OPTIMAL GUIDANCE AND CONTROL OF HETEROGENEOUS SWARMS FOR IN-ORBIT SELF-ASSEMBLY OF LARGE SPACE STRUCTURES: ALGORITHMS AND EXPERIMENTS

BY

REBECCA C. FOUST

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor Soon-Jo Chung, Chair and Director of Research
Professor Petros Voulgaris
Assistant Professor Koki Ho
Dr. Fred Hadaegh, Jet Propulsion Laboratory, California Institute of Technology

Abstract

Satellite design has been harshly constrained by surviving entry into space though the majority of the satellite's lifetime exists in much calmer conditions. Significant study has recently gone into assembling satellites and space structures in-orbit. Several methods have been proposed involving an assembler robot or astronaut which puts the parts together, but in the interest of saving resources we believe that it is advantageous to make this process autonomous and robust by leveraging existing optimal guidance and control schemes for a self-assembling swarm. This approach avoids single-point failures, requires significantly less ground support, provides increased reliability due to redundancy, increased flexibility, the ability to reconfigure for future missions, and the ability to self-repair. Since the satellites required could be mass-produced from a small set of different component types, the benefit from economy of scale would reduce the overall mission cost when compared to monolithic satellites. This dissertation details an optimal guidance and control scheme to enable in-orbit self-assembly of a large structure from a heterogeneous swarm of satellites. In the proposed scheme, the component satellites for the heterogeneous swarm are chosen to promote flexibility in final shape inspired by crystal structures and Islamic tile art. After the ideal fundamental building blocks are selected, basic nanosatellite-class satellite designs are presented to enable accurate attitude control simulations. The Swarm Orbital Construction Algorithm (SOCA) is a guidance and control algorithm that allows for the limited type heterogeneity and docking ability required for in-orbit assembly. The algorithm was tested in a simulated perturbed 6-DOF spacecraft dynamic environment for planar and out-of-plane final structures. The algorithm is then experimentally validated coarsely on omnidirectional wheeled robots and precisely on-board the M-STAR robots in the precision flat floor facility in the Caltech Aerospace Robotics and Control lab, the largest of its kind at any university.

In support of this effort, a better way of handling nonlinear dynamics constraints within sequential convex programs was developed. SCP is a useful tool in obtaining real-time solutions to direct optimal control, but it is unable to adequately model nonlinear dynamics due to the linearization and discretization required. As nonlinear program solvers are not yet functioning in real-time, a tool is needed to bridge the gap between satisfying the nonlinear dynamics and completing execution fast enough to be useful. Two methods are proposed, sequential convex programming with nonlinear dynamics correction (SCPn) and modified SCPn (M-SCPn), which mixes SCP and SCPn to reduce runtime and improve algorithmic robustness. Both methods are proven to generate optimal state and control trajectories that satisfy the nonlinear dynamics. Simulations are presented to validate the efficacy of the methods as compared to SCP.

In addition, several autonomous rendezvous and docking (AR&D) technologies were studied because in-orbit self-assembly requires repeated, reliable autonomous docking to ensure success. Docking small satellites in space is a high-risk operation due to the uncertainty in relative position and orientation and the lack of mature docking technologies. This is particularly true for missions that involve multiple docking and undocking procedures like swarm-based construction and reconfiguration. A tether-based docking system was evaluated in simulation as compared to traditional propulsive methods. The tether-based method provides a way to reduce the risk of the dock, since the docking maneuver is performed with a much smaller satellite and the reeling maneuver can be done gently. Tether-based methods still require some actuation on the docking end of the tether, and propulsion on such small systems is inexact. An electromagnetic docking system was investigated to address these issues. Designed with reconfigurable self-assembly in mind, the gripping mechanism is androgynous, able to dock at a variety of relative orientations, and tolerant of small misalignments. The electromagnetic system can be used either on the end of a tether or on the main spacecraft itself since the electromagnet is well controlled and the measurement of the ambient electromagnetic field can be used as to improve the intersatellite distance estimate enough to reduce the risk of docking to the main spacecraft. The performance of this system was validated experimentally on-board the M-STARs. The performance of the electromagnetic docking system on-board the simulators is then compared against a propulsive docking system tested in the same way. Overall, this dissertation provides optimal guidance and control algorithms for nonlinear systems to enable in-orbit self-assembly of heterogeneous swarms.

Acknowledgments

This work would not have been possible without the support and understanding of many people. First, I would like to thank my advisor, Prof. Soon-Jo Chung for his technical and personal guidance. Without Dr. Chung's support and encouragement I would certainly not have made it to this point. I would also like to thank my NASA advisor Dr. Fred Hadaegh for the professional advice and encouragement. Thank you also to my committee members, Prof. Voulgaris and Prof. Ho for their assistance and support.

I would like to thank my colleagues and friends Kyunam Kim, Yashwanth Nakka, Patrick Spieler, Kai Matsuka, Benjamin Riviere, Xichen Shi, Salar Rahili, Wolfgang Hoenig, Michael O'Connell, Vincenzo Capuano, Ellande Tang, Amir Rahmani, Jean-Pierre de la Croix, and Giri Subramanian for making the long journey much more pleasant and fruitful. Thanks especially to Sorina Lupu and Saptarshi Bandyopadhyay who have been a great comfort and assistance through the highs and lows.

Thanks also to Shelly Kaur, Kris Benke, and Zalak Thaker for helping me survive Illinois, and my friends back home Katie Guzman, Beth Leumas, Catharine Love, Janice Rattigan, Lizzy Wunsch and Tina Thomas for their support. Most of all, I would like to thank my family who can always make me laugh, and who never doubted that I could succeed.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AR&D	Autonomous Rendezvous and Docking
DOF	Degree of Freedom
FDIR	Fault Detection Identification and Recovery
GNC	Guidance Navigation and Control
HCW	Hill-Clohessy-Wiltshire
LVLH	Local Vertical Local Horizontal
MIB	Minimum Impulse Bit
MPC	Model Predictive Control
M-SCPn	Modified Sequential Convex Programming with Nonlinear Dynamics Correction
M-STAR	Multi-Spacecraft Testbed for Autonomy Research
PWM	Pulse Width Modulated
ROS	Robot Operating System
RPM	Rotation Per Minute
SCP	Sequential Convex Programming
SCPn	Sequential Convex Programming with Nonlinear Dynamics Correction
SE-SCP	Spherical Expansion, Sequential Convex Programming
SmallSat	Small Satellite
SOCA	Swarm Orbital Construction Algorithm

List of Symbols

Chapter 3

β	=	Trust region shrink rate
dt	=	Time step
$\mathbf{f}$	=	Nonlinear dynamics at time step k
$\mathcal{F}$	=	Cost integrand function
g_i	=	Convex inequality constraints
J	=	Cost function
k	=	Discrete time parameter
N	=	Number of agents
N_j	=	Set of neighbor agents
p	=	Number of inequality constraints
R_{col}	=	Collision avoidance radius
R_{comm}	=	Communication radius
$\mathbf{x}_0$	=	Initial state constraint
$\mathbf{x}_f$	=	Terminal state constraint
$\mathbf{x}_k^{(w)}$	=	State trajectory at the k-th time step for the w-th iteration
$\mathbf{x}_{n,k}^{(w)}$	=	Nominal, nonlinear state trajectory at the k-th time step for the w-th iteration
t	=	Time
t_0	=	Initial time
t_f	=	Final time
T	=	Final discrete time step

$\mathbf{u}_k^{(w)}$	=	Control input trajectory at the k-th time step for the w-th iteration
U_{max}	=	Maximum control input
w	=	Iteration number
υ_k	=	Numerical integration quadrature weight

Chapter 4

N	=	number of spacecraft
N_j	=	number of docks required at a target
n_i	=	maximum number of docks a satellite can perform based on its type
t	=	time
t_f	=	final time ($t_f = T\Delta t$)
t_{run}	=	time required to compute the optimization
Δt	=	length of time step
$\mathcal{N}_{[j]}$	=	set of the closed neighborhood
$\mathcal{D}_j$	=	set of agents that are assigned to dock with agent j
$\mathcal{P}_j$	=	set of agents that have a higher priority than j
R_{col}	=	minimum distance between spacecraft to avoid a collision in the optimization
R_{bl}	=	distance at which the agent chooses to dock or avoid an approaching agent
R_{dock}	=	physical separation of centroids of docking agents
$\bar{R}_{\text{col}}$	=	minimum distance between spacecraft to avoid a collision in reality
R_{comm}	=	maximum distance a spacecraft can communicate ($R_{\text{comm}} > R_{\text{col}}$)
T	=	total number of time steps
T_H	=	number of time steps in the model predictive control horizon
U_{max}	=	maximum allowable magnitude of the control vector
V_{max}	=	maximum allowable magnitude of the relative velocity vector
$h_j(\mathbf{x}[k], k])$	=	cost to transfer a spacecraft from $\mathbf{x}[k]$ at time k to $\mathbf{x}_f$ at T
k	=	time step k

k_0	=	time step at the start of the model predictive control horizon
$\mathbf{x}_0$	=	state vector at initial time
$\mathbf{x}_f$	=	state vector at final time
$\mathbf{x}_j$	=	state vector of spacecraft j
$\bar{\mathbf{x}}$	=	nominal state vector
B	=	Control influence matrix
CG	=	XYZ position of center of gravity
d_{act}	=	Actuator direction wrt body frame
f_{des}	=	Vector of desired XYZ forces and torques
I_3	=	3x3 Identity matrix
${}^{A}J$	=	Moment of Inertia tensor wrt frame A
m	=	Mass
NT	=	Number of thrusters
${}^{A}r_b$	=	Vector b wrt frame A
${}^{B}r_b$	=	Vector from CG of b to dock port in frame B
${}^{A}R^{B}$	=	Rotation from frame B to frame A

Chapter 5

θ	=	Heading Angle
μ_0	=	Magnetic Permeability of Free Space
ρ	=	Density of the Electromagnet Coil Material
σ	=	Coil Axial Turn Density
B	=	Control Influence Matrix
N	=	Number of Windings
T	=	Array of Thruster Commands
f	=	Electromagnet Force
h	=	Coil Separation Distance

i	=	Current
l	=	Coil Thickness
m	=	Electromagnet mass
m_z	=	Moment about Z-Axis
p	=	Power Consumption of the Electromagnet
r	=	Coil Radius
v	=	Drive Voltage of the Electromagnet

Chapter 1

Introduction

Design and construction of large space systems is often constrained by factors that have more to do with launch than the intended mission, like launch vehicle fairing size or ability to withstand launch loading. Satellites constructed in space would not experience these design constraints, allowing for lighter, more capable satellites. Start to finish construction in-orbit is not yet possible, but improvements can still be made through recent advances in swarm spacecraft guidance and control [70, 99, 100] and autonomous rendezvous and docking [51].

By leveraging the above swarm guidance and control algorithms, a large space structure can be constructed from a swarm of component satellites. The advantages of such a mission are clear: increased reliability due to redundancy, increased flexibility, ability to reconfigure for future missions, and ability to self-repair [143]. Applications for such missions range from the small scale, where the components are microsatellites building a support structure for a distributed telescope or a solar sail, to the large scale, where components are habitat modules building a space colony. The mission concept is illustrated in Fig. 1.1 using two types of agents. The mission steps are as follows:

Step 1 The components enter into loose formation to stay close to other components until they are used (e.g. see collision-free J_2 invariant passive relative orbits in [99]).

Step 2 The components determine their desired final position in the assembly and move to take the position using SOCA.

Step 3 Along the path to the final position, components assigned to neighboring positions

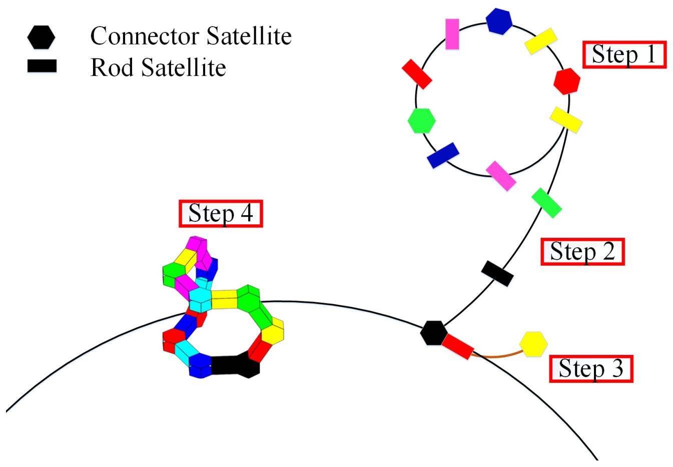


Figure 1.1: Outline of Mission Steps

dock and proceed combined.

Step 4 Finally, a complete structure is made once all components have reached their final destination.

Since the satellites required could be mass-produced from a small set of different component types, the benefit from economy of scale would reduce the overall mission cost when compared to monolithic satellites. Autonomous rendezvous and docking (AR&D) technologies for small satellites are necessary to enable in-orbit assembly schemes. SmallSats have the distinct advantages of low cost and weight, which make them particularly attractive to multi-agent missions like in-orbit assembly. With these benefits come restrictions, however. SmallSats have reduced performance across the board, in power generation, computing, pose determination and control. These difficulties are even more pronounced as the size of the satellite is decreased. Nanosatellite and CubeSat scale sensors and actuators are limited in performance and have few options available [19]. The most successful proximity operations CubeSat mission, the CanX-4&5 mission, was able to achieve centimeter-level position determination and sub-meter level control [24]. For a docking mission, this level of accuracy could lead to dock failure or even collision. These sensing and actuation gaps make activities like AR&D much more difficult, but through the use of well-designed controllers and docking actuators these difficulties can be overcome.

1.1 Literature Review

To reach the potential of this swarm, it is imperative that the proposed guidance and control algorithm allows each agent in the swarm to act independently, without global knowledge of the swarm. Centralized algorithms are disadvantageous because they require all-to-all or all-to-one communication, which is difficult in large spacecraft swarms, either highly taxes the communication systems or introduces a single-point failure. This requirement means the algorithm must be decentralized, so each agent decides its own trajectory based on information from the agents with which it can communicate. Also, to increase the flexibility in potential final structures, it is beneficial to use multiple types of agents in the construction swarm, so the guidance and control algorithm must be able to handle heterogeneous agents. Finally, the proposed algorithm must control both position and attitude of the spacecraft since docking is required.

In literature, many examples of decentralized swarm guidance schemes exist, but the swarms are typically homogeneous [144, 136, 137, 111]. The heterogeneous swarm guidance schemes typically use centralized algorithms. A similar modular swarm construction mission was demonstrated using a homogeneous swarm of rectangular boats constructed in a brick pattern [128]. Though this demonstration involved a homogeneous swarm with a planar construction and centralized guidance and assignment, the assembly scheme docks along the way to the final location, similar to the present paper. Another team created a satellite assembly guidance and control scheme for a homogenous swarm to a predefined final formation using a glideslope algorithm to guide each satellite to a dock relative to other satellite, with collision avoidance and particular constraints on relative velocity at certain waypoints along the trajectory [109]. The approach is suboptimal; it uses linearized dynamics and neglects perturbations and relative attitude but succeeds in building the formation at a low fuel cost [109].

The field of robotic self-assembly has many interesting and innovative mechanisms. The systems that are applied to space assembly are typically multi-use robots with multiple endeffectors like the MoleCubes [146], which have a useful reconfiguration technique where the cubes rotate along a diagonal axis to switch the location of two faces. This actuation type could be very useful in the in-space construction scheme we have defined. Another interesting actuation type with space applications is used by the MIT M-Blocks. The M-Blocks are cubes that have magnetic edges and an internal flywheel which allows them to pop up and latch on to make various configurations [119]. Self-assembling robotics applied to space applications is very limited. The Transformable Robotic Infrastructure-Generating Object Network (Trigon) system uses robotic self-assembly for in-space construction to facilitate human planetary missions [72]. The Trigon system is multi-use and can build structures from rovers to habitats using a "kit-of-parts", a set of Trigon parts. Each Trigon part is essentially a face with actuators along the edges that can interact to self-assemble by moving parts along the structure [72]. In orbit, some methods propose a free-flying tether robot which can dock with components to combine them into an overall structure [49, 129]. CalPoly's PolyBots perform self-assembly using two types of agents with hermaphorditic docking ports. The two agent types are similar to our design, a node and a segment. Though the system architecture is similar to our concept, PolyBots are mainly intended for surface operations and can be connected to form an arbitrary robot. The flexibility of the segment agent allows the PolyBot chains to be used for locomotion and manipulation [142].

The guidance and control algorithm detailed in this dissertation expands upon prior work, the Swarm Assignment and Trajectory Optimization (SATO) algorithm. SATO was used to solve a target assignment and collision-free path planning problem by implementing a decentralized auction algorithm with a trajectory planner which implemented model predictive control using sequential convex programming (MPC-SCP)[100, 101]. The two algorithms are run sequentially over the course of the SATO algorithm so that the initial assignments and trajectories can be updated as agent connectivity changes or collision avoidance is needed. All agents are assumed to know the set of target locations, and have a limited communication radius. SATO was designed for a homogeneous swarm targeting to a disconnected, free-flying formation and did not incorporate docking or assembly of any kind. While SATO performs admirably for this type of homogeneous swarm formation building, it must be altered for the proposed heterogeneous construction swarm. In addition to the heterogeneity logic, the collision avoidance logic in MPC-SCP must be carefully relaxed to allow docking agents to come within the collision avoidance radius.

Enabling such a mission also require guidance and control algorithms that are advanced enough to operate autonomously in complex dynamics environments with a large number of agents. Optimal guidance and control methods for nonlinear systems are currently lacking in real-time implementations, so other avenues must be taken to achieve fast, efficient motion on-board advanced aerial and space vehicles. In previous work, sequential convex programming was used to successively linearize the nonlinear dynamics about a trajectory, but the error in linearization and discretization of the dynamics adds up and results in a trajectory that may no longer satisfy the original constraints.

Several methods exist to tackle nonlinear optimal control problems, but they often fall short in capability when it comes to real-time implementation for multi-agent systems. Pseudospectral methods are well suited to handle the nonlinear dynamics, but grow computationally prohibitive as the number of agents increases and are not yet implemented in real-time [50, 116]. Nonlinear solvers are getting fast enough to implement onboard for well posed problems but do not adapt well and are not suitable for multi-agent problems thus far [112]. Typical real-time quadrotor trajectory generation and control implementations rely on simplifications like assuming differentially flat trajectories, using path primitives, or decoupling the states to reduce the optimization burden [71, 91]. These methods are effective for individual quadrotors, some even in obstacle-rich environments, but are not designed to handle multi-agent systems. Mixed integer linear programming can also be implemented in real-time if certain assumptions are valid, but also scales poorly with the number of agents [118, 61, 44].

Convex optimization problems are easily and efficiently solved, but many common constraints on trajectory generation and control problems are nonconvex and many dynamical systems are nonlinear [83]. Through convexification, relaxation, and approximation some such problems may be fully solved [9, 138]. In other cases though, the adjustments restrict the set of possible solutions to the point where the new problem is infeasible though the original nonconvex problem is feasible.

Sequential convex programming (SCP) parses a nonconvex problem into a sequence of convex programs with convexified or linearized constraints. This sequence of convex problems allows the convexification to be more closely tailored to the original nonconvex problem, allowing more solutions to be found, while still making use of efficient convex optimization solvers. SCP is used for a plethora of applications relating to optimal path planning and control, gaining popularity with the collision-free path planning for multiple quadrotors or spacecraft, developed independently by two research teams [14, 97, 98, 101]. SCP has been used before these works on several other applications, like generalized nonconvex optimization and single agent optimal control [20, 45]. SCP has also been used by roboticists for problems like team-based path planning in non-convex environments [30] and complex robot, complex environment optimal path planning [124]. Recently, SCP has been applied to more complex problems with tight constraints like optimal entry and landing [81, 134, 84]. Often, the optimal state trajectory found with SCP is implemented with a tracking controller rather than determining the optimal control concurrently with the optimal state [37].

Previous work [97, 98, 101] has used SCP for nonlinear problems ranging from collision-free guidance and control of swarms to optimal robotic motion planning in cluttered environments [17] to in-orbit satellite self-assembly [54, 52]. In some cases, it is possible to relax the original nonlinear nonconvex problem using a slack variable such that the relaxed problem is solvable using SCP and the solution is exact with respect to the nonlinear dynamics under certain assumptions [86]. This solution type is preferable, but requires rigorous analysis to prove the equivalence between the relaxed and original problems, and may not be possible for all problems. More often though, the nonlinear dynamics constraint is linearized and the nonconvex inequality constraints are convexified in order to make the nonconvex optimal guidance and control problem convex. The linearization and discretization of the dynamics cause error to accumulate with each successive iteration of SCP.

Typical implementations of SCP use an approximation of the dynamics without addressing this gap between the approximated and actual dynamics since the optimal state trajectory is used more frequently [82, 83, 23]. Frequently, trust regions are used to keep the successive solutions sufficiently close to mitigate the buildup of error due to linearization [88, 87]. These are helpful in achieving convergence, but ultimately still allow an accretion of error in the system, particularly for hard problems where several iterations are necessary. To fully trust the optimal state and control trajectories, this error must be corrected, especially before implementation on fragile problems like multi-agent docking or aggressive, cluttered-workspace trajectory planning where small errors in the commanded trajectory can result in mission failure.

As the agents in the swarm dock and move to their desired final positions, they must be able to control the structures they create. The act of docking makes certain thrusters unavailable due to the plume impingement on the now docked spacecraft, and the control allocation algorithm must be able to dynamically adjust the control influence matrix to model these new connections. Similar adaptations exist to allocate control in the case of actuator failure, which is the crux the recovery portion of FDIR [133]. In this case however, actuators must be added and the mass properties must be changed. This has been handled in most commonly in previous docked satellites through gain scheduling, where the controller gains are pre-determined for each configuration and stored in a table [94, 74]. This is disadvantageous for an in-orbit assembly scheme because the sheer number of gains to be computed and stored is intractable. Another common method is system identification, where the docked spacecraft characterize the mass properties by actuating thrusters and calculating the response. This is fuel and time intensive to generate high-fidelity models. The final option is online model calculation, in which each agent stores its mass properties

Mission Name	Lead Organization	Status	Docking Actuation
AAReST [28]	Caltech and Surrey Space Center	Scheduled, 2019	Electromagnets
STARS 1 [106]	Kagawa University	Jan. 23, 2009	Pre-Tethered
STARS 2 [107]	Kagawa University	Feb. 28, 2014	Pre-Tethered
CPOD[26]	Tyvak Nano-Satellite Systems	Awaiting Launch	Thrusters
Rascal [7]	St. Louis University	Cancelled	Thrusters
CleanSpace One [6]	EPFL	Gathering Funds	Thrusters
Spheres [108]	MIT	May-November 2006	Thrusters
STRaND 2 [28]	Surrey Space Center	Not Available	Electromagnets

Table 1.1: Review of SmallSat Docking Missions

and as they dock, they calculate the new mass properties of the combined system [94]. This is beneficial because it does not require much data storage or any fuel usage, but it can be sensitive to errors in docking alignment. Very few sources in literature have looked into this online model calculation problem and solutions typically end at determining the new mass and control properties to use with the controller and neglect to automate the removal of blocked actuators from the allocation problem [95].

In addition to the guidance and control algorithms, it is necessary that the docking system used in the in-orbit construction scheme be well-chosen. Autonomous rendezvous and docking missions have stringent requirements and the technology is often fraught with mechanism failures, making it one of the highest risk space operations [25]. Current AR&D methods also lead to extensive fuel consumption. Careful trajectory planning and improved sensors can reduce the fuel cost somewhat, but since docking necessitates two satellites coming in contact, the failure of any subsystem can easily lead to mission failure. For example, the DART mission was intended to show the rendezvous and docking of two satellites, but instead ended in mission failure when a relatively small navigation error led to excessive fuel consumption and collision with the docking target [3].

Performing proximity operations and docking with nanosatellites and CubeSats is particularly difficult. Nanosatellite-scale sensors and actuators are generally inaccurate and with few options, limited by availability, space, and power consumption [19].

In literature, several solutions have been proposed to perform docking in space for SmallSats. The main categories include thruster-based docking, tether-based docking, and magnet-based docking. A selection of SmallSat missions involving docking that flew into space or are proposed to fly into space is presented in Table 1.1.

Thruster-based docking is the most common form of docking and has a long history of success in space. Thrusters provide the most maneuverability of any of the docking types discussed. With thrusters, the initial separation can be as large as the propellant storage of the satellite allows.

However, thruster-based docking has drawbacks like the propellant consumption. The MIB of the thruster affects the resolution of the maneuver and drives consumption up for precise motion like docking. This also dictates the minimum impact velocity. Thruster-based docking also comes with the risk of thruster plume impingement, which can lead to disturbance forces and undue heating, causing dock and component failure [25, 51, 56]. These limitations can make missions like assembly of a structure difficult.

Tether-based docking is advantageous since it is repeatable and the main two spacecraft stay separated until the tether begins to reel, so docking failures can often be fixed with little risk to the main spacecraft. Additionally, with some extra sensors the tether can be used to improve the relative navigation accuracy [38, 33]. Tether-based docking poses disadvantages such as tether entanglement if tension is not maintained, as well as increased complexity in design. However, it is more robust to failure since the tether can be rolled back and a new attempt can be made.

Several studies have examined controlling tethered satellites and docking via tethers, but generally the satellites are launched with the tethers connected [102, 106]. In some concept missions without pre-tethered satellites, the tether is ejected forcefully to intercept the target like shooting a harpoon gun. To add more flexibility, some studies use an electromagnetic end effector to aide in the capture [110, 115, 47]. While this helps in final capture, it increases complexity, cost, and weight.

Magnet-based docking comes in two forms, permanent magnets and electromagnets. The permanent magnets create a hard dock and have a small capture range, but are simple to implement. In [113], permanent magnets are used for the docking mechanism, but the capture range issue is overcome by using thrusters to bring the CubeSats into close proximity.

Electromagnet-based docking can provide a smoother, more accurate docking. In addition, it does not require consumables to operate, and does not create plumes. As a disadvantage, more current and thus more mass is required to bring satellites together from longer distances. As superconductor technology improves though, this limitation can be ameliorated. Several studies have suggested electromagnetic docking, in addition to the ARReST and STRaND 2 missions from Table 1.1, but these docks are typically hard docks to ensure dock completion. Electromagnets have been also studied for formation flying applications, like in references [127], [79], [13], [80], or in contact-free docking applications such as [76].

1.2 Main Contributions

This dissertation investigates optimal guidance and control for robots with nonlinear dynamics, for heterogeneous swarm self-assembly in-orbit, and for robots traversing cluttered environments. The main contributions of this paper are as follows:

- Two novel algorithms, SCPn and M-SCPn, are presented which correct for the linearization and discretization error caused when nonlinear dynamics are used in sequential convex programming. This prevents the error from building up and allows the algorithm to converge to a solution that satisfies the nonlinear dynamics and the optimization constraints. A new constraint is added to SCPn to ensure that the resulting optimal trajectory is a feasible solution for the original nonconvex problem. Using this constraint, it is proven that SCPn converges to the optimal solution and the solution remains feasible for the nonlinear dynamics. Due to runtime and robustness issues with SCPn, a second algorithm, modified SCPn (M-SCPn) is presented which mixes SCP and SCPn to great effect, achieving solutions that are true to the nonlinear dynamics in shorter times for a wider set of test cases. Both algorithms are tested through simulations involving a quadrotor traversing a simple obstacle field.

- Novel modular spacecraft simulating robots are presented, capable of frictionless motion with spacecraft-like actuation methods for testing multi-agent spacecraft guidance and control algorithms in a variety of configurations. These robots are also capable of testing actual spacecraft hardware and fully integrated CubeSats. The robots are controlled well enough to be suitable for verifying proximity operations and docking maneuvers. Experimental results are presented to demonstrate the utility of the system.

- An in-orbit construction algorithm is presented which uses a decentralized auction algorithm with MPC-SCPn and is suitable for limited type heterogeneity in the swarm and allows for docking satellites while avoiding undesired collisions. The algorithm takes in a shape without pre-assigned target positions and solves the optimal assignment and collision-free trajectory generation together. The assignment is performed using a distributed auction with a variable number of targets in case of agent loss, and strict bonding rules to address the heterogeneity. MPC-SCPn is used to generate the collision-free trajectories, with modifications to relax collision constraints on agents targeting neighboring positions to allow the agents to dock before reaching the target. Simulation and experimental results are presented for three final configurations with up to 54 agents.

- A trade study over several docking methods is presented to demonstrate the benefits of tether-based and electromagnet-based autonomous docking over traditional propulsive docking. This study is broken into two parts, one where tether and propulsive docking are compared in a simulation study and one where electromagnet and propulsive docking are compared in simulation and experiments. Both tethers and electromagnets are propellant-free, which makes them desirable from a lifetime standpoint. The electromagnet-based docking system is capable of very low terminal velocity docking, which is ideal for in-orbit assembly applications.

1.3 Organization

The organization and flow of this dissertation are shown in Fig. 1.2. Chapter 2 discusses the development of the Multi-Spacecraft Testbed for Autonomy Research spacecraft simulator robots. This testbed is used for experimental validation across subsequent chapters. Chapter 3 details the development of the sequential convex programming algorithm which is designed to function with nonlinear dynamics constraints. The resulting algorithms are then verified against an uncorrected algorithm and against the true system dynamics. In Chapter 4, the in-orbit construction algorithm for heterogeneous swarms is presented and validated in simulation and on the aforementioned M-STARs. Chapter 5 details a study of several different kinds of docking schemes, which are tested against each other in simulation and in experiments on the M-STARs. Chapter 6 explores optimal motion planning in a cluttered environment by leveraging sequential convex programming with nonlinear dynamics. The dissertation concludes in chapter 7.

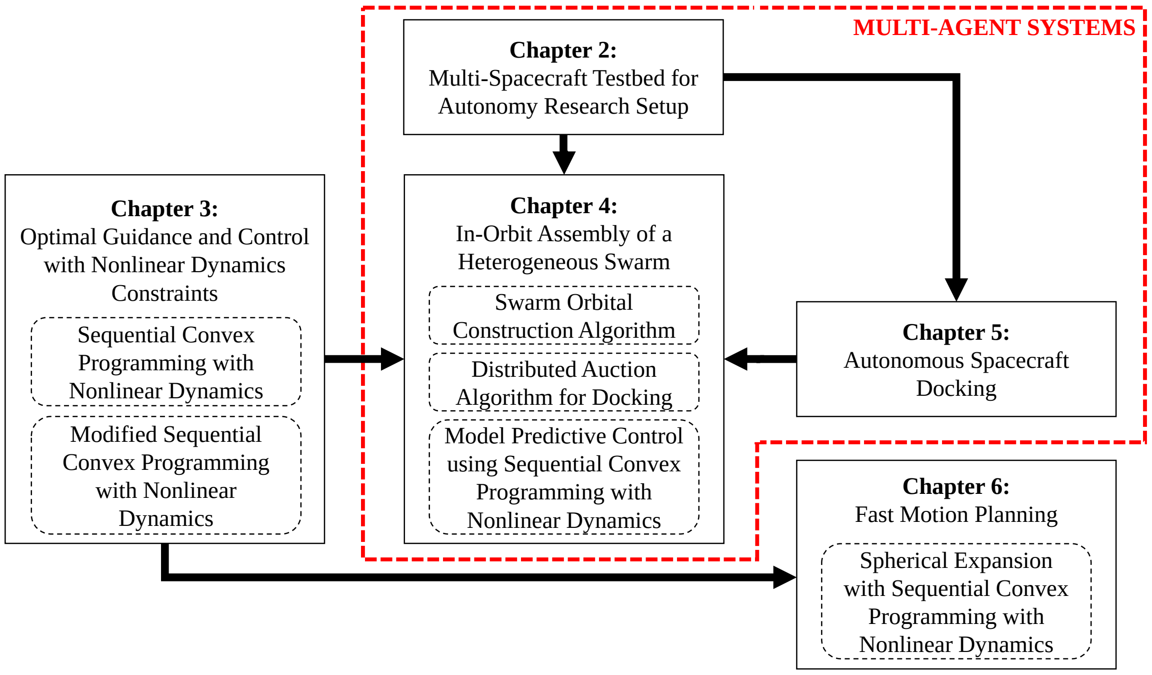

MULTI-AGENT SYSTEMS
Chapter 2:
Multi-Spacecraft Testbed for Autonomy Research Setup
Chapter 3:
Optimal Guidance and Control with Nonlinear Dynamics Constraints
Sequential Convex Programming with Nonlinear Dynamics
Modified Sequential Convex Programming with Nonlinear Dynamics
Chapter 4:
In-Orbit Assembly of a Heterogeneous Swarm
Swarm Orbital Construction Algorithm
Distributed Auction Algorithm for Docking
Model Predictive Control using Sequential Convex Programming with Nonlinear Dynamics
Chapter 5:
Autonomous Spacecraft Docking
Chapter 6:
Fast Motion Planning
Spherical Expansion with Sequential Convex Programming with Nonlinear Dynamics

Figure 1.2: Dissertation Organizational Diagram

Chapter 2

Multi-Spacecraft Testbed for Autonomy Research Setup

2.1 Motivation

Historically, air bearing [125] platforms have been a popular choice to build spacecraft dynamics simulators. Air bearing spacecraft simulation platforms were developed by several research laboratories [48, 126, 93, 36, 10, 58, 121, 122, 117, 145, 135, 59, 141, 131, 139, 123, 33]; a selection of these simulation platforms is shown in Table 2.1. Existing air bearing platforms can be classified into four types based on the mode of operation: 3 degrees-of-freedom (DOF) planar [145, 131, 93], 3-DOF attitude [126, 131, 36], 5-DOF planar and attitude [48, 122, 135, 123], and 6-DOF planar and attitude with gravity-axis motion [139, 121, 141]. The air bearing system acts as a ground-based simulator platform for flight-like actuators and sensors, which provides an opportunity to test flight algorithms and emulate space dynamics [40].

In this chapter, we describe the development of a new 6-DOF spacecraft simulator, the Multi-Spacecraft Testbed for Autonomy Research (M-STAR), that is designed to be modular and accommodates 3-DOF, 4-DOF, 5-DOF, and 6-DOF operation with minimal mechanical modifications. The spacecraft simulator hardware was designed to have decentralized control and information sharing capabilities with neighboring agents in view of the future goal of testing multi-agent GNC algorithms using up to five of these simulators. Each spacecraft has 16 thrusters and 4 reaction wheels to study fault-tolerant control.

In view of the model-based GNC algorithms a detailed nonlinear dynamic model for the 5-DOF system was derived by modelling it as a 3D pendulum on a gliding planar platform with a center of gravity offset in the 3D pendulum. The nonlinear dynamic model

Organization	Name	DOF
Naval Postgraduate School	POSEIDYN	3
Georgia Institute of Technology	ASTROS	5
Florida Institute of Technology	ORION	6
University of Florida	ADAMUS	6
Yonsei University	ASTERIX	5
NASA Jet Propulsion Laboratory (JPL)	FCT SSDT	5 3
German Aerospace Center (DLR)	TEAMS	3 and 5
Massachusetts Institute of Technology	SPHERES ARGOS	3 3 (attitude)

Table 2.1: A sample of spacecraft simulators from other institutions. [48, 126, 93, 36, 10, 58, 121, 122, 117, 145, 135, 59, 141, 131, 139, 123, 33]

is decoupled by assuming a small center of gravity offset. A nonlinear hierarchical control law is proposed for fast attitude dynamics and slower position dynamics due to the time-scale separation. The control law computes forces and torques collocated to the dynamics. Control allocation [75] is done to map the collocated control signal to the actuator signal. Optimization formulations [22] can be used to solve the control allocation problem, typically formulated as a linear program. For the M-STAR control allocation, we implement a generalized pseudo-inverse method for control allocation with a weighted influence matrix to account for actuator saturation limits, as the optimization formulations are computationally expensive for real-time implementation.

The position control of the M-STAR is performed using on-off solenoids, which are inherently nonlinear due to mechanical delays and varying pressure in the manifold that supplies compressed air to the solenoids. The solenoids are characterized [29] by measuring the force produced for varying on-off time, using a calibrated load cell. A linear model to compute the on time of a thruster is developed using the measured data for a given force requirement at each time step. The control law, control allocation scheme, and thruster model are tested for position tracking using a Robot Operating System (ROS) based software framework.

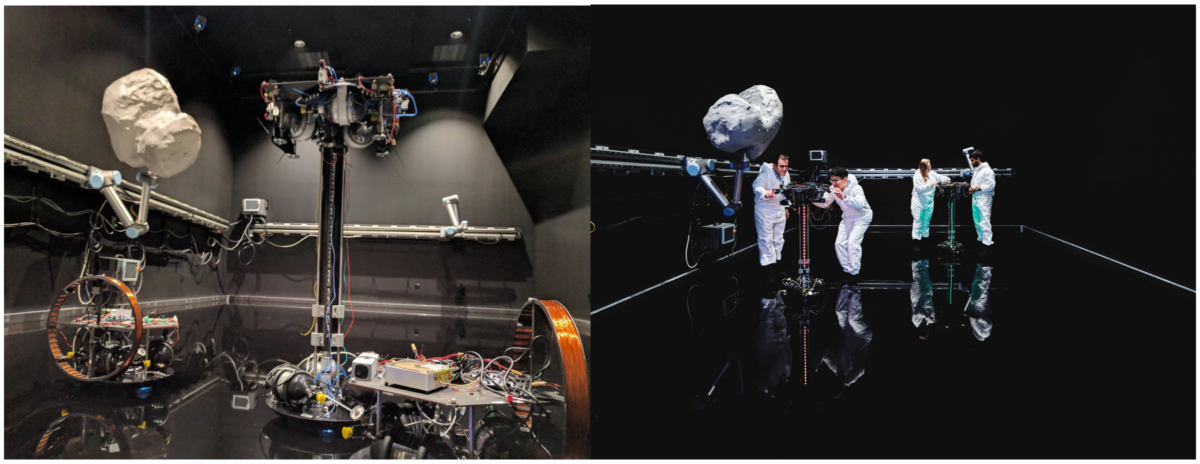


Figure 2.1: Multiple 6-DOF M-STAR spacecraft at Caltech's Aerospace Robotics and Control Lab.

2.2 Overview of the Facility

The spacecraft simulator facility requires the following three components to be operational: the epoxy flat floor, the compressed air filling station, and the M-STARs. The epoxy flat floor is a high precision floor with flatness within 0.001 inches for frictionless translation of the spacecraft dynamics simulator using three flat air-bearing pads. Fig. 2.1 shows the facility with multiple M-STAR spacecraft simulators and protection for collisions on the outer edge of the floor. The full 6-DOF spacecraft simulator can be seen in the middle with two 3-DOF simulators on the sides.The second component, the filling station, is comprised of an industrial air compressor and two 6,000 psi storage tanks. The filling station is used to fill the on-board air cylinders that supply air to the flat air bearings, spherical air bearing, and 16 on-off non-latching solenoid valves that act as thrusters on the simulator. The M-STAR shown in Fig. 2.2 acts as the dynamic simulation platform for a SmallSat and includes all the necessary on-board sensors, actuator systems, and computing to achieve full 6-DOF control. The pose of the spacecraft simulator is estimated using 14 motion capture cameras mounted on the ceiling of the facility. In the following section, we elaborate on the subsystem hardware of the simulator.

2.3 M-STAR Spacecraft Simulator Hardware

The Caltech Aerospace Robotics and Controls Lab's 6-DOF spacecraft dynamics simulator for spacecraft formation control research was designed to accommodate up to a 12U CubeSat as a payload. The floating test bed simulates 5-DOF dynamic motion and 1-DOF kinematic motion along the gravity direction, with translation and attitude stages decoupled via a spherical air bearing. The translation stage floats frictionlessly on the precision floor using three flat round air bearings. The attitude stage has a hemispherical air bearing ball that floats frictionlessly on the cup mounted at the top of the linear actuator on the translation stage. Tables 2.2 and 2.3 list the hardware components on both the translation stage and attitude stages respectively. The hardware on each stage is divided into three subsystems: 1) mechanical, including structural and pneumatic components; 2) electrical, including power, computing, and low level controller boards; and 3) actuation, to impart torque or impulse in the required degree of freedom. Each of these components plays an essential role in achieving torque-free controlled motion.

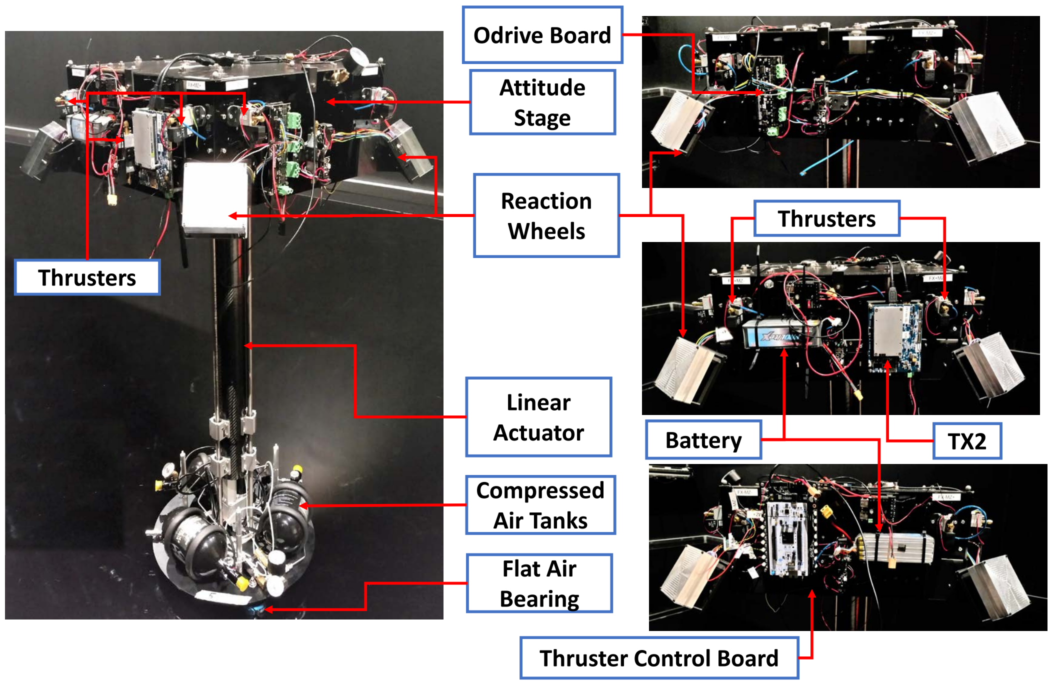


Figure 2.2: M-STAR spacecraft dynamics simulator.

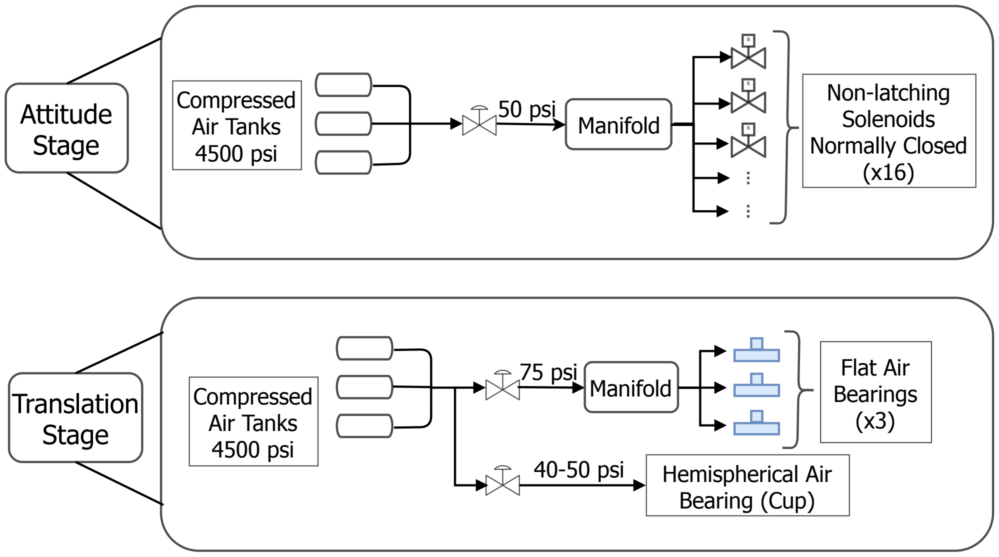

Attitude Stage
Compressed Air Tanks 4500 psi
50 psi
Manifold
Non-latching Solenoids Normally Closed (x16)
Translation Stage
Compressed Air Tanks 4500 psi
75 psi
Manifold
Flat Air Bearings (x3)
40-50 psi
Hemispherical Air Bearing (Cup)

Figure 2.3: Flowchart of pneumatic system on translation and attitude stage.

2.3.1 Translation Stage.

The translation stage provides frictionless in-plane motion for the whole simulator using three linear flat round air bearings. It consists of three compressed air cylinders running at 4500 psi, a spherical air bearing cup, pneumatic components for pressure regulation, and tubing required to supply air for the bearings. The pneumatic system on the translation stage is shown in Fig. 2.3. In addition, it is equipped with a linear actuator, a brushless DC linear motor for achieving motion in the gravity direction with supporting control electronics.

