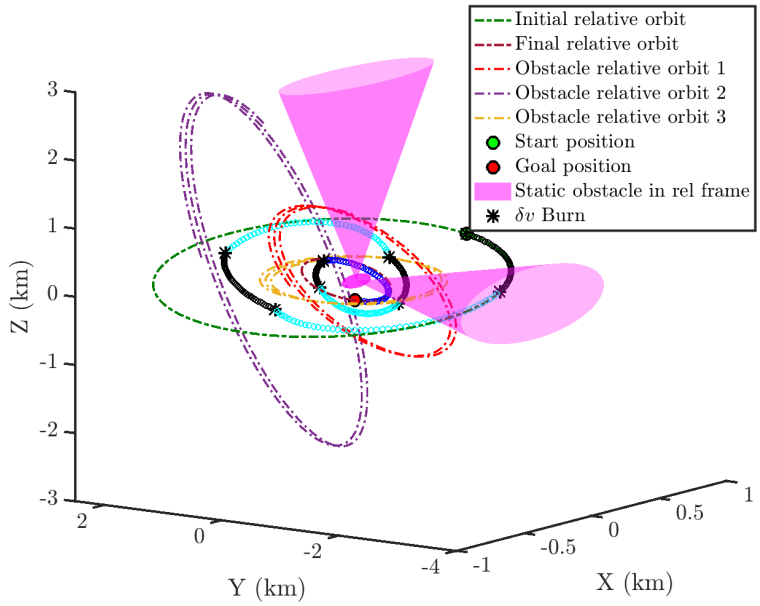
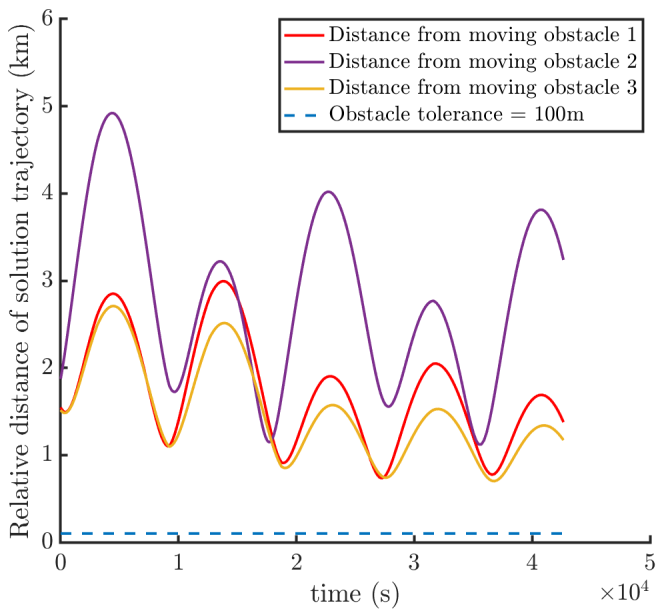


Figure 5.2: Relative orbit transfer scenario with both static (static in the chief-centered relative frame) and moving obstacles to demonstrate the working of AIKMP algorithm with infused passive collision avoidance capabilities.

As can be seen from Figures 5.3a and 5.3b, the AIKMP algorithm was able to compute a final transfer trajectory that maintains the desired separation (obs_tol = 100m) from the moving obstacles present in the scenario and is collision-free from all obstacles for the entire duration of the transfer. In this case, the motion planner was allowed to run for a total of 50000 iterations and the algorithm took $\approx$ 20 minutes to complete the simulation. A total of $\approx$ 2700 transfer solutions were



(a) Transfer solution in 3D relative position space.



(b) Relative separation between the transfer solution and moving obstacles over the duration of the transfer

Figure 5.3: Transfer solution as computed by the efficient AIKMP algorithm infused with passive collision avoidance for tree extension. The total Δv required for the transfer is 43.76 cm/s.

computed during this time and the best cost transfer computed (Δv = 43.76 cm/sec) is shown in

Figures 5.3 and 5.4.

To compare the performance of this efficient version of the AIKMP algorithm (uses LLS steering law + Pruning) that uses a tree extension strategy slightly different than the basic version of the AIKMP algorithm (uses Lambert steering law without pruning) as explained in Chapter 3, the later version was also allowed to compute a solution trajectory in this transfer scenario. The best cost transfer solution ($\Delta v = 29.26$ cm/sec) computed by this version is shown in Figures 5.5 and 5.6. In this case, the motion planner took ≈ 4 hours to run 1500 iterations of the algorithm and computed ≈ 40 transfer solutions.

Figures 5.5 and 5.6 in contrast to Figures 5.3 and 5.4 highlight an important point that – in complex transfer scenarios like the one demonstrated here (Figure 5.2), using all the prescribed ways to improve the computation and storage efficiency of the AIKMP algorithm may not always guarantee a fuel-efficient transfer solution in a given number of algorithm iterations. Particularly since cluttered environments like this offer very restricted state space for random exploration, the outcomes of using the different tree extension strategies for improved computation efficiency are evident in these results. The basic version of the AIKMP algorithm (that uses Lambert steering law), although took a longer simulation time, yielded a better cost solution in a lesser number of algorithm iterations in this scenario because this version uses the PSO-based tree extension strategy that optimizes the cost of each intermediate transfer arc resulting in an overall smoother transfer solution.

Given the complexity of the defined transfer scenario, the efficient version of the AIKMP algorithm was allowed to run for a larger number of algorithm iterations (150000 iterations) with slightly modified values for the AIKMP radius parameters ($\delta_{near} = 3 \times 10^{-5}$ and $\delta_{best} = 2 \times 10^{-5}$). The results can be seen in Figures 5.7 and 5.8. The algorithm took ≈ 3 hours (still less than the time taken by the Lambert steering law-based AIKMP) to complete the 150000 algorithm iterations. A total of ≈ 3200 transfer solutions were computed during this time and the best cost transfer computed ($\Delta v = 28.65$ cm/sec) is shown in Figures 5.7 and 5.8 which is, in fact, a better cost solution as compared to the one computed using Lambert steering law-based AIKMP algorithm

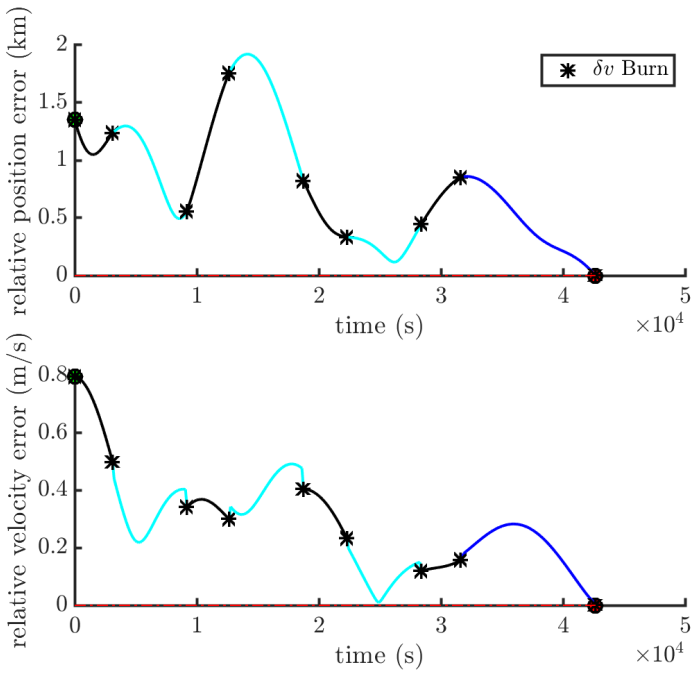
(Δv = 29.26 cm/sec).

Thus, given the complexity of the transfer scenario, the efficient version of the AIKMP algorithm may require a larger number of iterations to compute a better cost solution. However, if the goal is simply to generate a valid transfer solution for a given relative transfer scenario, the overall AIKMP algorithm along with the pruning module can be used to quickly compute feasible collision-free transfer solutions. The best part of the presented AIKMP framework along with the pruning module and the passive collision-avoidance strategy is that they can be adapted with any steering law and any modified tree extension strategy depending on the requirement of the mission.

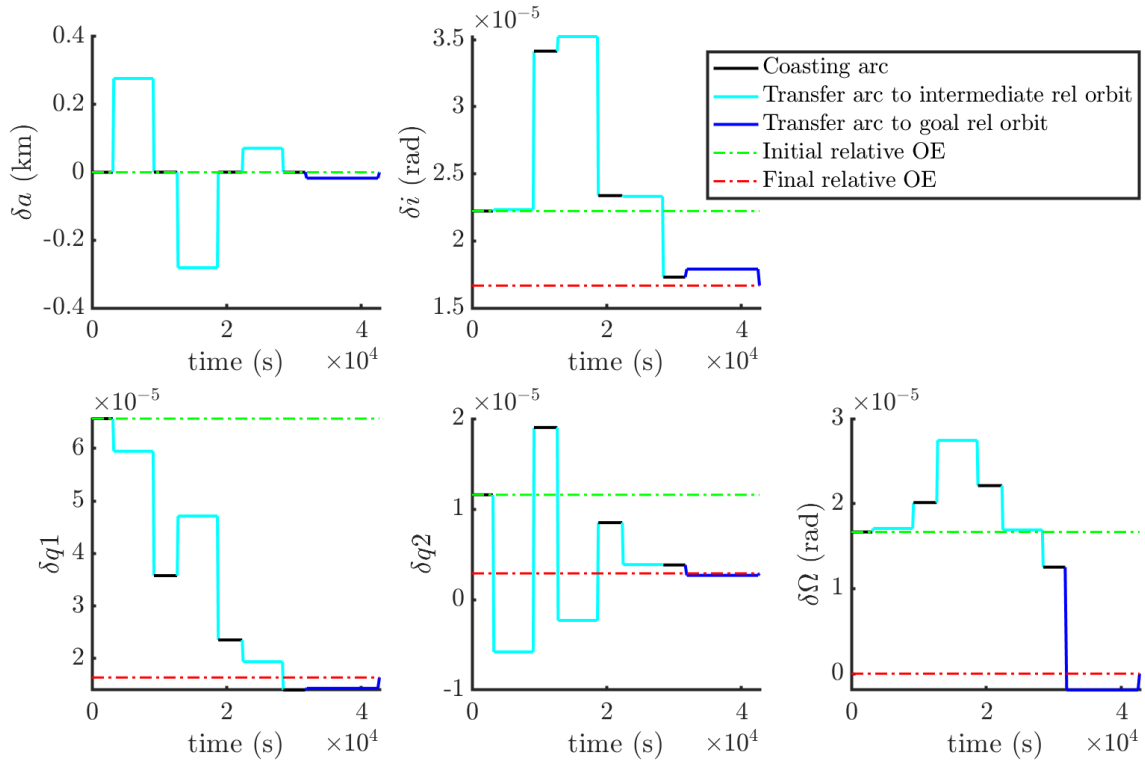
If more moving obstacles are present, the probability of having to draw more random relative orbits for the tree extension step to guarantee collision avoidance, increases. To demonstrate this, different numbers of obstacles were considered in the scenario, and for each case, a relative orbit was drawn at random and checked for collision safety (using approach 2) for a total time of flight of 5 Periods (chosen for demonstration purpose). For different numbers of obstacles, the number of random "draws" to find the first safe relative orbit was recorded and this process was repeated for a total of 10 "attempts". Finally, the average of the number of draws for the different cases (defined by the total number of obstacles) was computed and the results are as shown in Figure 5.9. Figures 5.9a and 5.9b show the average number of draws and the average time (out of the 10 attempts) required in this example to find the first safe relative orbit for the different cases. Figure 5.9c highlights the minimum and the maximum number of draws (out of the 10 attempts) required by each case. When contrasted with the average number of draws, Figure 5.9c clearly indicates that even though this approach of picking a safe relative orbit may take as low as a single draw (minimum number of draws for each case was 1 in this example), with a higher number of obstacles to avoid, the probability of requiring a higher number of draws increases. The required number of draws may increase or decrease depending on how complex the search space for the random sampling becomes based on the relative motion of the moving obstacles.