The different operation modes of operation (3-DOF, 4-DOF, 5-DOF, and 6-DOF) can be achieved as follows:

- 3-DOF: spherical air bearing turned off and linear actuator replaced with a passive tube
- 4-DOF: spherical air bearing turned off
- 5-DOF: linear actuator replaced with a passive tube
- 6-DOF: all actuators active

This provides flexibility in operation and allows the construction of algorithms with increased complexity. The compressed air storage tanks' capacity was designed to achieve at least 25-30 minutes of flotation time at the operating pressure in 6-DOF mode. Several custom-designed add-ons can be incorporated on the translation stage such as docking ports and reaction wheels for attitude control.

Subsystem	Component
Mechanical	NewWay Air Bearing Compressed Air Cylinders Structure Design Spherical Air Bearing Regulator
Actuator	Progressive Automation Linear Actuator
Electronics and Power	Battery Linear Actuator Controller Raspberry Pi

Table 2.2: List of components on the translation stage.

2.3.2 Attitude Stage.

The attitude stage structure was designed using carbon fiber composites and honeycomb materials, optimized to provide a flotation time of up to 30 minutes with a payload of 12 kilograms. It has a box structure and acts as a platform for a potential payload, such as a 12U CubeSat. The attitude stage structure has the hemispherical ball of the air bearing pair and floats on the translation stage to provide 3-DOF frictionless attitude motion. This stage has 16 on-off non-latching solenoids with custom made nozzles and four in-house reaction wheels as actuators. The power distribution board for the attitude stage and the low-level controller for the thrusters are designed at Caltech. The schematic of the pneumatic subsystem for supplying regulated compressed air to the thrusters is shown in Fig. 2.3. It includes three compressed air cylinders, a regulator, and a manifold for air distribution. The regulated pressure is supplied to all the thrusters through the manifold to maintain the pressure across them. The operating pressure of the thrusters is decided based on experimental characterization of the solenoids. The electrical subsystem of the attitude stage is shown in Fig. 2.4. We chose an NVIDIA Jetson TX2 as the main computer to run the GNC and perception algorithms. The computer communicates the control signal to the low level boards as shown in Fig. 2.4. The subsystem components of the stages are listed in the Table 2.3.

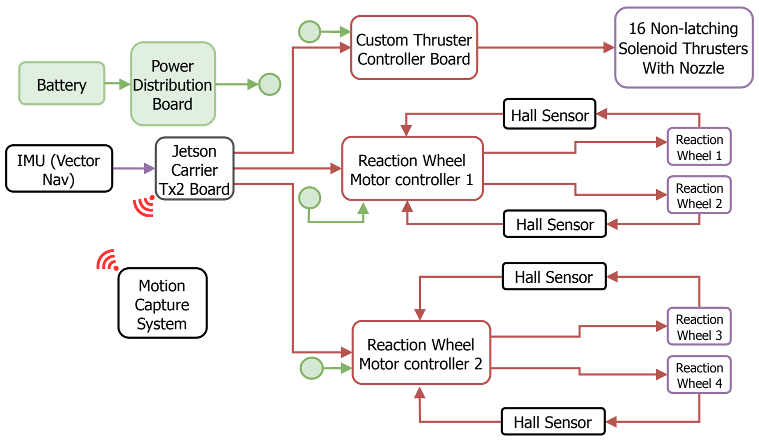


Figure 2.4: Attitude stage architecture.

Subsystem	Component
Mechanical	Structure Nozzles Pneumatics
Actuator	Thrusters Custom Reaction Wheel Assembly
Electronics and Power	Battery Power Distribution Board Thruster Control Board ODRIVE Reaction Wheel Driver Maxon Motor Reaction Wheel Motor NVIDIA Jetson TX2 Computer

Table 2.3: List of components on the attitude stage.

2.3.3 M-STAR Software Architecture

The software for the simulator was designed to allow for interchangeable guidance, navigation, and control modules. The architecture is implemented in C++ using abstract base classes for the three modules, with virtual loop functions for subclasses to implement. As illustrated in Fig. 2.5, navigation subclasses are responsible for generating updated state data for the guidance system and controller. The guidance system maintains a trajectory of desired states, from which the controller selects a target state for the current time step and implements the required dynamics. The current experimental setup features waypoint guidance, motion capture camera based navigation, and the 5-DOF controller outlined in the next section. However, these could respectively be swapped for an arbitrary motion-planning algorithm, pose feedback from integrated on-board sensor data, and controllers for the four configurations of the simulator.

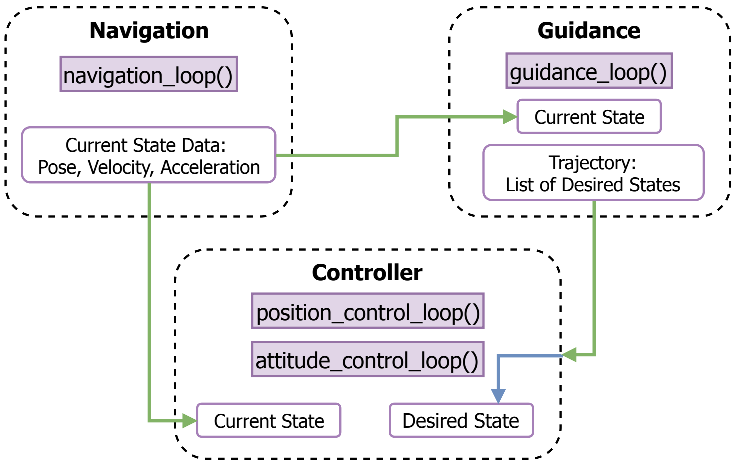


Figure 2.5: Software architecture design.

The architecture is built on Robotic Operating System (ROS) framework, which allows for each loop to be scheduled at a unique rate that can be changed at run time. Data from other modules is automatically fetched before each loop runs. ROS also provides a messaging architecture for communicating with peripheral boards, the ability to create unique launch configurations for different module setups, and test logging.

2.4 Dynamics and Control

Each M-STAR has two links coupled using a spherical air bearing as a joint. This system can be modelled as a three dimensional pendulum on a floating platform with a ball joint to provide 3-DOF rotation of the pendulum (modelling the attitude of the spacecraft) and 2-DOF planar motion of the floating platform. Constraints on the 3D pendulum motion due to mechanical interference between the attitude stage and the translation stage are shown in Table 2.4. The coordinate systems used for deriving the kinematics and dynamics of the system are shown in Fig. 2.6. The inertial reference frame on the test floor is defined by the coordinate system (X_i, Y_i, Z_i) with origin O_i . A non-rotating reference frame (X_{ib}, Y_{ib}, Z_{ib}) that is parallel to the inertial frame, is attached to the attitude stage with origin O_b at the center of the hemispherical bearing to define the orientation of the attitude stage. The attitude stage dynamics are derived in terms of the angular rates in the body frame (X_b, Y_b, Z_b) at origin O_b. Before proceeding to the discussion on the dynamics and control implementation, the attitude representation used for describing the motion of the 3D pendulum in SO(3) space is discussed.

Pitch (rotation about X_{ib})	$\pm 45°$
Roll (rotation about Y_{ib})	$\pm 45°$
Yaw (rotation about Z_{ib})	$\pm 180°$

Table 2.4: Constraints on the angular motion of the attitude stage.

2.4.1 Attitude Kinematics

The attitude of the 3D pendulum can be represented by any attitude representations including quaternions [89], Modified Rodrigues Parameters (MRPs) [89], and SO(3) rotation matrix. For example, the MRPs $p \in \mathcal{R}^3$ are stereographic projections of the unit quaternions [89], $q \in \mathcal{H}$, where $\mathcal{H}$ is the Hamiltonian space and have a bijective mapping to the quaternion sphere are used here. The attitude representation in MRPs takes into account the unit norm of the quaternions. The attitude kinematics equation is given using the body angular rates $\omega \in \mathcal{R}^3$. The kinematics of MRPs are given as follows.

$$\dot{p} = Z(p)\omega; \text{ where } Z(p) = \frac{1}{2}\left(I_3\left(\frac{1 - p^T p}{2}\right) + pp^T + S(p)\right), S(p) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix} \tag{2.1}$$

The rotation matrix $R(p)^\top$ to transform from the frame (X_{ib}, Y_{ib}, Z_{ib}) to the body frame (X_b, Y_b, Z_b) in terms of the MRPs is given as:

$$R(p)^\top = I_{3\times 3} - \frac{4(1 - p^T p)}{(1 + p^T p)^2} S(p) + \frac{8}{(1 + p^T p)^2} S(p)^2 \tag{2.2}$$

The transformation $R(p)$ is used in mapping the external force due to thrusters in the body frame to the inertial frame for controlling the translation dynamics.

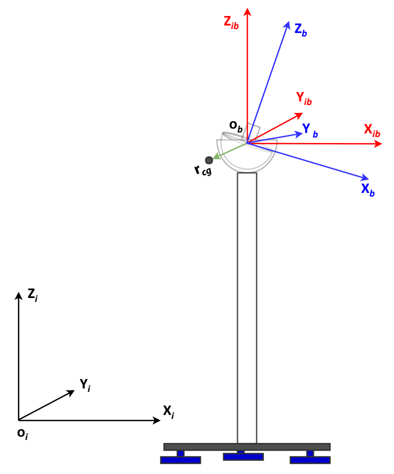


Figure 2.6: Coordinate Systems used for the derivation of the dynamic model.

2.4.2 Nonlinear Dynamic Model

The dynamics of the 5-DOF system with the velocity v_b at the centre of rotation of the attitude stage and angular rates of the attitude stage ω in body frame (X_b, Y_b, Z_b) is given in the Eq. (2.4), where, r_{cg} is the center of gravity offset from the center of rotation of the attitude stage in the body frame coordinates, J is the mass moment of inertia of the attitude stage about the center of rotation in the body frame, $R(p)$ is defined in Eq. (2.2), (x, y) is the planar location of the center of rotation from the inertial frame origin, m_a is the mass of the attitude stage, and m_t is the mass of the translation stage. In the following equations, $a_1 = [1; 0; 0]$, $a_2 = [0; 1; 0]$ and $a_3 = [0; 0; 1]$ are unit vectors in the reference frame (X_{ib}, Y_{ib}, Z_{ib}).

$$\dot{\bar{P}} = R(p)v_b \quad \text{where} \quad \bar{P} = (x, y, 0)^\top, \quad D = (a_1^\top; a_2^\top; 0) \tag{2.3}$$

$$M_b(p)\begin{bmatrix}\dot{\omega}\\ \dot{v}_b\end{bmatrix} + C_b\begin{bmatrix}\omega\\ v_b\end{bmatrix} + H_b = \tau_b \tag{2.4}$$

$$M_b(p) = \begin{bmatrix} J & m_a S(r_{cg})R(p)^\top D R(p) \\ m_a\left(S(r_{cg})R(p)^\top D R(p)\right)^\top & (m_a + m_t) \end{bmatrix} \tag{2.5}$$

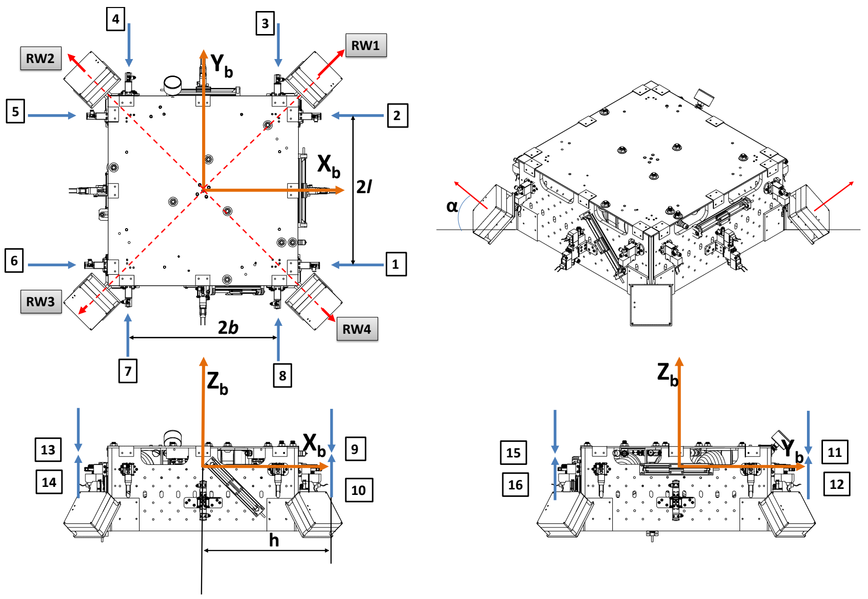


Figure 2.7: Attitude Stage with actuator configuration and nomenclature in the body frame.

$$C_b = \begin{bmatrix} -S(J\omega) & m_a S(r_{cg})R(p)^\top D R(p) S(\omega) \\ -m_a R(p)^\top D R(p) S(\omega) S(r_{cg}) & (m_t + m_a) S(\omega) \end{bmatrix} \tag{2.6}$$

$$H_b = \begin{bmatrix} -m_a g S(r_{cg}) R(p)^\top a_3 \\ 0 \end{bmatrix} \tag{2.7}$$

The control inputs to the system are represented by $\tau_b = \left[\begin{smallmatrix}\tau_f \\ \tau_t\end{smallmatrix}\right]$, which include forces due to thrusters τ_f and torques τ_t due to thrusters and reaction wheels in body frame. The control design is done in body frame. The forces computed in body frame τ_f are transformed to forces in inertial frame $\tau_p = R(p)\tau_f$ for implementation of the position control law. The implementation of the transformation and the influence of thrusters in the body frame on the position dynamics in inertial frame is discussed in the following sections. In the body frame, for the 5-DOF dynamics in Eq. (2.4) it can be proved that $\dot{M}_b - (C_b + C_b^T) = 0$ and that $\dot{M}_b - 2C_b$ is a skew-symmetric matrix. The matrix form in Eq. (2.4) will be used in the following section to derive a controller that globally exponentially tracks a given position and almost globally exponentially tracks an attitude trajectory.

2.4.3 Control Design for Full Nonlinear Dynamics

The objective of the control design is to ensure that the 5-DOF of M-STAR, $[\bar{P}(t), p(t)]$ given in Eq. (2.3), exponentially tracks a given trajectory $[\bar{P}_d(t), p_d(t)] \in C^2([0,\infty])$. The following theorem states the nonlinear control law and proves the global exponential stability of the closed-loop system in Eq. (2.10). Here the variables $s_\omega = \omega - \omega_r$ and $s_v = v_b - v_{b_r}$ define the states for virtual dynamics. The variables w_r and v_{b_r} define the reference signal computed from filtered desired states dynamics given in the following Eq. (2.8).

$$\begin{aligned} \omega_r &= Z^{-1}(p)\dot{p}_d(t) + Z^{-1}(p)\Lambda_\omega(p_d(t) - p) \\ v_{b_r} &= R^\top \dot{\bar{P}}_d(t) + R^\top \Lambda_v(\bar{P}_d(t) - \bar{P}) \end{aligned} \tag{2.8}$$

The closed-loop system in terms of virtual states s_ω, s_v, given in Eq. (2.10), with the control law Eqs. (2.8–2.9), is globally exponentially stable in the sense of the Euclidean norm, assuming the feedback gains $K_\omega, K_v, \Lambda_\omega, \Lambda_v > 0$ and the inertia matrix M_b is positive definite and uniformly bounded with lower bound $\lambda_{\min}$ and upper bound $\lambda_{\max}$. For the proof, please see [105].

$$\tau_b = M_b \begin{bmatrix} \dot{\omega}_r \\ \dot{v}_{b_r} \end{bmatrix} + C_b \begin{bmatrix} \omega_r \\ v_{b_r} \end{bmatrix} + H_b - \begin{bmatrix} K_\omega & 0 \\ 0 & K_v \end{bmatrix} \begin{bmatrix} s_\omega \\ s_v \end{bmatrix} \tag{2.9}$$

$$M_b \begin{bmatrix} \dot{s}_\omega \\ \dot{s}_v \end{bmatrix} + C_b \begin{bmatrix} s_\omega \\ s_v \end{bmatrix} + \begin{bmatrix} K_\omega & 0 \\ 0 & K_v \end{bmatrix} \begin{bmatrix} s_\omega \\ s_v \end{bmatrix} = 0 \tag{2.10}$$

2.4.4 Control Implementation

For the control implementation, it is assumed that the attitude stage is coarsely balanced with small r_{cg}. Eq. (2.11) shows the decoupled translation dynamics in inertial frame and rotational dynamics in body frame with small center of gravity offset. The terms in the dynamics corresponding to the r_{cg} act as a bounded disturbance at the input $d(t) = \begin{bmatrix} d_w(t) \\ d_p(t) \end{bmatrix}$

for small accelerations.

$$\begin{bmatrix} J & 0 & 0 \\ 0 & m_a + m_t & 0 \\ 0 & 0 & m_a + m_t \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} \omega \times J\omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -m_a g S(r_{cg}) R(p)^\top a_3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \tau_r \\ \tau_p \end{bmatrix} + \begin{bmatrix} d_w(t) \\ d_p(t) \end{bmatrix} \tag{2.11}$$

$$\begin{aligned} d_w(t) &= -m_a S(r_{cg}) R(p)^\top a_1 \ddot{x} - m_a S(r_{cg}) R(p)^\top a_2 \ddot{y} \\ d_p(t) &= \begin{bmatrix} -m_a (S(r_{cg}) R(p)^\top a_1)^\top \dot{\omega} - m_a a_1^\top R(p) S(\omega)^2 r_{cg} \\ -m_a (S(r_{cg}) R(p)^\top a_2)^\top \dot{\omega} - m_a a_2^\top R(p) S(\omega)^2 r_{cg} \end{bmatrix} \end{aligned} \tag{2.12}$$

A hierarchical control law was implemented with an inner attitude control loop and an outer position control loop because of the timescale separation between the two dynamics, Eq. (2.11). Given a desired position trajectory, $[x_d(t), y_d(t)] \in \mathcal{R}^2$, and attitude trajectory represented in MRPs, $p_d(t) \in \mathcal{R}^3$, the control law presented below exponentially tracks both position and attitude trajectories using smooth control inputs for the decoupled dynamics for no disturbance. In the case with a bounded disturbance at the input, the closed-loop system is finite-gain $\mathcal{L}_p$ stable. The control input to the position dynamics is simplified from Eq. (2.9) and is given by Eq. (2.13).

$$\tau_p = (m_t + m_a) \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \end{bmatrix} - K_d \begin{bmatrix} \dot{x} - \dot{x}_d \\ \dot{y} - \dot{y}_d \end{bmatrix} - K_p \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix} \tag{2.13}$$

$$(m_t + m_a) \begin{bmatrix} \ddot{x} - \ddot{x}_d \\ \ddot{y} - \ddot{y}_d \end{bmatrix} - K_d \begin{bmatrix} \dot{x} - \dot{x}_d \\ \dot{y} - \dot{y}_d \end{bmatrix} - K_p \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix} = d_p(t) \tag{2.14}$$

The closed-loop position dynamics with the control law in Eq. (2.13) are given in Eq. (2.14). The gain values K_d and K_p are chosen to achieve the required position tracking performance. The attitude controller in Eq. (2.15) is exponentially stable [18] with no disturbance and tracks a given desired attitude trajectory that is $\mathcal{C}^2$ continuous. It can be shown that this control law is simplified form of the controller proposed in Eq. (2.9). The nonlinear controller is finite-gain $\mathcal{L}_p$ stable with bounded disturbance at the input.

$$\begin{aligned} \tau_r &= J\dot{\omega}_r - S(J\omega)\omega_r - K_r(\omega - \omega_r) - m_a g S(r_{cg}) R(p)^\top a_3 \\ \omega_r &= Z^{-1}(p)\dot{p}_d(t) + Z^{-1}(p)\Lambda_r(p_d(t) - p) \end{aligned} \tag{2.15}$$

$$J(\dot{\omega} - \dot{\omega}_r) - S(J\omega)(\omega - \omega_r) - K_r(\omega - \omega_r) = d_w(t) \tag{2.16}$$

The closed-loop attitude dynamics are given in the Eq. (2.16). The matrices Λ_r and K_r are positive definite and are chosen to achieve required tracking performance. The control laws presented above compute control signals which are at least $\mathcal{C}^2$ continuous and the number of control inputs are collocated with the states. Considering the overactuated design of the simulator and the impulse actuation of the thrusters, a transformation from the continuous control signal to the thruster on-off times is required to achieve equivalent performance with non-smooth control inputs. In the following two sections, we discuss the actuator models for thrusters and reaction wheels to make this transformation, along with the influence matrices due to the location of the actuators.

2.4.5 Thruster Model and Influence Matrix

Influence Matrix.

Eqs. (2.13) and (2.15) give force and torque inputs that need to be applied collocated with the five degrees of freedom of the system. The spacecraft has 16 thrusters mounted in the configuration shown in Fig. 2.7, with thrusters 1-8 used for position and yaw angle control, and 9-16 used for roll and pitch angle control. The collocated force and torque inputs from the control law are transformed to the force input requirements on each of the 16 actuators through control allocation using an influence matrix. For the position controller, the following is the actuator input to control input mapping called the influence matrix.

$$\tau_p = R(p) B_p F_1 \tag{2.17}$$

In the equation 2.17, $R(p)$ transforms the actuator input in the body frame to the inertial frame. B_p corresponds to the influence matrix given by Eq. (2.18) for position control. The force vector, $F_1 = [f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7\ f_8]^\top$, acts as the input to the spacecraft dynamics simulator thrusters mounted for position and yaw control. The actuator numbering is shown in Fig. 2.7.

$$B_p = \begin{bmatrix} -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.18}$$

For attitude control, the thruster force to control input mapping is given as follows.

$$\tau_r = B_r F \quad \text{where } B_r = [B_1\ B_2] \text{ and } F = \begin{bmatrix} F_1^\top & F_2^\top \end{bmatrix}^\top \tag{2.19}$$

where $F_2 = [f_9\ f_{10}\ f_{11}\ f_{12}\ f_{13}\ f_{14}\ f_{15}\ f_{16}]^\top$. Also, see Fig. 2.7 for the thruster numbering and nomenclature of ℓ, b, and h.

$$\begin{aligned} B_1 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\ell & \ell & -b & b & -\ell & \ell & -b & b \end{bmatrix}, \\ B_2 &= \begin{bmatrix} 0 & 0 & h & -h & 0 & 0 & -h & h \\ h & -h & 0 & 0 & -h & h & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{2.20}$$

Control Allocation.

The control allocation scheme for the position controller computes the 8 dimensional thruster forces F_1 given the transformation matrix $R(p)$ and the influence matrix B_p. A generalized right psuedo-inverse solution to the control allocation problem that minimizes the $\mathcal{L}^2$-norm of the control effort is given by $F_1 = B_p^\top (B_p B_p^\top)^{-1} R(p)^{-1} \tau_p$ and weighted pseudo-inverse is given in Eq. (2.21), with a diagonal weighing matrix W. We use this algorithm for real-time implementation.

$$F_1 = W(B_p W)^\top (B_p W (B_p W)^\top)^{-1} R(p)^{-1} \tau_p \tag{2.21}$$

The elements of the diagonal matrix W can be chosen to take into account actuator saturation limits. For example, given the maximum u_{max} and minimum u_{min} thrust that can be produced by the thruster f_1, we choose the corresponding diagonal element in W as $\frac{1}{|u_{\text{max}} - u_{\text{min}}|}$. For attitude control using thrusters and reaction wheels, the same approach can be used for computing the actuator force.

Thruster Firing Time.

The continuous actuator force computed using the control allocation scheme needs to be transformed to the thruster firing times because the thrusters on the spacecraft simulator are on-off non-latching solenoids. The on time of the thrusters is controlled using a PWM signal with the duty cycle mapped to the on time requirements. Consider a PWM signal with frequency f_{pwm} with duty cycle corresponding to firing time Δt, and continuous force F_r that needs to be applied by a thruster at time step t. Let F_a be the force applied by the thruster when open/on and the control loop frequency be f_{cl}. It is assumed that control frequency is same as the PWM signal frequency. The firing time is given in the following equation.

$$\Delta t = \frac{F_r}{f_{\text{cl}} F_a} \tag{2.22}$$

The equation above assumes that the thruster produces the same force for all firing times. To verify this claim and validate the model, an experimental setup was built. Methods and results can be found in Ref. [105]

2.4.6 Reaction Wheel Configuration and Model

The simulator is equipped with four reaction wheels for attitude control arranged in a pyramid configuration (see Fig. 2.7). The angle α made by the axis of the wheel and the (X_b, Y_b) plane is chosen to have maximum momentum storage [90], $\alpha = 35.26°$. The overactuated configuration will be used to study the fault detection, isolation, and recovery of reaction wheels, which is a major source of failure [42] in flight missions. The attitude dynamics with four reaction wheels in the pyramid configuration and no gravity torques is given in Eq. (2.23). The influence matrix is given by G in Eq. (2.24).

$$J\dot{\omega} + \omega \times J\omega = -GJ_w\dot{\Omega} - \omega \times GJ_w\Omega \tag{2.23}$$

$$G = \begin{bmatrix} c(\alpha)c(45°) & -c(\alpha)c(45°) & -c(\alpha)c(45°) & c(\alpha)c(45°) \\ c(\alpha)s(45°) & c(\alpha)s(45°) & -c(\alpha)s(45°) & -c(\alpha)s(45°) \\ s(\alpha) & s(\alpha) & s(\alpha) & s(\alpha) \end{bmatrix} \tag{2.24}$$

$$J_w = \begin{bmatrix} J_{w1} & 0 & 0 & 0 \\ 0 & J_{w2} & 0 & 0 \\ 0 & 0 & J_{w3} & 0 \\ 0 & 0 & 0 & J_{w4} \end{bmatrix} \tag{2.25}$$

In the above equation, J is the mass moment of inertia including the four wheels, J_w is a diagonal matrix with the mass moment of inertia of the wheels about the rotation axis, $\Omega = [\Omega_1\ \Omega_2\ \Omega_3\ \Omega_4]^\top$ is the rotation speed of the each of the four wheels, and $s(\cdot), c(\cdot)$ denote the sine and cosine of a given angle, respectively. For the numbering and location of the wheels with respect to body frame see Fig. 2.7. The term $-GJ_w\dot{\Omega}$ is the control input to the attitude dynamics. The attitude controller in Eq. (2.15), is modified to cancel the cross-coupling term $-\omega \times GJ_w\Omega$ by feeding the wheel speed to the control law. The final control law is given in the Eq. 2.9d.

$$\tau_b = M_b \begin{bmatrix} \dot{\omega}_r \\ \dot{v}_{b_r} \end{bmatrix} + C_b \begin{bmatrix} \omega_r \\ v_{b_r} \end{bmatrix} + H_b - \begin{bmatrix} S(GJ_w\Omega) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_r \\ v_{b_r} \end{bmatrix} - \begin{bmatrix} K_\omega & 0 \\ 0 & K_v \end{bmatrix} \begin{bmatrix} s_\omega \\ s_v \end{bmatrix} \tag{2.26}$$

2.4.7 Hardware Implementation of the Hierarchical Control Law

In this section, we elaborate on the implementation of the hierarchical control law discussed earlier. The schematic of the control law is shown in Fig. 2.8. The attitude control is done in the inner-loop with control frequency between $80 - 100$ Hz using reaction wheels. The thrusters can be used to do coarse attitude control, or desaturate the reaction wheels. The X, Y position controller is done using thrusters, it is coupled with the attitude dynamics by a rotation matrix to map the actuator force in the body frame to the inertial frame. The position dynamics are slow compared to the attitude dynamics, so it is run as an outer-loop with feedback on position data for control computations and attitude data for control allocation at control frequency between $1 - 10$ Hz.

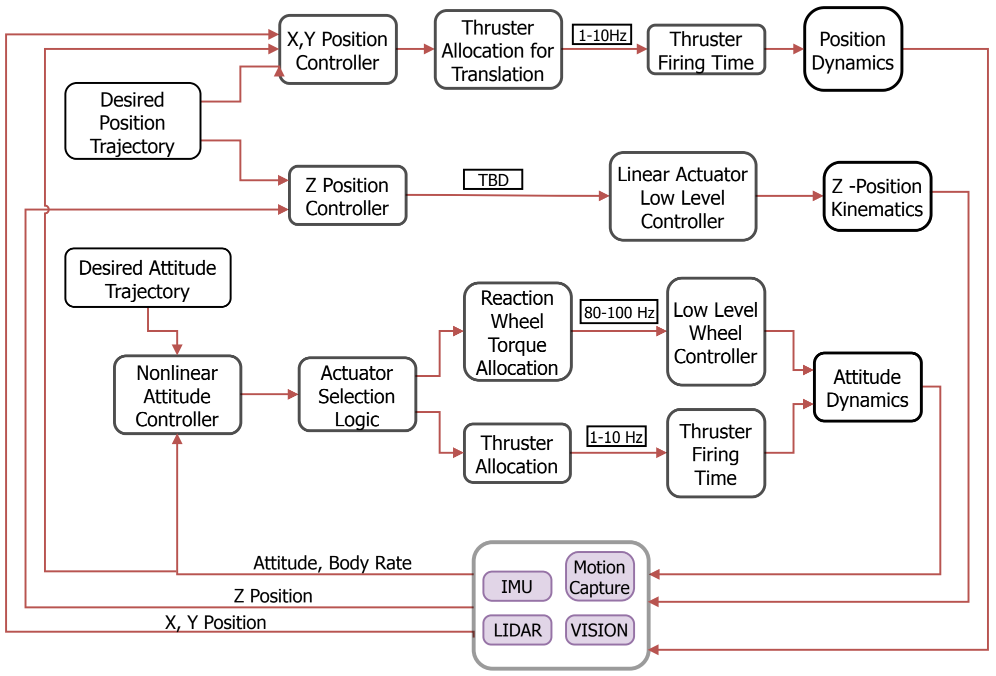


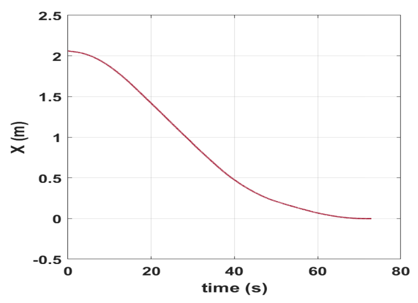
Figure 2.8: Closed-loop control implementation for the 6DOF simulator.

2.5 Experiments

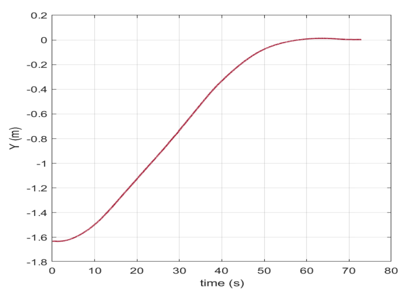
In this section, we present the preliminary experimental results for the position tracking controller discussed earlier. Here, we try to track a step input and demanding harmonic trajectories using the control law, control allocation and firing time schemes developed in the chapter. The position and orientation data of the simulator is measured using the motion capture camera system running at 100 Hz. The thrusters are operated at 50 psi. The tracking results are discussed in the following section.

2.5.1 Results

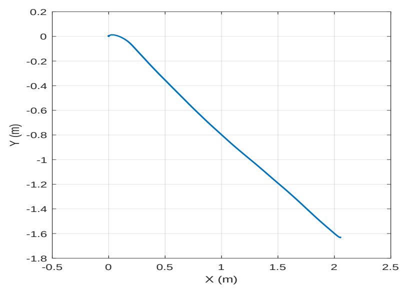
Fig.s 2.9 and 2.10 show preliminary results of waypoint reaching experiments. The task for the controller was to reach origin of the inertial frame and stay there until a further command was communicated. The controller performs well for the two presented cases. The current position controller can be easily extended for tracking a trajectory with coarse way points. The steady-state error in both of the cases was less than the assigned value of 5cm. In this particular test the yaw angle attitude was coarsely maintained around 0, except when the system faced perturbations from uneven flow and varying pressure in the pressure manifold that supplies air to the thrusters, which caused a couple on the simulator due to firing forces that do not balance. Further investigation into characterizing the viscous friction due to air gap between the simulator and the epoxy floor, and the dead zone of the thrusters needs to be done to improve the performance of the controller.



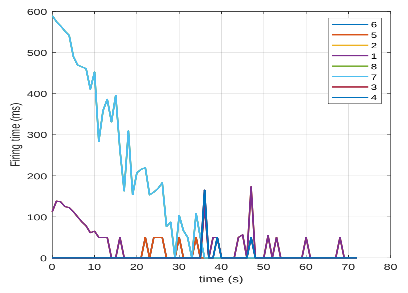
(a) x position (m) vs. time(s)



(b) y position (m) vs. time(s)

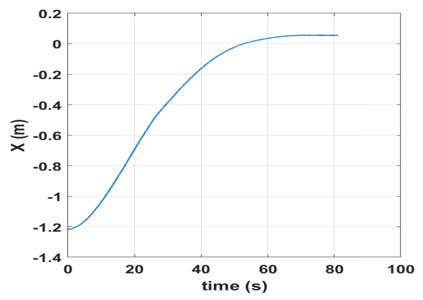


(c) (x,y) trajectory(m)

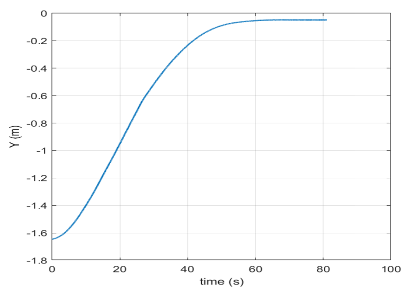


(d) Firing time vs. time

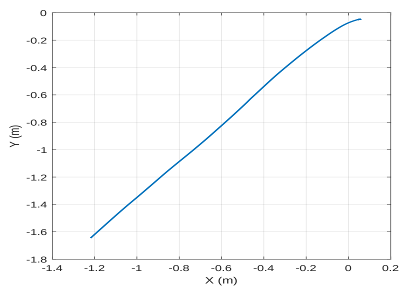
Figure 2.9: Closed-loop waypoint reaching experimental result- test case 1.



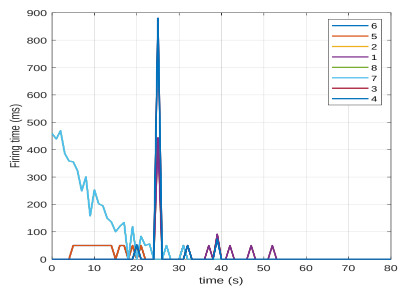
(a) x position (m) vs. time(s)



(b) y position (m) vs. time(s)



(c) (x,y) trajectory in m



(d) Firing time vs. time

Figure 2.10: Closed-loop waypoint reaching experimental result- test case 2.

2.6 Chapter Summary

In this chapter, we discussed the hardware development of a 6-DOF robotic spacecraft simulator M-STAR for testing formation guidance, navigation and control algorithms. The simulator has 6-DOF with translation and attitude stages decoupled using a spherical air bearing. The translation stage floats on the epoxy flat floor using three flat round air bearings. The hardware architecture of M-STAR and its subsystems including mechanical structure, pneumatic system for flat air bearings, spherical air bearing required to achieve frictionless and disturbance torque free motion of the simulator were discussed in detail. The low level control architecture for thrusters and reaction wheels was mentioned for controlling the dynamics.

A nonlinear dynamic model of M-STAR was presented by modelling the system as a 3D pendulum on a floating platform. A hierarchical model-based control law for the nonlinear system was discussed for tracking a given position and attitude trajectory. A generalized pseudo-inverse control allocation scheme, with a thruster actuator model developed using experiments, was used to implement the control law in a ROS based software framework for testing position control.

Chapter 3

Optimal Guidance and Control with Nonlinear Dynamics Constraints

This chapter presents an efficient upgrade to the widely used SCP method which allows the method to be used to more accurately solve optimal control problems with nonlinear dynamics constraints. The new algorithm, SCPn, corrects for the linearization and discretization error in each SCP loop. This prevents the error from building up and allows the algorithm to converge to a solution that satisfies the nonlinear dynamics and the optimization constraints. A new constraint is added to SCPn to ensure that the resulting optimal trajectory is a feasible solution for the original nonconvex problem. Using this constraint, it is proven that SCPn converges to the optimal solution and the solution remains feasible for the nonlinear dynamics. Due to runtime and robustness issues with SCPn, a second algorithm, modified SCPn (M-SCPn) is presented which mixes SCP and SCPn to great effect, achieving solutions that are true to the nonlinear dynamics in shorter times for a wider set of test cases. Both algorithms are tested through simulations involving a quadrotor traversing a simple obstacle field. Preliminary results, reported in [53], are broadened and strengthened with optimality proofs for M-SCPn and extensive simulations across a broad set of parameters chosen to identify weaknesses in the algorithms.

Previous work [97, 100, 101] has used SCP for nonlinear problems ranging from collision-free guidance and control of swarms to optimal robotic motion planning in cluttered environments [17] to in-orbit satellite self-assembly [54, 52]. In order to make the nonconvex optimal guidance and control problem convex, the nonlinear dynamics constraint is linearized and the nonconvex inequality constraints are convexified. The linearization and discretization of the dynamics cause error to accumulate with each successive iteration of SCP. This error causes the linearized/discretized optimal trajectory to diverge from the nonlinear dynamics and can lead to failed constraints, as seen in Fig. 3.1. In the figure, the black line represents the SCP optimal trajectory and the colored circles represent the error between the SCP trajectory and the nonlinear trajectory, which grows over the course of the trajectory. The error is large enough that the terminal constraint is missed by a substantial margin.

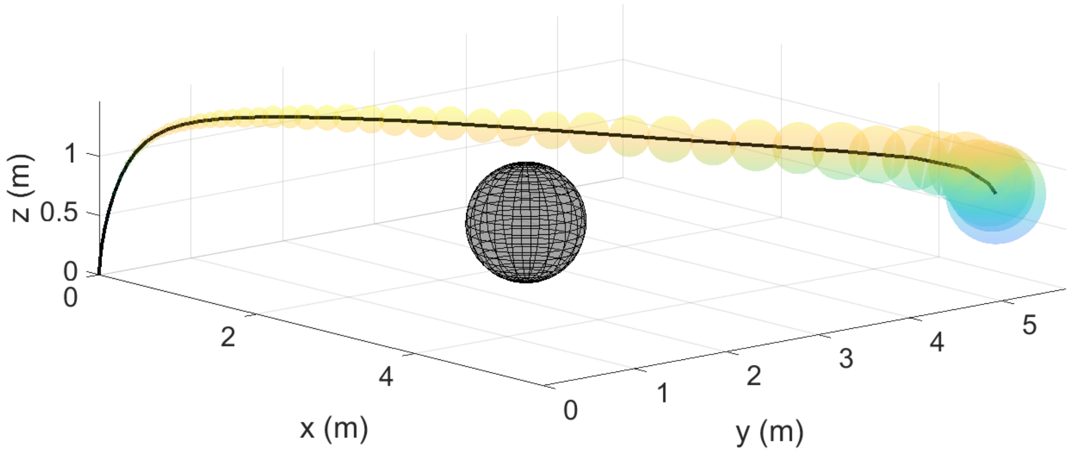


Figure 3.1: Linearization and discretization error accumulates over time when trajectories are found using SCP. Mean trajectory error resulting from SCP is shown by the colored circles along the nonlinear trajectory

3.1 Problem Statement

In this section, the original continuous-time nonlinear optimal control problem is presented.

3.1.1 Nonlinear Optimal Control Problem

We define the original finite-horizon optimal control problem for $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ as follows.

Problem 1 (Constrained, Nonlinear Optimal Control).

$$\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \qquad \int_{t_0}^{t_f} \mathcal{F}(\mathbf{x}(t),\mathbf{u}(t))dt \quad \text{subject to} \tag{3.1}$$

$$\dot{\mathbf{x}}(t) - \mathbf{f}_c(\mathbf{x}(t),\mathbf{u}(t)) = \mathbf{0} \qquad \forall t \in [t_0, t_f] \tag{3.2}$$

$$\tilde{\mathbf{g}}_i(\mathbf{x}(t),\mathbf{u}(t)) \leq \mathbf{0}, \qquad i = 1,\ldots,r, \ \ \forall t \in [t_0, t_f] \tag{3.3}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \qquad \mathbf{x}(t_f) = \mathbf{x}_f, \tag{3.4}$$

where Eqs. (3.2)-(3.3) represent the continuous dynamics constraint ($\dot{\mathbf{x}} = \mathbf{f}_c$) and the general inequality constraints ($\tilde{\mathbf{g}}_i \leq 0$) of the state ($\mathbf{x}$) and the control vector ($\mathbf{u}$), respectively, and Eq. (3.4) represents the initial and terminal constraints. Examples of the cost integrand function include $\mathcal{F}(\mathbf{x}(t),\mathbf{u}(t)) = \|\mathbf{u}(t)\|_q$. Then, the $\mathcal{L}_1$ integration of $\mathcal{F}$ shown in Eq. (3.1) correctly captures the fuel cost of a spacecraft dynamic model with $q \in \{1, 2, \infty\}$.

Example 1 (State and Control Constraint).

If Problem 1 involves multiple (N) agents [97, 98, 101] such that $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$ and $\mathbf{u} = (\mathbf{u}_1, \cdots, \mathbf{u}_N)$, examples of $\tilde{g}_i(\mathbf{x}(t),\mathbf{u}(t)) \leq 0$ in Eq. (3.3) include the following maximum control constraint and collision avoidance constraint:

$$\|\mathbf{u}(t)\|_r \leq U_{\max} \qquad \forall t \in [t_0, t_f], \qquad r \in \{1, 2, \infty\}, \qquad j = 1,\ldots,N \tag{3.5}$$

$$\|G[\mathbf{x}_j(t) - \mathbf{x}_m(t)]\|_2 \geq R_{\text{col}} \qquad \forall t \in [t_0, t_f], \qquad m \in \mathcal{N}_{[j]}, \qquad j = 1,\ldots,N \tag{3.6}$$

$$\mathcal{N}_{[j]} = \{m | \ \|\mathbf{x}_j(t) - \mathbf{x}_m(t)\|_2 \leq R_{\text{comm}}\} \tag{3.7}$$

where $G = [\mathbf{I}_{3\times 3} \quad \mathbf{0}_{3\times 3}]$ is used to take the position state if each $\mathbf{x}_j(t)$ is composed of both the position and velocity states. Also, R_{comm} is the communication radius of each agent, $U_{\max}$ denotes the maximum control magnitude, and R_{col} denotes the minimum allowable distance between two agents. Note that the q and r in the norms $\|\mathbf{u}\|_q$ in Eq. (3.1) and $\|\mathbf{u}\|_r$ in Eq. (3.5) could be different depending on the thruster/actuator architecture. Note that the collision avoidance constraints in Eq. (3.6) can be convexified into a convex polytope around the nominal position of the spacecraft, drawn from the intersection of half-space approximations of Eq. (3.6) [97, 98, 101].

3.2 Convex Optimization with Direct Transcription of Dynamics

To solve Problem 1 efficiently, the state and control constraints Eq. (3.3) are assumed to be convexified and decoupled so that each agent can use SCP to determine its optimal trajectories.

3.2.1 Non-Convex Optimization Problem Using Discretization

The first step in the process of converting Eq. (3.2) into a constraint that can be used in direct optimization is to convert the ordinary differential equation in Eq. (3.2) to a finite number of algebraic constraints by using a zero-order hold discretization approach such that $\mathbf{x}_k = \mathbf{x}(t_k)$, $\mathbf{u}_k = \mathbf{u}(t_k)$ for the $\mathbf{x}(t)$ and $\mathbf{u}(t)$ values at $t \in [t_k, t_{k+1})$, $k = k_0, \ldots, T-1$, where T is the final discrete time step ($t_T = t_f$) and $t_{k_0} = t_0$. The stacked vector is denoted by $\mathbf{x}_{k_0:T} = (\mathbf{x}_{k_0}, \ldots, \mathbf{x}_T)$ and $\mathbf{u}_{k_0:T-1} = (\mathbf{u}_{k_0}, \ldots, \mathbf{u}_{T-1})$. Furthermore, we assume that the integrand cost function $\mathcal{F}$ in Eq. (3.1) is written as

$$\mathcal{F}(\mathbf{x}(t), \mathbf{u}(t)) = \mathcal{F}_x(\mathbf{x}(t)) + \mathcal{F}_u(\mathbf{u}(t)) \tag{3.8}$$

where $\mathcal{F}_x : \mathbb{R}^n \mapsto \mathbb{R}^1$ and $\mathcal{F}_u : \mathbb{R}^m \mapsto \mathbb{R}^1$ is a convex function. First, we consider the problem with $\mathcal{F}_u$ only before generalizing the cost function to Eq. (3.8). The discretized version of Problem 1 is written as the following optimization.

Problem 2 (Non-Convex Program (NCP) with Nonlinear Dynamics).

$$\underset{\mathbf{u}_{k_0:T-1}}{\text{minimize}} \quad \sum_{k=k_0}^{T-1} \mathcal{F}_u(\mathbf{u}_k) v_k \quad \text{subject to} \tag{3.9}$$

$$\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \qquad \mathbf{x}_{k_0} = \mathbf{x}_0, \qquad k = k_0, \ldots, T-1 \tag{3.10}$$

$$g_i(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \qquad i = 1, \ldots, p, \quad k = k_0, \ldots, T \tag{3.11}$$

where $\mathbf{f}$ represents the discretized nonlinear dynamics and v_k denotes the quadrature weight of numerical integration (e.g., $v_k = \Delta t = t_{k+1} - t_k$ for the Euler method or see [62] for pseudospectral integration). Note that $g_i \leq 0$, $i = 1, \ldots, p$ includes all of the convex inequality constraints (Eq. (3.3)) as well as the terminal conditions of (Eq. (3.4)) relaxed as an inequality constraint (e.g., $\|\mathbf{x}_{k=T} - \mathbf{x}_f\| \leq \epsilon$) so the number of inequality constraints $p = r + 1$, where r is the number of inequality constraints in Problem 1. Hence, Problem 2 is a nonconvex optimization problem simply because of Eq. (3.10).

Example 2 (Discretization of Dynamics in Eq. (3.2)).

The discrete-time nonlinear dynamic model (Eq. (3.10)) of the dynamic model of Eq. (3.2) can be derived from the fourth-order Runge-Kutta (RK4) integration scheme as follows

$$\begin{aligned}
\mathbf{y_1}_k &= \mathbf{f}_c(\mathbf{x}_k, \mathbf{u}_k) \\
\mathbf{y_2}_k &= \mathbf{f}_c(\mathbf{x}_k + \tfrac{\Delta t}{2}\mathbf{y_1}_k, \mathbf{u}_{k+\frac{1}{2}}) \\
\mathbf{y_3}_k &= \mathbf{f}_c(\mathbf{x}_k + \tfrac{\Delta t}{2}\mathbf{y_2}_k, \mathbf{u}_{k+\frac{1}{2}}) \\
\mathbf{y_4}_k &= \mathbf{f}_c(\mathbf{x}_k + \Delta t \mathbf{y_3}_k, \mathbf{u}_{k+1}) \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{\Delta t}{6}\left(\mathbf{y_1}_k + 2\mathbf{y_2}_k + 2\mathbf{y_3}_k + \mathbf{y_4}_k\right)
\end{aligned} \tag{3.12}$$

where $\mathbf{u}_{k+\frac{1}{2}} = \frac{\mathbf{u}_k + \mathbf{u}_{k+1}}{2}$. SCPn is independent of the type of numerical integration/discretization method used, for further options see [120, 62, 60].

3.2.2 Sequential Convex Programming with Linearized Constraints

In prior work [97, 98], the dynamics and nonconvex constraints in Problem 2 are sequentially linearized about some nominal trajectory to form a convex program. The basic problem formulation is presented in Problem 3 and the method is presented in Method 1. The first nominal trajectory is an initial guess of the solution, then after the first iteration the solution of the previous iteration of SCP is used for all subsequent nominal trajectories. The nominal trajectory in SCP is only used to linearize the dynamics and other convexified constraints that require linearization. For more details see references [97, 98, 101].

Problem 3 ((w)-th Sequential Convex Program: $\text{SCP}^{(w)}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ given:
$\bar{\mathbf{x}}_{k_0:T} = \mathbf{x}_{k_0:T}^{(w-1)}, \bar{\mathbf{u}}_{k_0:T-1} = \mathbf{u}_{k_0:T-1}^{(w-1)}$).

$$\underset{\mathbf{u}_{k_0:T-1}}{\text{minimize}} \qquad \sum_{k=k_0}^{T-1} \mathcal{F}_u(\mathbf{u}_k) v_k \qquad \text{subject to} \tag{3.13}$$

$$\mathbf{x}_{k+1} - A(\bar{\mathbf{x}}_k)\mathbf{x}_k - B(\bar{\mathbf{u}}_k)\mathbf{u}_k - z(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k) = \mathbf{0}, \qquad \mathbf{x}_{k_0} = \mathbf{x}_0, \qquad k = k_0, \ldots, T-1 \tag{3.14}$$

$$g_i(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \qquad i = 1, \ldots, p, \quad k = k_0, \ldots, T \tag{3.15}$$

where $A(\bar{\mathbf{x}}_{n,k}) = \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, $B(\bar{\mathbf{u}}_k) = \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, and $z(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) = \mathbf{f}(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) - A(\bar{\mathbf{x}}_{n,k})\bar{\mathbf{x}}_{n,k} - B(\bar{\mathbf{u}}_k)\bar{\mathbf{u}}_k$. The Lipschitz and convex functions $g_i(\mathbf{x}_k, \mathbf{u}_k)$, $i = 1, \ldots p$ are from Eq. (3.11).

Remark 1 There are several ways to find a potential initial nominal trajectory, $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$. For many instances, SCP can be initialized with null trajectories. Problems with tougher constraints may have trouble converging with this, in which case a straight line trajectory or other simplified trajectory can be used. For highly restrictive problems, it is helpful to run a simplified version of SCP without some of the constraints for a few iterations.

Method 1 Sequential Convex Programming with Linearized Constraints (SCP Method)

```
1: $\mathbf{x}_k^{(0)}$ := feasible solution to Problem 3, $\mathbf{u}_k^{(0)} := \mathbf{0}$, $\forall k$
2: $\bar{\mathbf{x}}_k := \mathbf{x}_k^{(0)}$, $\bar{\mathbf{u}}_k := \mathbf{u}_k^{(0)}$, $\forall k$
3: $w := 1$
4: while $\|\mathbf{x}_k^{(w)} - \mathbf{x}_k^{(w-1)}\| < \epsilon \ \forall k$ do
5:     $\mathbf{x}_k^{(w)}, \mathbf{u}_k^{(w)}$ :=the solution to Problem 3, SCP$(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ $\forall k$
6:     $\bar{\mathbf{x}}_k := \mathbf{x}_k^{(w)}$, $\bar{\mathbf{u}}_k := \mathbf{u}_k^{(w)}$, $\forall k$
7:     $w := w + 1$
8: end while
9: $(\mathbf{x}_k^{(w-1)}, \mathbf{u}_k^{(w-1)})$ is the approximate solution to Problem 2
```

3.2.3 Sequential Convex Programming with Nonlinear Dynamics Constraints

In contrast with prior work, the contribution of this paper is to show that the numerical integration of nonlinear dynamics equations between each SCP iteration is essential to the optimality of the SCP solutions that use sequential linearizations. The solution $(\mathbf{x}_{k_0:T}, \mathbf{u}_{k_0:T-1})$ of the (w)-th iteration of the following convex programming approximation of the nonconvex program in Problem 2 is denoted as $\mathbf{x}_{k+1} = \mathbf{x}_k^{(w)}$, $\mathbf{u}_k = \mathbf{u}_k^{(w)}$, $k = k_0, \dots, T-1$.

Problem 4 ((w)-th Sequential Convex Program: SCPn$^{(w)}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ given: $\bar{\mathbf{x}}_{n,k_0:T} = \mathbf{x}_{n,k_0:T}^{(w-1)}, \bar{\mathbf{u}}_{k_0:T-1} = \mathbf{u}_{k_0:T-1}^{(w-1)}$).

$$\underset{\mathbf{u}_{k_0:T-1}}{\text{minimize}} \quad \sum_{k=k_0}^{T-1} \mathcal{F}_u(\mathbf{u}_k) v_k \qquad \text{subject to} \tag{3.16}$$

$$\mathbf{x}_{k+1} - A(\bar{\mathbf{x}}_{n,k})\mathbf{x}_k - B(\bar{\mathbf{u}}_k)\mathbf{u}_k - z(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) = \mathbf{0}, \qquad \mathbf{x}_{k_0} = \mathbf{x}_0, \qquad k = k_0, \dots, T-1 \tag{3.17}$$

$$g_i(\mathbf{x}_k, \mathbf{u}_k) + \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k}\|\mathbf{u}_j - \bar{\mathbf{u}}_j\| \le 0, \qquad i = 1, \dots, p, \quad k = k_0, \dots, T \tag{3.18}$$

$$\|\mathbf{u}_k - \bar{\mathbf{u}}_k\| \le (\beta)^{w-1}\mathcal{T}_0, \qquad \forall k = k_0, \dots, T-1 \tag{3.19}$$

where $A(\bar{\mathbf{x}}_{n,k}) = \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, $B(\bar{\mathbf{u}}_k) = \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, $z(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) = \mathbf{f}(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) - A(\bar{\mathbf{x}}_{n,k})\bar{\mathbf{x}}_{n,k} - B(\bar{\mathbf{u}}_k)\bar{\mathbf{u}}_k$, and β is chosen to allow the cost to converge first. The Lipschitz and convex functions $g_i(\mathbf{x}_k, \mathbf{u}_k)$, $i = 1, \dots p$ are from Eq. (3.11), and a positive constant $\mathcal{L}_{i,j,k}$ is defined as:

$$\mathcal{L}_{i,j,k} = 2\|\overline{B}_k\|(\|\overline{A}_k\|)^{k-j-1} \sup_{\substack{\|\mathbf{x}_k - \mathbf{x}_k^{(w)}\| \leq \epsilon_x \\ \|\mathbf{u}_k - \mathbf{u}_k^{(w)}\| \leq \epsilon_u}} \left\| \frac{\partial g_i(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} \right\| \tag{3.20}$$

where $\|\overline{A}_k\| = \sup_{\|\mathbf{x}_k - \mathbf{x}_k^{(w)}\| \leq \epsilon_x} \|A(\mathbf{x}_k)\|$, $\|\overline{B}_k\| = \sup_{\|\mathbf{u}_k - \mathbf{u}_k^{(w)}\| \leq \epsilon_u} \|B(\mathbf{u}_k)\|$ are the local Lipschitz constants of $\mathbf{f}(\mathbf{x}, \mathbf{u})$, $\epsilon_u = \beta^{w-1}\mathcal{T}_0$, the control trust region size and ϵ_x is the state trust region size, defined in Proposition 1. This definition of $\mathcal{L}_{i,j,k}$ is justified in Theorem 2 of Section 3.3. Moreover, the nominal trajectories $\bar{\mathbf{x}}_k = \mathbf{x}_{n,k}^{(w-1)}$ for the current (w)-th SCPn iteration are obtained by integrating the original nonlinear dynamics (Eq. (3.2)) starting from the initial condition $(\mathbf{x}_0)$ using the input trajectory $(\mathbf{u}_k^{(w-1)})$, $\forall k$ of the previous $(w-1)$-th SCPn iteration:

$$\mathbf{x}_{n,k+1}^{(w-1)} = \mathbf{f}(\mathbf{x}_{n,k}^{(w-1)}, \mathbf{u}_k^{(w-1)}), \qquad k = k_0, \ldots, T-1, \qquad \text{and} \qquad \mathbf{x}_{n,k_0}^{(w-1)} = \mathbf{x}_{k_0}^{(w-1)} = \mathbf{x}_0 \tag{3.21}$$

This SCPn optimization along with nonlinear dynamic correction Eq. (3.21) is repeated until the sequence of trajectories converges. The nominal trajectory $\mathbf{x}_{n,k+1}^{(w)}$, $\forall k$ for the $(w+1)$-th SCPn is obtained by integrating $\mathbf{x}_{n,k+1}^{(w)} = \mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})$, $\mathbf{x}_{n,k_0}^{(w)} = \mathbf{x}_{k_0}^{(w)}$ using the (w)-th SCPn solution $\mathbf{u}_{k_0:T-1}^{(w)}$, similar to Eq. (3.21). The convergence of SCPn solutions to an optimal solution is proven in Theorem 2 by showing that the costs in subsequent iterations are non-increasing and convergent.

The SCPn method is described in Method 2. First, an initial approximate trajectory is generated with or without state and control constraints (line 1). Then, the iterative process begins with the agent solving for its optimal state and control trajectories (line 5) and numerically integrating the control trajectory to get the nominal, nonlinear state trajectory for the next iteration (line 7). Finally, iteration is continued until the trajectories converge and satisfy all the constraints (line 4).

Remark 2 The state and control trust region conditions can be added into the convergence criteria in line 4 as well, but the user must be careful to choose the trust region parameters

β and $\mathcal{T}_0$ such that the cost is allowed to converge before the state and control. Allowing the state or control to converge first will result in suboptimal solutions.

Method 2 Sequential Convex Programming with Nonlinear Dynamic Correction (SCPn Method)

```
 1: $\mathbf{x}_k^{(0)}$ := feasible solution to Problem 4, $\mathbf{u}_k^{(0)} := \mathbf{0}$, $\forall k$
 2: $\bar{\mathbf{x}}_k := \mathbf{x}_k^{(0)}$, $\bar{\mathbf{u}}_k := \mathbf{u}_k^{(0)}$, $\forall k$
 3: $w := 1$
 4: while $\|J^{(w)} - J^{(w-1)}\| < \epsilon\ \forall k$, (See Remark 2) do
 5:     $\mathbf{x}_k^{(w)}, \mathbf{u}_k^{(w)}$ :=the solution to Problem 4, SCPn($\bar{\mathbf{x}}, \bar{\mathbf{u}}$) $\forall k$
 6:     $\mathbf{x}_{n,k}^{(w)}$ :=numericalIntegrate($\mathbf{f}_c, \mathbf{x}_0, \mathbf{u}_k^{(w)}$), $\forall k$
 7:     $\bar{\mathbf{x}}_k := \mathbf{x}_{n,k}^{(w)}, \bar{\mathbf{u}}_k := \mathbf{u}_k^{(w)}$, $\forall k$
 8:     $w := w + 1$
 9: end while
10: ($\mathbf{x}_{n,k}^{(w-1)}, \mathbf{u}_k^{(w-1)}$) is the approximate solution to Problem 2
```

3.3 Convergence and Optimality of SCPn

In this section, we will show that SCPn (Method 2) converges to a point, which satisfies the Karush-Kuhn-Tucker (KKT) conditions for optimality of the nonconvex optimization in Problem 2. First, we will show that SCPn converges as the number of iterations tends to infinity ($w \to \infty$).

Proposition 1 (Equivalence and Convergence of SCPn Trust Regions). *The error at the k-th time corrected by nonlinear integration (Eq.* (3.21)*) for the* $(w+1)$*-th SCPn is given as* $\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_k^{(w+1)}$*, while the accuracy of the linearized dynamics Eq.* (3.17) *for the* $(w+1)$*-th SCPn is* $\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}$*. Since the trust region* $\|\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}\| \leq \beta^w \mathcal{T}_0,\ \forall k$ *Eq.* (3.19) *is shrinking exponentially fast as* $w \to \infty$*, the following holds for* $k = k_0, \dots, T$*,*

$$\lim_{w\to\infty} \|\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| = 0, \qquad \lim_{w\to\infty} \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| = 0, \qquad \lim_{w\to\infty} \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_k^{(w+1)}\| = 0. \tag{3.22}$$

and at any given iteration, the size of the trust region around the state is related to the size of the trust region around the control as follows:

$$\epsilon_x = \beta^w T_0 \sum_{j=k_0}^{k-1} (\|\overline{A}_k\|)^{k-j-1} \|\overline{B}_k\| \tag{3.23}$$

Proof. Eq. (3.17) for the $(w+1)$-th SCPn becomes

$$\mathbf{x}_{k+1}^{(w+1)} - \mathbf{x}_{n,k+1}^{(w)} = A(\mathbf{x}_{n,k}^{(w)})(\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}) + B(\mathbf{u}_k^{(w)})(\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}) \tag{3.24}$$

whose fixed initial condition $\mathbf{x}_{k_0}^{(w+1)} = \mathbf{x}_{n,k_0}^{(w)}$ leads to

$$\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)} = \sum_{j=k_0}^{k-2} \left(\prod_{i=j+1}^{k-1} A(\mathbf{x}_{n,k+j-i}^{(w)}) \right) B(\mathbf{u}_j^{(w)})(\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}) + B(\mathbf{u}_{k-1}^{(w)})(\mathbf{u}_{k-1}^{(w+1)} - \mathbf{u}_{k-1}^{(w)}) \tag{3.25}$$

Hence, due to $\|\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}\| \to 0$, $\forall k$ as $w \to \infty$, $\|\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| \to 0$, $\forall k$ is proven.

The error corrected by nonlinear integration at the $(w+1)$-th SCPn and $(k+1)$-th time is given as

$$\begin{aligned} \mathbf{x}_{n,k+1}^{(w+1)} - \mathbf{x}_{k+1}^{(w+1)} &= \mathbf{x}_{n,k+1}^{(w+1)} - \mathbf{x}_{n,k+1}^{(w)} - \left(\mathbf{x}_{k+1}^{(w+1)} - \mathbf{x}_{n,k+1}^{(w)}\right) \\ &= \mathbf{f}(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)}) - \mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)}) - \left(\mathbf{x}_{k+1}^{(w+1)} - \mathbf{x}_{n,k+1}^{(w)}\right) \end{aligned} \tag{3.26}$$

If $\mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})$ has bounded, continuous partial derivatives in the convex domain (Eqs. (3.18)-(3.19)), $\mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})$ is Lipschitz [78]:

$$\begin{aligned} \|\mathbf{x}_{n,k+1}^{(w+1)} - \mathbf{x}_{n,k+1}^{(w)}\| &= \|\mathbf{f}(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)}) - \mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})\| \\ &\le \|\overline{A}_k\| \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| + \|\overline{B}_k\| \|\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}\| \end{aligned} \tag{3.27}$$

where $\|\overline{A}_k\| = \sup_{\|\mathbf{x}_k - \mathbf{x}_k^{(w)}\| \le \epsilon_x} \|A(\mathbf{x}_k)\|$, $\|\overline{B}_k\| = \sup_{\|\mathbf{u}_k - \mathbf{u}_k^{(w)}\| \le \epsilon_u} \|B(\mathbf{u}_k)\|$.

This can be expressed as a function of $\|\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}\|$ using $\mathbf{x}_{n,k_0}^{(w+1)} = \mathbf{x}_{n,k_0}^{(w)}$ as follows

$$\|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| \leq \sum_{j=k_0}^{k-1} (\|\overline{A}_k\|)^{k-j-1} \|\overline{B}_k\| \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{3.28}$$

Hence, $\lim_{w\to\infty} \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| = 0$ as $\lim_{w\to\infty} \|\mathbf{u}_k^{(w+1)} - \mathbf{u}_k^{(w)}\| = 0$. Also, the first equality of Eq. (3.26) verifies $\|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_k^{(w+1)}\| \leq \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| + \|\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\|$. Therefore, $\lim_{w\to\infty} \|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_k^{(w+1)}\| = 0$ holds due to the limit results mentioned immediately below Eqs. (3.25) and (3.28).

Combining the control trust region Eq. (3.19) with Eq. (3.28) establishes the state trust region:

$$\|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| \leq \beta^w T_0 \sum_{j=k_0}^{k-1} (\|\overline{A}_k\|)^{k-j-1} \|\overline{B}_k\| \tag{3.29}$$

and thus the size of the state trust region is given by:

$$\epsilon_x = \beta^w T_0 \sum_{j=k_0}^{k-1} (\|\overline{A}_k\|)^{k-j-1} \|\overline{B}_k\| \tag{3.30}$$

□

Definition 1 ($\mathcal{OS}$ and $\mathcal{FS}$ of SCPn$^{(w)}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ and NCP). We define SCPn$^{(w)}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ as the (w)-th Problem 4 where the nominal trajectories $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{u}}_k$ are used in Eqs. (3.17), (3.18), and (3.19). An optimal solution and a feasible solution to SCPn$(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ are denoted by $\mathcal{OS}$(SCPn$(\bar{\mathbf{x}}, \bar{\mathbf{u}})$) and $\mathcal{FS}$(SCPn$(\bar{\mathbf{x}}, \bar{\mathbf{u}})$), respectively. For example, the (w)-th and the $(w+1)$-th SCPn optimal solutions of Problem 4 yield

$$\begin{aligned} (\mathbf{x}_{k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}) &= \mathcal{OS}(\text{SCPn}^{(w)}(\mathbf{x}_{n,k_0:T}^{(w-1)}, \mathbf{u}_{k_0:T-1}^{(w-1)})) \\ (\mathbf{x}_{k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}) &= \mathcal{OS}(\text{SCPn}^{(w+1)}(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})) \end{aligned} \tag{3.31}$$

Similarly, $\mathcal{OS}$(NCP) and $\mathcal{FS}$(NCP) denote an optimal solution and a feasible solution of the nonconvex program in Problem 2, respectively.

As described in Method 2, the optimal solution of $\text{SCPn}^{(w)}(\mathbf{x}_{n,k_0:T}^{(w-1)}, \mathbf{u}_{k_0:T-1}^{(w-1)})$ is numerically integrated to obtain $\mathbf{x}_{n,k_0:T}^{(w)}$. This nonlinear trajectory is then used as the nominal trajectory for the next SCPn iteration, $\text{SCPn}^{(w+1)}(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ and can be shown to be a feasible solution to the nonconvex problem.

Proposition 2 *A nominal trajectory, $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$, which is a feasible solution to the NCP (Problem 2), is also a feasible solution to the $(w+1)$-th iteration of SCPn.*

Proof. It follows that $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ is a feasible solution to $\text{SCPn}^{(w+1)}(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ because substituting $(\mathbf{x}_k = \mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k = \mathbf{u}_k^{(w)})$ into Eqs. (3.17) and (3.18) for the $(w+1)$-th SCPn straightforwardly shows that Eqs. (3.17) and (3.18) reduce to the corresponding constraints in Problem 2: Eqs. (3.10) and (3.11). Also, the trust region condition, Eq. (3.19) automatically holds. $\square$

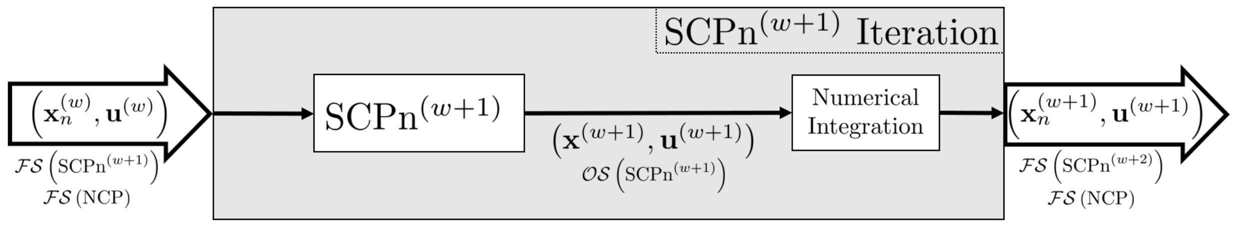
In summary, starting from some feasible solution $\mathcal{FS}(\text{NCP})$, we establish

$$(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}) = \mathcal{FS}(\text{NCP}) \tag{3.32}$$

$$(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}) = \mathcal{FS}(\text{SCPn}^{(w+1)}(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})) \tag{3.33}$$

$$(\mathbf{x}_{k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}) = \mathcal{OS}(\text{SCPn}^{(w+1)}(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})) \tag{3.34}$$

These relationships can also be seen in Fig. 3.2.



$\text{SCPn}^{(w+1)}$ Iteration
$\left(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)}\right)$
$\mathcal{FS}\left(\text{SCPn}^{(w+1)}\right)$
$\mathcal{FS}(\text{NCP})$
$\text{SCPn}^{(w+1)}$
$\left(\mathbf{x}^{(w+1)}, \mathbf{u}^{(w+1)}\right)$
$\mathcal{OS}\left(\text{SCPn}^{(w+1)}\right)$
Numerical Integration
$\left(\mathbf{x}_n^{(w+1)}, \mathbf{u}^{(w+1)}\right)$
$\mathcal{FS}\left(\text{SCPn}^{(w+2)}\right)$
$\mathcal{FS}(\text{NCP})$

Figure 3.2: Relationship between SCPn components

Note that $\lim_{w\to\infty} \|\mathbf{u}_k - \mathbf{u}_k^{(w-1)}\| = 0$ from Eq. (3.19) and Proposition 1 implies that the inequality constraint function $g_i(\mathbf{x}_k, \mathbf{u}_k) + \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k}\|\mathbf{u}_j - \mathbf{u}_j^{(w-1)}\|$ in Eq. (3.18) increases as w increases, thereby expanding the size of the feasible region that tends toward that of

Eq. (3.11). The following theorem shows the cost further decreases through a sequence of convex optimization and nonlinear integration when the inequality constraint Eq. (3.11) of Problems 2 and 4 is restricted to functions of $\mathbf{u}_k$ only, before the main result, Theorem 2 is presented.

Theorem 1 (Decreasing Cost over Optimal SCPn Sequence with Restricted Constraints). *If Eq.* (3.11) *is replaced by*

$$g_i(\mathbf{u}_k) \leq 0, \quad i = 1, \ldots, p, \quad k = k_0, \ldots, T, \tag{3.35}$$

then Problems 2 and 4 are referred to as R-NCP and R-SCPn, respectively. Also, it follows from Eq. (3.20) *that* $\mathcal{L}_{i,j,k} = 0$.

If there exists a feasible solution to the restricted nonconvex problem such that $(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1}) = \mathcal{FS}(\text{R-NCP})$*, then*

$$J(\mathbf{u}^{(w+1)}_{k_0:T-1}) \leq J(\mathbf{u}^{(w)}_{k_0:T-1}) \tag{3.36}$$

where $J(\mathbf{x}, \mathbf{u})$ *is the cost function (Eq.* (3.16)*) of Problems 2 and 4.*

Proof. If the state and control trajectory at some w is a feasible solution to the restricted nonconvex problem, expressed as $(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1}) = \mathcal{FS}(\text{R-NCP})$, then the solution must satisfy the constraints of the restricted NCP, Eqs. (3.17) and (3.35). This feasible solution to R-NCP is used as a nominal trajectory for the $(w+1)$-th R-SCPn to yield an optimal solution: $(\mathbf{x}^{(w+1)}_{k_0:T}, \mathbf{u}^{(w+1)}_{k_0:T-1}) = \mathcal{OS}(\text{SCPn}^{(w+1)}(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1}))$.
Consequently, applying Proposition 2 and $J(\mathcal{OS}(\text{SCPn}^{(w+1)})) \leq J(\mathcal{FS}(\text{SCPn}^{(w+1)}))$, $\forall w$ to Eqs. (3.33) and (3.34) results in

$$J(\mathbf{u}^{(w+1)}_{k_0:T-1}) \leq J(\mathbf{u}^{(w)}_{k_0:T-1}) \tag{3.37}$$

where strict inequality is used unless $(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1}) = \mathcal{OS}\left(\text{SCPn}(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1})\right)$. Since the nonlinear integration does not change the input trajectory, the above relationship holds across SCPn iterations. Furthermore, this optimal input solution ($\mathbf{u}_{k_0:T-1}^{(w+1)}$) of SCPn$^{(w+1)}$ Eq. (3.32) is used to determine the nonlinear trajectory Eq. (3.21) from the initial condition $\mathbf{x}_{n,k_0}^{(w+1)} = \mathbf{x}_0$, thereby yielding the new nominal trajectory ($\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}$) as seen in Fig. 3.2. We show herein that this solution is a feasible solution to R-NCP. ($\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}$) already satisfies the nonlinear dynamics constraint Eq. (3.17) and because the restricted inequality constraints are independent of the state, satisfying Eq. (3.35) is the same for R-NCP and R-SCPn. Therefore, the constraints are satisfied and the solution is feasible to the restricted nonconvex problem. We conclude that given that ($\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}$) is a feasible solution to the R-NCP, when R-SCPn is applied, ($\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}$) is also a feasible solution to the R-NCP. $\square$

Next, we show that the same results can be obtained for the unrestricted problems (Problems 2 and 4) provided that a modification is made to the inequality constraint Eq. (3.11), as given in Eq. (3.18).

Theorem 2 (Decreasing Cost over Optimal SCPn Sequence). *If there exists a feasible solution to the original nonconvex problem (Problem 2) such that* ($\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}$) $= \mathcal{FS}(\text{NCP})$, *$\exists w$, then*

$$J(\mathbf{u}_{k_0:T-1}^{(w+1)}) \leq J(\mathbf{u}_{k_0:T-1}^{(w)}) \tag{3.38}$$

(where $J(\mathbf{u})$ is the cost function of Problems 2 and 4 given in Eq. (3.16)*) under the following condition for each $g_i(\mathbf{x}_k, \mathbf{u}_k)$ in Eq.* (3.18)

$$\mathcal{L}_{i,j,k} = 2\|\overline{B}_k\|(\|\overline{A}_k\|)^{k-j-1} \sup_{\substack{\|\mathbf{x}_k - \mathbf{x}_k^{(w)}\| \leq \epsilon_x \\ \|\mathbf{u}_k - \mathbf{u}_k^{(w)}\| \leq \epsilon_u}} \left\| \frac{\partial g_i(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} \right\| \tag{3.39}$$

where $\|\overline{A}_k\| = \sup_{\|\mathbf{x}_k - \mathbf{x}_k^{(w)}\| \leq \epsilon_x} \|A(\mathbf{x}_k)\|$, $\|\overline{B}_k\| = \sup_{\|\mathbf{u}_k - \mathbf{u}_k^{(w)}\| \leq \epsilon_u} \|B(\mathbf{u}_k)\|$.

The resulting nominal trajectory ($\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}$) *obtained from SCPn is also a feasible solution to the nonconvex problem.*

Proof. This theorem starts with some value of w in which there is a feasible solution to the original nonconvex problem (Problem 2) such that $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)}) = \mathcal{FS}(\text{NCP})$. This means that $\mathbf{x}_{n,k+1}^{(w)} = \mathbf{f}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})$ from Eq. (3.10) and $\mathbf{g}(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)}) \leq \mathbf{0}$ Eq. (3.11) hold for all appropriate values of k. Also, by Proposition 2, $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ is also a feasible solution of SCPn$^{(w+1)}$.

Therefore, applying $J(\mathcal{OS}(\text{SCPn}^{(w+1)})) \leq J(\mathcal{FS}(\text{SCPn}^{(w+1)}))$, $\forall w$ to Eq. (3.33) and Eq. (3.34) results in

$$J(\mathbf{u}_{k_0:T-1}^{(w+1)}) \leq J(\mathbf{u}_{k_0:T-1}^{(w)}) \tag{3.40}$$

Furthermore, this optimal input solution $(\mathbf{u}_{k_0:T-1}^{(w+1)})$ of SCPn$^{(w+1)}$ Eq. (3.32) is used to integrate $\mathbf{x}_{n,k+1}^{(w+1)} = \mathbf{f}(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)})$, $k = k_0, \ldots, T-1$ Eq. (3.21) from the initial condition $\mathbf{x}_{n,k_0}^{(w+1)} = \mathbf{x}_0$, thereby yielding the new nominal trajectory $(\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)})$. We show herein that this solution is a feasible solution to Problem 2 if Eq. (3.39) holds.

First, the convex inequality constraint Eq. (3.18) of SCPn$^{(w+1)}$ is given as

$$g_i(\mathbf{x}_k^{(w+1)}, \mathbf{u}_k^{(w+1)}) + \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \leq 0, \qquad i = 1, \ldots, p, \quad k = k_0, \ldots, T \tag{3.41}$$

The first-order condition of a convex function holds for each g_i as follows

$$g_i(\mathbf{x}_k^{(w+1)}, \mathbf{u}_k^{(w+1)}) \geq g_i(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)}) + \left.\frac{\partial g_i}{\partial \mathbf{x}_k}\right|_{(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)})} (\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w+1)}) \tag{3.42}$$

$$+ \left.\frac{\partial g_i}{\partial \mathbf{u}_k}\right|_{(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)})} (\mathbf{0}) \tag{3.43}$$

Combining Eqs. (3.41) and (3.43) results in

$$g_i(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)}) \leq \left.\frac{\partial g_i}{\partial \mathbf{x}_k}\right|_{(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)})} (\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_k^{(w+1)}) - \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{3.44}$$

In order to show $(\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)}) = \mathcal{FS}(\text{NCP})$, we need to prove $g_i(\mathbf{x}_{n,k}^{(w+1)}, \mathbf{u}_k^{(w+1)}) \leq \mathbf{0}$, whose sufficient condition can be given as

$$\left\|\frac{\partial g_i}{\partial \mathbf{x}_k}\right\| (\|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\| + \|\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\|) \leq \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{3.45}$$

Applying the property of submultiplicativity of norms to Eq. (3.25) shows that both $\|\mathbf{x}_{n,k}^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\|$ and $\|\mathbf{x}_k^{(w+1)} - \mathbf{x}_{n,k}^{(w)}\|$ possess the same upper-bound given in Eq. (3.28). Hence, substituting Eq. (3.28) into Eq. (3.45) shows that Eq. (3.45) is satisfied by

$$\sum_{j=k_0}^{k-1} 2 \left\|\frac{\partial g_i}{\partial \mathbf{x}_k}\right\| (\|\overline{A}_k\|)^{k-j-1} \|\overline{B}_k\| \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| = \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{3.46}$$

Consequently, the condition of $\mathcal{L}_{i,j,k}$ Eq. (3.39) is established. If the global Lipschitz constants are used in Eq. (3.46) instead of the local Lipschitz constants, the bound is overly conservative in certain dynamics due to the large value of the global state Lipschitz constant ($\|\overline{A}\|$), which restricts the feasible set to the point where solutions are not possible or quite difficult to find.

Since $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ already satisfies the nonlinear dynamics constraint Eq. (3.10), we conclude that given that $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ is a feasible solution to the NCP, when SCPn is applied, $(\mathbf{x}_{n,k_0:T}^{(w+1)}, \mathbf{u}_{k_0:T-1}^{(w+1)})$ is also a feasible solution to the NCP as can be seen in Fig. 3.2.

$\square$

This theorem shows that a sequence of SCPn optimal solutions $(\mathbf{x}_{k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ and feasible solutions $(\mathbf{x}_{n,k_0:T}^{(w)}, \mathbf{u}_{k_0:T-1}^{(w)})$ to NCP (Problem 2) has a nonincreasing cost. We will now prove that a sequence of optimal solutions exists and converges to an optimal solution (KKT point) of Problem 2.

Theorem 3 (Convergence of SCPn to KKT Point). *If* $(\mathbf{x}^{(w)}, \mathbf{u}^{(w)})$ *is a feasible solution to Problem 2 for some* $w = w_0$*, then a sequence of optimal solutions* $(\{\mathbf{x}^{(w)}\}, \{\mathbf{u}^{(w)}\})$ *exists. If each optimal solution is unique, the sequence converges to* $(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$*, which is a KKT point of Problem 2.*

Proof. Since Eqs. (3.10)-(3.11) form a closed and bounded set and the feasible solutions in the sequence $(\mathbf{x}^{(w)}_{n,k_0:T}, \mathbf{u}^{(w)}_{k_0:T-1})$ satisfy the equations, there exists an infinite subsequence $\left(\{\mathbf{x}^{(w_i)}\}, \{\mathbf{u}^{(w_i)}\}\right)$ that converges, where w_i is the i-th iteration and $(\mathbf{x}^{(w_i)}, \mathbf{u}^{(w_i)})$ is the i-th feasible solution at iteration w_i in the convergent subsequence. Let the convergence point be called $(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$. The Weierstrass theorem [21] establishes that a continuous function over a closed and bounded set achieves a minimum and a maximum on that set, so $J(\mathbf{u}^{(\infty)})$ exists in the set of feasible solutions. By completeness, this gives the sequence $J(\{\mathbf{u}^{(w_i)}\}) \to J(\mathbf{u}^{(\infty)})$.

The mapping $M(\mathbf{x})$ is equivalent to solving the KKT conditions of SCPn$(\mathbf{x}, \mathbf{u})$, which are continuous with respect to $\mathbf{x}$. Therefore, the mapping M is continuous. Since the subsequence $(\{\mathbf{x}^{(w_i)}\}, \{\mathbf{u}^{(w_i)}\}) \to (\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$ and the limit of a continuous function of a convergent sequence is the function of the limit of that sequence, the following is true:

$$\{M(\mathbf{x}^{(w_i)})\} \to M(\mathbf{x}^{(\infty)}) \tag{3.47}$$

Additionally, $\mathbf{x}^{(\infty)}$ is a fixed point and $\mathbf{x}^{(w_i+1)} = M(\mathbf{x}^{(w_i)})$. Therefore,

$$\{\mathbf{x}^{(w_i+1)}\} \to \mathbf{x}^{(\infty)} \tag{3.48}$$

Finally, we will show that $(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$, the fixed point that minimizes $J(u^{(w)})$ is a KKT point of Problem 2. From Proposition 1, we know that after convergence, $(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)}) = (\mathbf{x}^{(\infty)}_n, \mathbf{u}^{(\infty)})$. Since $\mathbf{x}^{(\infty)}$ is a fixed point of M, it is a solution to SCPn$(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$ and from Proposition 2, it is a feasible solution to Problem 2. Additionally, Problem 4 is convex so any solution to this problem is a KKT point $(\mathbf{x}^{(\infty)}, \mathbf{u}^{(\infty)})$=SCPn$(\mathbf{x}^{(\infty)}_n, \mathbf{u}^{(\infty)})$ and satisfies stationarity (Eq. (3.49)), complementary slackness (Eq. (3.50)), and dual feasibility

(Eq. (3.51)):

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix} = \begin{bmatrix}\sum_{i=1}^{p}\lambda_{k,i}^{(\infty)}\nabla_{\mathbf{x}_k} g_i(\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)})\\ \sum_{i=1}^{p}\lambda_{k,i}^{(\infty)}\nabla_{\mathbf{u}_k} g_i(\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)})\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{x}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)})\\ \nabla_{\mathbf{u}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)})\end{bmatrix}\boldsymbol{\mu}_k^{(\infty)} + \tag{3.49}$$

$$\begin{bmatrix}\nabla_{\mathbf{x}_k} J(\mathbf{u}^{(\infty)})\\ \nabla_{\mathbf{u}_k} J(\mathbf{u}^{(\infty)})\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{x}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_k^{(\infty)},\mathbf{x}_{k-1}^{(\infty)},\mathbf{u}_{k-1}^{(\infty)})\\ \mathbf{0}\end{bmatrix}\boldsymbol{\mu}_{k-1}^{(\infty)}, \quad k = k_0,\dots,T-1$$

$$0 = \lambda_{k,i}^{(\infty)} g_i(\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)}), \qquad i = 1,\dots,p, \quad k = k_0,\dots,T \tag{3.50}$$

$$\lambda_{k,i}^{(\infty)} \geq 0, \qquad i = 1,\dots,p, \quad k = k_0,\dots,T \tag{3.51}$$

where $\mathbf{h}_{\text{dyn}}$ is the dynamics equality constraint and $\sum_{j=k_0}^{k-1}\mathcal{L}_{i,j,k}\|\mathbf{u}_j - \mathbf{u}_j^{(w-1)}\| \to 0$ is used due to the exponentially shrinking trust region Eq. (3.19). Also, we can find from Eq. (3.17)

$$\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(w)},\mathbf{x}_k^{(w)},\mathbf{u}_k^{(w)}) = A(\mathbf{x}_{n,k}^{(w-1)})\mathbf{x}_k^{(w)} + B(\mathbf{u}_k^{(w-1)})\mathbf{u}_k^{(w)} + z(\mathbf{x}_{n,k}^{(w-1)},\mathbf{u}_k^{(w-1)}) - \mathbf{x}_{k+1}^{(w)} \tag{3.52}$$

$$\nabla_{\mathbf{x}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)}) = A(\mathbf{x}_{n,k}^{(\infty)}), \qquad \nabla_{\mathbf{x}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_k^{(\infty)},\mathbf{x}_{k-1}^{(\infty)},\mathbf{u}_{k-1}^{(\infty)}) = -\mathbf{I}, \tag{3.53}$$

$$\nabla_{\mathbf{u}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)}) = B(\mathbf{u}_{n,k}^{(\infty)})$$

Proposition 2 and Theorem 2 indicate that the original cost function Eq. (3.9) of Problem 2 is expressed as $J(\mathbf{u})$ for the fixed KKT point $(\mathbf{x}_n^{(\infty)},\mathbf{u}^{(\infty)}) = \mathcal{FS}(\text{SCPn}(\mathbf{x}_n^{(\infty)},\mathbf{u}^{(\infty)})) = \mathcal{FS}(\text{NCP})$. Now let $\boldsymbol{\lambda}_k^* = \boldsymbol{\lambda}_k^{(\infty)}$, $\boldsymbol{\nu}_k^* = \boldsymbol{\nu}_k^{(\infty)}$, $\boldsymbol{\mu}_{\mathbf{x}_k}^* = \boldsymbol{\mu}_{\mathbf{x}_k}^{(\infty)}$, and $\boldsymbol{\mu}_{\mathbf{u}_k}^* = \boldsymbol{\mu}_{\mathbf{u}_k}^{(\infty)}$, where the Lagrange multipliers with superscript ∞ are the KKT multipliers that satisfy the KKT conditions for the convex program. Then, we can show that Eqs. (3.49)-(3.51) become the KKT conditions for the nonconvex program (Problem 2) because $\nabla_{\mathbf{x}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)}) = A(\mathbf{x}_{n,k}^{(\infty)}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{n,k}^{(\infty)}}$ and $\nabla_{\mathbf{u}_k}\mathbf{h}_{\text{dyn}}(\mathbf{x}_{k+1}^{(\infty)},\mathbf{x}_k^{(\infty)},\mathbf{u}_k^{(\infty)}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{u}_k^{(\infty)}}$, $k = k_0,\dots,T-1$. Hence, all the KKT conditions Eqs. (3.49)-(3.51) remain the same for Problem 2 after substitution.

□

Now, we can relax the cost function to include convex functions of the state trajectory. In the following proposition, it is shown that cost function for each iteration can be expressed solely in terms of the control input trajectory for that iteration.

Proposition 3 (Convexity of Cost Function of Input). *The cost function Eq.* (3.16) *can be written as a convex function of only* $\mathbf{u}_{k_0:T-1}^{(w)} = (\mathbf{u}_{k_0}^{(w)}; \cdots; \mathbf{u}_{T-1}^{(w)})$ *such that*

$$J(\mathbf{u}_{k_0:T-1}^{(w)}) = \sum_{k=k_0}^{T-1} \left(\mathcal{F}_x(\mathbf{u}_k^{(w)}) + \mathcal{F}_u(\mathbf{u}_k^{(w)}) \right) v_k \tag{3.54}$$

Proof. Nesting Eq. (3.17) in on itself for $k = k_0, \ldots, T$ results in:

$$\mathbf{x}_{k+1}^{(w)} = \left(\prod_{j=k_0}^{k} A(\mathbf{x}_{n,j}^{(w-1)}) \right) \mathbf{x}_{k_0}^{(w)} + \sum_{j=k_0}^{k-1} \left(\prod_{i=j+1}^{k} A(\mathbf{x}_{n,i}^{(w-1)}) \right) \left(B(\mathbf{u}_j^{(w-1)})\mathbf{u}_j^{(w)} + z(\mathbf{x}_{n,j}^{(w-1)}, \mathbf{u}_j^{(w-1)}) \right) \tag{3.55}$$

$$+ B(\mathbf{u}_k^{(w-1)})\mathbf{u}_k^{(w)} + z(\mathbf{x}_{n,k}^{(w-1)}, \mathbf{u}_k^{(w-1)})$$

$$= \left(\prod_{j=k_0}^{k} A(\mathbf{x}_{n,j}^{(w-1)}) \right) \mathbf{x}_{k_0}^{(w)} + \tag{3.56}$$

$$\sum_{j=k_0}^{k-1} \left(\prod_{i=j+1}^{k} A(\mathbf{x}_{n,i}^{(w-1)}) \right) \left(B(\mathbf{u}_j^{(w-1)})(\mathbf{u}_j^{(w)} - \mathbf{u}_j^{(w-1)}) + \mathbf{x}_{n,j+1}^{(w-1)} - A(\mathbf{x}_{n,j}^{(w-1)})\mathbf{x}_{n,j}^{(w-1)} \right)$$

$$+ B(\mathbf{u}_k^{(w-1)})(\mathbf{u}_k^{(w)} - \mathbf{u}_k^{(w-1)}) + \mathbf{x}_{n,k+1}^{(w-1)} - A(\mathbf{x}_{n,k}^{(w-1)})\mathbf{x}_{n,k}^{(w-1)}$$

$$= \mathbf{b}_k^{(w)}(\mathbf{x}_{n,k_0:k}^{(w-1)}, \mathbf{u}_{k_0:k}^{(w-1)}) + \left[\mathbf{C}_k^{(w)}(\mathbf{x}_{n,k_0:k}^{(w-1)}, \mathbf{u}_{k_0:k}^{(w-1)}) \right] \mathbf{u}_{k_0:k}^{(w)} \tag{3.57}$$

where the initial condition $\mathbf{x}_{k_0}^{(w)}$ is fixed as given in Eq. (3.4). The variables $\mathbf{b}_k^{(w)}$ and $\mathbf{C}_k^{(w)}$ are functions of solutions from previous iterations, therefore $\mathbf{x}_k^{(w)}$ is an affine function of $\mathbf{u}_k^{(w)}$. Since $\mathcal{F}_x(\mathbf{x}_k^{(w)})$ in Eq. (3.16) is a convex function, a convex function of an affine function is convex, so the entire cost function can be expressed as a convex function of $\mathbf{u}_k^{(w)}$.