5.5 Summary

In the work presented in this chapter, passive collision avoidance capability was infused in the tree extension step of the AIKMP algorithm, expanding its application to relative motion planning problems with moving obstacles. Leveraging the theory behind E-I vector separation, an analytical solution was derived to describe the relative separation between two drifting ($\delta a \neq 0$) spacecraft in the radial-normal plane in terms of relative orbital elements. Based on the derived solution, two approaches were detailed that can be used to pick a safe relative orbit that is guaranteed to avoid collision with known moving obstacles for a desired amount of time. With the help of simulation results, it was shown that the resulting AIKMP algorithm picks the intermediate random nodes for tree extension in a more informed way (as compared to the previous versions described in Chapter 2 and Chapter 3), such that obstacle tolerance constraints are satisfied for the entire duration of the transfer in the presence of multiple static and moving obstacles – a capability that is crucial for spacecraft formation flying applications. With theoretical reasoning and simulation results, it was also pointed out that, given the complexity of the transfer scenario, the efficient version of the AIKMP algorithm may require a larger number of iterations to compute a better cost solution. However, if the goal is simply to generate a valid transfer solution for a given relative transfer scenario, the overall AIKMP algorithm along with the pruning module can be used to quickly compute feasible collision-free transfer solutions. The best part of the presented AIKMP framework along with the pruning module and the passive collision-avoidance strategy is that they can be adapted with any steering law and any modified tree extension strategy depending on the requirement of the mission. It is to be noted here that, since this formulation of δr_{RN_min} is derived from equations 5.8, 5.9 and 5.10 which in turn are derived from the HCW equations ([55, 34]), this passive collision avoidance strategy derived in this chapter are valid for circular and near-circular chiefs. To extend their application to elliptical chief orbits, a similar approach may be attempted with Tschauner Hempel's description of relative motionB.

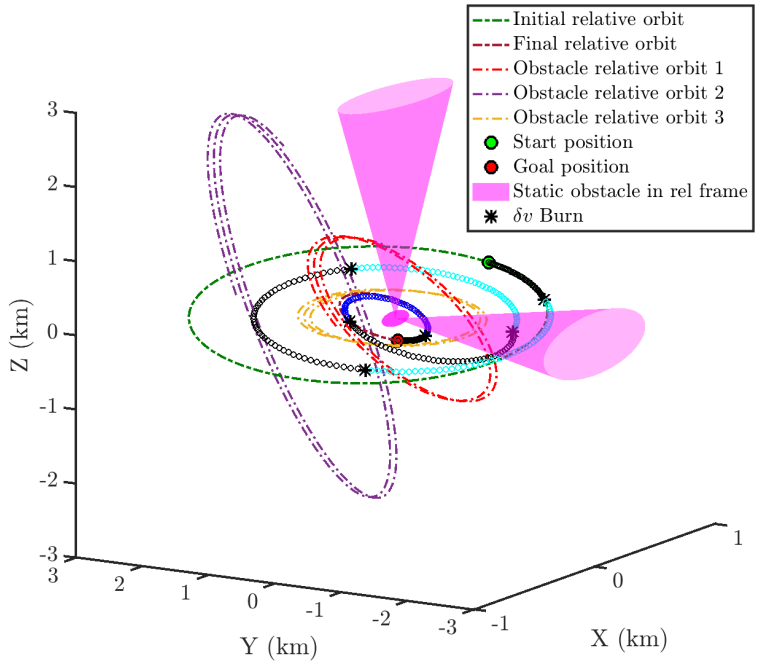


(a) Relative state error of solution trajectory with respect to the goal state.

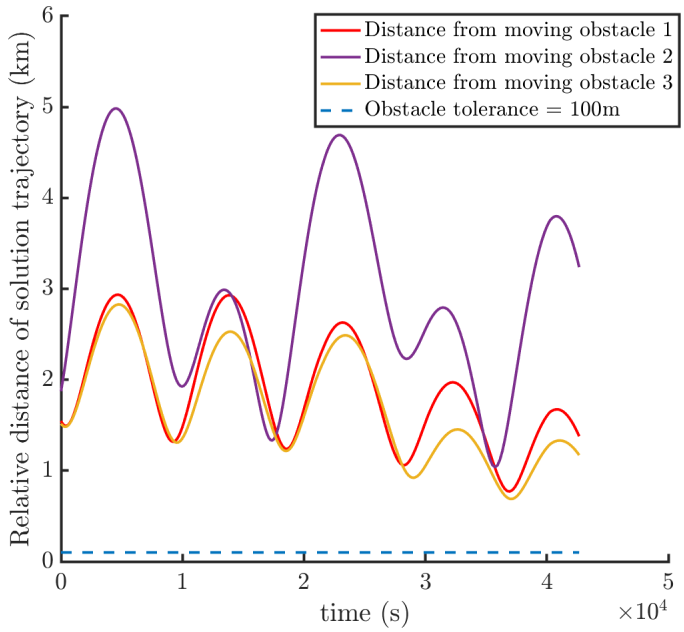


(b) Relative orbital elements vs time of the final AIKMP solution

Figure 5.4: Relative states and relative orbital elements of the deputy's transfer solution as computed by the passive collision-avoidance infused efficient AIKMP algorithm.

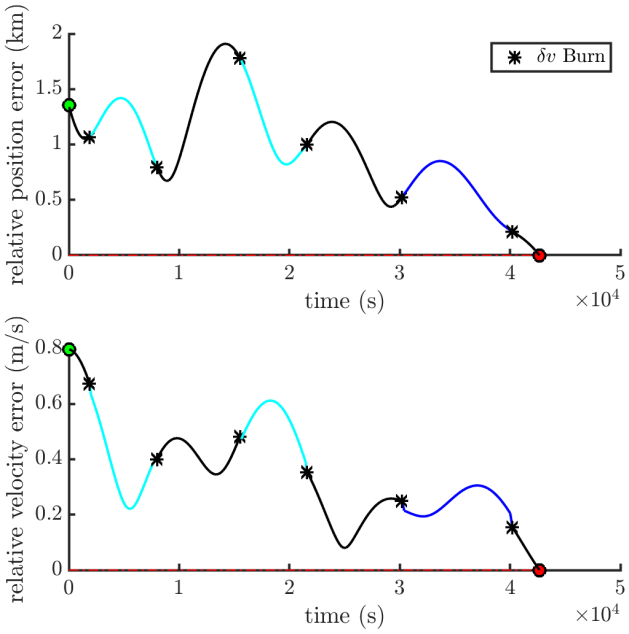


(a) Transfer solution in 3D relative position space.

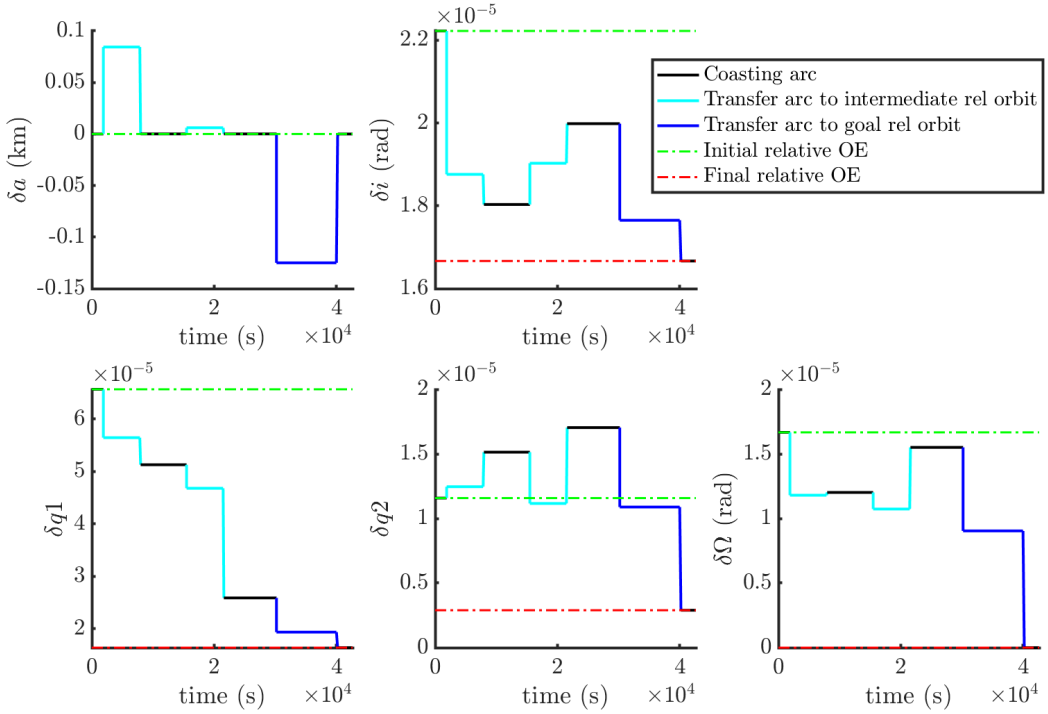


(b) Relative separation between the transfer solution and moving obstacles in the scenario over the duration of the transfer

Figure 5.5: Transfer solution as computed by the AIKMP algorithm (using Lambert steering without pruning) infused with passive collision avoidance for tree extension. The total Δv required for the transfer is 29.26 cm/s.

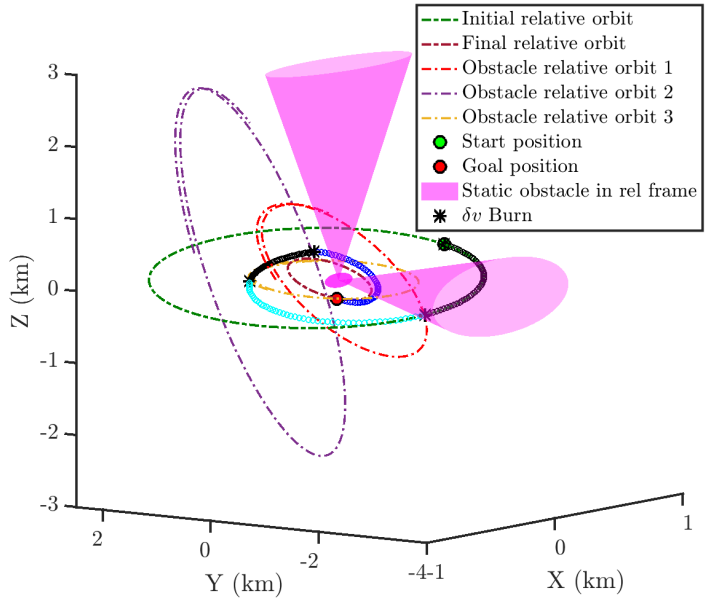


(a) Relative state error of solution trajectory with respect to the goal state.

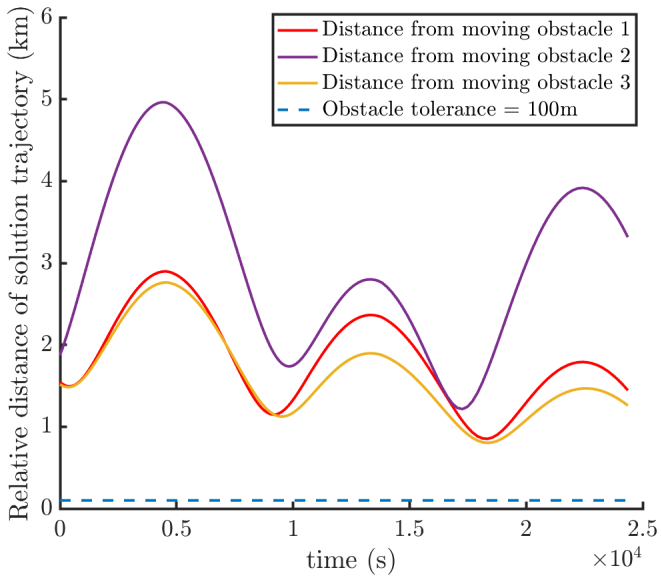


(b) Relative orbital elements vs time of the final AIKMP solution

Figure 5.6: Relative states and relative orbital elements of the deputy's transfer solution converging to the desired goal states as computed by the passive collision-avoidance infused AIKMP algorithm (using Lambert steering without pruning).

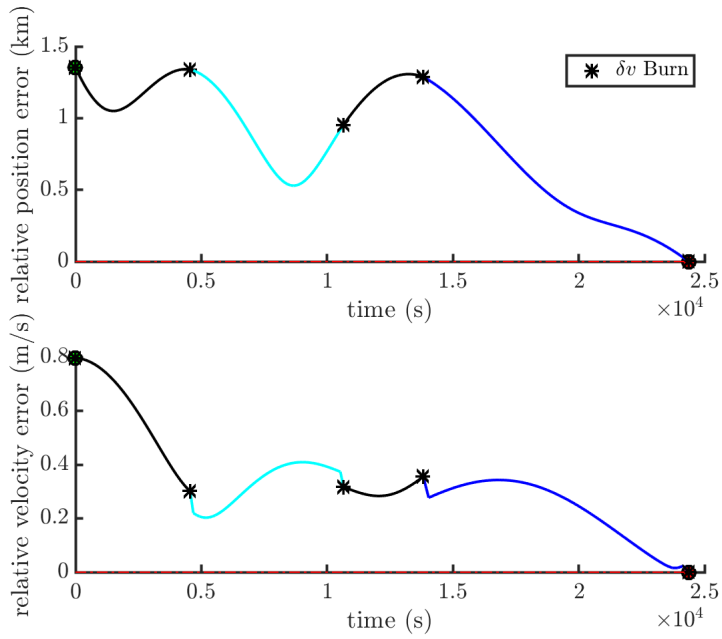


(a) Transfer solution in 3D relative position space.

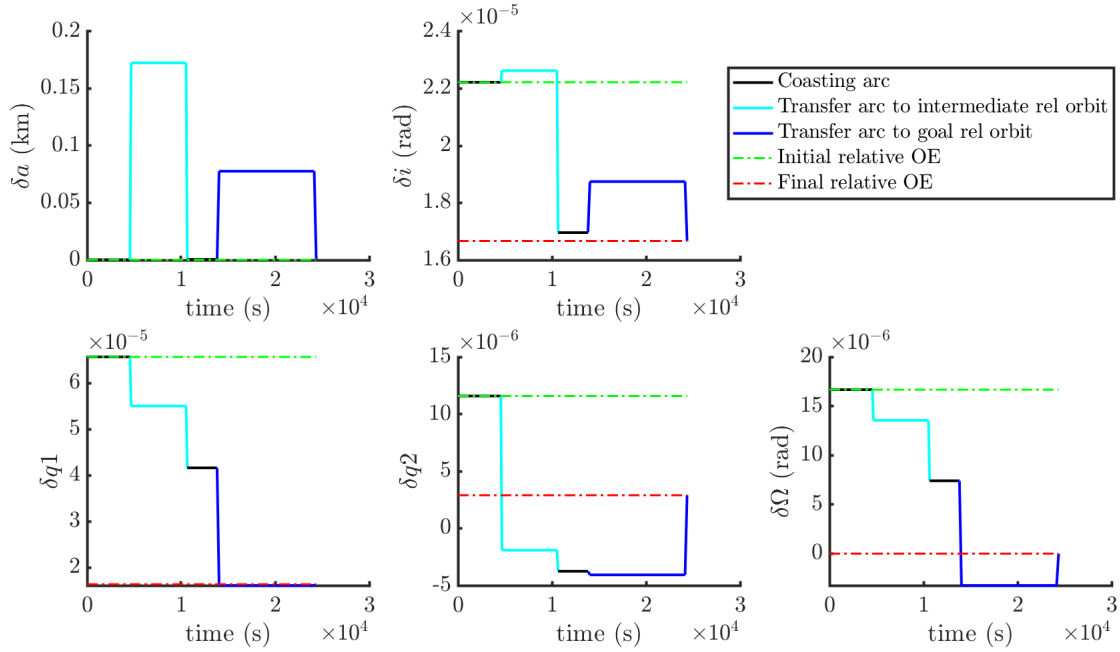


(b) Relative separation between the transfer solution and moving obstacles over the duration of the transfer

Figure 5.7: Transfer solution as computed by the efficient AIKMP algorithm infused with passive collision avoidance for tree extension. The total Δv required for the transfer is 28.65 cm/s.

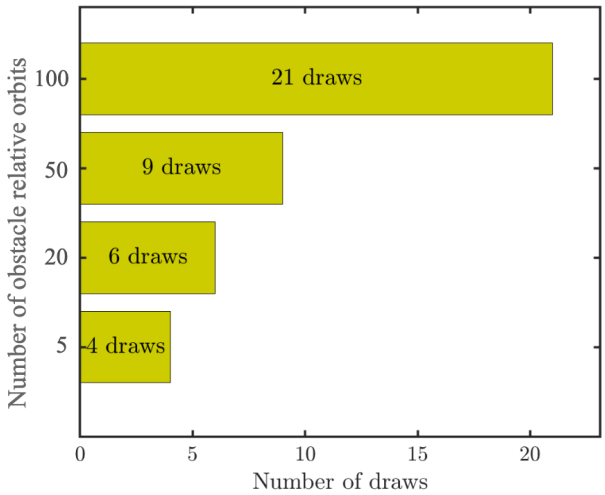


(a) Relative state error of solution trajectory with respect to the goal state.

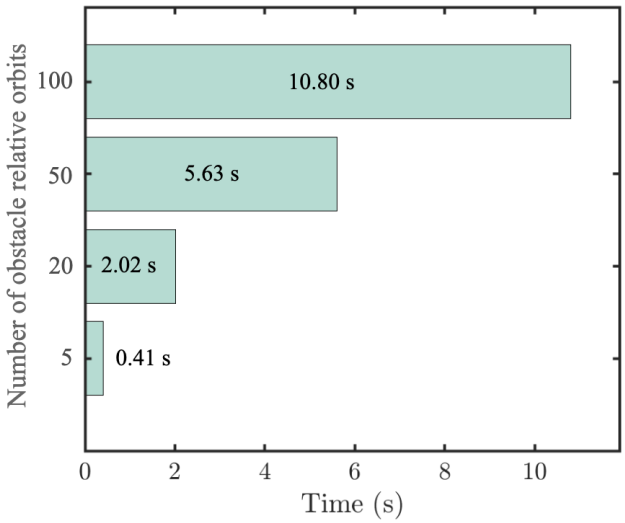


(b) Relative orbital elements vs time of the final AIKMP solution

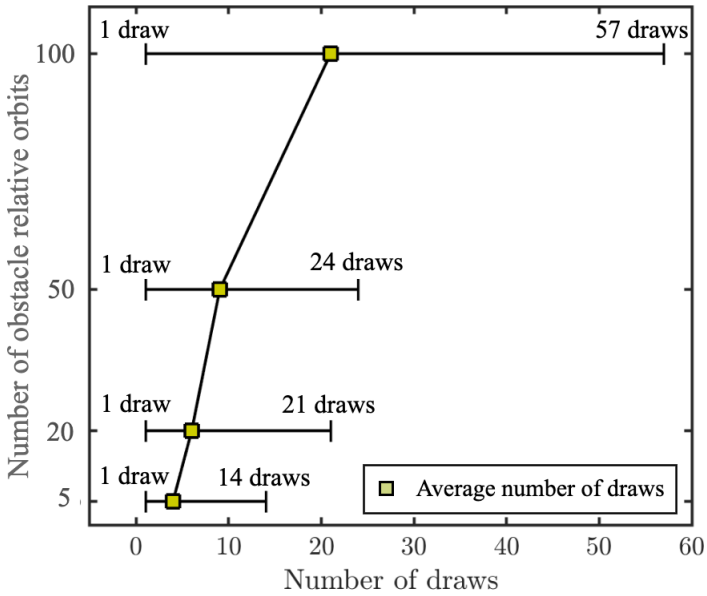
Figure 5.8: Relative states and relative orbital elements of the deputy's transfer solution as computed by the passive collision-avoidance infused efficient AIKMP algorithm.