Since we have already shown that the cost as a function of only the control input will converge, and that the difference between $\mathbf{x}_{n,k}^{(w)}$ and $\mathbf{x}_k^{(w)}$ will converge to zero, we can show that the cost as a function of $\mathbf{x}_{n,k}^{(w)}$ will converge. This does not mean that $J(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)})$ will be lower than $J(\mathbf{x}^{(w)}, \mathbf{u}^{(w)})$ for all w because the nonlinear dynamics correction can perturb the cost in either direction. □

To use convex programming to solve the trajectory optimization (Problems 2 and 4), the nonconvex constraints must be converted to convex or affine constraints.

Example 3 (Convexification of Collision Avoidance Constraint). *[97, 98]*

One standard definition of a collision avoidance constraint is to avoid a ball around the obstacle or other agent. This is defined as:

$$-\|\mathbf{p}_{j,k} - \bar{\mathbf{p}}_{i,k}\|_2 \leq R_{\text{col}} \qquad k = k_0, \ldots, T \tag{3.58}$$

where $\mathbf{p}_{j,k}$ is the agent's position at time k and $\bar{\mathbf{p}}_{i,k}$ is the nominal position of the other agent and obstacle for each agent and obstacle i. This constraint is nonconvex. The best convex approximation is an affine constraint, where the 3-D spherical prohibited region ($\mathbb{S}^2$) is replaced by a plane which is tangent to the sphere and perpendicular to the line segment connecting the nominal position of the other agent or obstacle.

$$-(\bar{\mathbf{p}}_{j,k} - \bar{\mathbf{p}}_{i,k})^T(\mathbf{p}_{j,k} - \bar{\mathbf{p}}_{i,k}) \leq R_{\text{col}}\|(\bar{\mathbf{p}}_{j,k} - \bar{\mathbf{p}}_{i,k})\|_2 \qquad k = k_0, \ldots, T \tag{3.59}$$

This hyperplane rotates as the agent moves which helps approximate the agent and obstacle more precisely and helps to prevent the workspace from becoming overly restricted.

Corollary 3.1 If $(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)})$ is a feasible solution to the the nonconvex problem (Problem 2), then $(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)})$ is a feasible solution to SCPn$^{(w+1)}$ even if the nonconvex problem contains nonconvex inequality constraints, $\mathbf{g}_{NC}(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)}) \leq 0$, that are convexified using a shifting hyperplane in SCPn as shown in Example 3. The hyperplane is defined as:

$$-\mathbf{a}_k^T\mathbf{x}_k \leq b_k \tag{3.60}$$

where $\mathbf{a}_k$ and b_k are chosen at each time step such that the nonconvex constraint region is tangent to the hyperplane without intersecting the hyperplane at any other points.

Proof. From Proposition 2, we know that $(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)}) = \mathcal{FS}(\text{SCPn}^{(w+1)})$ for problems with convex constraints. Relaxing this to include nonconvex inequality constraints convexified using successive hyperplanes as in [97] involves establishing the equivalency between satisfying the nonconvex constraint and the hyperplane-convexified constraint. By the construction of the hyperplane, it encompasses the full nonconvex constraint region and thus satisfying the hyperplane constraint must satisfy the nonconvex constraint.

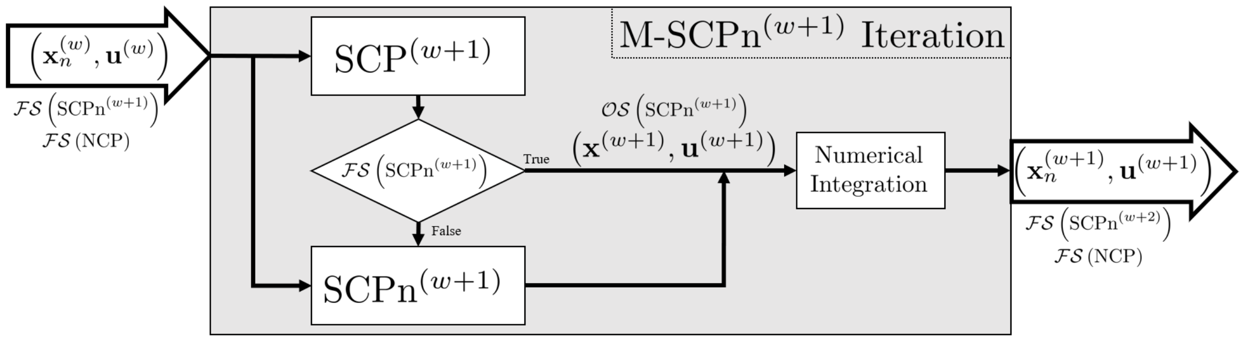
□

Remark 3 The optimization problems stated thus far have relied on a fixed terminal time but this is not required. If the terminal time is an optimization variable, even linear dynamics become nonlinear. Defining $\tau = \frac{t}{t_f}$ for $\tau \in [0, 1]$ and augmenting the state with t_f so that $\tilde{\mathbf{x}} = [x, t_f]$, the redefined linear system dynamics with respect to τ become the following:

$$\frac{\partial \mathbf{x}}{\partial \tau} = (A\mathbf{x} + B\mathbf{u})t_f \tag{3.61}$$
$$\frac{\partial t_f}{\partial \tau} = 0 \tag{3.62}$$

which is nonlinear in $\tilde{x}$. Since SCPn handles nonlinear dynamics, this broadens the applications of SCPn.

The addition of the barrier $\sum_{j=k_0}^{k-1} \mathcal{L}_{i,j,k} \|\mathbf{u}_j - \bar{\mathbf{u}}_j\|$ around the inequality constraints (Eq. (3.18)) is imposed to ensure conformity to the nonlinear dynamics, but it reduces the feasible set such that the number of iterations required to find a convergent solution is significantly higher than the previous SCP implementation when used with tough convex or convexified constraints like Example 3. To reduce the computational burden, a combined approach (Method 3) was formulated wherein the original SCP method is used until the solutions violate the constraints of the nonlinear problem, then the SCPn method is used. This method is only advisable when the inequality constraints are lenient enough that the added $\mathcal{L}_{i,j,k}$ term is not needed for several iterations in the trajectory generation process. The relationship between the components of the M-SCPn method can be seen in Fig. 3.3.



$\left(\mathbf{x}_n^{(w)}, \mathbf{u}^{(w)}\right)$
$\mathcal{FS}\left(\text{SCPn}^{(w+1)}\right)$
$\mathcal{FS}\,(\text{NCP})$
M-SCPn$^{(w+1)}$ Iteration
SCP$^{(w+1)}$
$\mathcal{FS}\left(\text{SCPn}^{(w+1)}\right)$
True
False
$\mathcal{OS}\left(\text{SCPn}^{(w+1)}\right)$
$\left(\mathbf{x}^{(w+1)}, \mathbf{u}^{(w+1)}\right)$
Numerical Integration
SCPn$^{(w+1)}$
$\left(\mathbf{x}_n^{(w+1)}, \mathbf{u}^{(w+1)}\right)$
$\mathcal{FS}\left(\text{SCPn}^{(w+2)}\right)$
$\mathcal{FS}\,(\text{NCP})$

Figure 3.3: Relationship between M-SCPn components

Method 3 Modified Sequential Convex Programming with Nonlinear Dynamic Correction (M-SCPn Method)

```
1: $\mathbf{x}_k^{(0)}$ := the solution to Problem 3 $\forall k$
2: $\bar{\mathbf{x}}_k := \mathbf{x}_k^{(0)}, \ \forall k$
3: $w := 1$
4: while $\|J^{(w)} - J^{(w-1)}\| < \epsilon \ \forall k$ do
5:     $\mathbf{x}_k^{(w)}, \mathbf{u}_k^{(w)}$ :=the solution to Problem 3 (Sequential Convex Program), $\forall k$
6:     if Eq. (3.18) does not hold then
7:         $\mathbf{x}_k^{(w)}, \mathbf{u}_k^{(w)}$ :=the solution to Problem 4 (Nonlinear Corrected Sequential Convex Program), $\forall k$
8:     end if
9:     $\mathbf{x}_{n,k}^{(w)}$ :=numericalIntegrate($\mathbf{f}_c, \mathbf{x}_0, \mathbf{u}_k^{(w)}$), $\forall k$
10:    $\bar{\mathbf{x}}_k := \mathbf{x}_{n,k}^{(w)}, \ \forall k$
11:    $w := w + 1$
12: end while
13: $\mathbf{x}_{n,k}^{(w-1)}$ is the approximate solution to Problem 2
```

Next, we will prove the stability of the combined method.

Corollary 3.2 (Convergence of M-SCPn to a KKT Point). If $(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)})$ is a feasible solution to Problem 2 for some w, then the M-SCPn Method (Method 3) will converge to a steady-state solution $(\mathbf{x}_{n,k}^{\infty}, \mathbf{u}_k^{\infty})$ which is feasible to the nonconvex problem and is a KKT point of Problem 2.

Proof. There are two possible ways the M-SCPn method can execute. Either SCP without dynamics integration returns a solution that is feasible to SCPn, or SCPn with dynamics integration executes and returns the solution. In either case, the resulting solution is feasible to SCPn, i.e. $(\mathbf{x}_k^{(w)}, \mathbf{u}_k^{(w)}) = \mathcal{FS}(\text{SCPn}^{(w+1)})$ then numerically integrated. From Theorem 2 we see that a solution of this sort must also be feasible to the nonconvex problem, Problem 2, so $(\mathbf{x}_{n,k}^{(w)}, \mathbf{u}_k^{(w)}) = \mathcal{FS}(\text{NCP})$. From Theorem 3 we know that the feasible set of the NCP is closed and bounded and thus the solutions of M-SCPn form a sequence which converges to a point $(\mathbf{x}_k^{(\infty)}, \mathbf{u}_k^{(\infty)})$. From the construction of the algorithm, we know that any solution $(\mathbf{x}_n^w, \mathbf{u}^w)$ is feasible to SCPn and so is also feasible to SCP since the feasible set of SCPn is a subset of the feasible set of SCP. To prove that the cost converges, we must first show that it decreases over M-SCPn iterations for both of the execution methods, SCP and SCPn. Assume the contrary, that the cost does not decrease over M-SCPn, $\neg J(\mathbf{u}^{(w+1)}) \leq J(\mathbf{u}^{(w)})$. This implies the following:

$$J\left(\mathcal{OS}(\text{SCP}^{(w+1)})\right) > J\left(\mathcal{FS}(\text{SCP}^{(w+1)})\right) \tag{3.63}$$

This is logically inconsistent since the cost of an optimal solution must be lower than or equal to that of a feasible solution. The same conclusion holds when SCPn runs, and thus the cost is nonincreasing over M-SCPn iterations.

Since the cost $J(\mathbf{u}^{(w)})$ is continuous and decreasing over a closed and bounded set, it converges by the Weierstrass theorem [21] to $J(\mathbf{u}^{(\infty)})$ as established in Theorem 2. The optimization problem solved in M-SCPn is either the same as SCPn or enforced to satisfy the constraints of SCPn, the KKT conditions remain the same as stated in Theorem 3. The conclusion in Theorem 3 holds for M-SCPn, considering that the state and cost converge as shown above. $\square$

3.4 Simulation Results

Simulations were performed to validate the SCPn and M-SCPn methods compared to SCP and the true nonlinear trajectory generated from the control trajectory for all three methods. All simulations were performed using quadrotor dynamics as described in [101]. The quadrotor was given the initial position $0, 0, 0$ m and the terminal position $5, 5, 1$ m with a 1-m diameter sphere in the middle of the straight-line path to engage the collision avoidance constraint described Example 3. Simulations used CVX, a MATLAB-based convex optimization engine running the Mosek solver [66, 64, 103]. The nonlinear trajectories were found by numerically integrating the nonlinear dynamics, as seen in Eq. (3.12), using the optimal control trajectory found using SCP, SCPn, and M-SCPn. To test the performance of the methods, the convergence tolerance and the initial nominal trajectories were varied. The convergence tolerance specifies at what point the method is considered converged based on the difference between successive state and control trajectories. This value was varied from 1e-2 to 1e-6. The initial nominal trajectories were found by running SCP without collision avoidance constraints for a set number of iterations. The number of iterations tested were $(5, 7, 10, 12, 15)$, where more iterations corresponds to a better initial guess. The algorithms were cut off at a maximum of 20 iterations even if unconverged to keep the batch runtime tractable.

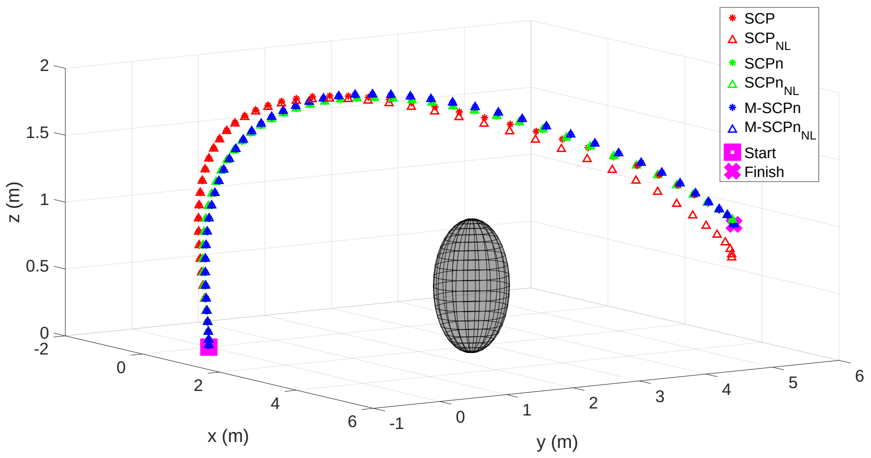


Figure 3.4: Quadrotor trajectories computed using SCP (red), SCPn (blue), and M-SCPn (black) in squares compared against the nonlinear trajectory using the respective control trajectory in triangles. The SCP trajectory clearly does not follow the nonlinear dynamics, but the SCPn and M-SCPn trajectories do.

Trajectories from one of the batch simulations are shown in Fig. 3.4. The SCP trajectory, shown in red, diverges from the nonlinear dynamics over the course of the trajectory to such an extent that the terminal constraint is not satisfied in the nonlinear dynamics. The SCPn and M-SCPn trajectories, shown in blue and black respectively, are perfectly matched to their nonlinear trajectories. A summary of the batch simulation results is shown in Figs. 3.5 and 3.6. Fig. 3.5 shows the error in the terminal position between the algorithm state trajectory and the nonlinear trajectory for all of the simulations. For SCP, the deviation from the nonlinear dynamics is independent of the convergence tolerance but varies drastically with the initial nominal trajectory. Poor initial trajectories result in lower error because fewer iterations means less accumulation of linearization and discretization error. For all SCP cases, the error in terminal position is unacceptably high and nowhere near the required tolerances.

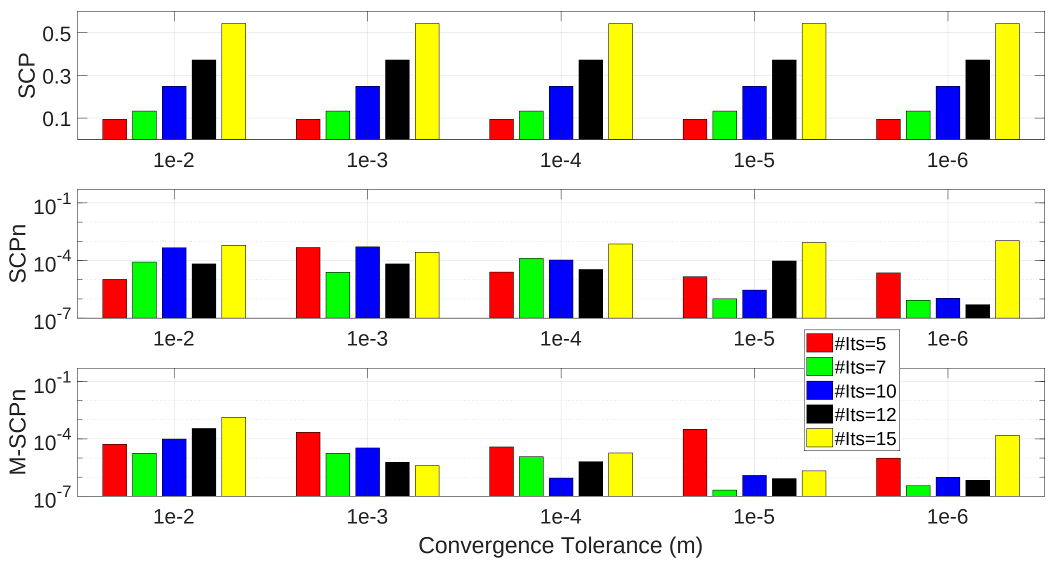


Figure 3.5: Terminal position error for each algorithm over the five different tolerances and five initial nominal trajectories, generated using [#Its] number of simplified SCP iterations.

In general the performance of SCPn and M-SCPn is comparable with millimeter-level tolerances and reasonably well-chosen initial nominal trajectories. The interesting differences come into play in the stress cases. When initialized with a poor trajectory and a stringent convergence tolerance, SCPn is much more likely to fail to converge than M-SCPn. Overall SCPn fails to sufficiently converge by the 20 iteration cutoff in 8 of the 25 test cases. M-SCPn fails just three of the cases.

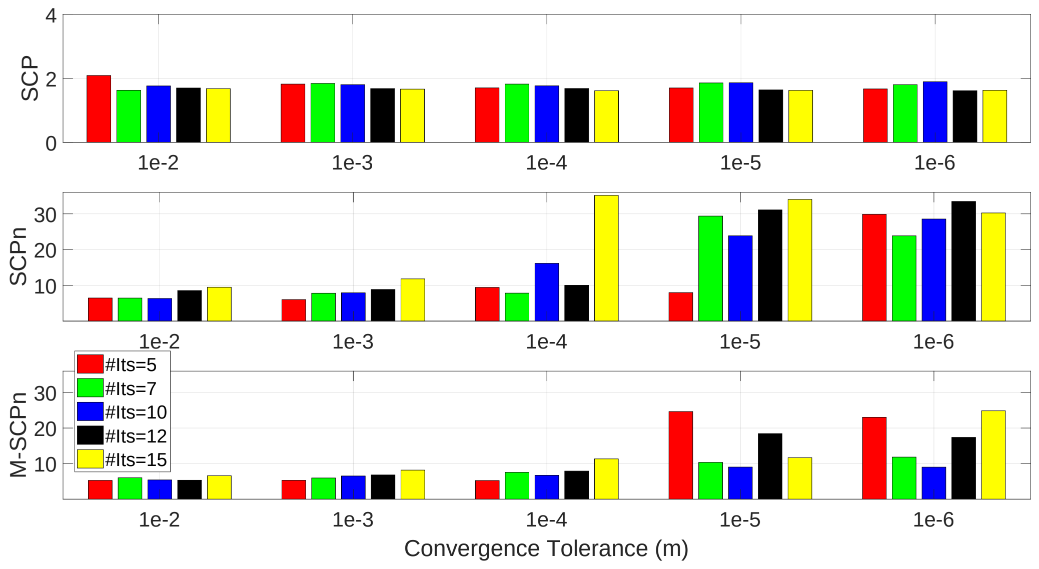


Figure 3.6: Algorithm runtimes (s) for each algorithm over the five different tolerances and five initial nominal trajectories, generated using [#Its] number of simplified SCP iterations.

Fig. 3.6 shows the runtimes for each of the simulations. In all cases, M-SCPn has a faster runtime than SCPn though it is still not as fast as SCP. From the runtimes it is clear that SCPn is very sensitive to the convergence tolerance as the runtime is much longer for all cases with sub-millimeter tolerances. For M-SCPn, the only test cases with excessively long runtimes were the tough cases, tight tolerances or poor nominal trajectories. The runtimes shown in the figure are too long to be implementable onboard most robots, but when these algorithms are implemented in C++ the runtimes reduce sufficiently to allow for onboard use.

To test these algorithms more thoroughly, more obstacles were added to the simulations. In this simulation, three spherical obstacles are placed along the quadrotor's desired path sequentially such that each sphere must be individually avoided. Results are shown in Fig. 3.7. None of the trajectories intersect with the obstacles, but the SCP nonlinear trajectory and the SCPn trajectories do not reach the terminal point. The additional constraints restrict the feasible set of SCPn too much to find an adequate solution. M-SCPn is the only successful algorithm, avoiding the obstacles while following the nonlinear dynamics and reaching the final point.

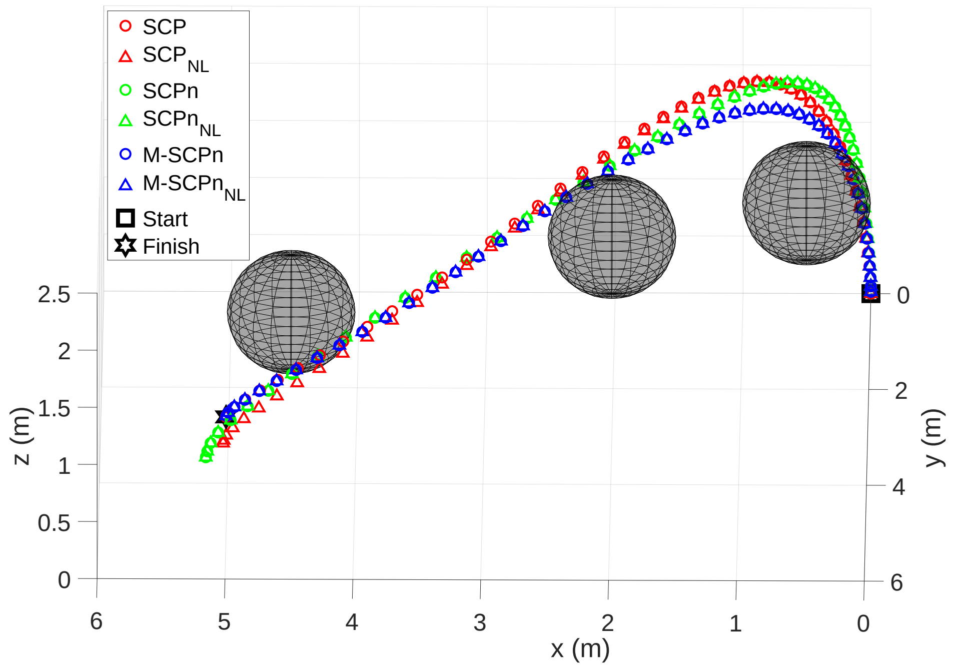


Figure 3.7: Simulations using three obstacles in SCP (red), SCPn (green), and M-SCPn (blue) in Xs compared against the nonlinear trajectory using the respective control trajectory in triangles.

M-SCPn has improved performance in every respect over SCPn, with faster and better convergence for a wider range of initializations. This is because the addition of the boundary around the inequality constraints in SCPn restricts the feasible set causing the algorithm to require more iterations to converge since the optimizations fail more frequently due to the shrinking trust region around the control restricting the feasible set even further. Since M-SCPn only optimizes using the boundary when needed, it is able to find more solutions faster.

3.5 Chapter Summary

In this chapter, two methods were presented which are capable of extending sequential convex programming for use with nonlinear dynamics. Previous implementations suffered from deviations from the true nonlinear dynamics due to the sequential linearizations and discretizations. The proposed methods, SCPn and M-SCPn, numerically integrate the control trajectory resulting from the optimization to obtain a corrected nominal trajectory, which the next iteration of the optimization then linearized the dynamics around. This way, the linearization and discretization errors do not compound in each successive iteration, keeping the obtained trajectory close to the nonlinear dynamic trajectory. To ensure the corrected solution stays feasible to the nonconvex optimization problem, an additional bound around the inequality constraints is added to the SCPn optimization. This bound allows the formation of theoretical guarantees. Both methods are shown to converge to a solution with decreasing cost, and to satisfy the KKT conditions of the original nonconvex optimization problem. These claims can also be extended to problems with hyperplane-convexified inequality constraints and problems with costs that are convex functions of both state and control. In addition to theoretical guarantees, extensive simulations were performed to establish the efficacy and robustness of the two algorithms as compared to the standard, uncorrected SCP algorithm. Simulations were performed using highly nonlinear quadrotor dynamics with a convexified collision avoidance constraint. These results show that both SCPn and M-SCPn solve the terminal constraint failure of SCP and yield trajectories which adhere to the nonlinear dynamics, but suffer a performance loss when compared to SCP due to the extra computation needed to enforce the nonlinear dynamics. M-SCPn has improved performance over SCPn in terms of computation time, convergence quality, and robustness to initialization quality.

Chapter 4

In-Orbit Self-Assembly of a Heterogeneous Swarm

This chapter will go over the development of the Swarm Orbital Construction Algorithm (SOCA). The result of this chapter is to present a coherent, robust, and correct algorithm for in-orbit construction using a decentralized algorithm to guide and control a heterogeneous, docking swarm of satellites. The algorithm presented in this dissertation is suitable for limited type heterogeneity in the swarm and allows for docking satellites while avoiding undesired collisions. The algorithm takes in a shape without pre-assigned target positions and solves the optimal assignment and collision-free trajectory generation together. The assignment is performed using a distributed auction with a variable number of targets in case of agent loss, and strict bonding rules to address the heterogeneity. MPC-SCPn is used to generate the collision-free trajectories, with modifications to relax collision constraints on agents targeting neighboring positions to allow the agents to dock before reaching the target.

4.1 Heterogeneous Target Assignment

4.1.1 Heterogeneous Docking Components

When designing the construction swarm, a lot of thought was put into the geometry of the agents and the resulting final constructions possible using those geometries. The desire for flexibility in final configuration led to the examination of Islamic tile art and the geometry of crystals for inspiration. Islamic art like that shown in Fig. 4.1 relies strongly on geometric shapes and symmetry, creating complicated and elegant periodic or aperiodic patterns [8, 11]. These patterns can be helpful in the investigation of large space structures for several reasons. Mainly, the vast array of complicated geometric patterns can be used as models for layouts of potential space structures because they are sufficiently complex to suit a variety of mission types and the basic geometric shapes can be constructed using standard satellite buses. The focus on symmetry is beneficial for space structures as well because symmetric structures are more likely to be controllable. By inspection of the rigid body Euler equations, it is clear that the attitude dynamics of asymmetric satellites are significantly more entwined than axisymmetric satellites, and in the proposed scheme it is highly possible that the center of gravity of an asymmetric assembly is outside the assembly, therefore the control points available may not be able to stabilize the assembly. When evaluating potential large space structures, controllability is a grave concern.

Islamic tile artists created quasi- and aperiodic tiling patterns centuries before western mathematicians like Roger Penrose formalized them [16]. Penrose tilings, or 5-way symmetric tilings do not have the translational symmetry of most tile patterns which makes them aperiodic [85, 114]. This means the tilings are not simple tessellations, but are complex and non-repeating patterns. Unfortunately these fascinating Penrose tile designs require non-convex geometries which are less desirable for satellite structural designs. Tessellation, which makes use of translational symmetry, is also helpful in increasing the packing efficiency of the rocket, components that tessellate will pack more densely into the rocket, enabling more agents to be brought to orbit in each launch. Crystals found in nature typically have a periodic geometric layout much like the above mentioned art. For example, the mineral beryl typically has a hexagonal prism shape in macro-scale. It also has a hexagonal void created at the nano-scale, created by the ring of silicon-oxygen bonds [46].

The chosen hexagonal prism connectors with rectangular prism rods can be combined to create a complex planar structure with a minimum degree of heterogeneity. Hexagonal and rectangular agent types can be used to create the following shapes: parallelogram, equilateral triangle, hexagon, any combination thereof, as shown in Fig. 4.2. Any one of these shapes can tessellate to cover the whole plane, but it is also possible to combine the shapes to cover

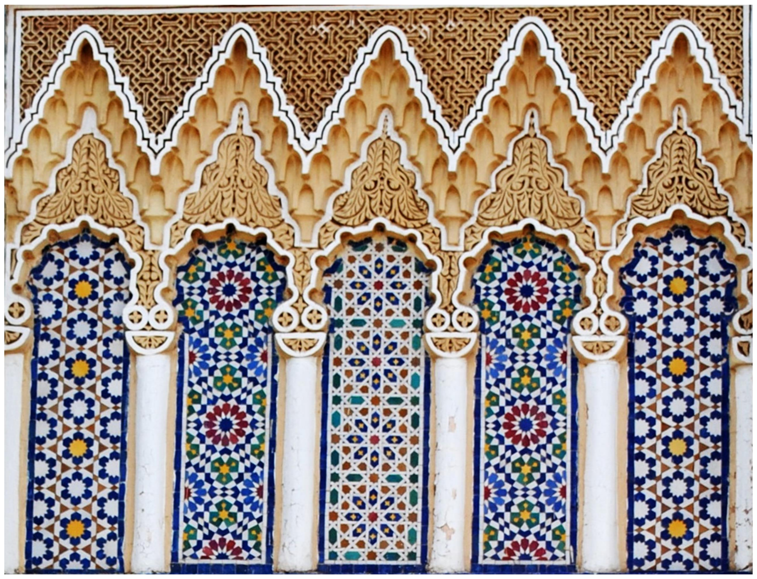


Figure 4.1: The Royal Palace Gates in Fes, Morocco served as inspiration for this work, along with the geometry in crystals found in nature

the plane in more useful ways tailored to particular missions. These shapes can be combined using semiregular or uniform tiling in methods similar to that of Islamic tile art and the work of M.C. Escher, where fixed geometric shapes are tessellated and decorated to create art [41]. The patterns and shapes can all be made at any scale, by adding more rectangular rod agents between the hexagonal connector agents.

The concept of tessellation and tiling is much more complicated in three dimensions, with only three regular geometries capable of filling 3-space, the cube, tetrahedron and octahedron [140]. Instead of complete 3-space tessellation, several planes of different tilings can be connected by out of plane agents to create 3D configurations. When designing these configurations, it would also be beneficial to examine space grid structures, an architectural feature akin to complicated trusses where patterns of struts combine in three dimensions to act as a single unit [31]. The ability of this system to achieve any three dimensional configurations relies heavily on the docking orientations allowed by the chosen docking port. In this chapter we will assume the docking system allows docking at a set of relative angles, 0 and 90°. This makes the generation of 3D shapes possible, though limiting the out of plane dock angles limits the possible final shapes.

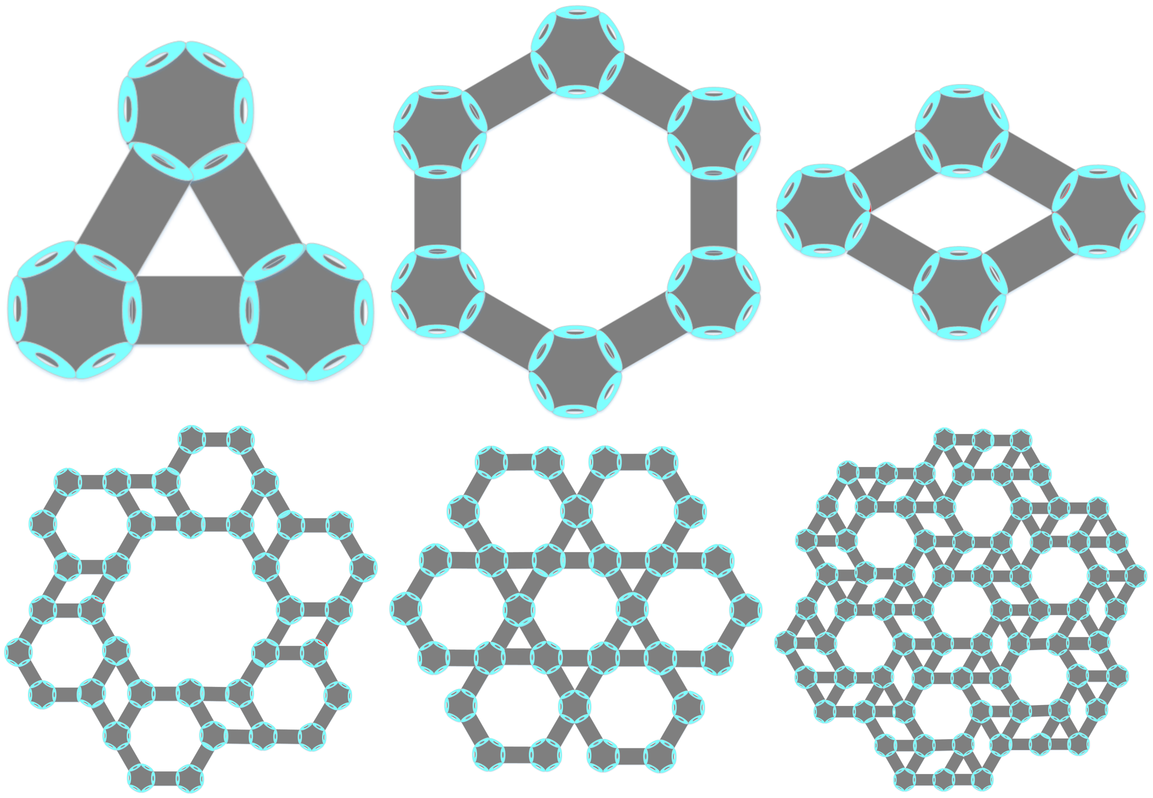
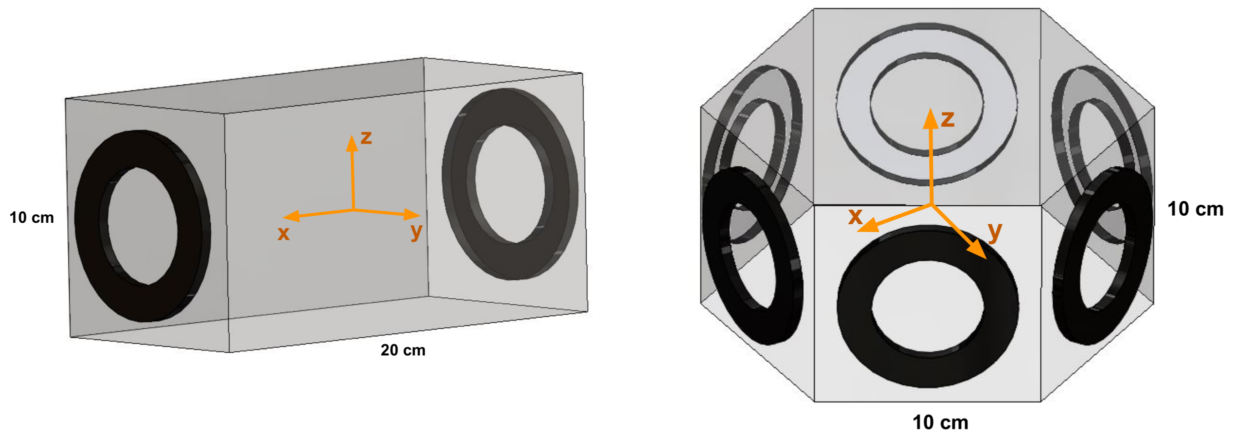


Figure 4.2: Examples of potential shapes and tiling patterns possible with the proposed mission, using only rectangular rod and hexagonal connector agents

4.1.2 Shape Parameters

The rod agent is a rectangular prism with two docking ports located on the ends. The connector agent is a regular hexagonal prism with six docking ports along the sides. In order to execute this algorithm while incorporating attitude control, assumptions about mass properties and systems engineering configurations for the agent types must be made. The mass and volume advantages of the swarm will be most effective if the agents are kept small, in the nanosatellite class. The rod agent can use a standard 2U CubeSat bus, with a mass of 2.6 kg and principal inertia parameters $(44.5, 111.3, 111.3) kg-cm^2$. Each side of the regular hexagonal prism must be the same as the face of a CubeSat to allow docking. The connector agent has a mass of 4 kg and principal inertia parameters $(116.7, 116.7, 166.7) kg-cm^2$. The body frame definitions and the docking port locations for the two agent types are shown in Fig. 4.3.



(a) Rod Agent

(b) Connector Agent

Figure 4.3: Definitions of the Rod and Connector agent types, with docking ports shown in red

The docking ports will combine features of existing magnetic docking ports [104, 2]. The docking ports are assumed to be hermaphroditic, electromagnet-based with a rigidizing component so the electromagnets can be turned off.

4.1.3 Assignment with Conflict Resolution for Heterogeneous Agents

The target assignment algorithm, called Distributed Auction Algorithm for Docking (DAA-D), assumes each agent in the swarm knows the location of all the possible targets. Since the algorithm is decentralized and distributed, all bidding information is communicated only to agents within the communication radius. Each agent calculates its cost to each target using some cost function like the cost of a minimum-fuel optimal trajectory or the distance between the two points. The agent then uses this cost to bid for target locations with the agents within its communication network. The auction is designed to allow all bids to propagate completely through the communication network. As the agents move to the targets, the communication graph becomes connected which ensures that over time the optimal assignment will be reached [101].

DAA-D has to be designed to ensure that each agent type is assigned to an appropriate location. For the current agent definitions, the agent types have different number and location of docking ports but the same radius, as seen in Fig. 4.3. This means that potential target locations can be differentiated by the number and the angle between docks required at each location. Algorithmically, this involves changing the cost function used in the auction to make improper assignments prohibitively expensive. This is achieved through the use of barrier functions. The target information known by every agent must now indicate the location of the target and how many docks must be performed at that location.

A barrier function can be used to prevent agents from successfully bidding on targeting locations they cannot accommodate. An example barrier function is:

$$B(n, N_T(\mathbf{x}_f)) = \begin{cases} -\log(a(n, N_T(\mathbf{x}_f)) & n \geq N_T(\mathbf{x}_f) \\ \text{Inf} & n < N_T(\mathbf{x}_f) \end{cases} \tag{4.1}$$

with $N_T(\mathbf{x}_f)$ is the number of docks required at a target and n is the maximum number of docks an agent can perform based on its type (6 for connectors, 2 for rods). The barrier function is chosen to give $\mathcal{B}(n, N_T(\mathbf{x}_f))$ infinite value when the number of docks at a target exceeds the number of docks the agent can perform.
It is also possible to use a different sigmoid function to change the performance of the assignment by changing the function $a(n, N_T(\mathbf{x}_f))$.

Three different $a(n, N_T(\mathbf{x}_f))$ are shown in Fig. 4.4. Using $a(n, N_T(\mathbf{x}_f)) = 1$ gives a simple sorting of agents. Changing $a(n, N_T(\mathbf{x}_f))$ to some decreasing positive function of $n - N_T(\mathbf{x}_f)$ like $1/(n - N_T(\mathbf{x}_f) + 1)$ makes agents inclined towards positions where they are most useful. If no barrier function is used then the problem is the same as the standard auction from SATO [101]. The chosen barrier function uses $a(n, N_T(\mathbf{x}_f)) = n - N_T(\mathbf{x}_f)$ to dissuade agents from using all of their docking ports.

The barrier function used above would not be sufficient to properly assign all input

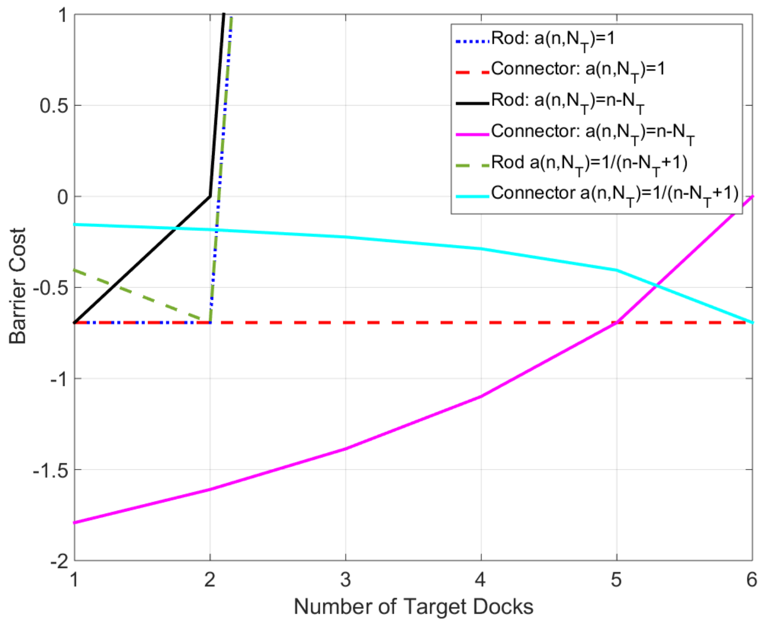


Figure 4.4: Possible barrier functions to use in assignment

configurations. A potential improper assignment is illustrated in Fig. 4.5. In assignments with underutilized connectors, with one or two docks required, it is necessary to augment the barrier function to include the angle between the docks so that improper assignments are avoided. This requires that the angle of the docks are encoded in the desired final configuration that all agents have access to.

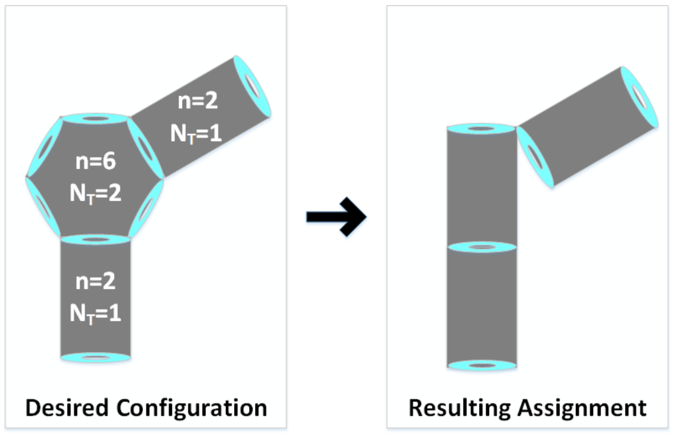


Figure 4.5: Bad Assignment: the desired configuration on the left has an underutilized connector, with only two docks required. This allows the initial barrier function to mis-assign a rod agent to the connector location, resulting in a disconnected structure.

The angle barrier function becomes more complicated in 3D configurations, and the choice of barrier function depends on the geometries of the chosen swarm. For planar configurations, the angle barrier function can act on the angle between docking agents, some multiple of 60° for connectors, or 180° for rods

$$\mathcal{B}_{ang}(\theta, \Theta_T(\mathbf{x}_f)) = -\log(g(\theta, \Theta_T(\mathbf{x}_f)) + 1) \tag{4.2}$$

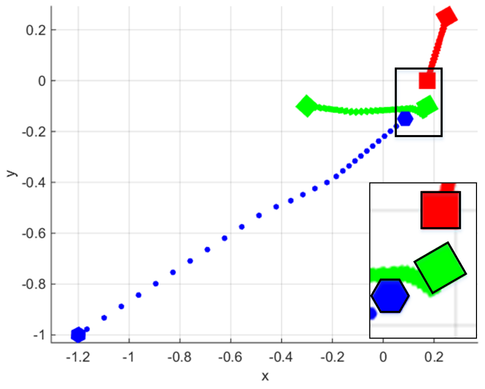
for some sigmoid $g(\theta, \Theta_T(\mathbf{x}_f))$ like

$$g(\theta, \Theta_T(\mathbf{x}_f)) = \begin{cases} \theta - \Theta_T(\mathbf{x}_f) & \theta \geq \Theta_T(\mathbf{x}_f) \\ -1 & \theta < \Theta_T(\mathbf{x}_f) \end{cases} \tag{4.3}$$

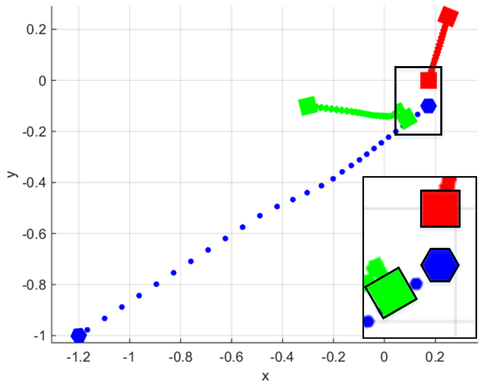
where θ is the dock angle allowed by the agent type, either 60° for connectors or 180° for rods, and $\Theta(\mathbf{x}_f)$ is the angle required for docking at terminal position $\mathbf{x}_f$. In 3D configurations, the connector geometry we have chosen is the same as 2D, but we opt to utilize the all of the dock orientations allowed by the hermaphroditic docking port. The barrier function for the planar case can also be used for this configuration since the out of plane angle does not affect the choice between the rod and the connector. Adding this barrier function and providing the required dock angles successfully eliminates this problem, as shown in Fig. 4.6. Without the angle barrier function, the agents can assign to improper target locations. The addition of the angle barrier function prevents rods from assigning to terminal positions which require a connector because of the dock angle.

To apply SOCA to a swarm with a higher degree of heterogeneity, the logic in the above barrier functions would need to be altered to properly sort between the characteristics of the new agent types.

The algorithm to solve the optimal assignment problem is presented in Algorithm 4. See [101] for an optimality proof of this auction algorithm.