(a) Average number of random draws required to find the first safe relative orbit for different numbers of obstacles



(b) Average time required to find the first safe relative orbit for different numbers of obstacles



(c) Minimum and the maximum number of draws (out of 10 different "attempts") as contrasted with the average number of draws required by each case.

Figure 5.9: Number of random draws (out of total 10 total "attempts") required to find the first safe relative orbit for different numbers of moving obstacles.

Chapter 6

Conclusion and Future Work

6.1 Conclusion on completed work

In this work, it was shown that by augmenting sampling-based motion planning ideas from the field of robotics with knowledge of astrodynamics, fuel-efficient and collision-free motion planning can be achieved in astrodynamics applications. The work presented an astrodynamics-informed version of the popular Rapidly Exploring Random Trees (RRTs) to make the algorithm more suitable for fuel-efficient orbital motion planning (Chapter 2). An astrodynamics-informed pruning module was developed to allow the motion planner to maintain and store a sparse set of nodes improving its overall computation efficiency by $\approx 98\%$ and storage efficiency by $\approx 80\%$ in the considered example scenario (Chapter 3). The overall motion planner with this pruning module comprises the proposed algorithm in this work, which is called the Astrodynamics-informed kinodynamic motion planning (AIKMP) algorithm. This AIKMP algorithm is probabilistically complete meaning that as the number of iterations of the algorithm tends to infinity, the motion planner will be able to find a feasible transfer solution with probability $= 1$, if a solution exists. This algorithm is a single-step sampling-based kinodynamic approach to orbital motion planning problems (as opposed to existing two-step sampling-based motion planning approaches in literature) and can quickly find solutions to spacecraft relative transfer problems in a cluttered environment – without requiring an initial guess of the solution.

This work also presented a novel extension of the linearized Lambert solution (LLS) called the closed-loop linearized Lambert guidance solution that allows a spacecraft to apply correction burns

during the transfer to improve the targeting accuracy in relative guidance problems in the presence of perturbations like Drag, J2, and Solar Radiation Pressure (SRP) (Chapter 4). Simulation results were used to demonstrate that closed-loop LLS can help save $\approx 26\%$ fuel over a duration of 50 days for a spacecraft formation flying scenario with maximum relative separation constraints. It was also shown that closed-loop LLS can be used in conjunction with orbital motion planners like the AIKMP - providing improved targeting accuracy of $\approx 93\%$ at a cost of $\approx 16\%$ fuel increase in the presence of external perturbations as compared to open-loop guidance.

Finally, an extension of the popular E/I vector separation method was derived for the case of drifting relative motion ($\delta a \neq 0$) to infuse passive collision avoidance capabilities in the AIKMP algorithm (Chapter 5). The resulting motion planning algorithm performs tree extension in a more informed way such that obstacle tolerance constraints are satisfied for the entire duration of the transfer in the presence of multiple static (static in moving reference frame) and moving obstacles – a capability that is crucial for spacecraft formation flying applications.

Overall, the proposed AIKMP algorithm describes the sufficient number of steps that can be adapted from a sampling-based robotic motion planner with required modifications to achieve fuel-efficient and collision-free sampling-based motion planning in astrodynamics applications. A notable aspect of this proposed motion planning framework lies in its modularity, as individual modules of the algorithm can be adapted to any existing sampling-based motion planner (tree-based planners or graph-based planners) to improve their applicability to orbital motion planning.

6.2 Possible extensions and future work

The work presented in this dissertation offers many opportunities for further extensions:

(1) **Investigate mathematical definitions and properties for the motion planner**: Chapters 2 and 3 described some basic rules of thumb that were used to pick the different parameters associated with a sampling-based orbital motion planner, such as state space for the random exploration as well as defining the radius parameters (δ_{near} and δ_{best}). The

definition of these parameters is critical for convergence of the motion planner and they mostly depend on the mission requirements. However, investigating the use of informed mathematical analysis (as done in [89]) to find a generic definition of these parameters will be very helpful for an orbital motion planning framework. Implementing self-adjustable δ_{near} and δ_{best} parameters depending on the dynamic environment can also be another viable option for these parameters [16].

Moreover, the tree pruning applied to the developed AIKMP algorithm is inspired by the asymptotically near-optimal stable sparse RRT (SST) robotic motion planning algorithm. One assumption contributing to this asymptotic near-optimal nature is the Lipchitz continuity of the cost function used [89]. Similar numerical analysis for evaluation of the asymptotic optimality of the AIKMP algorithm will be a useful next step especially when considering applications that do not have algorithm run-time constraints.

(2) **Automate selection of parameters in closed-loop LLS guidance**: In Chapter 4, for implementing closed-loop LLS for collision avoidance, picking a suitable refresh rate (frequency of constraint violation checking) is very important. If the refresh rate is too spread out/large, this may lead to constraint violation for a longer time. If the refresh rate is too small, this will lead to using too many correction burns because it takes some time for an implemented burn to move the trajectory such that the violation no longer occurs.

Similarly, the number of correction burns allowed for the closed-loop LLS implementation is another important parameter that will define the improvement in targeting accuracy that can be achieved by utilizing such a guidance algorithm. During this work, it was observed that although targeting accuracy can be improved by introducing correction burns, more correction burns do not always mean more improvement in targeting accuracy. In fact, if a correction burn is applied "too close" to the target state, this can in turn introduce a significant targeting error.

For the studies conducted in this dissertation, a considerable amount of time was spent

carefully selecting the refresh rate for collision checks and the placement of the correction burns in the defined scenarios for improved targeting accuracy. It will be very useful if the choice of these parameters can be automated depending on the fuel availability, propulsion system capability, and/or mission duration and requirements.

(3) **Extend passive collision avoidance capabilities to elliptical chief descriptions**: In chapter 5, an extension of the E/I vector separation method was derived for the case of drifting relative motion to infuse passive collision avoidance capabilities in the AIKMP algorithm. This formulation is based on the Hill-Clohessy-Wiltshire (HCW) equations [32, 131], meaning that the derived passive collision avoidance strategy is valid for only circular and near-circular chiefs [34, 55, 115, 37]. To extend their application to elliptical chief orbits, a similar approach may be attempted with Tschauner Hempel's description of relative motionB.

Bibliography

[1] In-space servicing, assembly, and manufacturing national strategy. 2022.

[2] Shakeeb Ahmad and J Sean Humbert. Efficient sampling-based planning for subterranean exploration. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7114–7121. IEEE, 2022.

[3] Keenan Albee and Brian Coltin. Kinodynamic-rrt for robotic free-flyers: Generating feasible trajectories for on-orbit mobile manipulation.

[4] Kyle Alfriend, Srinivas Rao Vadali, Pini Gurfil, Jonathan How, and Louis Breger. Spacecraft formation flying: Dynamics, control and navigation, volume 2. Elsevier, 2009.

[5] Kyle T Alfriend and et. al. Spacecraft formation flying: Dynamics, control and navigation, volume 2. Elsevier, 2009.

[6] Kyle T Alfriend, Dong-Woo Gim, and Hanspeter Schaub. Gravitational perturbations, nonlinearity and circular orbit assumption effects on formation flying control strategies. Guidance and control 2000, pages 139–158, 2000.

[7] Kyle T Alfriend and Hanspeter Schaub. Dynamic and control of spacecraft formations: challenges and some solutions. The Journal of the Astronautical Sciences, 48:249–267, 2000.

[8] Robert O Ambrose, Hal Aldridge, R Scott Askew, Robert R Burridge, William Bluethmann, Myron Diftler, Chris Lovchik, Darby Magruder, and Fredrik Rehnmark. Robonaut: Nasa's space humanoid. IEEE Intelligent Systems and Their Applications, 15(4):57–63, 2000.

[9] Nitin Arora, Ryan P. Russell, Nathan Strange, and David Ottesen. Partial derivatives of the solution to the lambert boundary value problem. Journal of Guidance, Control, and Dynamics, 38(9):1563–1572, 2015.

[10] Oktay Arslan and Panagiotis Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In 2013 IEEE International Conference on Robotics and Automation, pages 2421–2428. IEEE, 2013.

[11] M. Aung et al. An overview of formation flying technology development for the terrestrial planet finder mission. In 2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720), volume 4, pages 2667–2679 Vol.4, 2004.

[12] A. Badawy and C. McInnes. On-orbit assembly using superquadric potential fields. Journal of Guidance Control and Dynamics, 31:30–43, 2008.

[13] William Baker, Zachary Kingston, Mark Moll, Julia Badger, and Lydia E Kavraki. Robonaut 2 and you: Specifying and executing complex operations. In 2017 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), pages 1–8. IEEE, 2017.

[14] Mai Bando and Akira Ichikawa. Graphical generation of periodic orbits of tschauner-hempel equations. Journal of Guidance, Control, and Dynamics, 35(3):1002–1007, 2012.

[15] Saptarshi Bandyopadhyay, Francesca Baldini, Rebecca Foust, Soon-Jo Chung, Amir Rahmani, Jean-Pierre de la Croix, and Fred Y Hadaegh. Distributed fast motion planning for spacecraft swarms in cluttered environments using spherical expansions and sequence of convex optimization problems. 17(42), 2017.

[16] Saptarshi Bandyopadhyay, Francesca Baldini, Rebecca Foust, Amir Rahmani, Jean-Pierre de la Croix, Soon-Jo Chung, and Fred Hadaegh. Distributed spatiotemporal motion planning for spacecraft swarms in cluttered environments. In AIAA SPACE and Astronautics Forum and Exposition, page 5323, 2017.

[17] R. H. Battin and R. M. Vaughan. An elegant lambert algorithm. Journal of Guidance, Control, and Dynamics, 7(6):662–670, 1984.

[18] CA Beichman. The terrestrial planet finder (tpf): a nasa origins program to search for habitable planets/the tpf science working group; edited by ca beichman, nj woolf, and ca lindensmith.[washington, dc]: National aeronautics and space administration; pasadena, calif. Jet Propulsion Laboratory, California Institute of Technology,[1999](JPL publication, pages 99–3, 1999.