(a) Without angle barrier function



(b) With angle barrier function

Figure 4.6: 3-Agent example showing the efficacy of the angle barrier function in preventing improper assignments

Method 4 Distributed Auction Algorithm for Docking (DAA-D)

```
 1: $\mathcal{X}_f$ = terminal positions in desired shape
 2: $\mathbf{c}^i(s)$ = cost of agent $i$ choosing target $s$
 3: $\mathbf{B}^i(s)$ = docking barrier function for agent $i$ choosing target $s$
 4: $\mathbf{B}^i_{ang}(s)$ = angle barrier function for agent $i$ choosing target $s$
 5: $m^i$ = # of targets available for agent $i$ to bid on
 6: $\mathbf{p}^i = \mathbf{0}_{1\times m^i}$
 7: $\mathbf{p}^i_{\text{old}} = -\mathbf{1}_{1\times m^i}$
 8: $j^i = 1$
 9: $count^i = 0$
10: for all $i$ (run in parallel) do
11:     while $count^i < 2D_{\text{net}}$ do
12:         if $|\mathbf{p}^i(j^i)| > \mathbf{p}^i_{\text{old}}(j^i)$ ($i$ is outbid) then
13:             $m^i = \max(m^i, |\{s|\mathbf{p}^i(s) \neq 0\}|)$
```

```
14:         **if** $|\{s|\mathbf{p}^i(s) > 0\}| = m^i$ **then**
15:             $m^i = |\{s|\mathbf{p}^i(s) > 0\}| + 1$
16:             $\mathbf{p}^i(1:m^i) = -\left(|\mathbf{p}^i(1:m^i)| + \epsilon\right)$
17:         **end if**
18:         $v^i = \min_{s=1\dots m^i}\left(\mathbf{c}^i(s) + \mathbf{B}^i(s) + \mathbf{B}^i_{ang}(s) + |\mathbf{p}^i(s)|\right)$
19:         $j^i = \arg\min_{s=1\dots m^i}\left(\mathbf{c}^i(s) + \mathbf{B}^i(s) + \mathbf{B}^i_{ang}(s) + |\mathbf{p}^i(s)|\right)$
20:         $w^i = \min_{s=1\dots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + \mathbf{B}^i(s) + \mathbf{B}^i_{ang}(s) + |\mathbf{p}^i(s)|\right)$
21:         $\gamma^i = w^i - v^i + \epsilon$
22:         $\mathbf{p}^i(j^i) = |\mathbf{p}^i(j^i)| + \gamma^i$
23:         $count^i = 0$
24:     **else if** $\mathbf{p}^i \neq \mathbf{p}^i_{\text{old}}$ (*another agent is outbid*) **then**
25:         $m^i = \max\left(m^i, |\{s|\mathbf{p}^i(s) \neq 0\}|\right)$
26:         $count^i = 0$
27:     **else**
28:         $count^i = count^i + 1$
29:     **end if**
30:     $\mathbf{p}^i_{\text{old}} = \mathbf{p}^i$
31:     Communicate $\mathbf{p}^i$ to all agents in $\mathcal{N}_{[i]}$
32:     **for** $s = 1 \dots m^i$ **do**
33:         $\mathbf{p}^i(s) = \min_{q\in\arg\max_{q\in\mathcal{N}_{[i]}}(|\mathbf{p}^q(s)|)}\left(\mathbf{p}^q(s)\right)$
34:     **end for**
35:   **end while**
36:   *Optional:* $m^i = |\{j|\mathbf{p}^i(j) \neq 0\}|$
37:   *Optional:* Go back to line 6 and rerun with new $m^i$
38:   $\mathbf{x}_{i,f} = \mathcal{X}_f(j^i)$
39: **end for**
```

4.2 SOCA Problem Statements and Algorithms

4.2.1 Problem Statement

The updated SOCA assignment and trajectory generation problems with the changes discussed above are:

Problem 5 (Auction Cost).

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \tag{4.4}$$

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \quad k = k_0, \ldots, T-1, \quad j = 1, \ldots, N \tag{4.5}$$

$$\|\mathbf{u}_j[k]\|_\infty \leq U_{\max} \quad k = k_0, \ldots, T-1, \quad j = 1, \ldots, N \tag{4.6}$$

$$\|H\mathbf{x}_j[k]\|_2 \leq V_{\max} \quad H = [0_{3\times3} \quad I_{3\times3}], \quad k = k_0, \ldots, T, \quad j = 1, \ldots, N \tag{4.7}$$

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0} \tag{4.8}$$

$$\mathbf{x}_j[T] = \mathcal{X}_f(j) \tag{4.9}$$

The cost function used here and in the trajectory generation section was chosen to create spacecraft fuel optimal paths, though the vector norm chosen depends the spacecraft thruster configuration [100].

Problem 6 (Updated Assignment Problem).

$$\min_{\mathbf{x}_{j,f},\ j=1\ldots N} \sum_{j=1}^{N} [C(\mathbf{x}_{j,0}, \mathbf{x}_{j,f}) + \mathcal{B}(n_j, N_T(\mathbf{x}_{j,f})) + \mathcal{B}_{ang}(\theta_j, \Theta_T(\mathbf{x}_{j,f}))] \tag{4.10}$$

subject to the following constraints:

$$\mathbf{x}_{j,f} \in \mathcal{X}_f, \quad \mathbf{x}_{j,f} \neq \mathbf{x}_{i,f}, \quad \forall j = 1 \ldots N, \quad \forall i \neq j$$

where $\mathcal{B}(n_j, N_T(\mathbf{x}_{j,f}))$ is the docking port barrier function and $\mathcal{B}_{ang}(\theta_j, \Theta_T(\mathbf{x}_{j,f}))$ is the docking angle barrier function presented above in Eq. (4.1) and (4.2).

Problem 7 (Updated Trajectory Generation).

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \tag{4.11}$$

subject to the following constraints:

(a) the dynamics, state, and control constraints

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \quad k = k_0, \ldots, T-1, \quad j = 1, \ldots, N \tag{4.12}$$

$$\|\mathbf{u}_j[k] - \bar{\mathbf{u}}_j[k]\| \leq (\beta)^{w-1}\mathcal{T}_0, \quad \forall k = k_0, \ldots, T-1 \tag{4.13}$$

$$\|H\mathbf{x}_j[k]\|_2 \leq V_{\max} \quad H = [0_{3\times3} \quad I_{3\times3}], \quad k = k_0, \ldots, T, \quad j = 1, \ldots, N \tag{4.14}$$

(b) the initial and terminal conditions obtained from Problem 6

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0}, \quad \mathbf{x}_j[T] = \mathbf{x}_{j,f}, \quad j = 1, \ldots, N \tag{4.15}$$

(c) the new collision avoidance constraint

$$\begin{aligned} -(\bar{\mathbf{r}}_j[k] - \bar{\mathbf{r}}_i[k])^T \mathbf{r}_j[k] \leq &- R_{ij}(\|\tilde{\mathbf{r}}_j[k] - \tilde{\mathbf{r}}_i[k]\|_2 + 2\|P\|_2) + \|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ &- \|\tilde{\mathbf{r}}_j[k]^T P\|_2 + (\tilde{\mathbf{r}}_i[k] - \tilde{\mathbf{r}}_j[k])^T \tilde{\mathbf{r}}_i[k] \end{aligned} \tag{4.16}$$

$$G = [I_{3\times3} \quad 0_{3\times3}], \quad k = k_{bl}, \ldots, \min\{k_0 + T_H, T\}, \quad i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \setminus \mathcal{D}_j$$

where $\mathcal{N}_{[j]} = \{i| \ \|\mathbf{x}_j[k_0] - \mathbf{x}_i[k_0]\|_2 \leq R_{\text{comm}}\}$, R_{comm} is the communication radius of each agent, $\mathcal{D}_j$ is the set of agents that are assigned to dock with agent j and $\mathcal{P}_j$ is the set of agents that have a higher priority than j, P is the ellipsoidal error profile of the nominal trajectory, and R_{ij} depends on R_{col} or $2R_{dock}$ if agent i is in the assembly set of j

(d) the docking condition

if $\|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k]\|_2 \leq R_{\text{bl}}$:

$$\|G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k])\|_2 \leq R_{\text{cone}}(k), \tag{4.17}$$

$$k_{bl} = \arg\min_{k_0 \leq k \leq k_0 + T_H} \{\|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k]\|_2 - R_{\text{bl}}\} \tag{4.18}$$

$$R_{\text{cone}}(k) = R_{\text{dock}} + \frac{\|G(\mathbf{x}_j[k_{bl}] - \bar{\mathbf{x}}_i[k_{bl}])\|_2 - R_{\text{dock}}}{T - k_{bl}}(T - k),$$

$$k = k_{bl}, \ldots, \min\{k_0 + T_H, T\},\ i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \cap \mathcal{D}_j$$

After the problem above is solved using MPC-SCPn, the nonlinear correction step is applied:

$$\mathbf{x}_{n,j}^{w-1}[k+1] = \mathbf{f}_j[k](\mathbf{x}_{n,j}^{w-1}[k], \mathbf{u}_j^{w-1}[k]), \quad k = k_0, \ldots, T-1, \tag{4.19}$$

$$\mathbf{x}_{n,j}^{w-1}[k_0] = \mathbf{x}_j^{w-1}[k_0] = \mathbf{x}_0 \tag{4.20}$$

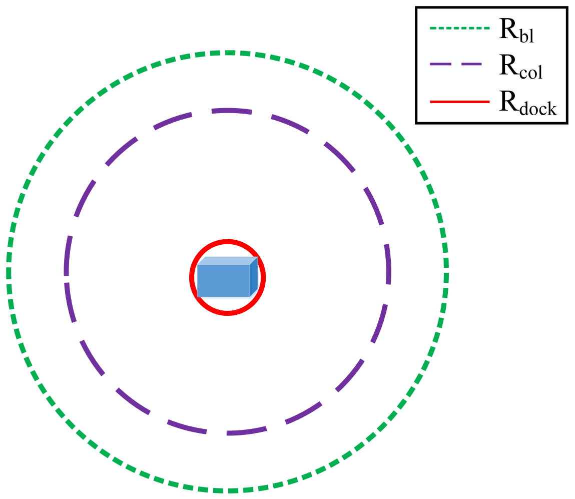
where $\mathbf{f}_j[k]$ is the Runge-Kutta integration of the state with the given control.

4.2.2 Optimal SOCA Trajectory Generation

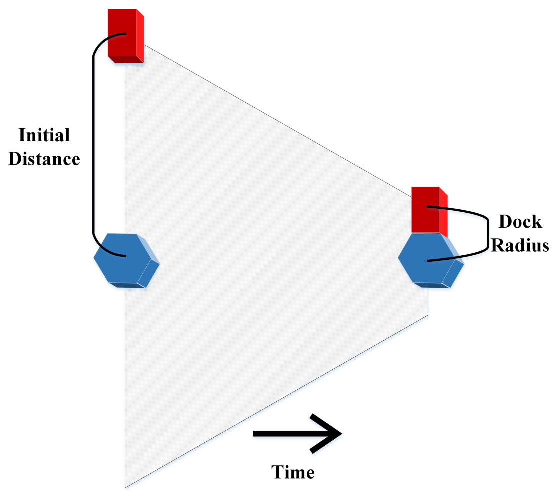
The next modification was to the trajectory generation algorithm. MPC-SCPn is used to create the optimal, collision-free trajectories to the targets selected by DAA-D. Initially, a nominal trajectory is generated without considering collision avoidance. Then each agent solves the MPC-SCPn problem with collision avoidance on a limited time horizon, with the knowledge of the nominal trajectories of the agents within its communication radius. The collision avoidance constraint is not convex, but by approximating the other agents' collision avoidance spheres as hyperplanes orthogonal to the surface, convexity can be obtained.

Because the agents need to dock for construction, the collision avoidance constraint must be suspended for agents that are attempting to dock. This is achieved using a boundary layer around the collision avoidance radius. The relative sizes of the radii are illustrated in

Fig. 4.7a. When an agent approaches within the boundary layer, the main agent checks to see if the approaching agent is targeted for a location neighboring the main agent's target. If it is, the agent follows the docking cone constraint. Otherwise, the collision avoidance constraint holds.



(a) Relative sizes of docking, collision avoidance and boundary layer radii



(b) Docking Cone

Figure 4.7: Description of docking radii and cone. In (a), the relative sizes of the docking, collision avoidance and boundary layer radii. Once another agent comes within the boundary layer radius, the main agent decides whether to dock or avoid it. If the agents are docking, the docking cone constraint shown in (b) goes into effect. The initial distance is the distance when the docking constraint is first applied, when the red agent comes within the boundary layer of the blue agent or vice versa.

In order to allow the agents some flexibility in docking, the docking constraint requires the agent to maintain a shrinking distance to the agent to be docked. This means that over time the main agent will stay within a cone defined by the other agent. The cone radius begins as the initial separation and ends as the agent docking radius, as seen in Fig. 4.7b. This docking cone forces the agents to come together by the final time. This way, the agents are allowed to dock before they reach the target if it is beneficial to their trajectories, or they can wait until the final position to dock.

Avoiding Collision with Docked Agents

This implementation requires that agents avoid docking with an approaching agent that is not in its docking set, even if that agent is already docked with an agent in its docking set. This causes agents to avoid docking until the final time step to avoid collision with the other agent. This is fixed by simply adjusting the collision avoidance radius of agents in this position. Each agent, j, knows the set of agents it intends to dock with based on the assigned terminal positions. As agent j docks with another agent i, agent i is added to the assembly set of agent j, A^j_{set} and vice versa. This set is communicated to other agents in j's dock set, which then reduce the collision avoidance radius for those agents to double the docking radius. This does not affect the convexity of the collision avoidance constraint because the radius changes between time steps. To illustrate this problem and the solution, Fig. 4.8 shows the agents smoothly travelling to the equilateral triangle final configuration. The blue circles in the figure represent the collision avoidance radius of each agent. In this final configuration, the collision avoidance radii overlap. Without the adjusted collision avoidance for agents in the assembly set, the yellow, magenta, and cyan agents would avoid each other causing irregularities in the trajectories. The magenta agent can be seen avoiding the cyan agent by the adjusted smaller collision avoidance radius as it passes near the end of the trajectory.

Second-Order Cone Programming

The convexified collision avoidance constraint in Problem 7 introduces a lot of error in the estimation of the other agents' trajectories. One way to make the algorithm robust to that error is to use second order cone programming to frame the constraint [27]. This method uses ellipsoidal error profiles around the estimated trajectory.

Lemma 1 Let the nominal trajectories $\bar{\mathbf{x}}$ of each agent i approaching agent j be distorted by an ellipsoidal error P, which is the same for all agents and defined as the square root of the measurement covariance. Under these conditions the collision avoidance constraint

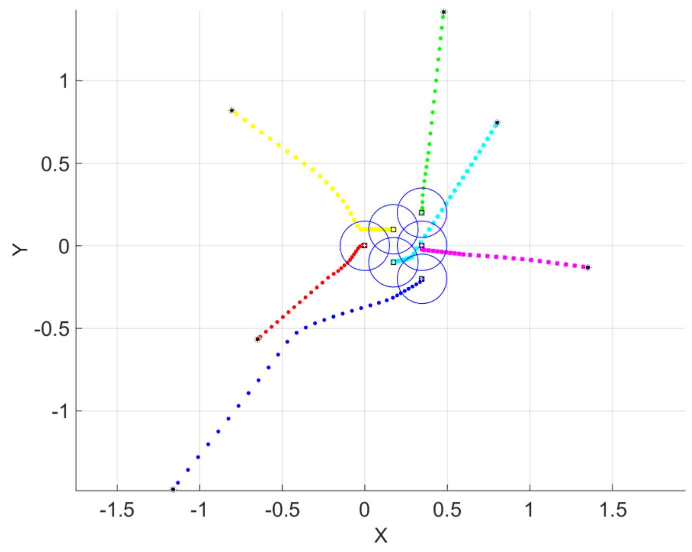


Figure 4.8: With the use of the assembly set condition, six agents can easily form a planar equilateral triangle without the collision avoidance issue. This example is performed using planar position and attitude double integrator dynamics for clarity.

becomes:

$$\begin{aligned} -(\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\mathbf{r}_j[k] \leq &- R_{ij}(\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]\|_2 + 2\|P\|_2) + \|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ &- \|\tilde{\mathbf{r}}_j[k]^TP\|_2 + (\tilde{\mathbf{r}}_i[k]-\tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k] \end{aligned} \tag{4.21}$$

Proof. The collision avoidance constraint is defined as:

$$(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])^TC^TC(\mathbf{x}_j[k]-\bar{\mathbf{x}}_i[k]) \geq R_{ij}\|C(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])\|_2 \tag{4.22}$$

for each agent j with neighbor agents i, where the nominal trajectories are denoted with bars. Since all but $\mathbf{x}_j[k]$ is constant and $\mathbf{r}[k] = C\mathbf{x}[k]$, this can be expressed as a linear inequality, $\mathbf{a}_j\mathbf{r}_j \leq \mathbf{b}_j$, with the constants given as:

$$\mathbf{a}_j = (\bar{\mathbf{r}}_i[k]-\bar{\mathbf{r}}_j[k])^T \tag{4.23}$$

$$\mathbf{b}_j = -R_{ij}\|\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k]\|_2 - (\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\bar{\mathbf{r}}_i[k] \tag{4.24}$$

We then introduce error ellipsoids into this linear inequality to increase robustness, with the ellipsoids defined using:

$$\sup\{\mathbf{a}_j\mathbf{x}_j|\mathbf{a}_j \in \varepsilon_j\} \le \inf \mathbf{b}_j \tag{4.25}$$

This implies the Second Order Cone Constraint [27]:

$$\bar{\mathbf{a}}_j x_j + \|P_j\mathbf{x}_j\|_2 \le \inf\{b_j\} \tag{4.26}$$

Then for any error direction $\mathbf{q}$:

$$\mathbf{a}_j = \{\bar{\mathbf{a}}_j + P_j\mathbf{q} | \|\mathbf{q}\|_2 \le 1\} \tag{4.27}$$

$$\mathbf{a}_j - \bar{\mathbf{a}}_j = P_j\mathbf{q} \tag{4.28}$$

$$(\mathbf{a}_j - \bar{\mathbf{a}}_j)^T P_j^{-T} P_j^{-1}(\mathbf{a}_j - \bar{\mathbf{a}}_j) = \mathbf{q}^T\mathbf{q} \le 1 \tag{4.29}$$

Assume $\|\mathbf{q}\| \le 1$, $\bar{\mathbf{r}}_j[k] = \tilde{\mathbf{r}}_j[k] + P_j\mathbf{q}_j$, $\bar{\mathbf{r}}_i[k] = \tilde{\mathbf{r}}_i[k] + P_i\mathbf{q}_i$, where $\tilde{\mathbf{r}}$ denotes the actual nominal trajectory. Using these assumptions and plugging Eq. (4.23) into the left hand side of Eq. (4.25), we get:

$$\begin{aligned}\sup_{\|q_{j,i}\|_2 \le 1}(\mathbf{a}_j\mathbf{r}_j[k]) &= (\bar{\mathbf{r}}_i[k] - \bar{\mathbf{r}}_j[k])^T\mathbf{r}_j[k] + \sup_{\|q_{1,2}\|_2 \le 1}(\mathbf{q}_j^T P_j^T \mathbf{r}_j[k] - \mathbf{q}_i^T P_i^T \mathbf{r}_j[k]) \\ &= (\bar{\mathbf{r}}_i[k] - \bar{\mathbf{r}}_j[k])^T\mathbf{r}_j[k] + 2\|P_j^T\mathbf{r}_j[k]\|_2\end{aligned} \tag{4.30}$$

Substituting the assumed error ellipse constraints and assuming equal error profiles for both agents

$$\begin{aligned}\mathbf{b}_j &= -R_{ij}\|\tilde{\mathbf{r}}_j[k] - \tilde{\mathbf{r}}_i[k] + P_j\mathbf{q}_j - P_i\mathbf{q}_i\|_2 \\ &\quad - (\tilde{\mathbf{r}}_j[k] + P_j\mathbf{q}_j - \tilde{\mathbf{r}}_i[k] - P_i\mathbf{q}_i)^T(\tilde{\mathbf{r}}_i[k] + P_i\mathbf{q}_i) \\ &\le -R_{ij}(\|\tilde{\mathbf{r}}_j[k] - \tilde{\mathbf{r}}_i[k]\|_2 + 2\|P\|_2) + \|P^T\tilde{\mathbf{r}}_i[k]\|_2 - \|\tilde{\mathbf{r}}_j[k]^T P\|_2 \\ &\quad + (\tilde{\mathbf{r}}_i[k] - \tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k]\end{aligned} \tag{4.31}$$

Through manipulation, we arrive at the new collision avoidance constraint:

$$\begin{aligned} -(\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\mathbf{r}_j[k] \leq & - R_{ij}(\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]\|_2 + 2\|P\|_2) + \|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ & - \|\tilde{\mathbf{r}}_j[k]^T P\|_2 + (\tilde{\mathbf{r}}_i[k]-\tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k] \end{aligned} \tag{4.32}$$

□

The ellipsoidal error P allows the error in the nominal trajectory to be fitted to the on-board sensors more realistically. This is particularly important with the swarm configuration since small satellite sensors are less capable. Using the same profile for all agents is appropriate since most of the error would come from sensing, which is the same for all agents.

4.2.3 MPC-SCPn Nonlinear Dynamics Correction

Though MPC-SCP trajectory generation converged on a solution, the solution did not necessarily follow the actual spacecraft dynamics. This was due to the linearization done in the optimization loop. The losses due to linearization accumulated over the trajectory and could become substantial over long duration simulations. The constraints were modified and a nonlinear correction step was added to the MPC-SCPn portion of SOCA to reduce these errors, as seen in Chapter 3.

For a discretized non-convex problem with nonlinear dynamics and convex or convexified constraints like spacecraft guidance with collision avoidance, the solution can be approximated using sequential convex programming. This is done by linearizing the dynamics and generating a solution iteratively using the previous solution as a nominal trajectory until the optimization converges on a solution, like in MPC-SCP. For SCP with nonlinear correction, the nominal trajectory for the next SCP iteration is taken as the numerically integrated nonlinear dynamics using the initial conditions and the current SCP-generated control trajectory:

$$\mathbf{x}^w_{n,j}[k+1] = \mathbf{f}_j[k](\mathbf{x}^w_{n,j}[k], \mathbf{u}^w_j[k]), \quad k = k_0, \ldots, T-1,$$
$$\text{and} \quad \mathbf{x}^w_{n,j}[k_0] = \mathbf{x}^w_j[k_0] = \mathbf{x}_0 \tag{4.33}$$

where $f_j[k]$ is the nonlinear dynamics, w is the SCP iteration, and the subscript n indicates a nominal, nonlinear trajectory. This SCP optimization process along with nonlinear dynamic correction step (4.33) is repeated until the sequence of trajectories converges. To ensure convergence and optimality, the problem's inequality constraints must be modified so that the corrected solution will be provably feasible to the nonconvex problem, generalized to Problem 2. The convergence and optimality of SCPn is shown in Chapter 3.

The algorithm to solve Problem 7 for the optimal trajectories allowing both docking and collision avoidance is shown below in Algorithm 5. Note that the nonlinear correction step 11 occurs after each iteration of MPC-SCP.

Method 5 Guidance and Control using Sequential Convex Programming

```
1:  $\bar{\mathbf{x}}_j[k] := \mathbf{0}_{6\times 1},\ \forall j, k$
2:  $\mathbf{x}_{j,0}[k] :=$ the solution to Problem 7 (Trajectory Generation) with $\mathcal{P}_j = \emptyset,\ \forall j, k$
3:  $\bar{\mathbf{x}}_j[k] := \mathbf{x}^0_j[k],\ \forall j, k$
4:  Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_{[j]}$)
5:  $\mathcal{K} := \{1, \ldots, N\}$
6:  $w := 1$
7:  while $\mathcal{K} \neq \emptyset$ do
8:      for all $j \in \mathcal{K}$ (run in parallel) do
9:          $\mathbf{x}^{nom}_{j,w}[k] :=$the solution to Problem 7 (Trajectory Generation), $\forall k$
10:     end for
11:     $\mathbf{x}_{j,w}[k] := \mathbf{f}_j[k](\mathbf{x}^{nom}_{j,w}[k], \mathbf{u}^{nom}_w[k])\quad \forall k$ (Nonlinear Correction)
```

```
12:     for all $j$ (run in parallel) do
13:         $\bar{\mathbf{x}}_j[k] := \mathbf{x}_{j,w}[k],\ \forall k$
14:         Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_{[j]}$)
15:         if $\|\mathbf{x}_{j,w}[k] - \mathbf{x}_{j,w-1}[k]\|_\infty < \epsilon_{\text{SCP}}\ \forall k$ and $\|G(\mathbf{x}_{j,w}[k] - \mathbf{x}_{i,w}[k])\|_2 > R_{\text{col}}\ \forall k \geq k_{bl}, \forall i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \setminus \mathcal{D}_j$ then
16:             Remove $j$ from $\mathcal{K}$
17:         end if
18:     end for
19:     $w := w + 1$
20: end while
```

A model predictive control implementation of SCPn (Algorithm 5) can be used to implement the DAA-D and SCPn methods in real time in order to simultaneously solve the SOCA problems. MPC uses a receding horizon to update the optimal target assignments for docking (Algorithm 4) and current trajectories obtained via communication with neighbors and on-board sensors. SOCA is described in Algorithm 6.

Method 6 Swarm Orbital Construction Algorithm (SOCA)

```
1: $k_0 = 0$
2: while $k_0 \leq T$ do
3:     for all $i = 1, \ldots, N$ (parallel) do
4:         for all $j = 1, \ldots, M$ do
5:             Solve Problem 5 using SCP (Algorithm 5)
6:             $\mathbf{c}^i(j)$ = cost of optimal solution to Problem 5
7:         end for
8:     end for
9:     Solve Problem 6 using DAA-D (Algorithm 4)
10:    $\mathbf{x}_{j,f}$ = solution to Problem 6, $\forall j$
```

```
11:     if # of bids has changed then
12:         $k_0 = 0$
13:     end if
14:     Solve Problem 7 using SCP (Algorithm 5)
15:     $\mathbf{u}_j[k]$ = control solution to Problem 7, $\forall j$, $k = k_0 \dots k_0 + T_H - 1$
16:     Apply $\mathbf{u}_j[k]$ for $k = k_0 \dots k_0 + T_H - 1$
17:     Update $k_0$ and $\mathbf{x}_{j,k_0}$ to current time
18: end while
```

4.3 Simulation of 3D Spacecraft Dynamics with Attitude

The algorithm with the above modifications was implemented in a simulation with 54 agents (24 connectors, 30 rods) using high-fidelity relative orbit dynamics [99] with J_2 perturbations with a virtual chief in a 500 km, 45° inclination orbit. The attitude dynamics used are Euler's rotational equation. The LVLH frame J_2 perturbed orbital dynamics of each agent j are described by the following equations [99]:

$$\ddot{x}_j = 2\dot{y}_j\omega_z - x_j(\eta_j^2 - w_z^2) + y_j\alpha_z - z_j\omega_x\omega_z - (\zeta_j - \zeta)\sin(i)\sin(\theta) - r(\eta_j^2 - \eta^2) \tag{4.34}$$

$$\begin{aligned}\ddot{y}_j =& 2\dot{x}_j\omega_z + 2\dot{z}_j\omega_x - x_j\alpha_z - y_j(\eta_j^2 - w_z^2 - \omega_x^2) + z_j\alpha_x \\ &- (\zeta_j - \zeta)\sin(i)\cos(\theta) + x_j\omega_z - z_j\omega_x)\end{aligned} \tag{4.35}$$

$$\ddot{z}_j = 2\dot{y}_j\omega_x - x_j\omega_x\omega_z - y_j\alpha_x - z_j(\eta_j^2 - w_x^2 - (\zeta_j - \zeta)\cos(i)) \tag{4.36}$$

with parameters as defined in Ref. [99]. These equations are then linearized with respect to a nominal orbit found for each agent j for use in MPC-SCP. The attitude dynamics use

Euler's rotational equation and attitude kinematics:

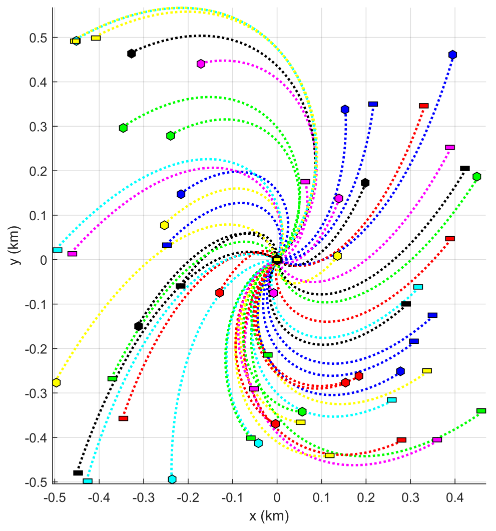
$$\mathbf{I}\dot{\omega} + \omega \times (\mathbf{I} \cdot \omega) = \tau_{\text{ext}} \tag{4.37}$$

$$\dot{\theta} = \mathbf{Z}(\theta)\omega \tag{4.38}$$

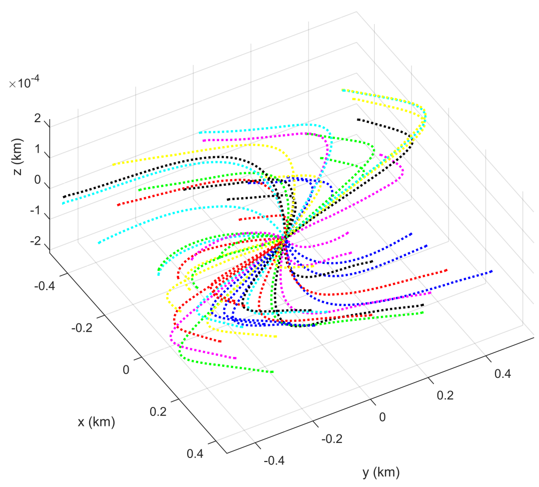
where $\mathbf{I}$ is the inertia matrix, ω is the angular velocity, $\tau_{(ext)}$ is the external torque vector acting on the body, θ is the vector of 3-2-1 Euler angles, and $\mathbf{Z}(\theta)$ is the corresponding kinematic transformation matrix for these Euler angles [18, 130]. These equations are linearized with respect to a nominal attitude trajectory.

4.3.1 Simulation Results

The first simulation uses 24 connectors and 30 rods targeted to a planar flower shape, though the trajectories to achieve the arrangement are three dimensional. In the figures presenting the results, rod agents are represented by rectangular prisms and connector agents are represented by hexagonal prisms, rotated to the trajectory orientation. Initial positions are enlarged to show orientation. The agents begin in J_2 invariant relative orbits, which greatly reduce the energy required to maintain the orbit and would likely be used for agents awaiting docking. The agents have an initial separation of up to 1.5 kilometer and final separation of twenty centimeters.



(a) **2D View of Trajectories**



(b) **3D View of Trajectories**

Figure 4.9: 54 agents (30 rods, 24 connectors) combine from up to 1.5 km apart to make a planar hexagon with a 20 cm separation. Note the out of plane motion is all within 0.2 meters.

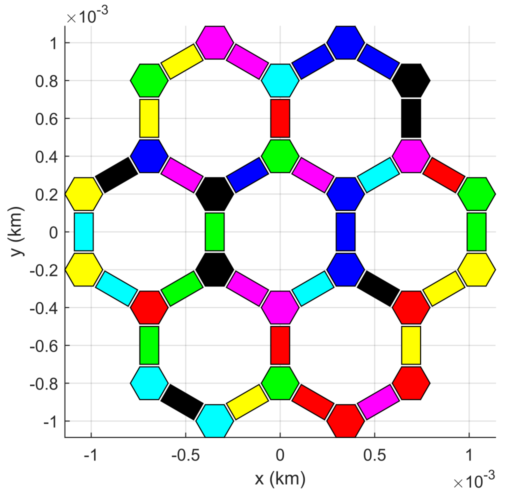
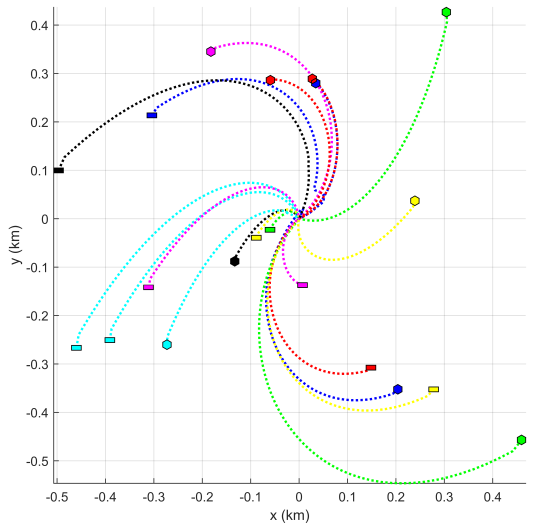


Figure 4.10: Zoomed in view of the final time step of all 54 agents in the final flower shape

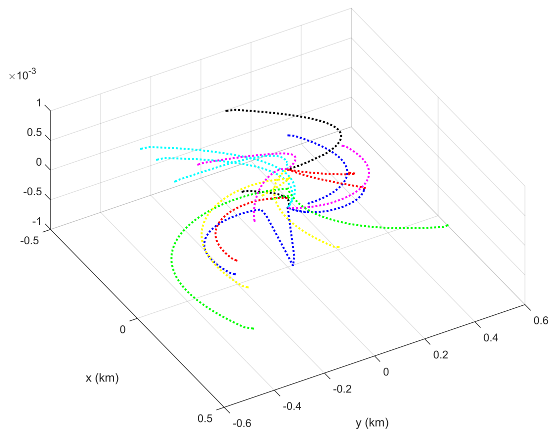
Fig. 4.9a shows the 2D view of the overall trajectories of the agents over the duration of the simulation. The 3D trajectories are shown in Fig. 4.9b. Though the agents are allowed to move in three dimensions, the out of plane motion is minimal (<20 cm) due to the expense of out of plane motion and the planar nature of the target configuration.

The scale difference is too large in these figures to show the agents achieve the target configuration since the final separation is so small. Fig. 4.10 shows the position and orientation of all agents at the final time step. The agents reach the desired terminal configuration.

The second simulation uses 10 connectors and 10 rods targeted to a three dimensional folded hexagon shape. Fig. 4.11a shows the 2D view of the overall trajectories of the agents over the duration of the simulation. The 3D trajectories are shown in Fig. 4.11b. The out of plane motion in this simulation is larger than the previous simulation, about one meter. This is still very small, just above the 65 centimeters required of the target shape.
Fig. 4.12 shows the position and orientation of all agents at the final time step. Again, all agents reach the desired terminal configuration.



(a) 2D View of Trajectories



(b) 3D View of Trajectories

Figure 4.11: 20 agents (10 rods, 10 connectors) combine from up to 1.5 km apart to make a folded hexagon with a 20 cm separation. Note the out of plane motion is still small, but up to one meter

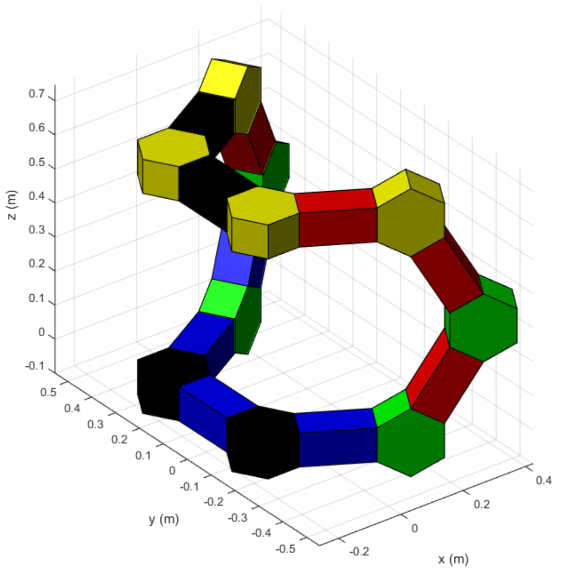


Figure 4.12: Zoomed in view of the final time step of all 20 agents in the 3D folded hexagon shape

4.4 3DOF Experiment on Omni-Directional Wheeled Robots and M-STAR Spacecraft Simulators

Experimental validation of the algorithm was performed first on wheeled robots in a motion capture environment then on spacecraft simulators in the flat floor facility at Caltech. The SOCA algorithm can generate optimal collision-free trajectories from any set of initial conditions within the workspace to create the desired shape at the specified final time, but in these experiments the algorithm is run offline so the robots are initialized in the same configuration each run.

4.4.1 Omni-Wheeled Robot Experimental Validation

Initial experiments were performed using six NEXUS 3-Wheeled Compact Omni-Directional Arduino Compatible Mobile Robots (shown in Fig. 4.13). The robots are controlled using

Arduinos with XBee communication devices. The SOCA is run on a separate computer, which uses ROS to access the motion capture system, run the algorithm, and send commands to the robots. This computer then communicates commands to the robots through the Xbee serial wireless modules attached to each of the robots. SOCA generates trajectories which require motion capture feedback to follow, so each robot is given its current position and orientation, along with the trajectory waypoint. Onboard the robot, a PD controller is used to generate the desired wheel RPMs, then map those values to the actual robot wheel commands, a set of three PWM values. A six-robot system was tested in a 2-meter by 2-meter motion capture space, where each robot read in its trajectory, given by a list of waypoints provided by an offline system SOCA algorithm.

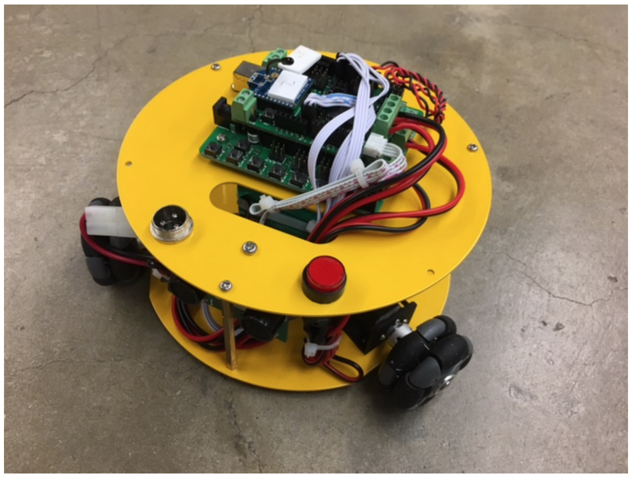


Figure 4.13: Omni-Directional 3-wheeled robot used to experimentally validate SOCA

Multiple preliminary tests were performed to characterize the relationship between the input PWM signal and the motor RPM. Inconsistencies were observed in the RPM at a constant PWM even on smooth surfaces. This problem was ameliorated by closing the loop onboard the robots using the motor encoders, which gives better control over the speed and path of the robot. The encoders are read at each control loop (about 0.1s) and the PWM is corrected based on the encoder readings. For more details see [57]. The experimental results show SOCA performing assignment and trajectory generation for 6 agents in planar final configuration with realistic 3DOF trajectories.

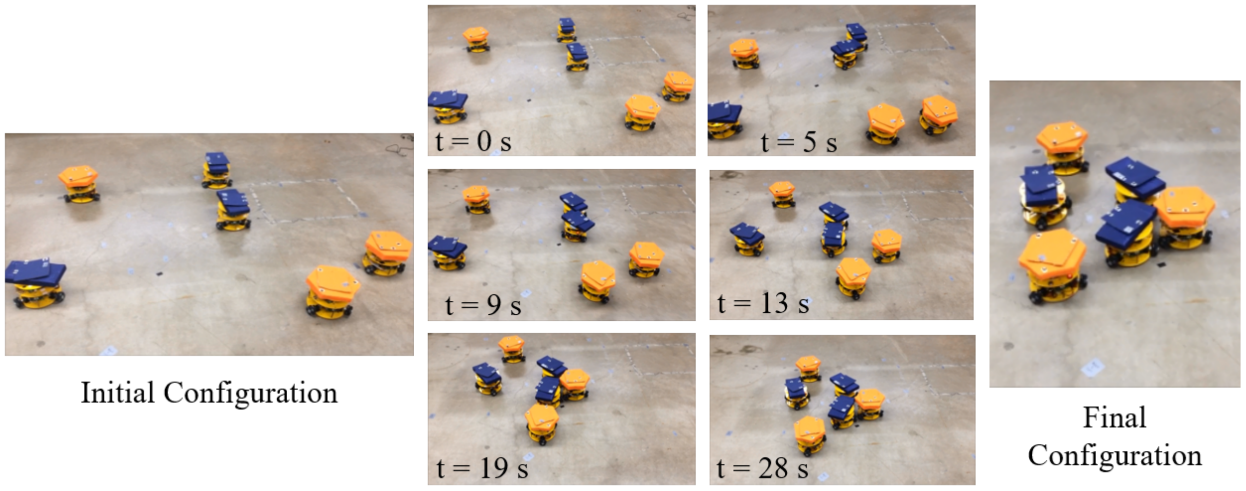


Figure 4.14: Time lapse of 6 omni-directional robots for SOCA experiment

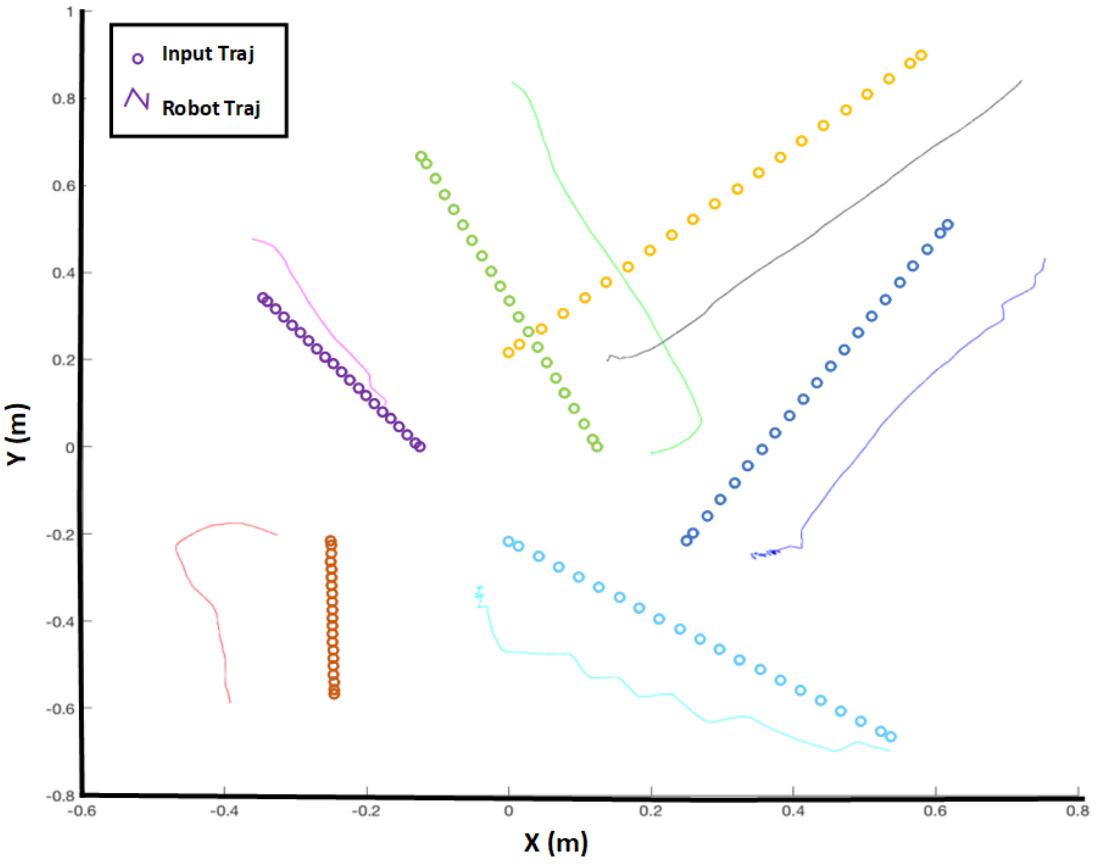


Figure 4.15: Actual vs desired robot trajectories

The off-board control computer ran the SOCA algorithm given intended starting points of each of the six robots and determined the optimal trajectory with collision avoidance. The starting positions of the robots and the time lapse of the full test is shown in Fig 4.14. The algorithm worked very well however the onboard controller on the robots could not track the trajectories sufficiently well to enable docking. The tracking error of the robots is approximately 10 cm, which is sufficient to eliminate the collision avoidance effects of SOCA. This causes the two rectangular agents to get stuck at t=5-10sec and the bottom rectangular agent to miss its final orientation. To within the tracking error, all of the robots follow the trajectories well. Further investigation is needed to conclusively determine the source of the error but the culprit is most likely the low-level motor controller which has very inconsistent performance due to the noisy wheel encoders. The intended trajectory is plotted in Fig. 4.15 with the actual trajectories achieved by the robots. For the most part, each of the robots achieves the correct direction of travel, but the tracking error prevents the trajectories from lining up perfectly. The wiggle seen in blue at the bottom of the figure is likely cause by this error in wheel actuation. The feedback keeps pulling the robot back towards the desired trajectory but the wheel errors keep causing deviations. The actuation errors in this ground robot system were a motivating factor in the development of the spacecraft simulator facility.

4.4.2 In-Orbit Construction Experiment on the M-STARs

For this experiment, the four M-STARS described in Chapter 2 made a T shape in the center of a 4 meter square. Three of the four agents were identical, with two docking ports placed on opposite sides of the square upper stage. The fourth M-STAR had four ports, one on each side of the upper stage. These specifications were input into SOCA to generate optimal, collision-free construction trajectories. Due to time limitations, the trajectories were computed offline then fed to the M-STARs. These trajectories were then followed using the onboard control scheme described in [55].

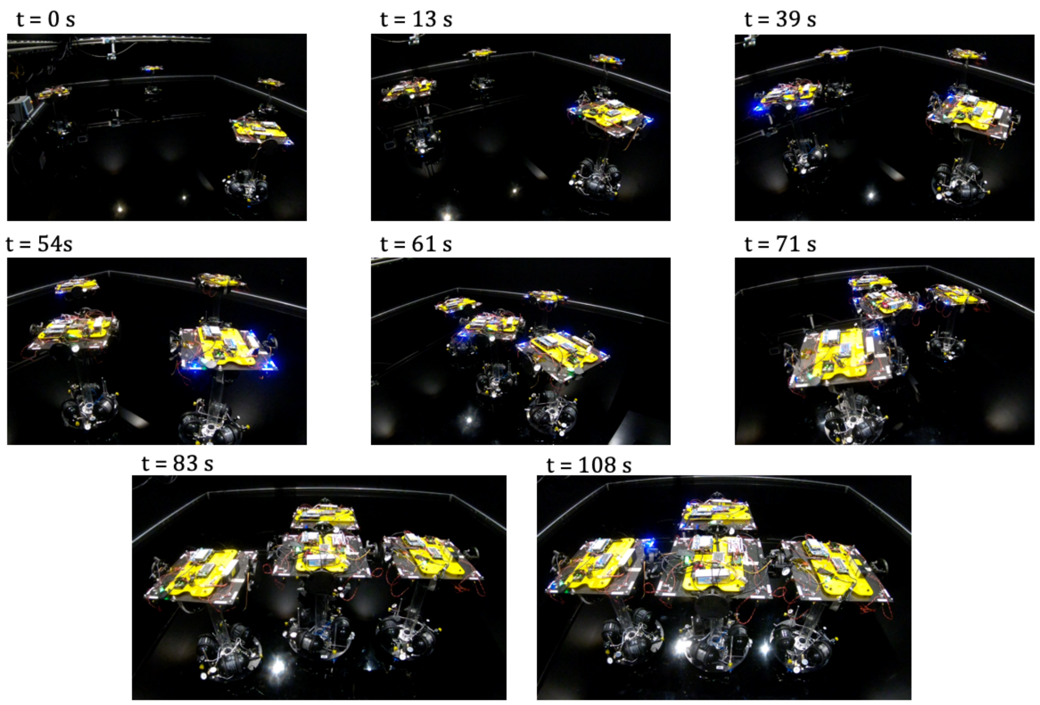


Figure 4.16: 4 M-STARs Following SOCA Optimal Trajectories in Spacecraft Simulator Facility

Due to the aggressive nature of the optimal trajectories, the simulators come very close. Errors in trajectory tracking caused several initial attempts to fail. After extensive gain tuning and cleaning of the flat air bearings and the floor, the construction was successful and repeatable. A sample experimental trajectory is seen in Fig. 4.17, and the discrepancy between the command and actual trajectory is shown in 4.18. Stills from the experiment are shown in Fig. 4.16. The biggest issue is when spacecraft 4 encounters some friction on the floor at 55 s. Links to videos for the three experiment types can be found in Appendix B.

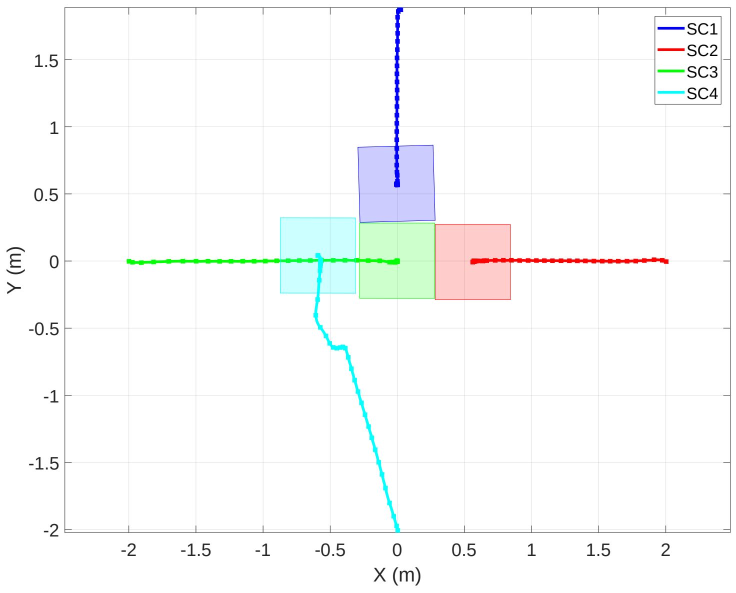


Figure 4.17: Full Motion Captured Trajectories of 4 M-STARs Following SOCA Optimal Trajectories

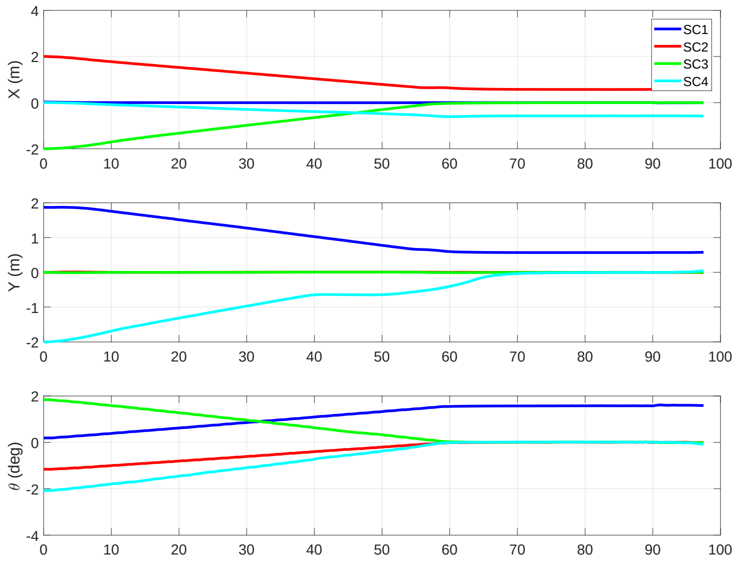


Figure 4.18: Experimental performance of M-STARS as compared to command

4.5 Control Allocation for Self-Assembling Structures

This section describes an online model aggregation and control allocation method which combines mass properties and removes actuators blocked by docking. This method will demonstrate the ability to prevent uncontrollable docks when used in conjunction with SOCA.

Combining the agent types above into any feasible structure requires a modularized approach to updating the model. This can be done by requiring that only one dock be performed at any given time. To effectively combine two models into one, we must first define a set of information that each model must contain:

- System mass
- Center of gravity
- Moment of inertia tensor
- Control influence matrix

We assume that each component satellite has its center of gravity at the geometric center of the object. Combining the masses and determining the new center of gravity is straightforward:

$$m_{ay} = m_1 + m_2 \tag{4.39}$$

$$^1CG_{ay} = \frac{m_2}{m_{ay}} \left[{}^1r_{1D} + {}^1r_{2D} \right] \tag{4.40}$$

From here the moment of inertia tensor can be calculated using the parallel axis theorem:

$$^1J_{ay,CG} = {}^1J_1 + m_1 \left(({}^1r_{1,CG})^2 I_3 - {}^1r_{1,CG}{}^1r_{1,CG}^T \right) + \tag{4.41}$$

$$^1R^{22}J_2({}^1R^2)^T + m_2 \left(({}^1r_{2,CG})^2 I_3 - {}^1r_{2,CG}{}^1r_{2,CG}^T \right) \tag{4.42}$$

Before calculating the changes to the control influence matrix, the control influence matrix of each agent type must be calculated.

$$B_2 = \begin{bmatrix} B_{Force} \\ B_{Torque} \end{bmatrix} = \begin{bmatrix} {}^2\left[d_{act,1}, d_{act,2}, \dots\right] \\ {}^2r_{act} \times B_{Force} \end{bmatrix} \tag{4.43}$$

This can then be transformed for each of the two docking components as follows:

$$B_{2_{ay}} = \begin{bmatrix} {}^1R^{22}[d_{act,1}, d_{act,2}, \dots] \\ ({}^1R^{22}r_{act} + {}^1r_{2,CG} - {}^1CG) \times B_{Force} \end{bmatrix} \tag{4.44}$$

$$B_{ay} = [B_{1_{ay}}, B_{2_{ay}}] \tag{4.45}$$

When a new agent is added, the position of the actuators in the A frame must be recalculated before solving for the B matrix. The new control influence matrix still needs some tweaking to remove actuators that have been blocked by the dock, or whose plumes would interact with other parts of the spacecraft. Actuators can be removed by creating an identity matrix of the size of the number of actuators, then zeroing out the row corresponding to the blocked actuator. This matrix then post-multiplies B to create the effective B matrix, B_{eff}. To determine which actuators are blocked, the configuration of the agents is leveraged. For both agent types, the blocked thrusters can easily be determined if the docking port in use is found. The thrusters associated with each docking port is predetermined and stored, then these agents are considered blocked if the docking port is in use. A useful extension here is finding the actuators whose plumes would interact with the assembled structure. Finding these actuators involves approximating the plumes as cones and checking for collision between these cones and the assembly. This is a simplistic model of thruster plumes, typically the three-dimensional characteristics of the plumes are taken into consideration and this may be addressed in future work. Collision checking methods from the field of robotics can be leveraged to efficiently determine if the plume cone interacts with the rest of the structure [32, 15]. After the unusable thrusters are pruned from the control allocation matrix, the control authority of this potential assembly must be checked. The control authority of the assembly is determined by calculating the singular value decomposition of the B matrix. This is then multiplied by the maximum achievable thrust to determine the maximum f_{des} that this assembly can achieve. For CubeSat scale actuators, a reasonable value for the maximum thrust of each actuator is 50 mN [4]. If this f_{des} is sufficient to complete the trajectory, the docking is allowed to occur, if not the agents must not dock and must avoid collision until one or both assemblies have docked with other agents. While the agents are docked, the control allocation is performed using the following linear program:

$$\min_u \sum_i u_i \text{s.t.} \tag{4.46}$$

$$\begin{bmatrix} u \\ -u \end{bmatrix} \geq \begin{bmatrix} 0 \\ -u_{max} \end{bmatrix} \tag{4.47}$$

$$Bu = f_{des} \tag{4.48}$$

where the B and u change for different assemblies as the number of control points change. This program minimizes the control effort exerted by all actuators subject to constraints on the maximum and minimum values.

4.5.1 Simulation Results

Simulations were performed in Matlab using CVX, a Matlab package for specifying convex programs, along with the Gurobi solver [63, 64, 68]. The simulation in Matlab uses SOCA to generate collision-free assembly trajectories in a J_2 perturbed spacecraft dynamical environment as described earlier in the chapter, then uses this method to allocate the control for the assembly as the various agents dock along their trajectories. A separate simulation was created using the maximum efforts required over a SOCA trajectory to determine which potential docking configurations were infeasible by examining the control allocation matrix at these configurations and the maximum control effort available to the thrusters.

A simple simulation was made using the maximum desired force and torque generated through an execution of the SOCA. This f_{des} was used to test three docking scenarios, agent to agent docking, agent to assembly docking, and assembly to assembly docking. The error in the allocation was determined as follows:

$$u_{err} = \|Bu - f_{des}\|_1 \tag{4.49}$$

Agent to Agent Docking

This test involved a rod agent and a connector agent docking to form an assembly. The resulting shape can be seen in Fig. 4.19a. A common-sense test was performed to see that each component of f_{des} was achieved with the correct combination of thrusters. This assembly was then used to allocate the maximum f_{des} over a SOCA trajectory as given above. The error in actuation is 2.89e-10 N with a total force of 0.138N using 16 thrusters of the available 52. One problem that is noted in using linear programs to allocate control is the inability to specify an impulse bit, the smallest thrust possible for the controller to produce. As a result, several agents are commanded to give very small thrusts, on the order of 1e-10. This is not achievable with the chosen thruster. To reduce these erroneous micro-fires, we will add the number of thrusters firing to the cost to be minimized. Nearly all of the control error is a result of these micro-fires.

Agent to Assembly Docking

The agent to assembly docking involves a connector agent docking with an assembly of a rod and a connector as seen in Fig. 4.19b. This and the more complex docks must be treated carefully, particularly in the calculation of $r_{2,CG}$, which becomes more complicated the more agents are involve. We circumvent this problem by calculating the value for each docking port upon assembly so that new assemblies can simply sum the vectors. For this assembly,

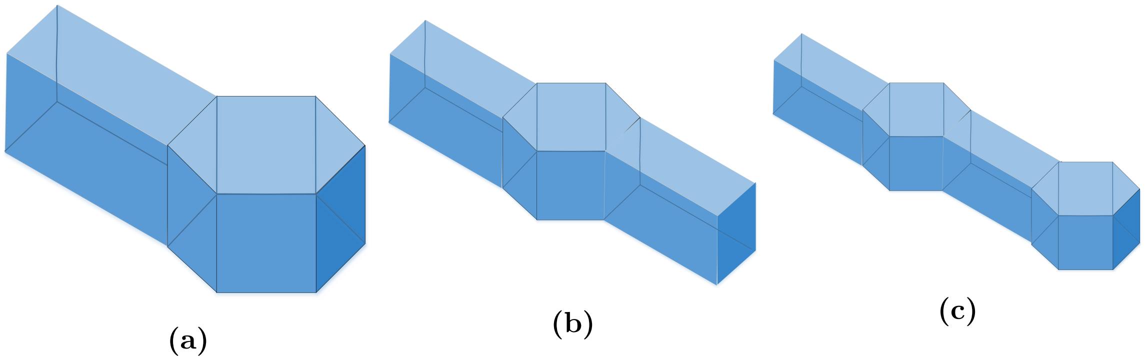


Figure 4.19: Description of the three basic control allocation tests run before incorporating the algorithm into SOCA

there are 96 thrusters with 16 blocked, giving 80 usable thrusters. The control allocation algorithm uses 22 thrusters a total force of 0.128N with a maximum error of 4.82e-10N.

Assembly to Assembly Docking

The assembly to assembly docking involves two assemblies of a rod and a connector as seen in Fig. 4.19c. For this assembly there are 120 thrusters with 24 blocked, leaving 96 thrusters. The maximum SOCA f_{des} is achieved using 40 thrusters at a total force of 0.129 N with a maximum error of 3.16e-10 N.

4.5.2 Incorporation Into SOCA

The model update and control allocation were incorporated into the SOCA to test the calculations over the course of a complete, 20 agent mission. Once agents enter within a predefined distance from each other, they determine if they will dock based on their assignment. If they are intended to dock, now they check the control authority of the assembly they would create. If it is sufficient, they dock. Otherwise, they avoid collision until a later time.

4.6 Chapter Summary

A distributed algorithm has been presented to allow for construction using a heterogeneous swarm of component satellites with limited communication radii. The agent types chosen can create a diverse set of final configurations which can cover the plane and build out-of-plane. This extends prior work in the field because it is both distributed and heterogeneous, can function in a complex dynamic environment, and accounts for relative attitude dynamics. The SOCA algorithm can correctly assign the heterogeneous agents for all target sets and avoid collision only where necessary. The algorithm was also made robust to uncertainty in the nominal trajectory. The handling of nonlinear dynamics was improved to make the trajectories commanded by SOCA realistic and achievable. An advanced control allocation scheme was presented to handle the changing shape parameters and actuator availability during self-assembly. The simulation results show SOCA performing assignment and trajectory generation for 20-54 agents in two and three dimensional final configurations with realistic trajectories. The algorithm is experimentally validated on 6 omni-wheeled robots and four spacecraft simulators. The wheeled robots were not sufficient to make claims about the success of the algorithm, but the spacecraft simulators were used to successfully experimentally validate the optimal construction algorithm. The proposed scheme is useful for missions ranging from sparse aperture interferometric telescope construction to space colony or station construction.

This algorithm can also be used for a higher degree of heterogeneity without substantially altering the algorithm, as discussed in the auction algorithm section. While adding more agents would expand the possible final shapes even further, it would also increase system manufacturing cost and does not affect the design or execution of the algorithm substantially.

Chapter 5

Autonomous Spacecraft Docking

This chapter investigates three types of docking for use in in-orbit self-assembly as shown in Fig. 5.1. The first half of the chapter focuses on comparing tether-based docking with traditional thruster-based methods in simulation. The second half is a more thorough treatment of an electromagnetic docking system as compared to thruster-based methods in simulation and experimentation on the M-STARs. The three docking schemes presented are capable of very low terminal velocity docking, which is ideal for in-orbit assembly applications. Ultra-soft docking is defined in this chapter as a dock where the relative velocity is less than 1 mm/s, soft docking has a relative velocity of less than 1 cm/s, and hard docking is above 1 cm/s. Both tether and electromagnet -based methods are capable of achieving docks without using propellant. The tether method required advanced control schemes to perform this dock without propellant and with an ultra-soft approach velocity.

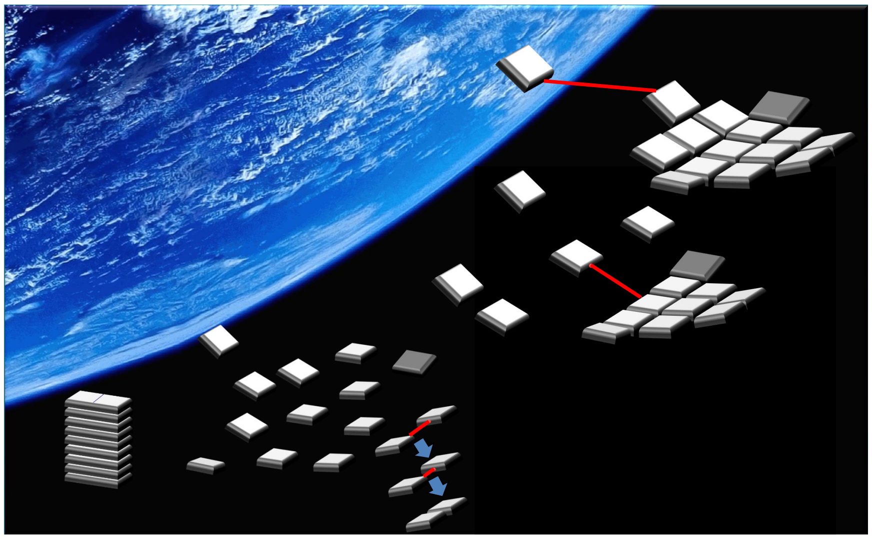


Figure 5.1: Concept Mission Application for Tethered Formation Flying-based AR&D

5.1 Tether-Based Autonomous Docking

5.1.1 Sample Mission Overview

For this mission, the chaser vehicle performs all the maneuvering for docking with a target vehicle located at the origin of the LVLH frame. Both the chaser and the target are 30 cm cubes with a mass of 10 kg, and are assumed to be launched in the same vehicle to a 500 km orbit, so the chaser vehicle begins within a 1 km sphere around the target due to delay in separation and drift. For the tether simulations, a child spacecraft/mass is attached by the tether to the chaser. The child is a 3 cm cube weighing 0.5 kg. The tether length is restricted to 10 m though longer tethers can be used. The tether length was chosen because of the multi-agent nature of the desired mission. With multiple satellites in close range or even docking simultaneously, long tethers are a liability. This system allows for easy satellite replacement or addition to the tether-connected system. If a satellite is being replaced, it releases the tethers holding it to the others, then moves away. The new agent then moves into position and the other spacecraft release tethers to dock with it.

Nanosatellite-scale thruster options are extremely limited, but the child spacecraft could make use of a modified version of the VACCO 0.14U thruster with 5.4 mN thrust and 70 second I_{sp} [4]. The same thruster system is available at multiple scales so these parameters are also used for the parent.

The mission begins with a 1 km separation between the two satellites. At this point the initial proximity maneuver begins and places the chaser into the desired berthing maneuver position. The berthing maneuver is then performed using either thrusters or the tether, with some docking mechanism making the final rigid connection. The satellites used will be referred to as target(T), parent(P) and child(C) in fig/tether.

5.1.2 Initial Proximity Maneuver

Traditional docking strategies focus on approaching the target using the radial or velocity directions, R-bar and V-bar respectively, or using the glideslope algorithm which merges the two while maintaining a line of sight to the target. The glideslope algorithm is typically preferred because it achieves good terminal relative velocities while staying within line of sight of the docking port, however it does perform several unoptimized burns so it can be quite expensive. Optimized versions of the glideslope algorithm exist, and achieve a great reduction in fuel cost, but these methods can have worse performance on other docking constraints like a higher terminal velocity [12]. In this chapter we will use the glideslope algorithm as proposed by Hablani et al. to maintain vehicle safety through a low terminal velocity [69]. As a comparison, we will briefly show optimal trajectory results generated with Model Predictive Control Using Sequential Convex Programming (MPC-SCP) [100, 54].

Glideslope Trajectory

The glideslope algorithm is based on the Hill-Clohessy-Wiltshire (HCW) equations shown in Eq. 5.1, which are a set of dynamic equations describing the relative motion of satellites in a circular orbit near to each other in the Local Vertical/Local Horizontal (LVLH) frame. The HCW equations depend only on initial state, orbit angular frequency ω, and time and are only accurate within about 30 km of the origin of the frame due to the linearization [51]. The definition for the LVLH frame can be found in Fig. 5.2. Eq. 5.4 shows a simplified notation for Eq. 5.1.

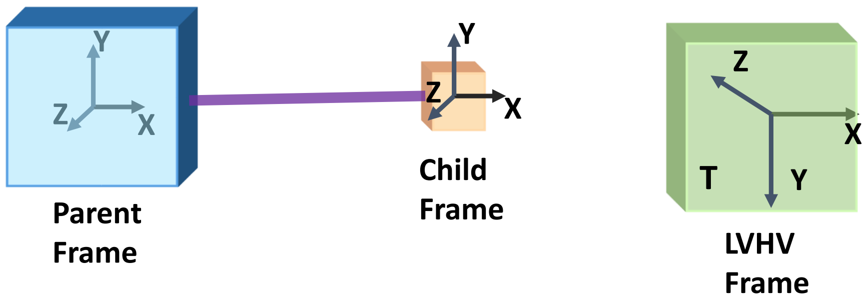


Figure 5.2: The frame conventions for the child and parent spacecraft

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} z(t)\dot{x}(t)\dot{y}(t)\dot{z}(t) = \tag{5.1}$$

$$\begin{bmatrix} 1 & 0 & 6(\omega t - \sin(\omega t)) & \frac{4}{\omega}\sin(\omega t) - 3t & 0 & \frac{2}{\omega}(1-\cos(\omega t)) \\ 0 & \cos(\omega t) & 0 & 0 & \frac{\sin(\omega t)}{\omega} & 0 \\ 0 & 0 & 4 - 3\cos(\omega t) & -\frac{2}{\omega}(1-\cos(\omega t)) & 0 & \frac{\sin(\omega t)}{\omega} \\ 0 & 0 & 6\omega(1-\cos(\omega t)) & -3 + 4\cos(\omega t) & 0 & 2\sin(\omega t) \\ 0 & -\omega\sin(\omega t) & 0 & 0 & \cos(\omega t) & 0 \\ 0 & 0 & 3\omega\sin(\omega t) & -2\sin(\omega t) & 0 & \cos(\omega t) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} z_0 \times \tag{5.2}$$

$$\dot{x}_0\dot{y}_0\dot{z}_0 \tag{5.3}$$

$$\left[\begin{array}{c} r(t) \\ \hline v(t) \end{array}\right] = \left[\begin{array}{c|c} \Phi_{rr}(t) & \Phi_{rv}(t) \\ \hline \Phi_{vr}(t) & \Phi_{vv}(t) \end{array}\right] \left[\begin{array}{c} r_0 \\ \hline v_0 \end{array}\right] \tag{5.4}$$

The goal of the glideslope algorithm is to approximate a straight-line path, ρ, to the target using a chosen number of impulsive burns while satisfying line-of-sight constraints and minimizing burns near the target to reduce plume impingement. This version of the algorithm can find a trajectory even for initial and terminal positions in different planes. Thrusts performed are assumed to be impulsive and the range rate is assumed to be linearly related to the range, though this is nonlinear for close maneuvers.

First we find the vector u_0 as a function of the position at $t = 0$ and $t = T$, the terminal time.

$$u_0 = \frac{1}{\rho_0} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - \begin{bmatrix} x_T \\ y_T \\ z_T \end{bmatrix} \tag{5.5}$$

The XYZ position along the desired vector can be found by multiplying u_0 by the range over time. To find the range over time, we assume the range rate $\dot{\rho}$ is linearly related to the range with slope m, making the range rate equation:

$$\dot{\rho} = m\rho + \dot{\rho}_T \tag{5.6}$$

where $\dot{\rho}_T$ is the desired arrival speed and m can be found using this and the initial speed.

$$m = \frac{\dot{\rho}_0 - \dot{\rho}_T}{\rho_0} \tag{5.7}$$

Solving these equations yields an equation for range over time:

$$\rho(t) = \rho_0 e^{mt} + \left(\frac{\dot{\rho}_T}{m}\right)(e^{mt} - 1) \tag{5.8}$$

with total transfer time, T:

$$T = \frac{1}{m} \ln\left(\frac{\dot{\rho}_T}{\dot{\rho}_0}\right) \tag{5.9}$$

To ensure safety, the arrival speed is chosen to be much less than the initial speed. The actual glideslope trajectory is found by performing a user-selected N burns with initial and final points along ρ. The burns are separated by $\Delta t = \frac{T}{N}$ seconds with each burn time denoted as t_b. The range of each burn $\rho_b = \rho(t_b)$ is used to calculate the XYZ position of each burn, r_b:

$$r_b = r_T + \rho_b u_0 \tag{5.10}$$

The ΔV expended along this trajectory can be found by leveraging the HCW equations to find the change in velocity needed to get from one r_b to the next. The $+$ superscript denotes velocity after a burn and the $-$ superscript denotes velocity before a burn.

$$v_b^+ = \Phi_{rv}^{-1}(\Delta t)(r_{b+1} - \Phi_{rr}(\Delta t) r_b) \tag{5.11}$$

$$v_{b+1}^- = \Phi_{vr}(\Delta t) r_b + \Phi_{vv}(\Delta t) v_b^+ \tag{5.12}$$

The total ΔV is then found by:

$$\Delta V = \sum_{b=1}^{N} \Phi_{rv}^{-1}(\Delta t)(r_{b+1} - \Phi_{rr}(\Delta t) r_b) - \Phi_{vr}(\Delta t) r_{b-1} + \Phi_{vv}(\Delta t) v_{b-1}^{+} \tag{5.13}$$

where v_0^+ is simply v_0 and the position and velocity at any point along this trajectory are found by concatenating the position and velocities found using the HCW equations for each thruster firing:

$$\left[\begin{array}{c} r_b(t-t_b) \\ \hline v_b(t-t_b) \end{array}\right] = \left[\begin{array}{c|c} \Phi_{rr}(t-t_b) & \Phi_{rv}(t-t_b) \\ \hline \Phi_{vr}(t-t_b) & \Phi_{vv}(t-t_b) \end{array}\right] \left[\begin{array}{c} r_{b-1} \\ \hline v_{b-1}^{+} \end{array}\right] \tag{5.14}$$

The free variables in this algorithm are $\dot{\rho}_0$, $\dot{\rho}_T$, N, and x_f assuming the initial position and velocity are randomly generated within the aforementioned bounds. More thruster firings leads to a lower ΔV, but increases the probability of a thruster-related failure, like a stuck open valve or over/under actuation. For this simulation, we use an initial relative speed of 30 m/s and a desired terminal relative speed of 0.1 m/s, with 3 thruster firings. The final position is 10 m in the -X direction from the target spacecraft. The result of the glideslope algorithm is shown in Fig. 5.3. The maneuver had a total ΔV of 38.9 m/s and a total elapsed time of 188.8 seconds. The final velocity overshot the desired final speed at $[0.30, 0, -0.02]$ m/s. This can either be cleaned up with another burn at the final position or rolled in to the next step of the process. Here we elect to use this residual velocity as the initial velocity for the next phase. Using the Tsiolkovsky rocket equation [43], we can determine the propellant mass required for this maneuver:

$$M_p = M_0(e^{\frac{\Delta V}{I_{sp} g_0}} - 1) \tag{5.15}$$

For the thruster parameters and spacecraft parameters chosen above, the required fuel mass is 612 g. This seems small, but this is already more than half of the propellant available to the largest thruster of the chosen type and it does not account for drift corrections, station-keeping, momentum dumps, etc.

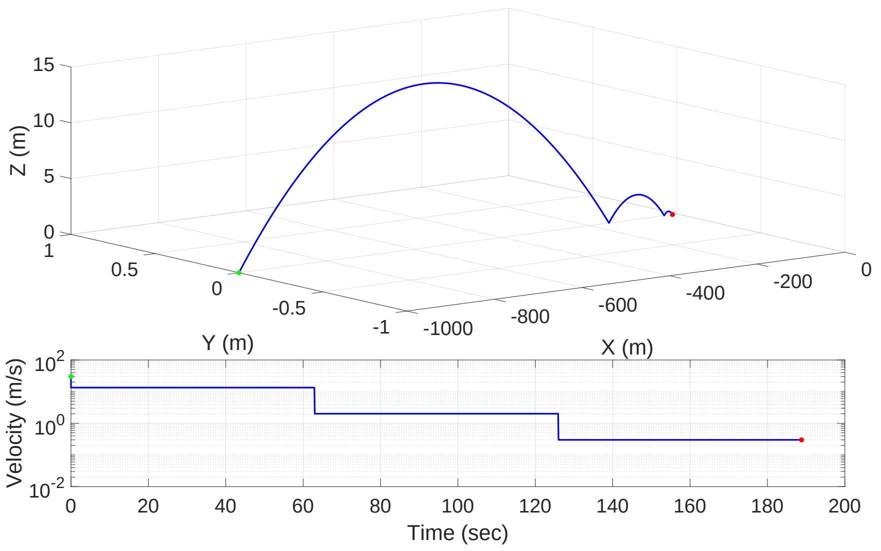


Figure 5.3: The glideslope approach trajectory for the parent spacecraft approaching the berthing position

Optimal Trajectory

Model Predictive Control using Sequential Convex Programming performs a series of optimizations based on linearized dynamics and convexified constraints. The series of optimizations is proven to converge on the optimal solution for a properly stated problem. More details on this trajectory generation method can be found in the cited references [100, 101]. One drawback of MPC-SCP is the loss of precision due to the linearization. This was solved by adding the nonlinear correction step between optimization iterations to match the nonlinear dynamics [54]. The pseudocode for a single-agent version of the algorithm is presented below.

Method 7 Model Predictive Control using Sequential Convex Programming

```
1:  $\bar{\mathbf{x}}[k] := \mathbf{0}_{6\times 1},\ \forall k$
2:  $\mathbf{x}_0[k] :=$ reference trajectory $\forall k$
3:  $\bar{\mathbf{x}}[k] := \mathbf{x}^0[k],\ \forall k$
4:  $w := 1$
5:  while Not Converged do
6:      $\mathbf{x}_w^{nom}[k]$ :=the solution to convex problem, $\forall k$
7:      $\mathbf{x}_w[k] := \mathbf{f}_k(\mathbf{x}_w^{nom}[k], \mathbf{u}_w^{nom}[k])\quad \forall k$ (Nonlinear Correction)
8:      $\bar{\mathbf{x}}[k] := \mathbf{x}_w[k],\ \forall k$
9:      if $\|\mathbf{x}_w[k] - \mathbf{x}_{w-1}[k]\|_\infty < \epsilon_{\text{SCP}}\ \forall k$ then
10:         Converged, break
11:     end if
12:     $w := w + 1$
13: end while
```

To generate this berthing trajectory, the convex problem can be defined as minimizing the fuel consumption subject to the J_2 perturbed relative dynamics as stated in Eq. 5.1, along with constraints on the initial and final position and velocity, maximum control effort, and maximum velocity. The J_2 perturbations cause nonlinear dynamics for both the target and the parent. The resulting trajectory is shown in Fig. 5.4 with the velocities and controls in Fig. 5.5. The total impulse over the trajectory is 67.6 mN-s, which corresponds to a fuel consumption of 98.44 μg using the thruster parameters defined above.

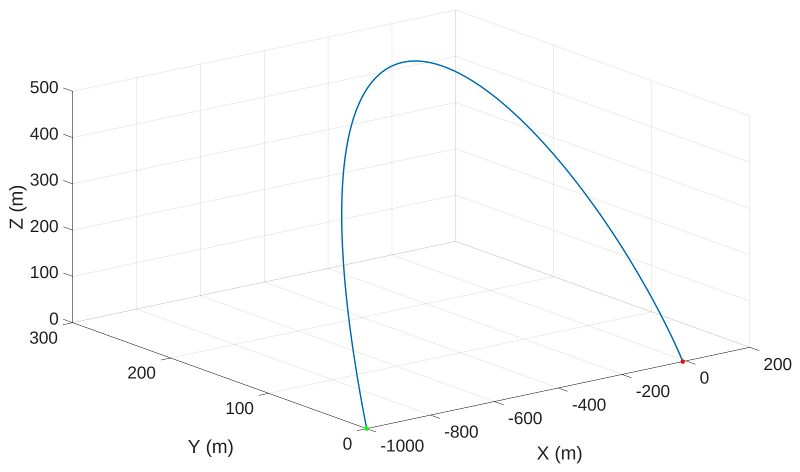


Figure 5.4: Optimal Approach Trajectory for the Parent Spacecraft

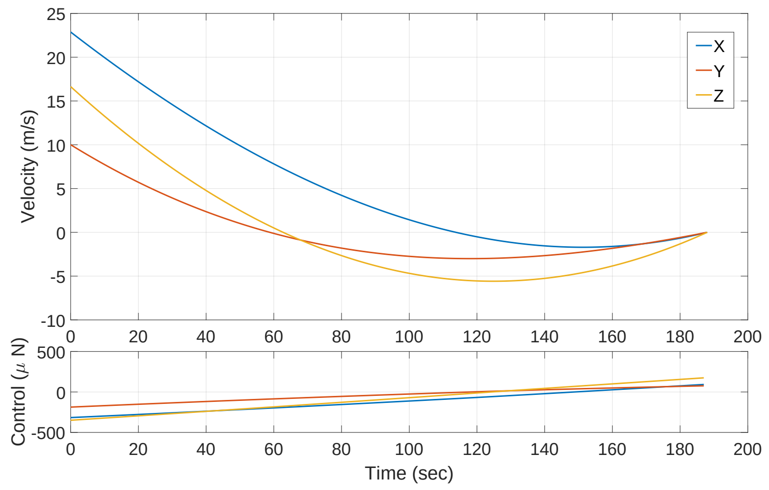


Figure 5.5: Velocity and Thrust over Time for the Optimal Approach Maneuver

5.1.3 Berthing Maneuver

The target and the parent are now separated by 10 m in the -X direction, with the parent behind the target. The final approach phase is modeled as propulsive, child-based tethered, and spin-based tethered docking.

Thruster-based Docking

The thruster-based docking maneuver is done using the same glideslope algorithm as the initial proximity maneuver. For this version, the initial relative speed was 0.3 m/s and the desired final speed was 0.1 mm/s, using 3 burns. The glideslope trajectory can be seen in Fig. 5.6

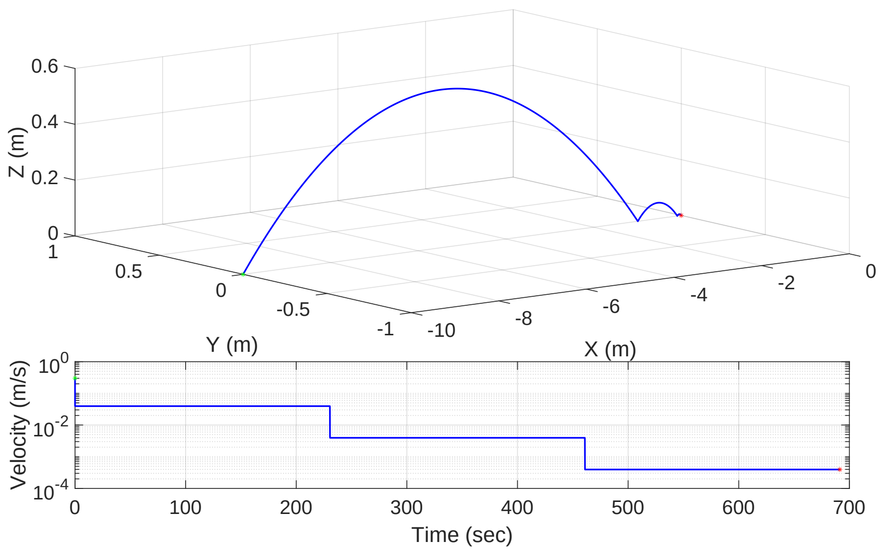


Figure 5.6: The glideslope approach trajectory for the child spacecraft approaching the target spacecraft's position

The maneuver had a total ΔV of 0.30 m/s and a total elapsed time of 266.4 seconds. The final velocity was $[0.5, 0, -0.05]$ mm/s which overshot the desired final speed again. This is why the desired final speed was set so low, this overshoot cannot be cleaned up with a burn at the final time. The last burn performed occurs only 4 cm away from the target spacecraft which is highly undesirable but the speed at this location is still 7 mm/s. Depending on the capabilities of the capture mechanism, this speed may be acceptable in which case the final burn would be canceled. The fuel mass for this maneuver is 4.4 g.

The optimal maneuver is found using MPC-SCP for the same input parameters as the glideslope algorithm. The resulting trajectory can be seen in Fig. 5.7. The velocity profile and the control effort can be found in Fig. 5.8. The total control impulse over the whole maneuver is 0.89 mN-s. For this small a thrust requirement, an electric propulsion device would be more suitable, but several are available at this form-factor. There are even some flight-proven CubeSat electric propulsion technologies, like the Busek pulsed plasma thrusters [5]. Using the thruster parameters already specified though, this leads to a propellant consumption of approximately 2.7 μg. This type of trajectory involves burning continuously and since the relative velocities are so low (on the order of mm/s within 20 cm of the target) it may possible to neglect the final burns.

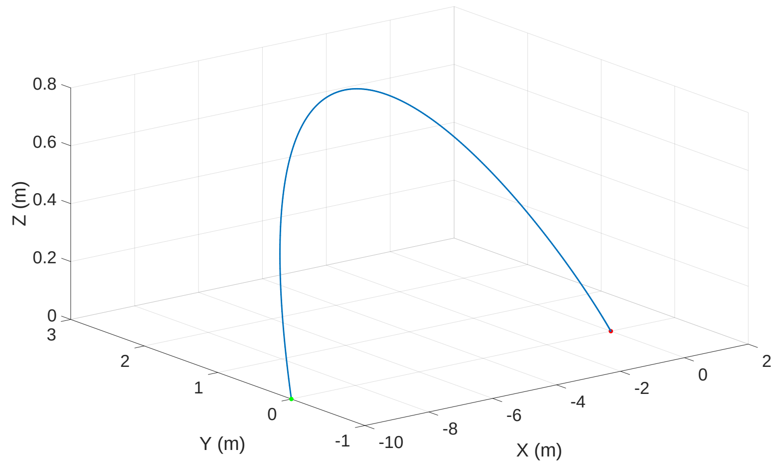


Figure 5.7: Optimal Trajectory for the Parent Spacecraft

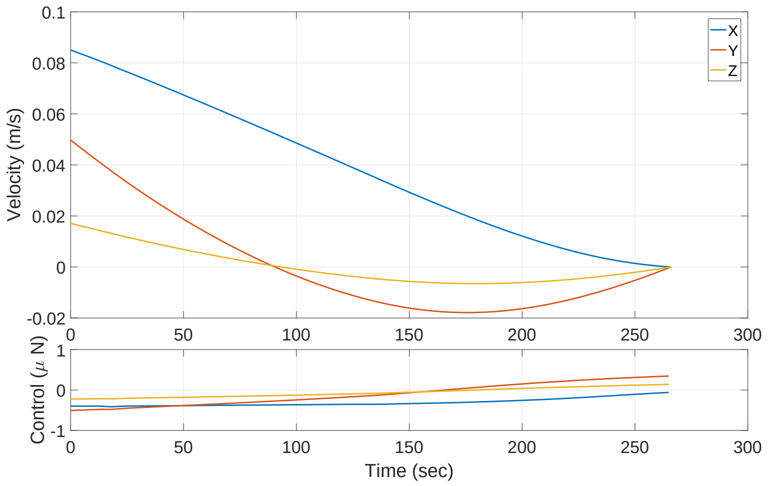


Figure 5.8: Velocity and Thrust over Time for the Optimal Maneuver

Tethered Docking

When AR&D begins, the parent spacecraft expels the child spacecraft connected to the parent via the tether. The child spacecraft has a varying amount of position and attitude control, subject to study. Two capture strategies are considered for this scheme, one where the child spacecraft has full maneuverability and one where the child spacecraft is unactuated. These strategies are illustrated in Fig. 5.9. The complexity of the child and target satellites are highly dependent on which capture strategy is chosen. A thruster-based approach adds most of the complexity to the child spacecraft, requiring thrusters and attitude control, while a spin-up approach depends upon the development of a robust, reusable catching mechanism for the target. The software model of these strategies was generated using the Euler-Lagrange formulation of the system. To simplify the modeling, the tether is assumed to be in tension for the duration of the mission.

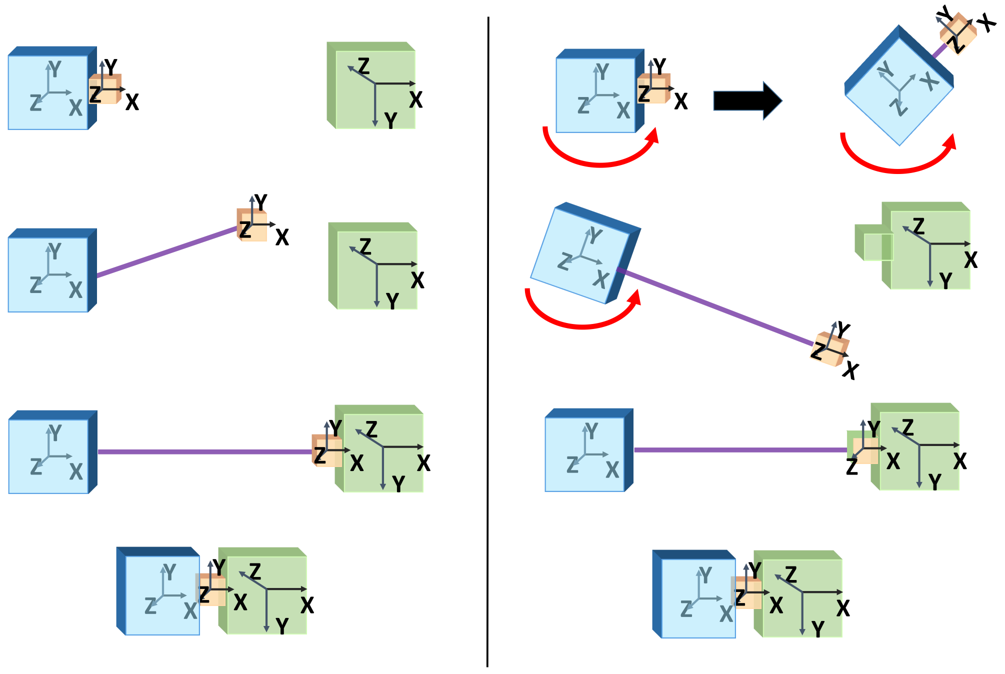


Figure 5.9: Left: Free-flying child tether-based capture and docking scenario, Right: Spin-up released tether-based capture and docking scenario

The model has three translational degrees of freedom on the parent and a rotational degree of freedom about the body Z axis. The parent and child are connected by a variable-length, massless tether with a rotational degree of freedom at the root. The dynamics are presented in Appendix A. This planar model is sufficient to capture most child motion since during spin-up, the out-of-plane angle is decoupled from the in-plane motion. This model results in complex pendulum behavior which is excited by the contraction of the tether during reel-in maneuvers. This model can be leveraged to achieve improved intersatellite position knowledge through the use of a force torque sensor at the root of the tether. This sensor would allow the parent to know the precise distance and angle to the target once docked [34]. When the child spacecraft collides and attaches with the target spacecraft the change in linear momentum is significantly smaller than that of traditional docking, because the child spacecraft is much lower mass than the target spacecraft. This scheme also reduces the risk of mission failure due to docking because the parent spacecraft could have several sets of tethers and masses in case of failure.