[19] Arnaud Boutonnet, Bénédicte Escudier, and Vincent Martinot. Direct approach for spacecraft formation-flying positioning in quasicircular low earth orbit. In AIAA/AAS Astrodynamics Specialist Conference and Exhibit, page 4638, 2002.

[20] Owen Brown, Paul Eremenko, and Booz Allen Hamilton. Fractionated space architectures: a vision for responsive space. In 4th Responsive Space Conference, volume 2006. AIAA Los Angeles, 2006.

[21] Martin Buehler et al. The DARPA urban challenge: autonomous vehicles in city traffic, volume 56. springer, 2009.

[22] R. Burns et al. Techsat 21: formation design, control, and simulation. In 2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484), volume 7, pages 19–25 vol.7, 2000.

[23] Mark E Campbell. Planning algorithm for multiple satellite clusters. Journal of Guidance, Control, and Dynamics, 26(5):770–780, 2003.

[24] F Kenneth Chan. Spacecraft collision probability. American Institute of Aeronautics and Astronautics, Inc., 2008.

[25] F Kenneth Chan et al. Spacecraft collision probability. Aerospace Press El Segundo, CA, 2008.

[26] Aijun Chen, Jiadong Ren, Zhenhua Wang, and Yi Shen. Optimal satellite formation reconfiguration based on the uncertainty and disturbance estimator. Advances in Space Research, 70(7):2013–2020, 2022.

[27] David F Chichka. Dynamics of clustered satellites via orbital elements. Astrodynamics 1999, pages 147–161, 2000.

[28] Howie Choset et al. Principles of Robot Motion: Theory, Algorithms and Implementation, pages 77–106. The MIT Press, 2005.

[29] Howie Choset et al. Principles of Robot Motion: Theory, Algorithms and Implementation. The MIT Press, 2005.

[30] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE transactions on Evolutionary Computation, 6(1):58–73, 2002.

[31] WH Clohessy and RS Wiltshire. Terminal guidance system for satellite rendezvous. Journal of the aerospace sciences, 27(9):653–658, 1960.

[32] B. A. Conway and J. E. Prussing. Orbital Mechanics. Oxford University Press, 1993.

[33] Bruce A. Conway. Spacecraft Trajectory Optimization, pages 263–291. Cambridge University Press, 2010.

[34] Simone D'Amico. Autonomous formation flying in low earth orbit. PhD thesis, TU Delft, 2010.

[35] Simone D'Amico and J Russell Carpenter. Satellite formation flying and rendezvous. Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications, 2:1921–1946, 2020.

[36] Simone D'Amico and J Russell Carpenter. Satellite formation flying and rendezvous. Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications, 2:1921–1946, 2020.

[37] Simone D'Amico and Oliver Montenbruck. Proximity operations of formation-flying spacecraft using an eccentricity/inclination vector separation. Journal of Guidance, Control, and Dynamics, 29(3):554–563, 2006.

[38] Karsten Danzmann. Lisa mission overview. Advances in Space Research, 25(6):1129–1136, 2000.

[39] Luigi d'Arcio et al. Search for extraterrestrial planets: the darwin mission. In International Conference on Space Optics—ICSO 2006, volume 10567, page 105670H. International Society for Optics and Photonics, 2017.

[40] Alok Das et al. Techsat 21-a revolutionary concept in distributed space based sensing. In AIAA defense and civil space programs conference and exhibit, page 5255, 1998.

[41] T. Deka et al. Closed-loop linearized lambert solution (lls) for on-board formation control and targeting. In AAS/AIAA Astrodynamics Specialist Conference, 2020.

[42] T. Deka et al. Closed-loop linearized lambert solution for onboard formation control and targeting. Journal of Guidance, Control, and Dynamics (under review), 2023.

[43] T. Deka and J. McMahon. Efficient astrodynamics-informed kinodynamic motion planning for relative spacecraft motion. Advances in Space Research (under review), 2022.

[44] T. Deka and J. W. McMahon. Astrodynamics-Informed Kinodynamic Sampling-Based Motion Planning for Relative Spacecraft Motion. AAS/AIAA Astrodynamics Specialist Conference, 2021.

[45] T. Deka and J. W. McMahon. Astrodynamics-informed sparse kinodynamic motion planning for relative spacecraft motion. 11th International Workshop on Satellite Constellations Formation Flying, 2022.

[46] T. Deka and J. W. McMahon. Efficient astrodynamics-informed kinodynamic motion planning for safe relative spacecraft motion. AAS/AIAA Astrodynamics Specialist Conference, 2022.

[47] T. Deka and J. W. McMahon. Astrodynamics-informed kinodynamic sampling-based motion planning for relative spacecraft motion. Journal of Guidance, Control, and Dynamics (under review), 2023.

[48] Elad Denenberg. Satellite closest approach calculation through chebyshev proxy polynomials. Acta Astronautica, 170:55 – 65, 2020.

[49] SV Dhurandhar et al. Fundamentals of the lisa stable flight formation. Classical and Quantum Gravity, 22(3):481, 2005.

[50] G. Di Mauro, D. Spiller, R. Bevilacqua, and S. D'Amico. Spacecraft formation flying reconfiguration with extended and impulsive maneuvers. Journal of the Franklin Institute, 356(6):3474–3507, 2019.

[51] Giuseppe Di Mauro and et. al. Survey on guidance navigation and control requirements for spacecraft formation-flying missions. Journal of Guidance, Control, and Dynamics, 41(3):581–602, 2018.

[52] Giuseppe Di Mauro, Dario Spiller, Riccardo Bevilacqua, and Simone D'Amico. Spacecraft formation flying reconfiguration with extended and impulsive maneuvers. Journal of the Franklin Institute, 356(6):3474–3507, 2019.

[53] Bryce Doerr and Richard Linares. Motion planning and control for on-orbit assembly using lqr-rrt* and nonlinear mpc. arXiv preprint arXiv:2008.02846, 2020.

[54] Bruce Donald et al. Kinodynamic motion planning. Journal of the Association for Computing Machinery, 40(5):1048–1066, 1993.

[55] Simone D'Amico. Relative orbital elements as integration constants of hill's equations. DLR, TN, pages 05–08, 2005.

[56] MC Eckstein, CK Rajasingh, and P Blumer. Colocation strategy and collision avoidance for the geostationary satellites at 19 degrees west. In International Symposium on Space Flight Dynamics, volume 6, 1989.

[57] Ahmed Elshamli et al. Genetic algorithm for dynamic path planning. In Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513), volume 2, pages 677–680. IEEE, 2004.

[58] Sepideh Faghihi, Siavash Tavana, and Anton HJ de Ruiter. Multiple spacecraft coordination and motion planning for full-coverage inspection of large complex space structures. Acta Astronautica, 202:119–129, 2023.

[59] Angel Flores-Abad, Ou Ma, Khanh Pham, and Steve Ulrich. A review of space robotics technologies for on-orbit servicing. Progress in aerospace sciences, 68:1–26, 2014.

[60] David Folta and Albin Hawkins. Results of nasa's first autonomous formation flying experiment: earth observing-1 (eo-1). In AIAA/AAS Astrodynamics Specialist Conference and Exhibit, page 4743, 2002.

[61] Emilio Frazzoli. Quasi-random algorithms for real-time spacecraft motion planning and coordination. Acta Astronautica, 53(4-10):485–495, 2003.

[62] CVM Fridlund. Darwin-the infrared space interferometry mission. ESA bulletin, 103(3):20–25, 2000.

[63] Jonathan D Gammell and Marlin P Strub. Asymptotically optimal sampling-based motion planning methods. Annual Review of Control, Robotics, and Autonomous Systems, 4:295–318, 2021.

[64] E Gill et al. Gemini: A milestone towards autonomous formation flying. In ESA Workshop on on-board Autonomy, volume 419. NoordwijkL: ES-TEC, 2001.

[65] Xuefei Gong et al. A scientific case study of an advanced lisa mission. Classical and Quantum Gravity, 28(9):094012, 2011.

[66] R. H. Gooding. A procedure for the solution of lambert's orbital boundary-value problem. Celestial Mechanics and Dynamical Astronomy, 48(2):145–165, 1990.

[67] J.L. Goodman. History of space shuttle rendezvous and proximity operations. Journal of Spacecraft and Rockets, 43(5):944–959, 2006.

[68] NASA GSFC. On-orbit satellite servicing study project report. NASA Project Report NP-2010-08-162-GSFC, Greenbelt, MD, 2010.

[69] Tommaso Guffanti and Simone D'Amico. Passively safe and robust multi-agent optimal control with application to distributed space systems. Journal of Guidance, Control, and Dynamics, pages 1–22, 2023.

[70] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In Robotics: Science and Systems, volume 3, pages 1–8, 2013.

[71] Marcus Holzinger, Jeremiah DiMatteo, Jeremy Schwartz, and Mark Milam. Passively safe receding horizon control for satellite proximity operations. In 2008 47th IEEE Conference on Decision and Control, pages 3433–3440. IEEE, 2008.

[72] Dario Izzo and Lorenzo Pettazzi. Autonomous and distributed motion planning for satellite swarm. Journal of Guidance, Control, and Dynamics, 30(2):449–459, 2007.

[73] Lucas Janson et al. Deterministic sampling-based motion planning: Optimality, complexity, and performance. 2015.

[74] Lucas Janson et al. Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions, 2015.

[75] Marcelo Kallman and Maja Mataric. Motion planning using dynamic roadmaps. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, volume 5, pages 4399–4404. IEEE, 2004.

[76] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7):846–894, 2011.

[77] Lydia Kavraki et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Robotics and Automation, IEEE Transactions on, 12:566 – 580, 09 1996.

[78] Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Kaelbling, and Tomás Lozano-Pérez. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 9916–9924, 2020.

[79] Mykel J. Kochenderfer. Decision Making Under Uncertainty: Theory and Application. The MIT Press, 2015.

[80] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), volume 2, pages 995–1001 vol.2, 2000.

[81] Yoshiaki Kuwata et al. Real-time motion planning with applications to autonomous urban driving. IEEE Transactions on control systems technology, 17(5):1105–1118, 2009.

[82] ER Lancaster. Solution of lambert's problem for short arcs. Celestial mechanics, 2(1):60–63, 1970.

[83] Jean-Claude Latombe. Robot Motion Planning, pages 153–200. Springer Science+Business Media, New York, 2 edition, 1991.

[84] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.

[85] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[86] Steven M. LaValle and Jr. James J. Kuffner. Randomized kinodynamic planning. The International Journal of Robotics Research, 20(5):378–400, 2001.

[87] John Leonard et al. A perception-driven autonomous urban vehicle. Journal of Field Robotics, 25(10):727–774, 2008.

[88] Wei-Jie Li, Da-Yi Cheng, Xi-Gang Liu, Yao-Bing Wang, Wen-Hua Shi, Zi-Xin Tang, Feng Gao, Fu-Ming Zeng, Hong-You Chai, Wen-Bo Luo, et al. On-orbit service (oos) of spacecraft: A review of engineering developments. Progress in Aerospace Sciences, 108:32–120, 2019.

[89] Yanbo Li et al. Asymptotically optimal sampling-based kinodynamic planning. The International Journal of Robotics Research, 35(5):528–564, 2016.

[90] J Salvador Llorente et al. Proba-3: Precise formation flying demonstration mission. Acta Astronautica, 82(1):38–46, 2013.

[91] Maurice Martin and Michael Stallard. Distributed satellite missions and technologies-the techsat 21 program. In Space Technology Conference and Exposition, page 4479, 1999.

[92] Reza Mashayekhi et al. Informed rrt*-connect: An asymptotically optimal single-query path planning method. 2019.

[93] J. W. McMahon and D. J. Scheeres. Linearized lambert's problem solution. Journal of Guidance, Control, and Dynamics, 39(10):2205–2218, 2016.

[94] Nik A. Melchior and Reid Simmons. Particle rrt for path planning with uncertainty. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 1617–1624, 2007.

[95] Borna Monazzah Moghaddam and Robin Chhabra. On the guidance, navigation and control of in-orbit space robotic missions: A survey and prospective vision. Acta Astronautica, 184:70–100, 2021.

[96] Michael Montemerlo et al. Junior: The stanford entry in the urban challenge. Journal of field Robotics, 25(9):569–597, 2008.

[97] Mohsen Mosleh, Kia Dalili, and Babak Heydari. Distributed or monolithic? a computational architecture decision framework. IEEE Systems journal, 12(1):125–136, 2016.

[98] Michael T Ohradzansky, Eugene R Rush, Danny G Riley, Andrew B Mills, Shakeeb Ahmad, Steve McGuire, Harel Biggie, Kyle Harlow, Michael J Miles, Eric W Frew, et al. Multi-agent autonomy: Advancements and challenges in subterranean exploration. arXiv preprint arXiv:2110.04390, 2021.

[99] V Orekhov and T Chung. The darpa subterranean challenge: A synopsis of the circuits stage. Field Robotics, 2(1):735–747, 2022.

[100] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1(1):33–55, 2016.

[101] Hyeongjun Park et al. Model predictive control for spacecraft rendezvous and docking with a rotating/tumbling platform and for debris avoidance. In Proceedings of the 2011 American Control Conference, pages 1922–1927, 2011.

[102] R. P. Patera. Satellite collision probability for nonlinear relative motion. Journal of Guidance, Control and Dynamics, 26(5):728–733, 2003.

[103] Jeff Phillips et al. Spacecraft rendezvous and docking with real-time, randomized optimization. In AIAA Guidance, Navigation, and Control Conference and Exhibit, page 5511, 2003.

[104] John E. Prussing. Geometrical interpretation of the angles cy and b in lambert's problem. Journal of Guidance and Control, 2(5):442–443, 1979.

[105] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurnỳ, et al. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6, pages 274–290. Springer, 2020.

[106] RD Russell and Lawerence F Shampine. A collocation method for boundary value problems. Numerische Mathematik, 19:1–28, 1972.

[107] Ryan P. Russell. Complete lambert solver including second-order sensitivities. Journal of Guidance, Control, and Dynamics, 45(2):196–212, 2022.

[108] Daniel P Scharf, Fred Y Hadaegh, and Scott R Ploen. A survey of spacecraft formation flying guidance and control. part ii: control. In Proceedings of the 2004 American control conference, volume 4, pages 2976–2985. Ieee, 2004.

[109] DP Scharf, B Acikmese, SR Ploen, and FY Hadaegh. Three-dimensional reactive collision avoidance with multiple colliding spacecraft for deep-space and earth-orbiting formations. In 4th International Conference on Spacecraft Formation Flying Missions & Technologies, 2011.

[110] Hanspeter Schaub. Spacecraft relative orbit geometry description through orbit element description through orbit element differences. In 14 US National Congress of Theoretical and Applied Mechanics Blacksburg, pages 23–28, 2002.

[111] Hanspeter Schaub. Analytical Mechanics of Space Systems, chapter 14. American Institute of Aeronautics and Astronautics, 3rd edition, 2014.

[112] Ray Sedwick et al. Exploiting orbital dynamics and micropropulsion for aperture synthesis using distributed satellite systems-applications to techsat21. In AIAA Defense and Civil Space Programs Conference and Exhibit, page 5289, 1998.

[113] Daniel Selva, Alessandro Golkar, Olga Korobova, Ignasi Lluch i Cruz, Paul Collopy, and Olivier L de Weck. Distributed earth satellite systems: What is needed to move forward? Journal of Aerospace Information Systems, 14(8):412–438, 2017.

[114] Romain Serra et al. Fast and accurate computation of orbital collision probability for short-term encounters. Journal of Guidance, Control, and Dynamics, 39(5):1009–1021, 2016.

[115] Xiaowei Shao, Jihe Wang, Dexin Zhang, and Junli Chen. Optimal satellite-formation collision-avoidance maneuver based on relative e/i vectors. Journal of Aerospace Engineering, 29(6):04016048, 2016.

[116] Peng Shu, Zhen Yang, and Ya-Zhong Luo. Higher-order lambert problem solution based on differential algebra. Journal of Guidance, Control, and Dynamics, 45(10):1913–1926, 2022.

[117] G Singh and F Hadaegh. Collision avoidance guidance for formation-flying applications. In AIAA Guidance, Navigation, and Control Conference and Exhibit, page 4088, 2001.

[118] Trey Smith, Jonathan Barlow, Maria Bualat, Terrence Fong, Christopher Provencher, Hugo Sanchez, and Ernest Smith. Astrobee: A new platform for free-flying robotics on the international space station. In International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS), number ARC-E-DAA-TN31584, 2016.

[119] J. A. Starek et al. Real-time, propellant-optimized spacecraft motion planning under clohessy-wiltshire-hill dynamics. pages 1–16, 2016.

[120] Joseph A. Starek et al. Bidirectional fast marching trees: An optimal sampling-based algorithm for bidirectional motion planning. 2014.

[121] Joseph A. Starek, Edward Schmerling, Gabriel D. Maher, Brent W. Barbee, and Marco Pavone. Fast, safe, propellant-efficient spacecraft motion planning under clohessy–wiltshire–hill dynamics. Journal of Guidance, Control, and Dynamics, 40(2):418–438, 2017.

[122] Anthony Stentz. The d* algorithm for real-time planning of optimal traverses. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 1994.

[123] Anthony Stentz. Optimal and efficient path planning for partially known environments. In Intelligent unmanned ground vehicles, pages 203–220. Springer, 1997.

[124] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. Artificial Intelligence Review, 56(3):2497–2562, 2023.

[125] Lorenzo Tarabini Castellani et al. Proba-3 mission. International Journal of Space Science and Engineering 5, 1(4):349–366, 2013.

[126] Ronald L Ticker. 2000 Survey of Distributed Spacecraft Technologies and Architectures for NASA's Earth Science Enterprise in the 2010-2025 Timeframe. National Aeronautics and Space Administration, Goddard Space Flight Center, 2000.

[127] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. Science Robotics, 7(66):eabp9742, 2022.

[128] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters, 85(6):317–325, 2003.

[129] J. Tschauner and P. Hempel. Rendezvous zu einemin elliptischer bahn umlaufenden ziel. Acta Astronautica, 11(2):104–109, 1965.

[130] Chris Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. Journal of field Robotics, 25(8):425–466, 2008.

[131] D. A. Vallado. Fundamentals of Astrodynamics and Applications. McGraw–Hill, New York, 3rd edition, 2007.

[132] Rafael Vazquez and et al. Model predictive control for spacecraft rendezvous in elliptical orbits with on/off thrusters. In IFAC-PapersOnLine, volume 48, pages 251–256, 2015.

[133] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. Soft computing, 22:387–408, 2018.

[134] Jihe Wang, Chengxi Zhang, and Jinxiu Zhang. Analytical solution of satellite formation impulsive reconfiguration considering passive safety constraints. Aerospace Science and Technology, 119:107108, 2021.

[135] Yi Wang, Xiaoqian Chen, Dechao Ran, Yong Zhao, Yang Chen, and Yuzhu Bai. Spacecraft formation reconfiguration with multi-obstacle avoidance under navigation and control uncertainties using adaptive artificial potential function method. Astrodynamics, 4:41–56, 2020.

[136] Avishai Weiss et al. Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies. IEEE Transactions on Control Systems Technology, 23(4):1638–1647, 2015.

[137] David C Woffinden and David K Geller. Navigating the road to autonomous orbital rendezvous. Journal of Spacecraft and Rockets, 44(4):898–909, 2007.

[138] Koji Yamanaka and Finn Ankersen. New state transition matrix for relative motion on an arbitrary elliptical orbit. Journal of Guidance, Control, and Dynamics, 25(1), 2012.

Appendix A

Spacecraft relative motion and relative orbital element description

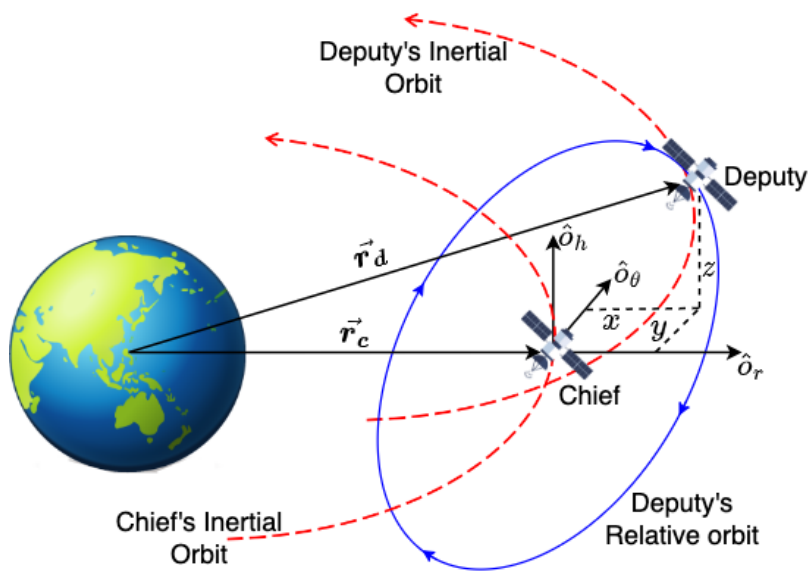


Figure A.1: Illustration of chief centered Local-Vertical Local-Horizontal frame and deputy's general relative motion with respect to the chief in this frame.