Tethered Child Spacecraft The actuated scheme shown in Fig. 5.9 uses thrusters on the child spacecraft to move it to the desired location on the target spacecraft, with the tether unreeling speed equaling the child spacecraft speed to keep the tether taut. The child spacecraft can follow the same glideslope trajectory as the thruster-based docking parent spacecraft, though the reduced mass affects the required propellant mass. The required propellant mass for the child spacecraft is 0.22 g. Following the MPC-SCP optimal trajectory, the required propellant mass is 0.135 μg. After the child has docked with the target, the parent spacecraft begins reeling in the tether.

Spin-Released Tether In the unactuated scheme (Fig 5.9), the parent spacecraft spins up which deploys the tether using centrifugal force, then the child spacecraft is captured using a retractable arm on the target. This strategy simplifies the child and parent spacecrafts because the child spacecraft does not need propulsion, but it complicates the target spacecraft because it requires a retractable arm to capture the child spacecraft. Once the spacecraft are connected with the tether, the tether attributes can be leveraged to aide in relative positioning.

CubeSat attitude determination and control is actually fairly advanced. The Blue Canyon Technologies XACT system claims attitude control within 0.003° [1, 132]. XACT can also sustain a slew rate of 10°/s. This slew rate was used to saturate the controller during spin-up. The controller used here is a simple proportional controller acting on the torque about the parent body Z axis.

$$\tau_Z = K_{p_L}(L - L_F) - K_{p_\phi} * (\phi - \phi_F) \tag{5.16}$$

A PD controller is used to stabilize the motion of the parent. The results of this simulation is shown in Fig. 5.10, with the control inputs shown in Fig. 5.11

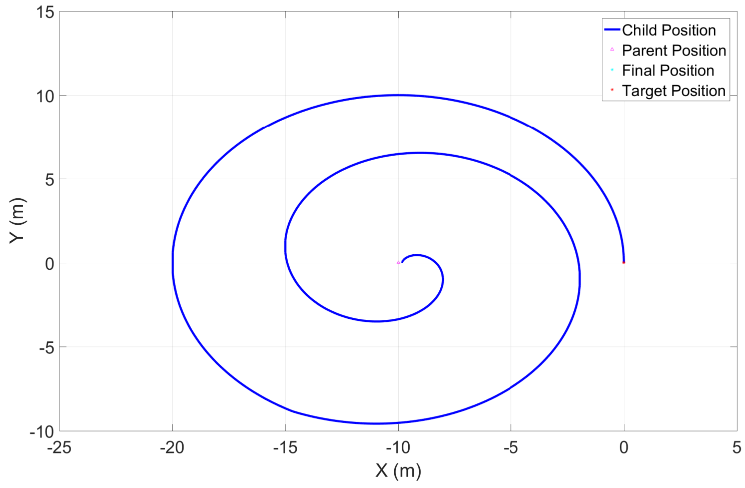


Figure 5.10: Parent and Child Spacecraft Trajectories for Spin-Up Based Dock

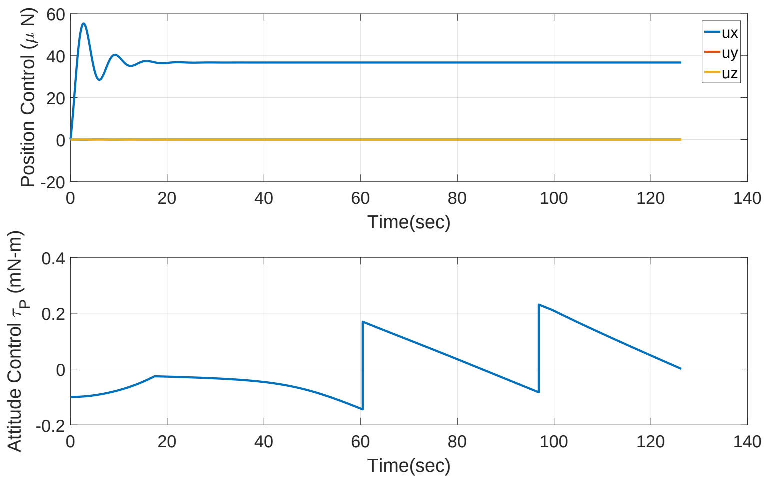


Figure 5.11: Control Inputs for the Spin-Up Based Dock

The spin-up maneuver takes 126.3 seconds, during which time the parent must actively thrust to maintain its position against the spacecraft dynamics and tether reaction forces. The child can achieve docking with the target very precisely (within 1 mm), as seen in the figure. The angular velocity at the time of docking is around 0.09 rad/s which corresponds to a linear speed at the child spacecraft of about 0.9 m/s. This is unacceptably high. Longer docking times and more precise control can bring this speed down, but the impact on the catching mechanism will likely still cause a significant torquing perturbation on the target.

Tether Reeling

Once the target and parent are connected by a tether, the tether motor in the parent spacecraft can reel in the target spacecraft without the use of propellant. This causes complex pendulum modes to form in the dumbbell-type system, but these modes can be controlled using only the reaction wheel assembly in both agents. Controlling these modes does require the cooperation of the target spacecraft [35, 39]. This reeling maneuver uses the same model as the spin-up maneuver, except here the controller applies a stabilizing torque and a reeling force:

$$\tau_Z = K_{p_\phi}(\phi - \phi_F) \tag{5.17}$$

$$u_L = K_{p_L}(L_F - L) \tag{5.18}$$

The reeling maneuver shown in Fig. 5.12 took 158 seconds with negligible final velocity and a maximum reel velocity of approximately 4.3 cm/s. The control applied is shown in Fig. 5.13.

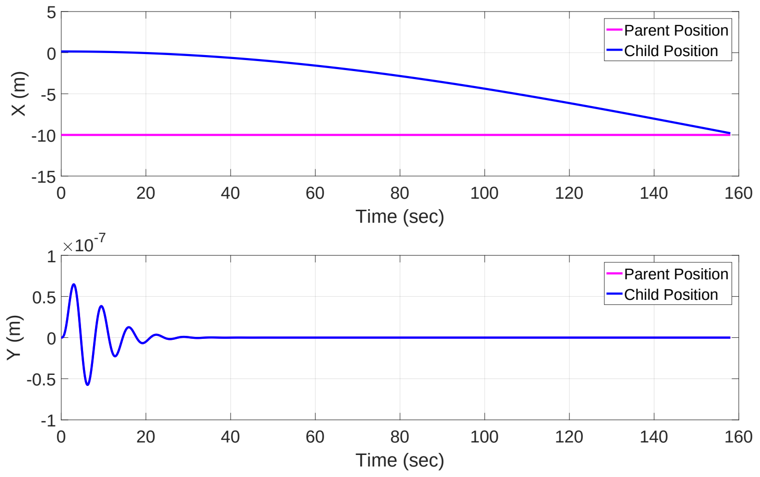


Figure 5.12: Parent and Child Spacecraft X and Y Positions for Reeling Maneuver

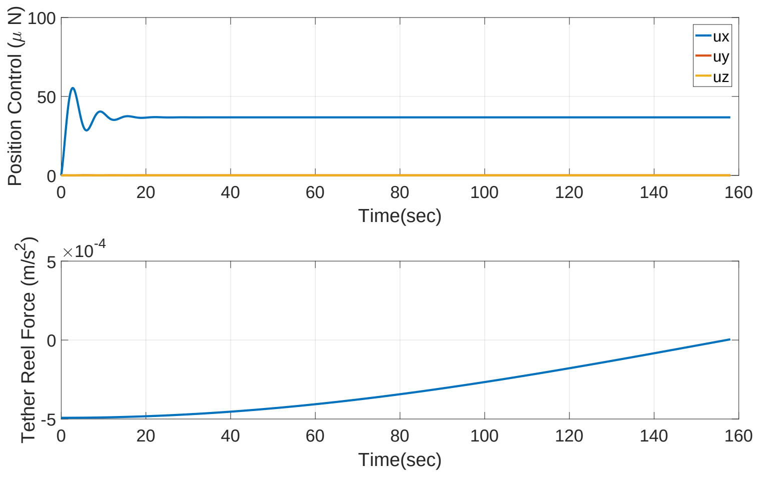


Figure 5.13: Control Inputs for the Reeling Maneuver

5.2 Electromagnetic and Thruster-based Docking

5.2.1 Electromagnet-based Docking

The electromagnet-based system is shown in Fig. 5.14 and is mounted on the lower stage of the spacecraft simulator. The main components of the electromagnet-based docking system are: electromagnets, docking ports, reaction wheels, gyroscopes, batteries, and driving electronics.

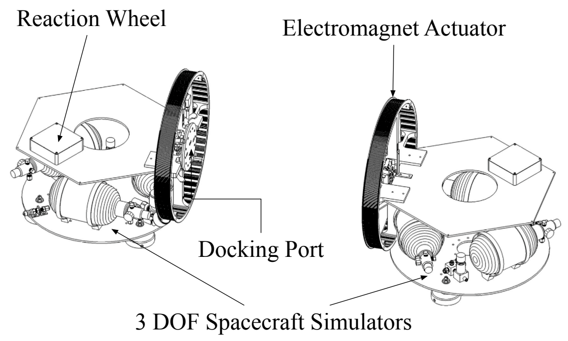


Figure 5.14: Two 3DOF M-STARs Equipped with Electromagnets, Docking Ports, and Reaction Wheels

Electromagnet Design

The electromagnets are designed to provide a force of $f_{lim} = 40$ mN at 40 cm coil-to-coil distance and take up less than 20% of the entire mass of the spacecraft simulator including the structure and docking port. The number of windings, coil radius, coil thickness, and maximum current specifications are calculated using a Sequential Least Squares Programming optimization constrained to meet the above requirements while minimizing mass and power consumption.

$$\begin{aligned} \min_{N,l,R,i} \quad & c_1 p(N,l,R,i) + c_2 m(N,l,R) \\ \text{s.t.} \quad & f = f_{lim} \\ & m \leq m_{max} \\ & p \leq p_{max} \end{aligned}$$

The design of the electromagnet can be seen in Fig. 5.15. The coils were mounted on a 3D printed frame with cutouts to increase airflow and prevent overheating, which causes an increase in the resistance of the wire.

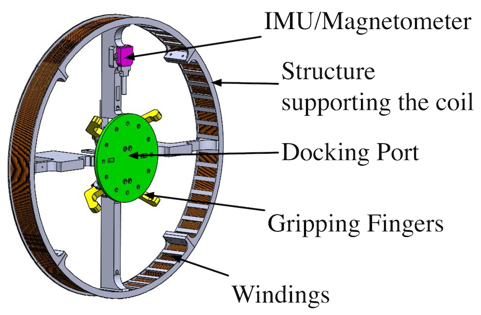


Figure 5.15: Electromagnet Design

The electromagnet-based docking system runs a real-time current controller, shown in Fig. 5.16, to compensate for the induced voltage generated when the two spacecraft move with respect to each other. The feedback controller runs on the microcontroller at 1 kHz. The controller takes a current setpoint and outputs a drive voltage for the electromagnet, using feedback from a precise current sensor based on a low-tolerance shunt resistor.

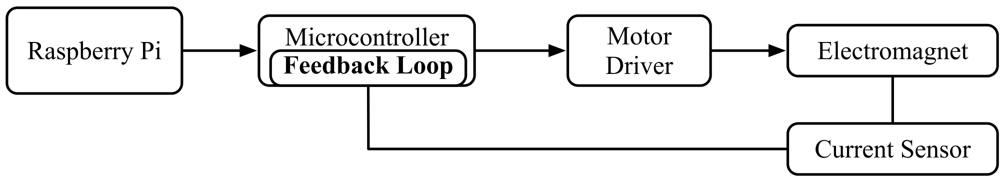


Figure 5.16: Electromagnet Current Controller

5.2.2 Thruster-based Docking

Thruster-based docking on the spacecraft simulators uses two 3DOF M-STARs, configured with a tube connecting the upper thruster stage with the lower stage. In Fig. 5.17, two docked

3DOF thruster M-STARs are shown. Each M-STAR has eight on-off solenoidal thrusters connected through a regulator to three compressed air cylinders, driving electronics, and four docking ports, presented in subsection 5.2.3, which are mounted on each side of spacecraft simulator upper stages. A more detailed description of the thruster system on-board the M-STARs can be seen in Ref. [105].

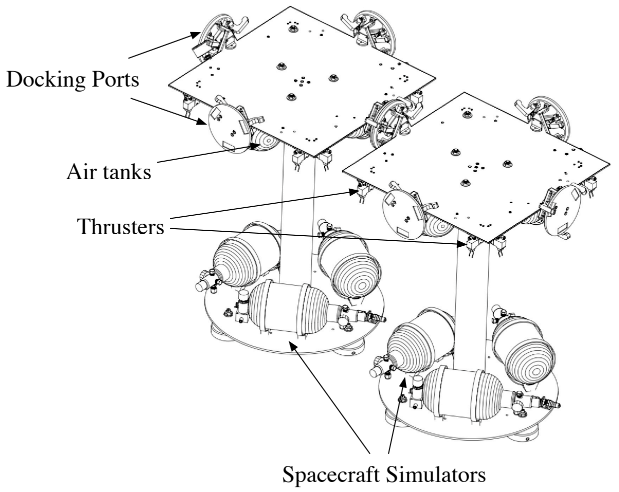


Figure 5.17: Two M-STARs With Thrusters and Four Docking Ports

5.2.3 Docking Port Design

Once the spacecraft are moved into proximity by the electromagnets, a gripping mechanism has to ridigize the system and allow the magnets to be turned off. The gripping mechanism is designed to be androgynous, so any spacecraft can dock to any other, and to lock so that no torque is needed on the motor to keep the gripper in closed position.

The design of the docking port can be seen in Fig. 5.18. The servomotor in the center of the gripper connects to the fingers using push rods, which lock into place at the closed configuration so that once the gripper is closed, the motor can be turned off. Additionally, a high friction surface is added to the docking port's plate to constrain the rotational motion.

The reaction wheel should be able compensate for any heading error that happens between the spacecraft while docking, but the gripper can accommodate for misalignment as

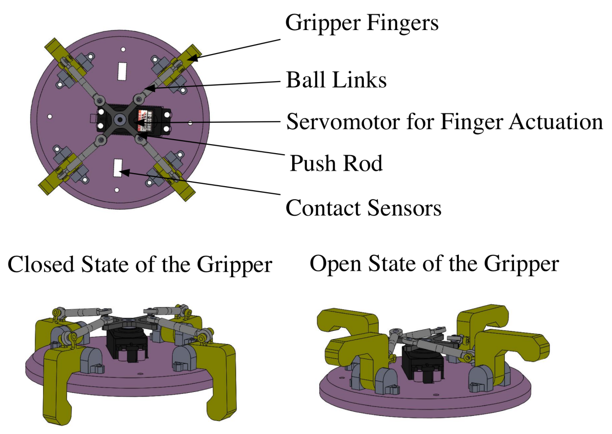

Gripper Fingers
Ball Links
Servomotor for Finger Actuation
Push Rod
Contact Sensors
Closed State of the Gripper
Open State of the Gripper

Figure 5.18: Kinematics of the Gripper Mechanism

well. The misalignment tolerance was measured experimentally by putting the two spacecraft at a certain separation and measuring the offset before gripping. The maximum distance between the two grippers where the dock succeeds is 2 mm. The maximum lateral misalignment between the two grippers is 12 mm. These values can be improved by changing the finger shape but are sufficient for this work.

5.3 Control

For the feedback controllers on both docking systems, the pose and the angular velocity for each spacecraft were determined by the motion capture system in the flat floor facility and the on-board gyroscope. The dock finalization was also the same for both systems, with the gripper closing when the two satellites are touching. Both systems relied on detailed models derived mathematically and experimentally. For more details, see [55].

5.3.1 Trajectory Planner

The trajectory that was input into both controllers is a constant acceleration profile. For electromagnet-based docking, an acceleration of 0.3 mm/s was input, while the velocity was limited to 45 mm/s. For thruster-based docking, the acceleration is 1 mm/s, while the velocity limit is infinity.

5.3.2 Electromagnet-Based Docking

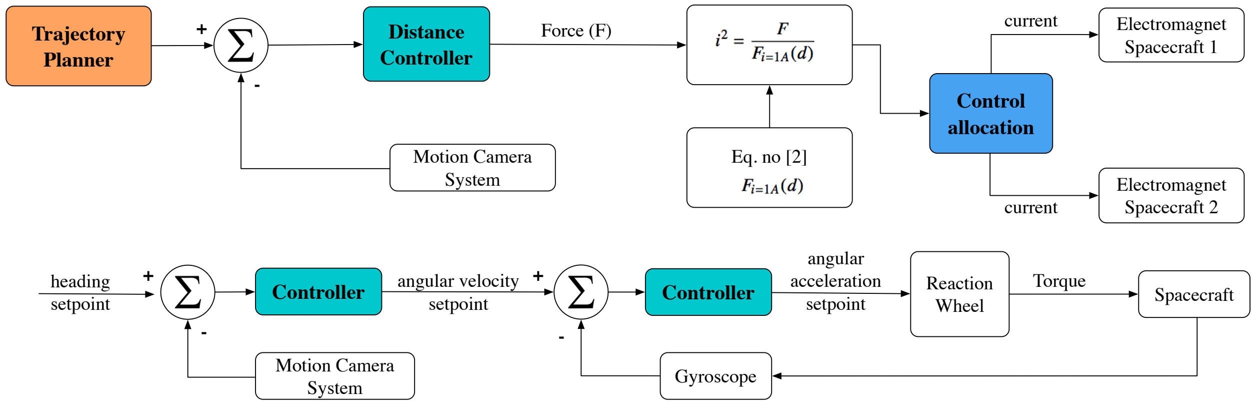


Figure 5.19: Distance Controller for Electromagnet-based Docking System

The electromagnet-based docking involves three controllers (Fig. 5.19). For attitude control, two PIDs in cascade are used, one that takes as input the heading and outputs an angular velocity, that is then fed into the second controller. For the distance control, a single PID was implemented. The input of the controller is the intersatellite distance. However, due to the non-linearity between the input and the output (the current in the electromagnets), the gain of the PI controller will have drastically different effect at different distances. Thus, the controller determines the required force, which is used as a virtual input for an inverse input nonlinear function which determines the current that is input into the electromagnets. The poses of the two spacecraft simulators obtained from the motion capture system were used to determine the intersatellite distance and the heading error. The distance controller uses the distance between the electromagnets as a setpoint and controls the electromagnet forces.

The required currents for the two electromagnets are computed from the force output by the controller. The force at any distance is proportional to the product of the current (i) of the two coils. Therefore, using Eq. (5.19), the required current product can be computed from the force given for 1 Ampere ($F_{i=1A}$) computed using the derived electromagnet model.

$$i_1 i_2 = \frac{F}{F_{i=1A}(d)} \tag{5.19}$$

The current is allocated between the two electromagnets using the following equations:

$$i_1 = \sqrt{|i_1 i_2|} \tag{5.20}$$

$$i_2 = sign(i_1 i_2)\sqrt{i_1 i_2} \tag{5.21}$$

One of the spacecraft was chosen to always have a positive current, while the second one was chosen to have the sign of the current product. In practical applications it may be preferable to have a control allocation scheme that trades between the two to prevent overheating.

5.3.3 Thruster-Based Docking

The thruster controller uses a PID in cascade with velocity feed-forward for position control and another PID for heading control. The controller determines the desired forces and moment that is then allocated to each individual thruster.

The forces and moment in body frame can be expressed as the follows:

$$\begin{bmatrix} f_x^B \\ f_y^B \\ m_z^B \end{bmatrix} = B\mathbf{F} \tag{5.22}$$

where $\mathbf{F}$ is the thrust vector and B is the control influence matrix defined as:

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.23}$$

The thruster control allocation is determined using the Moore-Penrose pseudoinverse of the control influence matrix.

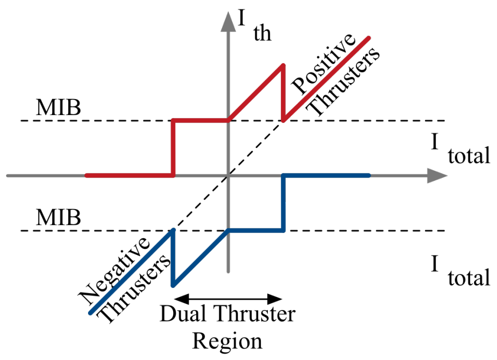


Figure 5.20: Thruster Allocation Based on MIB

For fine velocity and position control where the total desired impulse for the spacecraft is less than the MIB allows, two counteracting thrusters are allocated. The allocation scheme for each of the pair of counteracting thrusters can be seen in Fig. 5.20. This function is designed in such a way that each thruster has either 0 or more than its minimum impulse.

In this manner, very small net forces can be applied on the spacecraft for precise docking. This solution is expensive in terms of propellant because two counteracting thrusters need to fire in the same time, thus it is only done for close-approach maneuvers when distance between the spacecraft simulators is less than 20 cm.

5.4 Results

Experimental validation of the two docking systems was performed in the flat floor facility at Caltech. Due to the design of the facility and the M-STARs, friction between the floor and the linear air bearings is small.

5.4.1 Electromagnet-Based Docking

Tests for the electromagnet-based docking system were performed by placing the two simulators a set distance apart, then starting the position and heading controllers. Several tests were performed by keeping the distance constant at approximately 40 cm apart. Afterwards, the initial intersatellite distance was increased in each test until the electromagnets could no longer attract enough to overcome the static friction of the floor. The maximum intersatellite distance was found to be 55 cm. The results at 55 cm distance are discussed below.

The real trajectory of the two simulators is shown in Fig. 5.21, with the heading angle shown as the orientation of miniaturized hexagons at various points along the trajectory. The final orientation and position of the two simulators is shown by the life-sized hexagons.

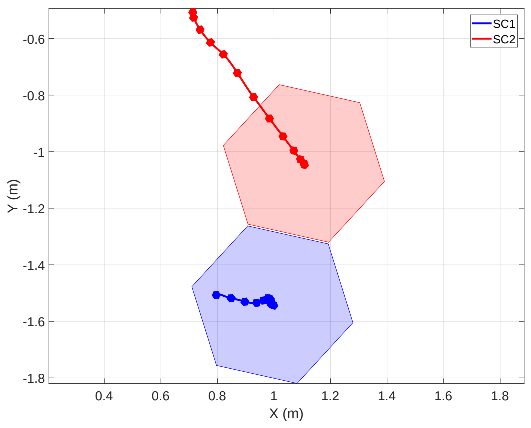


Figure 5.21: Electromagnet-based Docking Trajectory with Orientation Represented using Hexagons

It is clear to see that the electromagnets have quite a bit of drift in heading, which could be caused by imbalances in the simulator mass or issues with the reaction wheel like vibrations. The system is still able to maintain relative alignment though, so the docking is still successful.

The control input is the current in each of the electromagnets, seen in Fig. 5.22. Note that because this experiment has the highest possible intersatellite distance, the current is saturated to 4.5A which is the maximum the electromagnets can reach with the power supply used in this experiment (a 5-cell lithium polymer battery). The current can be increased by using more power, wire with slightly less resistance than copper such as silver wire, or superconductors.

The spacecraft point towards each other using the custom reaction wheels. In Fig. 5.23,

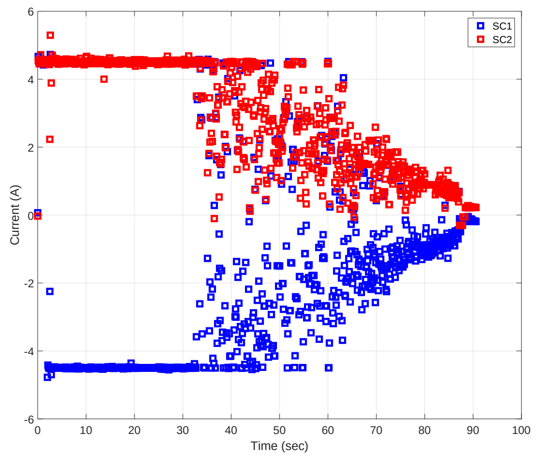


Figure 5.22: Electromagnet Current Commanded vs. Time

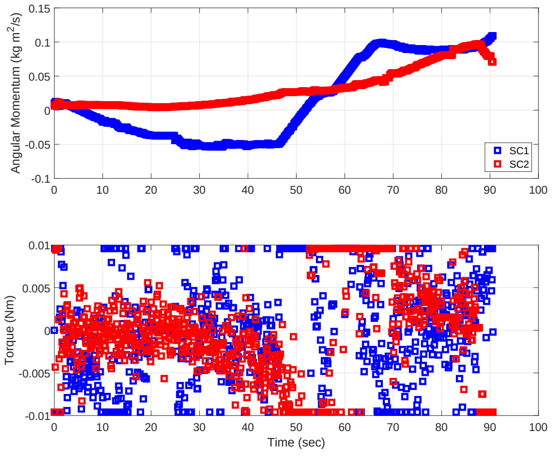


Figure 5.23: Angular Momentum and Torque of the Reaction Wheel vs Time

the angular momentum and torque of the reaction wheels are shown, while in Fig. 5.24, the heading of the two spacecraft is shown. The reaction wheel is saturating at 5000 rpm in this case.

The intersatellite distance tracking accuracy of the electromagnet-based docking system is shown in Fig. 5.25, which displays the standard deviation of the distance tracking error over time during the 14 test runs. It is seen that the controller is robust to position errors, as long as they are inside the attraction envelope of the electromagnet. The maximum standard deviation of the tracking error is 2.59 cm.

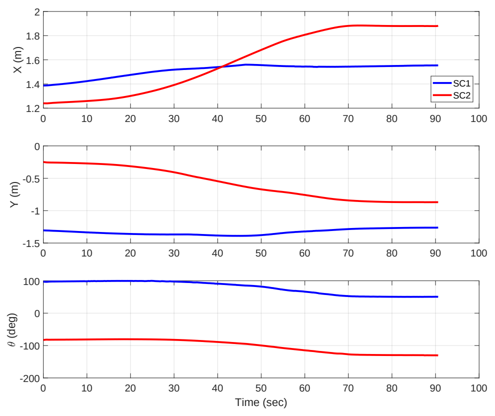


Figure 5.24: X Y Position and Heading of the Two M-STARs vs Time

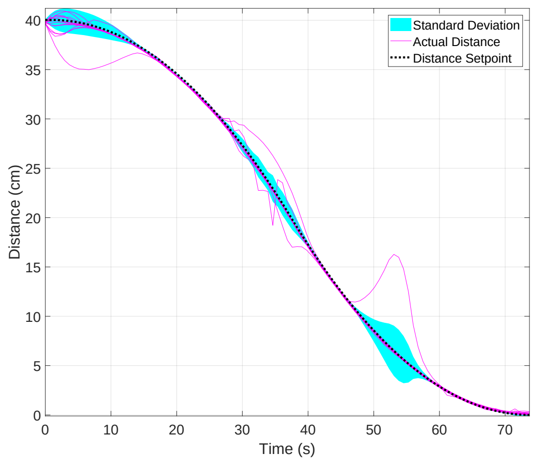


Figure 5.25: Electromagnet-based Docking System Tracking Error

5.4.2 Thruster-Based Docking

Since the thrusters are not limited by intersatellite distance but by propellant consumption, these experiments all started from much higher separations. The trajectory resulting from one of thruster-based docking tests is shown in Fig. 5.26. During this test, the spacecraft were placed 5 meters apart. The components of the velocities of the two spacecraft can be seen in Fig. 5.28.

The resultant force for each of the positive and negative thrusters during translation phase is seen in Fig. 5.27. It is observed that the thrusters are not firing less than the time corresponding to the MIB, which is 16 ms.

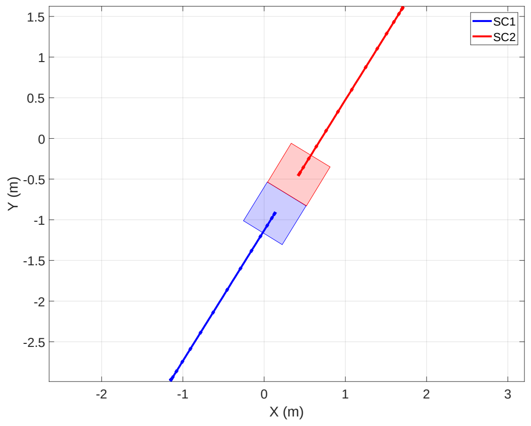


Figure 5.26: Thruster Docking Trajectory with Orientation Represented using Squares

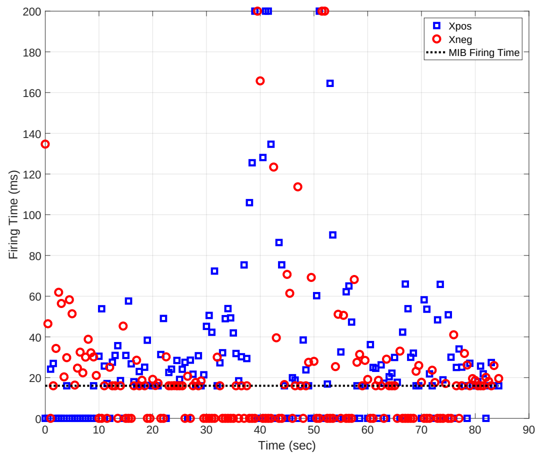


Figure 5.27: MIB

Fig. 5.29 shows the trajectory tracking accuracy during 12 of the 14 thruster-based docking test runs. Two runs were excluded because the initial separation was significantly below the average. The maximum standard deviation of the tracking error is 4.17 cm.

5.4.3 Experimental Results - Comparison

The two analyzed docking systems are compared by evaluating the initial separation, final relative velocity, consumable usage and docking success. Unlike other metrics, consumables are specific to each of the docking systems. For thruster-based docking, the total firing time of each of individual thrusters used for position and attitude control is computed. For

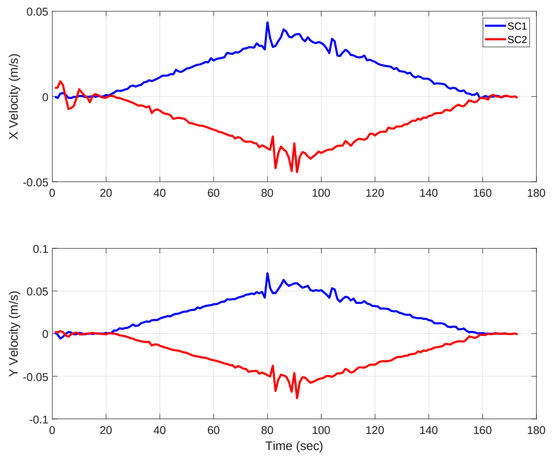


Figure 5.28: X and Y Velocities vs Time

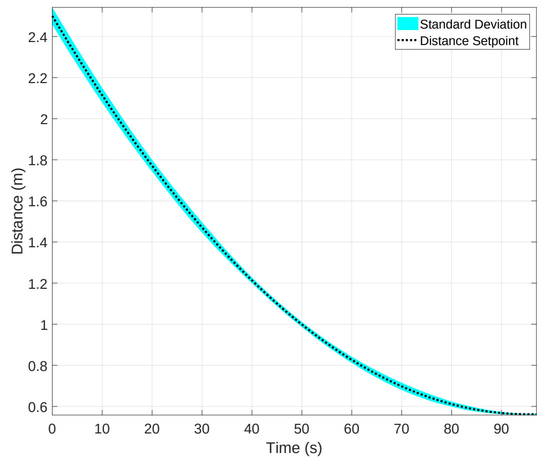


Figure 5.29: Thruster-based Docking System Tracking Error

electromagnet-based docking, the energy is computed as the average power multiplied by the docking time. The results from the two docking systems during the 14 tests are presented in the Tables B.1 and B.2 in Appendix B.

To assess the ultra-soft docking capability of both systems, in Fig. 5.30, the relative velocity between the two spacecraft simulators is plotted. The thrusters perform well compared to the electromagnets, however, at the cost of significant propellant use. During the first 8 runs, the electromagnets had an erroneous offset of 2.5 mm, thus the relative velocity is artificially higher than subsequent runs where the error was corrected. After this issue was solved, it can be seen that the electromagnets are under the ultra-soft docking velocity limit (less than 1 mm/s). Additionally, in this plot it is shown that Test 5 is a failed dock. The spacecraft were not able to dock in this test due to the aforementioned offset and the fact that the gripper is designed to grip at a maximum separation of 2 mm. This was the only electromagnet-based docking test where the gripper did not ridigize the two spacecraft. For thruster docking, a dock failure occurred in Test 13, caused by a misalignment.

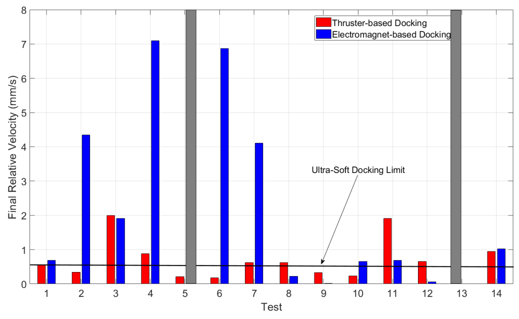


Figure 5.30: Relative Velocity for the 14 runs

5.5 Conclusion

Two propellant-free and ultra-low propellant docking systems were presented and compared against thruster-based methods. For the tether-based system, the savings in fuel for the tether system vs the thruster-based system using the optimal control was not considered large enough to justify the added mechanical complexity. The tether does offer a significant reduction in risk by its nature so overall is still considered favorable. A preliminary propellant-less maneuver was attempted with some degree of success, but to achieve the same ultra-soft dock would require further study.

The electromagnetic and thruster-based docking systems presented in the second half and experimentally validated. The electromagnet-based docking system performed admirably and was able to complete docks from up to a 55 cm intersatellite distance. The electromagnet-based docking system suffers from angular drift, most likely due to imbalances and manufacturing issues in the reaction wheels. The thruster-based docking system was able to successfully capture with a fairly low relative velocity but to achieve such a low velocity the propellant usage was high. Work remains to isolate the potential error sources in the electromagnetic docking system to reduce the drift, incorporate on-board navigation to remove the reliance on the motion capture system, and redesign the gripper to improve robustness to misalignment.

Chapter 6

Fast Motion Planning

Fast motion planning of an agile, autonomous vehicle (like quadrotor, micro aerial vehicle, rover, or spacecraft) has become an important topic of research over the last decade due to significant improvements in hardware and processing power, which enable a large range of applications that require faster planning for more fluid motion. For example, quadrotor and micro aerial vehicle are suitable for deploying mobile sensor platforms for intelligence, surveillance, and reconnaissance missions. Various space agencies have proposed missions to explore small bodies in space using small spacecraft. A common problem in these applications is that of quickly generating an optimal trajectory for the autonomous vehicle, which satisfies vehicle dynamics and collision avoidance constraints, in an environment cluttered with multiple geometrically fixed obstacles. This chapter discusses the development of a novel motion planning algorithm that addresses this problem. The work contained in this chapter was developped in collaboration with Dr. Saptarshi Bandyopadhyay and Francesca Baldini.

6.1 Problem Statement

The nonlinear optimal motion planning problem in continuous-time is first presented in Section 6.1.1. This problem is then recast into a discrete-time nonlinear optimal motion planning problem in Section 6.1.2.

6.1.1 Continuous-time Nonlinear Optimal Motion Planning Problem

Let $\mathcal{X} \subset \mathbb{R}^3$ denote the closed 3D workspace. Let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ represent the stationary obstacles in the workspace. Let δ denote the clearance that the vehicle must maintain from any obstacle. Let $\mathcal{X}_{\text{unsafe}} \subset \mathcal{X}$ denote the unsafe regions around the obstacles that the vehicle cannot enter. Therefore, the region where the vehicle can maneuver freely is given by $\mathcal{X}_{\text{free}} = \mathcal{X}/(\mathcal{X}_{\text{obs}} \cup \mathcal{X}_{\text{unsafe}})$. Let $X_{\text{init}} \in \mathcal{X}_{\text{free}}$ and $X_{\text{goal}} \in \mathcal{X}_{\text{free}}$ represent the starting position and the goal position of the vehicle at times t_0 and t_f respectively. Here t represents continuous time. The workspace, obstacles, and locations of interest are shown in Fig. 6.1. Although the figures are shown in 2D for ease of representing the ideas, the whole scheme is valid for 3D.

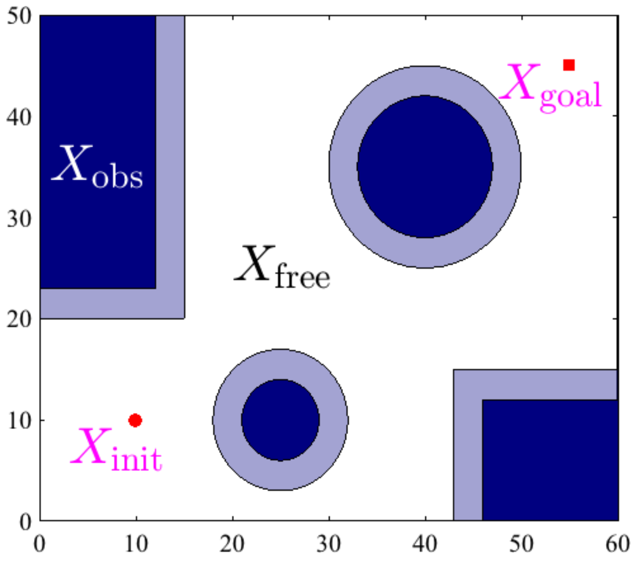


Figure 6.1: $\mathcal{X}_{\text{obs}}$, $\mathcal{X}_{\text{free}}$, $\mathcal{X}_{\text{init}}$, and $\mathcal{X}_{\text{goal}}$ are shown in this workspace. The obstacles are marked in blue.

Let $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ represent the vehicle's state vector and control input at time t. The vehicle's state vector can be further decomposed into $\boldsymbol{x}(t) = \left(\boldsymbol{p}(t)^T, \dot{\boldsymbol{p}}(t)^T, \boldsymbol{\theta}(t)^T, \dot{\boldsymbol{\theta}}(t)^T\right)^T$, where $\boldsymbol{p}(t) \in \mathbb{R}^3$ and $\boldsymbol{\theta}(t) \in \text{SO}(3)$ represent the vehicle's position and attitude vectors. We assume that the objective function of the states and control inputs $\mathcal{C}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$ is convex.

As shown in Problem 8, the objective of the optimal motion planning problem is to find a feasible trajectory for the vehicle that starts at X_{init} at time t_0, reaches X_{goal} at time t_f, and minimizes the given convex objective function. Here (6.2)–(6.3) represent the initial and terminal constraints and (6.4) ensures that the trajectory is within the safe region. The set $\mathcal{U} \subset \mathbb{R}^{n_u}$ in (6.5) represents the feasible range of control inputs which is assumed to be convex. The vehicle's dynamics are given by (6.6), where $\boldsymbol{f}(\cdot)$ is a smooth nonlinear function.

Problem 8 (Continuous-time Nonlinear Optimal Motion Planning Problem).

$$\underset{\boldsymbol{x}(t),\, \boldsymbol{u}(t),\, \forall t\in[t_0,t_f]}{\text{minimize}} \quad \int_{t_0}^{t_f} \mathcal{C}(\boldsymbol{x}(t), \qquad \boldsymbol{u}(t))\, dt\,, \tag{6.1}$$

$$\text{subject to} \quad \boldsymbol{p}(t_0) = X_{\text{init}}\,, \tag{6.2}$$

$$\boldsymbol{p}(t_f) = X_{\text{goal}}\,, \tag{6.3}$$

$$\boldsymbol{p}(t) \in \mathcal{X}_{\text{free}}\,, \qquad \forall t \in [t_0, t_f]\,, \tag{6.4}$$

$$\boldsymbol{u}(t) \in \mathcal{U}\,, \qquad \forall t \in [t_0, t_f]\,, \tag{6.5}$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)\,, \qquad \forall t \in [t_0, t_f]\,. \tag{6.6}$$

6.1.2 Discrete-time Nonlinear Optimal Motion Planning Problem

We first transform the continuous-time Problem 8 into a discrete-time problem so that it can be efficiently solved using SCP. Let Δ represent the fixed time step size. Let $\tau[k]$, $\forall k \in \{0, \ldots, T\}$ represent the discrete time instants, where T is the total number of discrete time steps. Therefore, we have $\tau[0] = t_0$, $\tau[T] = t_f$, and $\tau[k] = t_0 + k\Delta$ for all $k \in \{0, \ldots, T\}$. Therefore, $\Delta = \frac{t_f - t_0}{T}$ and $T = \frac{t_f - t_0}{\Delta}$.

The control inputs in Problem 8 are discretized using a zero-order hold approach such that $\forall k \in \{0, \ldots, T-1\}$:

$$\boldsymbol{u}(t) = \boldsymbol{u}[k], \quad t \in [\tau[k], \tau[k+1]) \ , \tag{6.7}$$

$$\boldsymbol{x}(t) = \boldsymbol{x}[k], \quad t \in [\tau[k], \tau[k+1]) \ , \tag{6.8}$$

$$\therefore \boldsymbol{p}(t) = \boldsymbol{p}[k] \, , \boldsymbol{\theta}(t) = \boldsymbol{\theta}[k] \, .$$

In Problem 8, the objective function (6.1) and the constraints (6.2)–(6.5) can be easily discretized using this zero-order hold approach. The linearization and discretization of the vehicle's dynamics equations (6.6) is discussed in the Appendix. Therefore, the transformed discrete-time optimal motion planning problem is given in Problem 9. Note that the constraints (6.10), (6.11), and (6.14) are linear and the constraint (6.13) is convex. Since the region $\mathcal{X}_{\text{free}}$ is usually not convex, the only non-convexity in Problem 9 arises from the safety constraint (6.12).

Problem 9 (Discrete-time Nonlinear (Non-Convex) Optimal Motion Planning Problem).

$$\underset{\boldsymbol{x}[k],\boldsymbol{u}[k]}{\text{minimize}} \sum_{k=0}^{T-1} \mathcal{C}\left(\boldsymbol{x}[k], \, \boldsymbol{u}[k]\right) \Delta \, , \text{subject to} \tag{6.9}$$

$$\boldsymbol{p}[0] = X_{\text{init}} \, , \tag{6.10}$$

$$\boldsymbol{p}[T] = X_{\text{goal}} \, , \tag{6.11}$$

$$\boldsymbol{p}[k] \in \mathcal{X}_{\text{free}} \, , \qquad \forall k \in \{0, \ldots, T\} \, , \tag{6.12}$$

$$\boldsymbol{u}[k] \in \mathcal{U} \, , \qquad \forall k \in \{0, \ldots, T-1\} \, , \tag{6.13}$$

$$\boldsymbol{x}[k+1] = F[k]\boldsymbol{x}[k] + G[k] \, \boldsymbol{u}[k] + H[k], \qquad \forall k \in \{0, \ldots, T-1\} \, . \tag{6.14}$$

The objective of this chapter is to find an efficient technique to solve Problem 9. Our SE–SCP algorithm first explores the workspace to approximate the region $\mathcal{X}_{\text{free}}$ using a graph and a combination of spheres. Once a feasible path is found on the graph, then we solve Problem 9 using a sequence of convex problems.

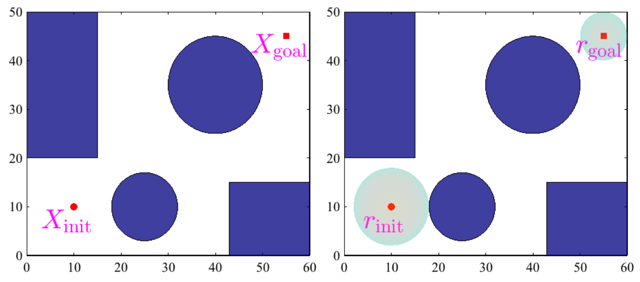
6.2 Spherical Expansion and Sequential Convex Programming (SE–SCP) Algorithm

The SE–SCP algorithm's pseudocode is presented in Algorithm 8. During the *Initialization step* (Section 6.2.1), the necessary data structures are created and initialized. After initialization, the *Spherical Expansion step* (Section 6.2.2) and the *Sequential Convex Programming step* (Section 6.2.3) are executed iteratively until the algorithm is stopped.

6.2.1 Initialization Step

The SE–SCP algorithm generates a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in the safe region $\mathcal{X}_{\text{free}}$, where $\mathcal{V}$ is the set of nodes (vertices) and $\mathcal{E}$ is the set of directed edges. Each node in $\mathcal{V}$ stores the position of the node and the minimum distance from that node to any obstacle. For the node X_{init}, the minimum distance to any obstacle r_{init} is obtained using the function `MinDistObs`$(X_{\text{init}}, \mathcal{X}_{\text{obs}})$, which takes into account the position of the node and the obstacles in the workspace and returns the radius of the largest sphere centered on that node which does not intersect with any obstacle (line 1). Similarly, the minimum distance to any obstacle r_{goal} is obtained for the node X_{goal} as shown in Fig. 6.2 (line 2). Then the set of nodes $\mathcal{V}$ is initialized with the nodes $X_{\text{init}}[r_{\text{init}}]$ and $X_{\text{goal}}[r_{\text{goal}}]$ (line 3). Each element in the set of edges $\mathcal{E}$ stores the edge's starting and ending nodes and the cost of traversing that edge. The set of edges $\mathcal{E}$ is initialized with the empty set (line 3). If the spheres of the nodes X_{init} and X_{goal} intersect, then edges are added between them (lines 4–9). The process of generating edges is discussed in Section 6.2.2.