Let, $\mathcal{O} : \{\hat{o}_r, \hat{o}_\theta, \hat{o}_h\}$ denote the chief centered orbit frame, also called the Local-Vertical Local-Horizontal (LVLH) frame as shown in Figure A.1. The unit vector $\hat{o}_r$ is aligned with the radial direction (positive outwards), $\hat{o}_h$ is parallel to the orbit momentum vector (positive in orbit normal direction), and $\hat{o}_\theta$ completes the right-hand coordinate system. Mathematically these unit

vectors are defined as follows:

$$\hat{o}_r = \frac{\vec{r_c}}{|\vec{r_c}|} \tag{A.1}$$

$$\hat{o}_h = \frac{\vec{r_c} \times \vec{v_c}}{|\vec{r_c} \times \vec{v_c}|} \tag{A.2}$$

$$\hat{o}_\theta = \hat{o}_r \times \hat{o}_h \tag{A.3}$$

where, $\vec{r_c}$ and $\vec{v_c}$ are chief's inertial position and velocity vectors respectively. A deputy spacecraft's relative position ($\vec{\rho}$) and relative velocity ($\dot{\vec{\rho}}$) with respect to the chief in this LVLH frame are described as:

$$\vec{\rho} = [x, y, z], \quad \dot{\vec{\rho}} = [\dot{x}, \dot{y}, \dot{z}] \tag{A.4}$$

In this work, the following orbital element set is chosen which is a common choice for circular non-equatorial orbits to avoid singularities:

$$OE = (a, \theta, i, q_1, q_2, \Omega) \tag{A.5}$$

$$\text{where, } \theta = \omega + f \tag{A.6}$$

$$q_1 = e_X = e\cos(\omega) \tag{A.7}$$

$$q_2 = e_Y = e\sin(\omega) \tag{A.8}$$

Here a is the semi-major axis, θ is the true latitude angle (Eq. A.6, where ω is argument of periapsis, f is the true anomaly angle), i is the orbit inclination angle, Ω is the argument of the ascending node, and q_1 and q_2 are as defined in Eqs. A.7 and A.8 respectively where e is the eccentricity [111].

Given this definition of orbital elements, the deputy's relative orbital element set is defined as:

$$\delta OE = OE_d - OE = (\delta a, \delta\theta, \delta i, \delta q_1, \delta q_2, \delta\Omega) \tag{A.9}$$

where OE_d and OE refer to deputy's and chief's orbital elements respectively. The subscript d is used to denote deputy-related quantities and to distinguish them from the orbital elements of the

chief (without a subscript). Relative latitude ($\delta\theta$) is defined as:

$$\delta\theta = \delta\omega + \delta f \tag{A.10}$$

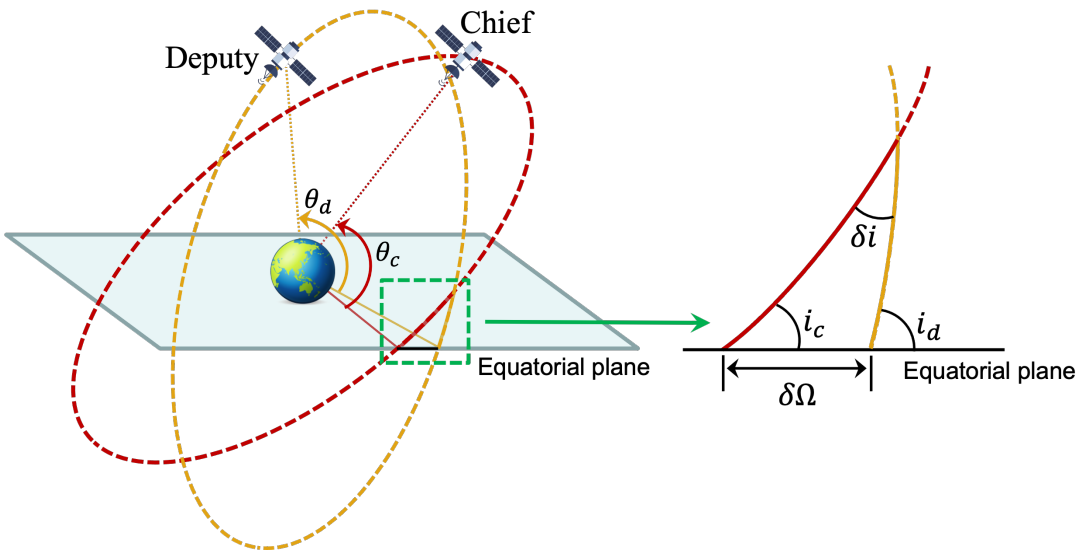
Relative eccentricity vector $\vec{\boldsymbol{\delta e}} = (\delta e_X, \delta e_Y)^T$ where,

$$\delta e_X = \delta q_1 = (\delta e + e)cos(\delta\omega + \omega) - ecos(\omega) \tag{A.11}$$

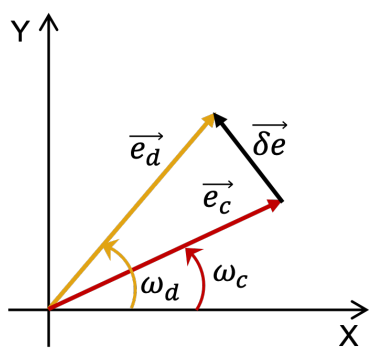
$$\delta e_Y = \delta q_2 = (\delta e + e)sin(\delta\omega + \omega) - esin(\omega) \tag{A.12}$$

A visual representation of the relative orbital elements is shown in Figure A.2.

It is important to realize here that, unlike the true anomaly (f) of a closed orbit, the relative true latitude ($\delta\theta$) of a closed relative orbit does not range between 0 to 2π. Varying the $\delta\theta$ means considering different phasing between the deputy and the chief that results in relative states that belong to different relative orbits around the chief as can be seen in Figure A.3.



(a) Relative inclination angle (δi), relative right ascension of the ascending node ($\delta\Omega$), and relative argument of latitude ($\delta\theta = \theta_d - \theta_c$)



(b) Relative eccentricity vector ($\vec{\delta e}$)

Figure A.2: Visual illustration of relative orbital elements.

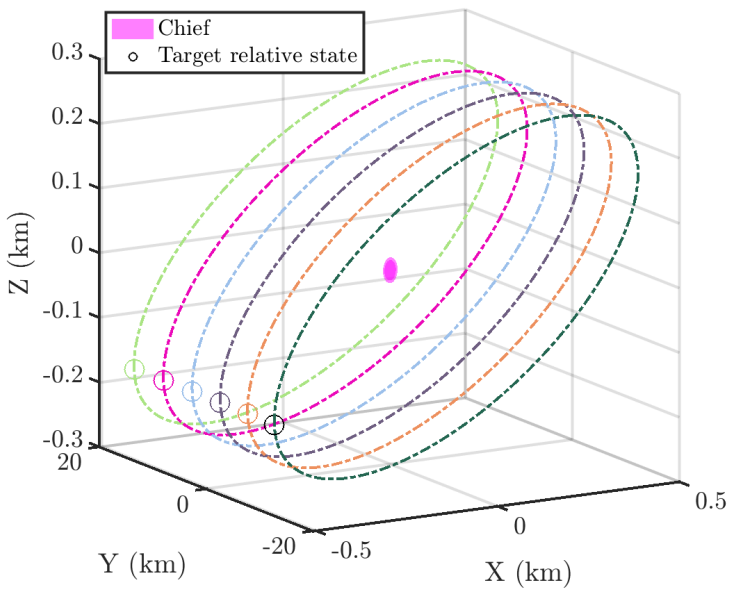


Figure A.3: Relative states corresponding to varying only the relative true latitude ($\delta\theta$) (keeping all other relative orbit elements the same) in the interval of [-0.05, 0.05] degrees for the same chief state.

Appendix B

Estimating the cost-to-go (Δv_{est}):

When the maximum relative separation between two spacecraft is relatively small compared to the Chief's orbit radius, we leveraged the Tschauner-Hempel (T-H) equations [129], which are a linear set of equations describing the motion of a Deputy around a generic eccentric chief orbit.

Tschauner-Hempel STM for relative spacecraft motion:

The non-dimensional T-H equations are provided below:

$$\begin{aligned} \bar{x}'' - 2\bar{y}' - \frac{3\bar{x}}{\rho} &= 0 \\ \bar{y}'' + 2\bar{x}' &= 0 \\ \bar{z}'' + \bar{z}' &= 0 \end{aligned} \tag{B.1}$$

$$\text{Where,} \quad \rho = 1 + e\cos(f), \tag{B.2}$$

$$(\bar{x}, \bar{y}, \bar{z}) = \frac{(x, y, z)}{R}, \tag{B.3}$$

$$R = \frac{1}{\rho}, \tag{B.4}$$

$$()' = \frac{\partial}{\partial f}() \tag{B.5}$$

here, e and f describe the Chief's eccentricity and true anomaly, respectively, and $\bar{()}$ notation refers to a non-dimensional state. The analytical solution for the relative motion has the following form

[138, 14].

$$\begin{aligned}
\bar{x}(f) &= K_1\rho sin(f) + K_2\rho cos(f) + K_3(2 - 3e\rho sin(f)\bar{k}(\tau - \tau_0)) \\
\bar{y}(f) &= K_1(\rho + 1)cos(f) - K_2(\rho + 1)sin(f) - 3K_3\rho^2\bar{k}(\tau - \tau_0) + K_4 \\
\bar{z}(f) &= K_5 sin(f) + K_6 cos(f)
\end{aligned} \tag{B.6}$$

where the parameters K_1, K_2, K_3, K_4 are constant of the motion, $\tau = nt$, n is the Chief's mean motion, and $\bar{k} = \sqrt{\mu}[a(1 - e^2)]^{-3/2}$ is derived from the constant of angular momentum (h) as reproduced here from [138]:

$$\text{Orbital rate } (\dot{f}) = (h/r_c^2) = (h/p^2)(1 + ecos(f))^2 = \bar{k}\rho^2 \tag{B.7}$$

where r_c is the magnitude of chief's inertial position vector, angular momentum magnitude $h = \sqrt{(}\mu p)$ and $p = a(1 - e^2)$.

Let $\rho cos(f) = c$, and $\rho sin(f) = s$, and $J(f)$ is defined as $\bar{k}(\tau - \tau_0) = \int_{f_0}^{f} \frac{1}{\rho(f)^2}\, df$. Then from equation B.6, we get:

$$\begin{aligned}
\bar{x}(f) &= K_1 s + K_2 c + K_3(2 - 3esJ) \\
\bar{x}'(f) &= K_1 s' + K_2 c' - 3eK_3(s'J + s/\rho^2) \\
\bar{y}(f) &= K_1(\frac{\rho + 1}{\rho})c - K_2(\frac{\rho + 1}{\rho})s - 3K_3\rho^2 J + K_4 \\
\bar{y}'(f) &= -2K_1 s + K_2(e - 2c) + -3K_3(1 - 2esJ) \\
\bar{z}(f) &= K_5 s/\rho + K_6 c/\rho \\
\bar{z}'(f) &= K_5 c/\rho - K_6 s/\rho
\end{aligned} \tag{B.8}$$

This gives,

$$\bar{\boldsymbol{x}}(\boldsymbol{f}) = \begin{bmatrix} s & c & 2-3esJ & 0 & 0 & 0 \\ (1+1/\rho)c & -(1+1/\rho)s & -3\rho^2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & s/\rho & c/\rho \\ s' & c' & -3e(s'J+s/\rho^2) & 0 & 0 & 0 \\ -2s & -2c+e & -3(1-2esJ) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c/\rho & -s/\rho \end{bmatrix} K \tag{B.9}$$

$$\implies \bar{\boldsymbol{x}}(\boldsymbol{f}) = \Phi_f K$$

where $K = [K_1 \quad K_2 \quad K_3 \quad K_4 \quad K_5 \quad K_6]^T$. Now given the initial state $\bar{\boldsymbol{x}}(\boldsymbol{f_0})$, K can be written as $K = \Phi_{f_0}^{-1}\bar{\boldsymbol{x}}(\boldsymbol{f_0})$. Substituting this in equation B.9 gives,

$$\begin{aligned} \bar{\boldsymbol{x}}(\boldsymbol{f}) &= \Phi_f \Phi_{f_0}^{-1} \bar{\boldsymbol{x}}(\boldsymbol{f_0}) \\ \implies \bar{\boldsymbol{x}}(\boldsymbol{f}) &= \Phi_{f_0}^{f} \bar{\boldsymbol{x}}(\boldsymbol{f_0}) \end{aligned} \tag{B.10}$$

where $\Phi_{f_0}^{f}$ is the STM of the relative motion that propagates the initial non-dimensional relative state ($\bar{\boldsymbol{x}}(\boldsymbol{f_0})$) to a final non-dimensional relative state ($\bar{\boldsymbol{x}}(\boldsymbol{f})$) for an arbitrary time. It is to be noted here that **bold notation** represents a vector.

Dimensionalizing non-dimensional states:

From equations B.3 and B.4, we have non-dimensional state's position vector $\bar{\boldsymbol{r}} = \frac{\boldsymbol{r}}{R} = \rho \boldsymbol{r}$.

This gives:

$$\boxed{\boldsymbol{r} = \frac{\bar{\boldsymbol{r}}}{1+ecos(f)} = \bar{\boldsymbol{r}}/\rho} \tag{B.11}$$

Now,

$$\dot{\boldsymbol{r}} = \boldsymbol{r'}\dot{f} \tag{B.12}$$

Using equations B.11, B.12 and substituting $\dot{f}$ from equation B.7, we get:

$$\begin{aligned} \dot{\boldsymbol{r}} &= \left(\frac{esin(f)\bar{\boldsymbol{r}}}{(1+ecos(f))^2} + \frac{\bar{\boldsymbol{r}}'}{1+ecos(f)} \right) \dot{f} \\ &= \left(\frac{esin(f)\bar{\boldsymbol{r}}}{\rho^2} + \frac{\bar{\boldsymbol{r}}'}{\rho} \right) \bar{k}\rho^2 \\ \implies & \boxed{\boldsymbol{v} = \bar{k}esin(f)\bar{\boldsymbol{r}} + \bar{k}\rho\bar{\boldsymbol{v}}} \end{aligned} \tag{B.13}$$

This gives the following relation between non-dimensional state $\bar{\boldsymbol{x}}$ and dimensional state $\boldsymbol{x}$:

$$\boldsymbol{x} = \begin{bmatrix} 1/\rho & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\rho & 0 & 0 & 0 \\ \bar{k}esin(f) & 0 & 0 & \bar{k}\rho & 0 & 0 \\ 0 & \bar{k}esin(f) & 0 & 0 & \bar{k}\rho & 0 \\ 0 & 0 & \bar{k}esin(f) & 0 & 0 & \bar{k}\rho \end{bmatrix} \bar{\boldsymbol{x}} \tag{B.14}$$

Cost-to-go estimate:

Equations B.10 can be written as

$$\begin{bmatrix} \bar{\boldsymbol{r}}(\boldsymbol{f}) \\ \bar{\boldsymbol{v}}(\boldsymbol{f}) \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{r}}(\boldsymbol{f_0}) \\ \bar{\boldsymbol{v}}(\boldsymbol{f_0}) \end{bmatrix} \tag{B.15}$$

where $\bar{\boldsymbol{r}}$ and $\bar{\boldsymbol{v}}$ refer to the non-dimensional relative position and velocity of the deputy with respect to the chief respectively.

The initial change in velocity ($\boldsymbol{\delta\bar{v}(t_0)}$) required to reach a target state is defined as

$$\boldsymbol{\delta\bar{v}(t_0)} = \boldsymbol{\bar{v}(t_0)}^{+} - \boldsymbol{\bar{v}(t_0)} \tag{B.16}$$

where $\boldsymbol{\bar{v}(t_0)}$ is the relative velocity of the deputy before executing the transfer and $\boldsymbol{\bar{v}(t_0)}^{+}$ is the velocity after applying the transfer impulse that can be solved from equation B.15 as follows:

$$\begin{aligned} \boldsymbol{\bar{r}(t)} &= \Phi_{rr}\boldsymbol{\bar{r}(t_0)} + \Phi_{rv}\boldsymbol{\bar{v}(t_0)}^{+} \\ \implies \boldsymbol{\bar{v}(t_0)}^{+} &= \Phi_{rv}^{-1}[\boldsymbol{\bar{r}(t)} - \Phi_{rr}\boldsymbol{\bar{r}(t_0)}] \end{aligned} \tag{B.17}$$

Similarly, the final change in velocity ($\boldsymbol{\delta\bar{v}(t)}$) given the relative orbital velocity on the arrival relative orbit ($\boldsymbol{\bar{v}(t)}$) can be computed as

$$\boldsymbol{\delta\bar{v}(t)} = \boldsymbol{\bar{v}(t)} - \boldsymbol{\bar{v}(t)}^{-} \tag{B.18}$$

where $\boldsymbol{\bar{v}(t)}^{-}$ is the final relative velocity achieved at the end of the transfer and can be solved from equation B.15 as follows:

$$\boldsymbol{\bar{v}(t)}^{-} = \Phi_{vr}\boldsymbol{\bar{r}(t_0)} + \Phi_{vv}\boldsymbol{\bar{v}(t_0)}^{+} \tag{B.19}$$

Both, $\boldsymbol{\delta\bar{v}(t_0)}$ and $\boldsymbol{\delta\bar{v}(t)}$ are dimensionalized using equation B.14 to give $\boldsymbol{\delta v(t_0)}$ and $\boldsymbol{\delta v(t)}$ respectively. This gives the final estimated total cost-to-go as:

$$\Delta v_{est} = \begin{cases} |\boldsymbol{\delta v(t_0)} + \boldsymbol{\delta v(t)}| + 100, & \text{if the estimated transfer arc is in violation of keep-out zones} \\ |\boldsymbol{\delta v(t_0)} + \boldsymbol{\delta v(t)}|, & \text{otherwise} \end{cases} \tag{B.20}$$

If the estimated transfer arc from the current node to the goal node is violating the keep-out zones, a high penalty (in this case, the penalty is picked as 100) is added to the estimated cost-to-go meaning it will be expensive to execute that transfer.

Appendix C

Review of Prussings and Conway's Lambert's Solution (reproduced from [93]):

Lambert's problem is to determine the orbit that allows an object to transfer between two terminal position vectors ($\boldsymbol{r_1}$ and $\boldsymbol{r_2}$) in a given time of flight (Δt) using Keplerian dynamics. The Prussings and Conway's Lambert's solution [32] is based on determining two Lagrange parameters, α and β [104] defined as:

$$\sin(\frac{\alpha}{2}) = \sqrt{\frac{s}{2a}} \tag{C.1}$$

$$\sin(\frac{\beta}{2}) = \sqrt{\frac{s-c}{2a}} \tag{C.2}$$

where a is the transfer orbit's semimajor axis, s is defined as

$$s = \frac{1}{2}(r_1 + r_2 + c) \tag{C.3}$$

where r_i are magnitudes of the position vectors and c is given by

$$c = |\boldsymbol{r_2} - \boldsymbol{r_1}| \tag{C.4}$$

To solve for α and β, we must know a, all of which must simultaneously satisfy the time-of-flight problem cast as

$$\Delta t = \frac{a^{3/2}}{\sqrt{\mu}}[\alpha - \beta - (\sin\alpha - \sin\beta)] \tag{C.5}$$

$$\implies \Delta t = g(\boldsymbol{r_1}, \boldsymbol{r_2}, a) \tag{C.6}$$

A full discussion of the solution to parameters a, α, and β are discussed in [93, 32]. Using these solutions, the terminal velocities can be computed as:

$$\boldsymbol{v_i} = f_i(\boldsymbol{r_1}, \boldsymbol{r_2}, a) \tag{C.7}$$

Specifically,

$$\boldsymbol{v_1} = (B + A)\boldsymbol{u_c} + (B - A)\boldsymbol{u_1} \tag{C.8}$$

$$\boldsymbol{v_2} = (B + A)\boldsymbol{u_c} - (B - A)\boldsymbol{u_2} \tag{C.9}$$

where,

$$A = \sqrt{(\frac{\mu}{4a})}\cot(\frac{\alpha}{2}) \tag{C.10}$$

$$B = \sqrt{(\frac{\mu}{4a})}\cot(\frac{\beta}{2}) \tag{C.11}$$

$$\boldsymbol{u_1} = \frac{\boldsymbol{r_1}}{r_1} \tag{C.12}$$

$$\boldsymbol{u_2} = \frac{\boldsymbol{r_2}}{r_2} \tag{C.13}$$

$$\boldsymbol{u_c} = \frac{\boldsymbol{r_2} - \boldsymbol{r_1}}{c} \tag{C.14}$$