The variable P_{old} is used to store the best path found till the current time instant, and the variable $c_{P_{\text{old}}}$ represents the cost of this path. The variable $c_{\boldsymbol{u}_{\text{old}}}$ represents the cost of the best trajectory found till the current time instant. During initialization, the path P_{old} is set to an empty set and the cost for the path $c_{P_{\text{old}}}$ and the trajectory $c_{\boldsymbol{u}_{\text{old}}}$ are set to infinity or a very large number (line 10). The variable N, which represents the number of samples, is also initialized to an appropriate value (line 11).



(a) X_{init} **and** X_{goal} **(b)** r_{init} **and** r_{goal}

Figure 6.2: Initialization step. The obstacles are marked in blue.

Method 8 SE–SCP Algorithm

```
1: $r_{\text{init}} \leftarrow \texttt{MinDistObs}(X_{\text{init}}, \mathcal{X}_{\text{obs}})$
2: $r_{\text{goal}} \leftarrow \texttt{MinDistObs}(X_{\text{goal}}, \mathcal{X}_{\text{obs}})$
3: $\mathcal{V} \leftarrow \{X_{\text{init}}[r_{\text{init}}], X_{\text{goal}}[r_{\text{goal}}]\}$, $\mathcal{E} \leftarrow \emptyset$
4: if $\|X_{\text{init}} - X_{\text{goal}}\|_2 \leq r_{\text{init}} + r_{\text{goal}}$ then
5:     $c_{\text{init,goal}} \leftarrow \texttt{EdgeCost}(X_{\text{init}}, X_{\text{goal}})$
6:     $\mathcal{E} \leftarrow \mathcal{E} \cup \{\overrightarrow{X_{\text{init}}\, X_{\text{goal}}}[c_{\text{init,goal}}]\}$
7:     $c_{\text{goal,init}} \leftarrow \texttt{EdgeCost}(X_{\text{goal}}, X_{\text{init}})$
8:     $\mathcal{E} \leftarrow \mathcal{E} \cup \{\overrightarrow{X_{\text{goal}}\, X_{\text{init}}}[c_{\text{goal,init}}]\}$
9: end if
10: $c_{P_{\text{old}}} \leftarrow \infty$, $c_{\boldsymbol{x}_{\text{old}}\boldsymbol{u}_{\text{old}}} \leftarrow \infty$, $P_{\text{old}} \leftarrow \emptyset$
11: for $i = 1, \ldots, N$ do
12:     $X_{\text{rand}} \leftarrow \texttt{GenerateSample}$
13:     $X_{\text{nearest}} \leftarrow \texttt{NearestVertex}(\mathcal{V}, X_{\text{rand}})$
14:     $X_{\text{new}} \leftarrow \texttt{Steer}(X_{\text{rand}}, X_{\text{nearest}})$
15:     $r_{\text{new}} \leftarrow \texttt{MinDistObs}(X_{\text{new}}, \mathcal{X}_{\text{obs}})$
16:     $\boldsymbol{X}_{\text{near}} \leftarrow \texttt{NearVertices}(\mathcal{V}, X_{\text{new}}, r_{\text{new}})$
17:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{X_{\text{new}}[r_{\text{new}}]\}$
18:     for all $X_{\text{n}} \in \boldsymbol{X}_{\text{near}}$ do
19:         $c_{\text{n,new}} \leftarrow \texttt{EdgeCost}(X_{\text{n}}, X_{\text{new}})$
20:         $\mathcal{E} \leftarrow \mathcal{E} \cup \{\overrightarrow{X_{\text{n}}\, X_{\text{new}}}[c_{\text{n,new}}]\}$
21:         $c_{\text{new,n}} \leftarrow \texttt{EdgeCost}(X_{\text{new}}, X_{\text{n}})$
22:         $\mathcal{E} \leftarrow \mathcal{E} \cup \{\overrightarrow{X_{\text{new}}\, X_{\text{n}}}[c_{\text{new,n}}]\}$
23:     end for
24:     $P_{\text{new}}, c_{P_{\text{new}}} \leftarrow \texttt{MinPath}(\mathcal{G}, X_{\text{init}}, X_{\text{goal}})$
25:     if $c_{P_{\text{new}}} < c_{P_{\text{old}}}$ and $P_{\text{new}} \neq P_{\text{old}}$ then
26:         $P_1 \leftarrow P_{\text{new}}$
27:         $\texttt{WS}_1 \leftarrow \texttt{PickSphere}(P_1)$
28:         $(\boldsymbol{x}_1^\star, \boldsymbol{u}_1^\star, c_{\boldsymbol{x}_1^\star\boldsymbol{u}_1^\star}) \leftarrow \texttt{OptTraj}(P_1, \texttt{WS}_1)$
29:         for j $= 2, \ldots, N_{SCP}$ do
30:             $(P_{\text{j}}, \texttt{WS}_{\text{j}}) \leftarrow$
                    $\texttt{GenPath}(\boldsymbol{x}_{\text{j}-1}^\star, P_{\text{j}-1}, \texttt{WS}_{\text{j}-1})$
31:             $(\boldsymbol{x}_{\text{j}}^\star, \boldsymbol{u}_{\text{j}}^\star, c_{\boldsymbol{x}_{\text{j}}^\star\boldsymbol{u}_{\text{j}}^\star}) \leftarrow$
                    $\texttt{OptTraj}(P_{\text{j}}, \texttt{WS}_{\text{j}}, \boldsymbol{x}_{\text{j}-1}^\star, \boldsymbol{u}_{\text{j}-1}^\star)$
32:         end for
33:         if $c_{\boldsymbol{x}_{\text{NSCP}}^\star\boldsymbol{u}_{\text{NSCP}}^\star} < c_{\boldsymbol{x}_{\text{old}}\boldsymbol{u}_{\text{old}}}$ then
34:             $\boldsymbol{x}_{\text{old}} \leftarrow \boldsymbol{x}_{\text{NSCP}}^\star$
35:             $\boldsymbol{u}_{\text{old}} \leftarrow \boldsymbol{u}_{\text{NSCP}}^\star$
36:             $c_{\boldsymbol{x}_{\text{old}}\boldsymbol{u}_{\text{old}}} \leftarrow c_{\boldsymbol{x}_{\text{NSCP}}^\star\boldsymbol{u}_{\text{NSCP}}^\star}$
37:             $c_{P_{\text{old}}} \leftarrow c_{P_{\text{new}}}$
38:         end if
39:         $P_{\text{old}} \leftarrow P_{\text{new}}$
40:     end if
41: end for
42: Return $\boldsymbol{x}_{\text{old}}$, $\boldsymbol{u}_{\text{old}}$, $c_{\boldsymbol{x}_{\text{old}}\boldsymbol{u}_{\text{old}}}$
```

6.2.2 Spherical Expansion Step

During this step, the workspace is explored using the sampling technique shown in lines 12–24 in Algorithm 8. The objective of this step is to populate the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and find the minimum-cost paths from the start position X_{init} to the goal position X_{goal}.

Generate Sample

: The SE–SCP algorithm can use both random or quasi-random (deterministic) sampling since the randomness of the samples is not crucial for motion planning applications ([73]). For a given sampling choice, the function `GenerateSample` returns a random sample $X_{\text{rand}} \in \mathcal{X}$ (line 12). If random sampling is chosen, then X_{rand} is drawn from a uniform distribution in $\mathcal{X}$ such that the random samples are independent and identically distributed. If quasi-random sampling is chosen, then X_{rand} is generated using the Halton sequence, which deterministically produces samples with low discrepancy and provides good coverage over the workspace with a limited number of samples ([73]).

Generate New Node

: Next, the function `NearestVertex`$(\mathcal{V}, X_{\text{rand}})$ takes into account the current set of nodes $\mathcal{V}$ and the random sample X_{rand} and returns the node X_{nearest} that is nearest to X_{rand} (line 13), i.e.,:

$$X_{\text{nearest}} := \underset{X \in \mathcal{V}}{\operatorname{argmin}} \left\| X - X_{\text{rand}} \right\|_2 \ . \tag{6.15}$$

Note that the minimum distance to any obstacle r_{nearest} from the node X_{nearest} is already stored in $\mathcal{V}$.

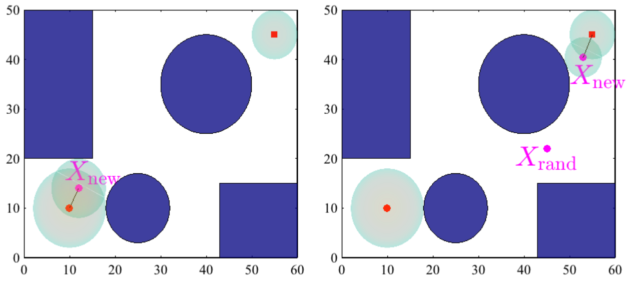
The function `Steer`$(X_{\text{rand}}, X_{\text{nearest}})$ generates the new node X_{new} according to either of the following two cases (line 14):

Case 1: If X_{rand} is inside X_{nearest}'s sphere (i.e., $\|X_{\text{nearest}} - X_{\text{rand}}\|_2 \leq r_{\text{nearest}}$) as shown in Fig. 6.3a, then the new node $X_{\text{new}} = X_{\text{rand}}$.

Case 2: Otherwise, X_{rand} is outside X_{nearest}'s sphere as shown in Fig. 6.3b. Then the new node X_{new} is on the boundary of X_{nearest}'s sphere and closest to the sample X_{rand}, i.e.,:

$$\begin{aligned} X_{\text{new}} &:= \operatorname*{argmin}_{\{X\in\mathcal{X}:\|X-X_{\text{nearest}}\|_2=r_{\text{nearest}}\}} \|X - X_{\text{rand}}\|_2 \\ &= X_{\text{nearest}} + r_{\text{nearest}} \frac{(X_{\text{rand}} - X_{\text{nearest}})}{\|X_{\text{rand}} - X_{\text{nearest}}\|_2} \,. \end{aligned} \tag{6.16}$$

The minimum distance from obstacles r_{new} for the node X_{new} is computed using the function `MinDistObs` (line 15). Thus the node X_{new} characterizes the convex obstacle-free region around itself using the closed sphere of radius r_{new}.



(a) Case 1 **(b) Case 2**

Figure 6.3: X_{new} and r_{new} are generated using the `Steer` and `MinDistObs` functions.

In contrast with the classical steering function in the literature ([77]), where the radius of the sphere is fixed and represents the step-size of the algorithm, the radius of the sphere in the SE–SCP algorithm is variable and adapts with the density of obstacles. Therefore, the SE–SCP algorithm usually finds a feasible path faster than other sampling-based algorithms.

The function `NearVertices`$(\mathcal{V}, X_{\text{new}}, r_{\text{new}})$ takes into account the current set of nodes $\mathcal{V}$, the new node X_{new} and its radius r_{new}, and generates a list $\boldsymbol{X}_{\text{near}}$ of all the nodes whose spheres intersect with X_{new}'s sphere (line 16), i.e.,:

$$\boldsymbol{X}_{\text{near}} := \{X_{\text{n}} \in \mathcal{V} : \|X_{\text{n}} - X_{\text{new}}\|_2 \leq r_{\text{n}} + r_{\text{new}}\} \,. \tag{6.17}$$

It follows from our construction that $X_{\text{nearest}} \in \boldsymbol{X}_{\text{near}}$. The new node $X_{\text{new}}[r_{\text{new}}]$ is added to the set of nodes $\mathcal{V}$ (line 17).

Generate Graph's Edges

: There exists a feasible collision-free path between each node in $\boldsymbol{X}_{\text{near}}$ and X_{new} because their spheres intersect. We now generate the edges that connect these nodes. For each vertex $X_{\text{n}} \in \boldsymbol{X}_{\text{near}}$, the function `EdgeCost`$(X_{\text{n}}, X_{\text{new}})$ takes into account the two nodes and outputs the cost of traversing the directed edge $c_{\text{n,new}}$ from X_{n} to X_{new}, which depends on the given convex objective function, such that the trajectory always remains inside the union of the two spheres (line 19). Then the directed edge $\overrightarrow{X_{\text{n}}\, X_{\text{new}}}[c_{\text{n,new}}]$ is added to the set of edges $\mathcal{E}$ (line 20). Similarly, the cost for traversing the directed edge $c_{\text{new,n}}$ from X_{new} to X_{n} is generated (line 21), and this new directed edge $\overrightarrow{X_{\text{new}}\, X_{\text{n}}}[c_{\text{new,n}}]$ is added to $\mathcal{E}$ (line 22).

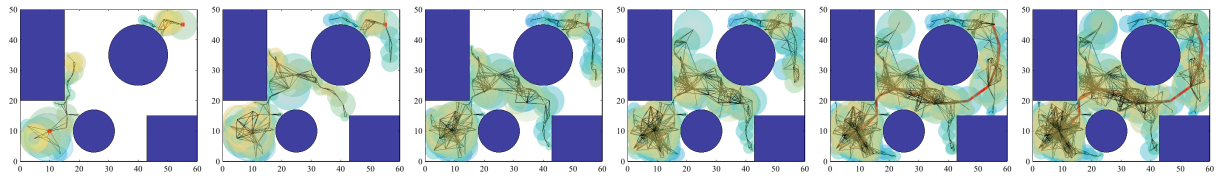

(a) $i = 50$ **(b)** $i = 100$ **(c)** $i = 150$ **(d)** $i = 200$ **(e)** $i = 250$ **(f)** $i = 300$

Figure 6.4: Multiple iterations of the spherical expansion step. The minimum-cost paths are shown in red. See Extension 1.

Generate Minimum-Cost Path

: The function `MinPath`$(\mathcal{G}, X_{\text{init}}, X_{\text{goal}})$ takes into account the current graph, the initial and goal nodes, and returns the new minimum-cost path P_{new} along with the cost of that path $c_{P_{\text{new}}}$ (line 24). The path $P_{\text{new}} = \{X_1[r_1], X_2[r_2], \ldots, X_{\text{n}}[r_{\text{n}}]\}$ is a sequence of n nodes with corresponding radii, where $|P_{\text{new}}| = n$, $X_1 = X_{\text{init}}$, and $X_{\text{n}} = X_{\text{goal}}$. An example path is shown in Fig. 6.5. The cost of the path $c_{P_{\text{new}}}$ is the sum of the directed edges along that path. Graph search algorithms like Dijkstra's algorithm can be used for this step. If no path exists, then $c_{P_{\text{new}}}$ is set to infinity.

In Fig. 6.4 and Extension 1, multiple iterations of the spherical expansion step are shown. During each iteration, a new node and multiple edges are added to the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the graph $\mathcal{G}$ grows from both the start and goal positions. The minimum-cost paths from

X_{init} to X_{goal} are shown in Fig. 6.4e and 6.4f.

The following SCP step of the algorithm does not start until at least one geometric path is found from X_{init} to X_{goal}. However, the spherical expansion step continues to explore the workspace, even after SCP step has started, to find better geometric paths which are then further optimized by the SCP step.

6.2.3 Sequential Convex Programming Step

During this step, the locally optimal trajectories from the start X_{init} to the goal position X_{goal} that satisfy vehicle's dynamics constraints are found using lines 24–40 in Algorithm 8.

Generate Optimal Trajectory

: If the cost of the new path $c_{P_{\text{new}}}$ is less than that of the old path $c_{P_{\text{old}}}$ and the new path is different from the old path, then the optimal trajectory is found along this new path.

We first set the time step size Δ such that $T \gg |P_{\text{new}}|$ and the linearization and discretization steps in Section 6.1.2 are valid. Initially, the T nodes are partitioned such that the first set lie inside the sphere of X_1, the next set lie in the sphere of X_2, and so on. Let us represent this partitioning using the array $\texttt{WS}_1$, where $\texttt{WS}_1[k]$ outputs the index of the sphere the k^{th} node is supposed to lie in. The function `PickSphere` is used to generate the array $\texttt{WS}_1$ from the path P_{new} (or P_1). Here we partition the nodes based on the distance between the spheres as shown in Algorithm 9. Although this initial partitioning of the nodes is done heuristically, it does not affect the convergence proof since it is only done during the first SCP loop. Furthermore, a better partitioning of the nodes can be obtained using a greedy optimization algorithm ([92]).

Problem 10 (Discrete-time Convex Optimal Motion Planning Problem $\big(CP(\boldsymbol{x}^{\star}_{\text{j}-1}, \boldsymbol{u}^{\star}_{\text{j}-1})\big)$).

$$\underset{\boldsymbol{x}_{\text{j}},\boldsymbol{u}_{\text{j}}}{\text{minimize}} \sum_{k=0}^{T-1} \mathcal{C}\left(\boldsymbol{x}_{\text{j}}[k],\, \boldsymbol{u}_{\text{j}}[k]\right)\Delta\,, \text{subject to} \tag{6.18}$$

Method 9 The function $\texttt{PickSphere}(P_1)$

```
 1: Given $P_1 = \{X_1[r_1], X_2[r_2], \dots, X_\mathrm{n}[r_\mathrm{n}]\}$
 2: $Z_\mathrm{old} \leftarrow X_1$
 3: for $\ell = 1, \dots, n$ do
 4:     if $\ell = n$ then $Z_\mathrm{new} = X_\mathrm{n}$
 5:     else $Z_\mathrm{new} \leftarrow X_\ell + (X_{\ell+1} - X_\ell) \cdot \left(\frac{r_\ell}{r_\ell + r_{\ell+1}}\right)$
 6:     end if
 7:     $\mathbf{D}[\ell] \leftarrow \|Z_\mathrm{new} - Z_\mathrm{old}\|_2$
 8:     $Z_\mathrm{old} \leftarrow Z_\mathrm{new}$
 9: end for
10: $\mathbf{D} \leftarrow \left(\frac{1}{\sum_{\ell=1}^{n} \mathbf{D}[\ell]}\right) \cdot \mathbf{D}$
11: for $k = 0, \dots, T$ do
12:     for $\ell = 1, \dots, n$ do
13:         if $\ell = 1$ and $\frac{k}{T} \leq \mathbf{D}[1]$ then
14:             $\texttt{WS}_1[k] \leftarrow 1$
15:         else if $\mathbf{D}[\ell] < \frac{k}{T}$ and $\frac{k}{T} \leq \mathbf{D}[\ell+1]$ then
16:             $\texttt{WS}_1[k] \leftarrow \ell + 1$
17:         end if
18:     end for
19: end for
```

$$\boldsymbol{p}_\mathrm{j}[0] = X_\mathrm{init}\,, \tag{6.19}$$

$$\boldsymbol{p}_\mathrm{j}[T] = X_\mathrm{goal}\,, \tag{6.20}$$

$$\|\boldsymbol{p}_\mathrm{j}[k] - X_{\texttt{WS}_\mathrm{j}[k]}\|_2 \leq r_{\texttt{WS}_\mathrm{j}[k]}\,, \qquad \forall k \in \{1, \dots, T-1\}\,, \tag{6.21}$$

$$\|\boldsymbol{p}_\mathrm{j}[k] - X_{\texttt{WS}_\mathrm{j}[k+1]}\|_2 \leq r_{\texttt{WS}_\mathrm{j}[k+1]}\,, \qquad \forall k \in \{1, \dots, T-1\}\,, \tag{6.22}$$

$$\boldsymbol{u}_\mathrm{j}[k] \in \mathcal{U}\,, \qquad \forall k \in \{0, \dots, T-1\}\,, \tag{6.23}$$

$$\boldsymbol{x}_\mathrm{j}[k+1] = F_{\mathrm{j}-1}[k]\,\boldsymbol{x}_\mathrm{j}[k] + G_{\mathrm{j}-1}[k]\,\boldsymbol{u}_\mathrm{j}[k] + H_{\mathrm{j}-1}[k]\,, \qquad \forall k \in \{0, \dots, T-1\}\,. \tag{6.24}$$

The function $\texttt{OptTraj}(P_1, \texttt{WS}_1)$ takes into account this new path P_1 and $\texttt{WS}_1$, and returns the optimal state trajectory $\boldsymbol{x}_1^\star$, optimal control trajectory $\boldsymbol{u}_1^\star$, and the cost of this trajectory $c_{\boldsymbol{x}_1^\star \boldsymbol{u}_1^\star} = \sum_{k=0}^{T-1} \mathcal{C}\left(\boldsymbol{x}_1^\star[k],\, \boldsymbol{u}_1^\star[k]\right) \Delta$ by solving an approximate version of Problem 9 (shown in Problem 10) (line 28). If an additional trajectory is provided to the function $\texttt{OptTraj}(P_\mathrm{j}, \texttt{WS}_\mathrm{j}, \boldsymbol{x}_{\mathrm{j}-1}^\star, \boldsymbol{u}_{\mathrm{j}-1}^\star)$ (as is the case in line 31), then the linearization and discretization of the vehicle's dynamics is performed around this trajectory $(\boldsymbol{x}_{\mathrm{j}-1}^\star, \boldsymbol{u}_{\mathrm{j}-1}^\star)$ to generate the matrices $F_{\mathrm{j}-1}[k]$, $G_{\mathrm{j}-1}[k]$, and $H_{\mathrm{j}-1}[k]$. If no additional trajectory is provided to the

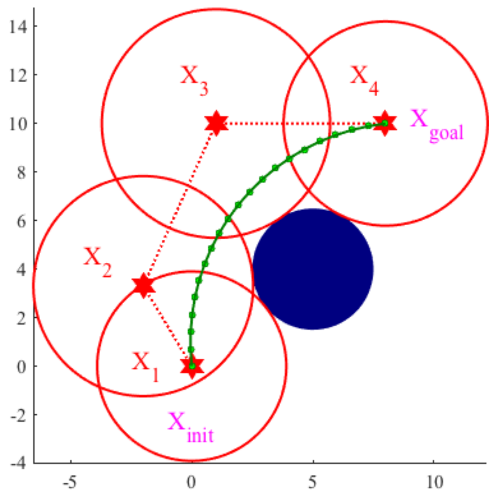


Figure 6.5: Visualization of the optimal trajectory (in green) for the first loop of SCP.

function `OptTraj`, then a nominal (pre-stored) trajectory $(\boldsymbol{x}_0^\star, \boldsymbol{u}_0^\star)$ is used for linearization and discretization.

Our key innovation is to use the sequence of spheres in the path P_1, as shown in Fig. 6.5, as an approximation of the region $\mathcal{X}_{\text{free}}$ in order to convexify the safety constraint (6.12) in Problem 9. The new motion planning problem is given in Problem 10. In order to convexify the safety constraint (6.12), we enforce the constraint (6.21) that all the nodes are inside their corresponding sphere. Nodes, with the next neighbor in the next sphere, have the additional constraint (6.22) that they must lie in the intersection of the two spheres. Note that Problem 10 is a convex optimization problem because the objective function is convex and all constraints are either affine or convex. The convex optimization problem can be solved using computationally-efficient interior-point-methods ([65, 67]) on resource-constrained hardware.

The globally optimal solution of Problem 10, namely state trajectory $\boldsymbol{x}_1^\star = \{\boldsymbol{x}_1[0], \boldsymbol{x}_1[1], \ldots, \boldsymbol{x}_1[T]\}$ and control trajectory $\boldsymbol{u}_1^\star = \{\boldsymbol{u}_1[0], \boldsymbol{u}_1[1], \ldots, \boldsymbol{u}_1[T-1]\}$ is used to setup a new optimization problem as shown in Fig. 6.6. The function $\texttt{GenPath}(\boldsymbol{x}_1^\star, P_1, \texttt{WS}_1)$ takes into account this optimal solution $\boldsymbol{x}_1^\star$, the old path P_1, and $\texttt{WS}_1$; and returns a new path P_2 and $\texttt{WS}_2$ as shown in Algorithm 10 (line 30 in Algorithm 8). If the spheres of two

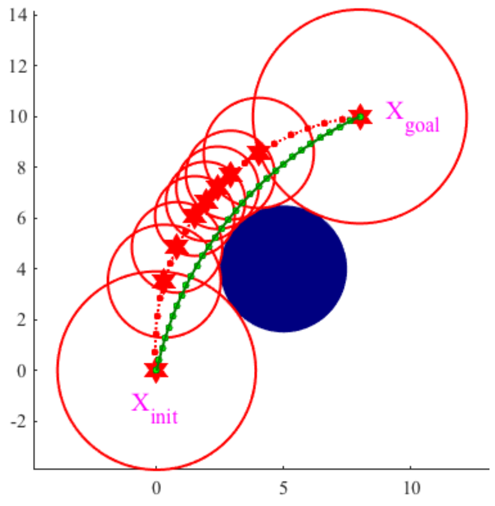


Figure 6.6: Visualization of the optimal trajectory (in green) for the second loop of SCP, where the old optimal trajectory (in red) from Fig. 6.5 is used to setup the optimization problem.

neighboring nodes in $\boldsymbol{x}_1^\star$ do not intersect, then the spheres from the previous step are used to fill in the gap.

Method 10 The function GenPath $(\boldsymbol{x}_{\mathrm{j}-1}^\star, P_{\mathrm{j}-1}, \mathtt{WS}_{\mathrm{j}-1})$

```
1:  Given $\boldsymbol{x}_{\mathrm{j}-1}^\star = \{\boldsymbol{x}_{\mathrm{j}-1}[0], \boldsymbol{x}_{\mathrm{j}-1}[1], \ldots, \boldsymbol{x}_{\mathrm{j}-1}[T]\}$, $\mathtt{WS}_{\mathrm{j}-1}$, and $P_{\mathrm{j}-1} = \{X_1[r_1], X_2[r_2], \ldots, X_{\mathrm{n}}[r_{\mathrm{n}}]\}$
2:  $\mathbf{S}[0] \leftarrow -1$
3:  for $k = 0, \ldots, T-1$ do
4:      if $\mathtt{WS}_{\mathrm{j}-1}[k] \neq \mathtt{WS}_{\mathrm{j}-1}[k+1]$ then
5:          $\mathbf{S}[\mathtt{WS}_{\mathrm{j}-1}[k]] \leftarrow k$
6:      end if
7:  end for
8:  $m \leftarrow 1$
9:  $P_{\mathrm{j}}[m] = P_{\mathrm{j}-1}[1]$
10: for $\ell = 1, \ldots, n-1$ do
11:     $(P_{\mathrm{j}}, m) \leftarrow$ AddToPath$(\mathbf{S}[\ell], m, \mathbf{S}[\ell-1]+1)$
12:     $\hat{k} \leftarrow \left\lceil \frac{\mathbf{S}[\ell]+\mathbf{S}[\ell+1]}{2} \right\rceil$
13:     $(P_{\mathrm{j}}, m) \leftarrow$ AddToPath$(\hat{k}, m, \mathbf{S}[\ell]+1)$
14: end for
```

Once again, the optimal trajectory $(\boldsymbol{x}_2^\star, \boldsymbol{u}_2^\star, c_{\boldsymbol{x}_2^\star \boldsymbol{u}_2^\star})$ is found using Problem 10, where the trajectory $(\boldsymbol{x}_1^\star, \boldsymbol{u}_1^\star)$ is used for linearization and discretization and

$c_{\boldsymbol{x}_2^\star \boldsymbol{u}_2^\star} = \sum_{k=0}^{T-1} \mathcal{C}\left(\boldsymbol{x}_2^\star[k],\, \boldsymbol{u}_2^\star[k]\right) \Delta$ (line 31). This optimization process using SCP is iteratively executed N_{SCP} times, where N_{SCP} depends on the desired convergence error and the complexity of the optimization problem (lines 29–32). Multiple iterations of the SCP step are shown in Fig. 6.7 and Extension 2. Then, the current trajectory cost is compared with the stored cost, and the best trajectory is stored (lines 33–39).

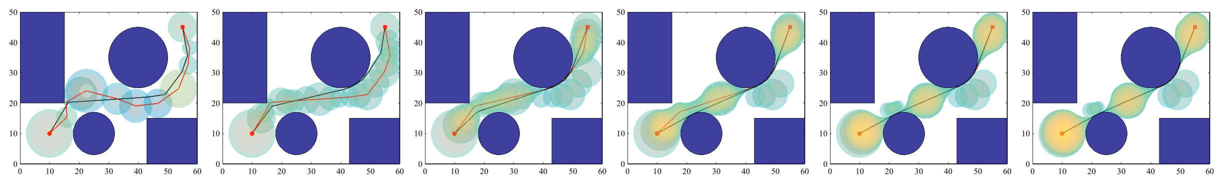

(a) $j = 1$ **(b)** $j = 2$ **(c)** $j = 3$ **(d)** $j = 4$ **(e)** $j = 5$ **(f)** $j = 6$

Figure 6.7: Multiple iterations of the SCP step. See Extension 2.

The algorithm stops after N nodes have been added to the graph (line 11), where N is significantly (orders-of-magnitude) smaller than state-of-the-art sampling-based algorithms. The algorithm outputs the best state trajectory $\boldsymbol{x}_{\text{old}}$, its control trajectory $\boldsymbol{u}_{\text{old}}$, and its cost $c_{\boldsymbol{x}_{\text{old}}\boldsymbol{u}_{\text{old}}}$ (line 42). If the algorithm is stopped before the for-loop is completed, then the current best state and control trajectories are stored in $\boldsymbol{x}_{\text{old}}$ and $\boldsymbol{u}_{\text{old}}$.

6.2.4 Nonlinear Dynamics in SE–SCP

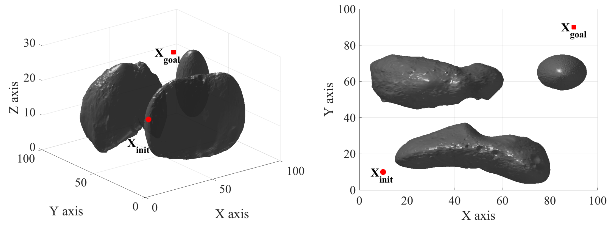
When used on platforms with nonlinear dynamics that are not well approximated by lower order dynamics, it is necessary to implement an approach similar to M-SCPn described in Chapter 3 in the sequential convex programming step to maintain the integrity of the solution. When SCPn or M-SCPn are used in lieu of SCP, the algorithm is referred to as SE-SCPn.

6.3 Numerical and Experimental Results

In this section, we demonstrate the effectiveness of the SE–SCP algorithms using numerical simulations and experimental results. We also present comparisons with existing algorithms like RRT* and PRM*.

Method 11 The function `AddToPath`(`Input1`, `Input2`, `Input3`)

```
1: $X_{\text{new}} \leftarrow \boldsymbol{p}_{\text{j}-1}[\texttt{Input1}]$
2: $r_{\text{new}} \leftarrow \texttt{MinDistObs}(X_{\text{new}}, \mathcal{X}_{\text{obs}})$
3: $m \leftarrow \texttt{Input2}$
4: $X_{\text{old}}[r_{\text{old}}] \leftarrow P_{\text{j}}[m]$
5: $k_{\text{min}} \leftarrow -1$, $k_{\text{max}} \leftarrow -1$
6: $k_{\text{new}} \leftarrow -1$, $k_{\text{old}} \leftarrow -1$
7: for $s = \texttt{Input3}, \ldots, \texttt{Input1}$ do
8:     if $\|X_{\text{new}} - \boldsymbol{p}_{\text{j}-1}[s]\|_2 \leq r_{\text{new}}$ and $\|X_{\text{old}} - \boldsymbol{p}_{\text{j}-1}[s]\|_2 \leq r_{\text{old}}$ then
9:         if $k_{\text{min}} = -1$ then
10:            $k_{\text{min}} \leftarrow s$, $k_{\text{max}} \leftarrow s$
11:        else if $k_{\text{max}} = s - 1$ then
12:            $k_{\text{max}} \leftarrow s$
13:        else
14:            $k_{\text{max}} \leftarrow k_{\text{min}}$
15:        end if
16:    end if
17:    if $\|X_{\text{old}} - \boldsymbol{p}_{\text{j}-1}[s]\|_2 \leq r_{\text{old}}$ then
18:        $k_{\text{old}} \leftarrow s$
19:    end if
20:    if $\|X_{\text{new}} - \boldsymbol{p}_{\text{j}-1}[s]\|_2 \leq r_{\text{new}}$ and $k_{\text{new}} = -1$ then
21:        $k_{\text{new}} \leftarrow s$
22:    end if
23: end for
24: if $\|X_{\text{old}} - X_{\text{new}}\|_2 \leq r_{\text{old}} + r_{\text{new}}$ and $k_{\text{min}} \geq 0$ then
25:    $m \leftarrow m + 1$
26:    $P_{\text{j}}[m] \leftarrow X_{\text{new}}[r_{\text{new}}]$
27:    $k_{\text{mid}} \leftarrow \left\lceil \frac{k_{\text{min}} + k_{\text{max}}}{2} \right\rceil$
28:    for $s = \texttt{Input3}, \ldots, \texttt{Input1}$ do
29:        if $s \leq k_{\text{mid}}$ then
30:            $\texttt{WS}_{\text{j}}[s] = m - 1$
31:        else
32:            $\texttt{WS}_{\text{j}}[s] = m$
33:        end if
34:    end for
35: else
36:    $m \leftarrow m + 1$
37:    $P_{\text{j}}[m] \leftarrow P_{\text{j}-1}[\ell]$
38:    $m \leftarrow m + 1$
39:    $P_{\text{j}}[m] \leftarrow X_{\text{new}}[r_{\text{new}}]$
40:    for $s = \texttt{Input3}, \ldots, \texttt{Input1}$ do
41:        if $s \leq k_{\text{old}}$ then
42:            $\texttt{WS}_{\text{j}}[s] = m - 2$
43:        else if $s > k_{\text{old}}$ and $s \leq k_{\text{new}}$ then
44:            $\texttt{WS}_{\text{j}}[s] = m - 1$
45:        else
46:            $\texttt{WS}_{\text{j}}[s] = m$
47:        end if
48:    end for
49: end if
```

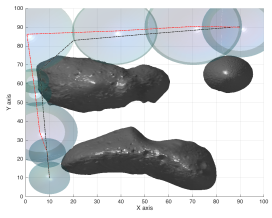


(a) 3D view (b) Top view

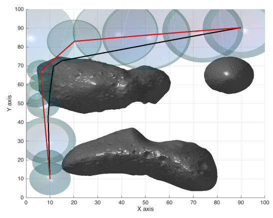
Figure 6.8: Debris environment.

6.3.1 Numerical Simulations with Spacecraft in Debris Field

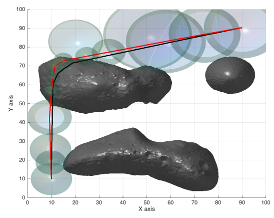
In this subsection, we show that the SE–SCP algorithm (Algorithm 8) can be used to generate a spacecraft's trajectory through a cluttered environment like a debris field. The debris environment, the start position X_{init}, and the goal position X_{goal} are shown in Fig. 6.8 and the nonlinear spacecraft dynamics are discussed in [96]. The objective is to find the spacecraft's trajectory from X_{init} to X_{goal} so that the fuel consumption is minimized.



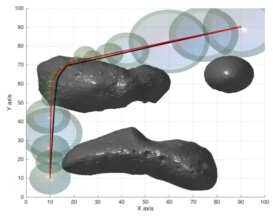
(a) Path P_{new}



(b) First SCP iteration



(c) Second SCP iteration



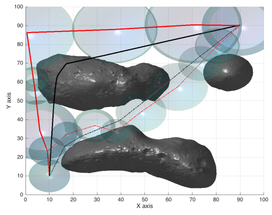
(d) Third SCP iteration

Figure 6.9: The first path P_{new} is found during the the spherical expansion step and its refinement using the SCP step are shown.

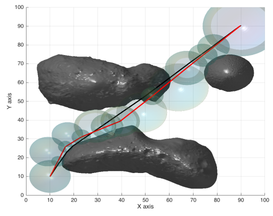
Fig. 6.9 shows the first path found during the spherical expansion step and three successive refinements using the SCP step. Fig. 6.10 shows a better path is found during the spherical expansion step and three successive refinements using the SCP step yield a better trajectory. These trajectories satisfy the spacecraft's dynamics constraints and are locally optimal with respect to fuel consumption.

6.3.2 Experimental Results with Quadrotor

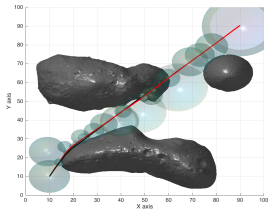
Experiments were performed with a custom-built quadrotor to show that the collision-free trajectories computed using the SE-SCP algorithm satisfy the vehicle's dynamics constraints. In this experiment, the collision-free trajectory is computed onbaord the quadrotor and then the quadrotor tracks this trajectory. These experiments were performed in collaboration with Dr. Kyunam Kim. The quadrotor used in this experiment weighs 2.5 kg, has an arm



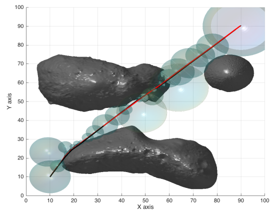
(a) Path P_{new}



(b) First SCP iteration



(c) Second SCP iteration



(d) Third SCP iteration

Figure 6.10: A better path P_{new} is found during the the spherical expansion step and its refinement using the SCP step are shown.

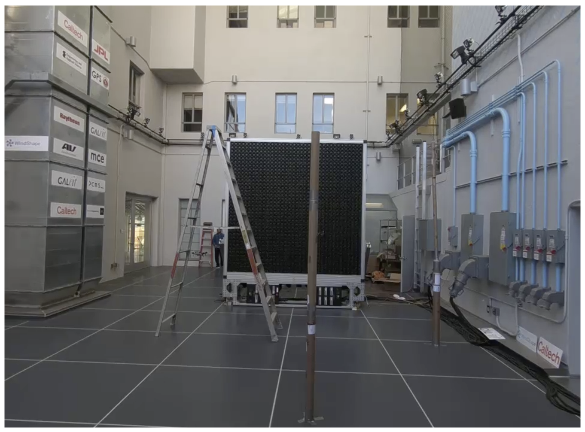
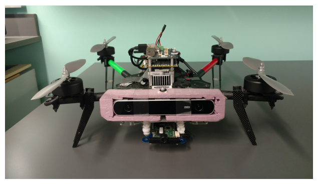
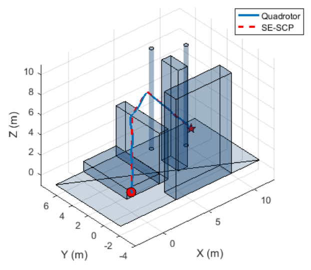


Figure 6.11: Caltech's CAST Drone Arena

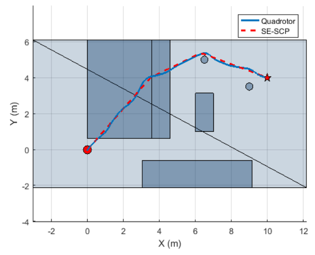
length of 24 cm, and is equipped with computers and sensors enabling fully autonomous flight (Fig. 6.12a).

The experiment is performed in the flight arena of Caltech's Center for Autonomous Systems and Technologies. The arena is approximately 9.3 m by 12.2 m by 12 m with multiple obstacles located both at the center of the space and close to the walls (Fig. 6.11). The environment is modeled in a simulation environment with the clearance distance from the obstacles set to $\delta = 0.5$ m. Although the arena has a plenty of vertical space, we limited

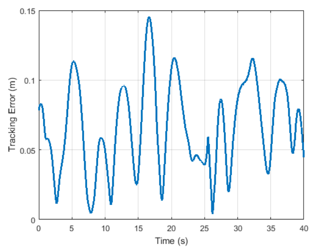

(a) Quadrotor



(b) Perspective View



(c) Top View



(d) Tracking Error

Figure 6.12: A map of the environment and trajectories computed by the SE-SCP algorithm (red dash line) and flown by a quadrotor (blue solid line). The initial and goal positions are marked with red circles and stars. See Extension 5.

the maximum height to 7 m to make the planning more challenging. The initial and goal locations are given as (0, 0, 1) and (10, 4, 1). Due to the position of the obstacles, the quadrotor has to pass through the narrow open space between the central obstacles.

The collision-free trajectory, computed onboard the quadrotor using by the SE-SCP algorithm, is shown in Fig. 6.12b, 6.12c. The quadrotor's actual flight trajectory, recorded with a motion capture system, is shown in Extension 5. The quadrotor is able to track the trajectory with the maximum position error of 0.15 m (Fig. 6.12d) and an average velocity of approximately 0.4 m/s while traversing the 16 m long trajectory. These results demonstrate that the SE–SCP algorithm can be executed onboard resource-constrained hardware and the collision-free trajectory generated by the SE–SCP algorithm satisfies the vehicle's dynamics constraints.

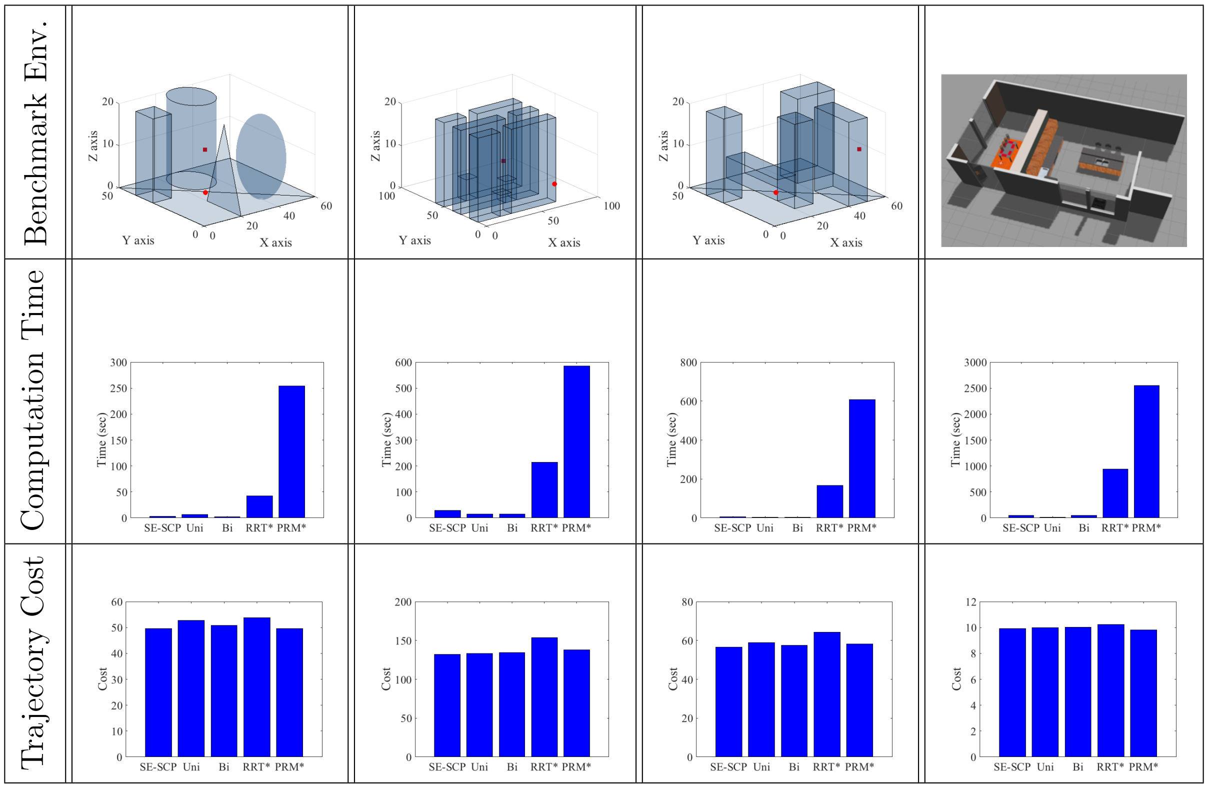


Figure 6.13: Benchmark Comparisons of the SE–SCP algorithms (original, uni-directional, and bi-directional) with RRT* and PRM*

6.3.3 Comparison with RRT* and PRM*

In this subsection, we compare the SE–SCP algorithms (Algorithm 8) with the asymptotically-optimal RRT* and PRM* algorithms ([77]). Since the RRT* and PRM* algorithms cannot directly incorporate the vehicle dynamics, we use the following discrete-time dynamics equation to compare these algorithms:

$$\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \boldsymbol{u}[k] \,. \tag{6.25}$$

The 3D benchmark environments are shown in Fig. 6.13. For comparison purposes, all the algorithms are implemented in MATLAB and each algorithm is executed 10 times. Average results of their computation time and trajectory cost (path distance) are presented as bar graphs in Fig. 6.13.

We conclude from these comparisons that the SE–SCP algorithm significantly outperform the the RRT* and PRM* algorithms in terms of the computation time for similar trajectory costs. The main reasons for the superior performance of the SE–SCP algorithms are as follows:

- The spherical expansion step in the SE–SCP algorithm adapts with the density of the obstacles in the environment. Since there is no pre-defined step-length in the SE–SCP algorithm, the algorithm can build larger spheres when there are no nearby obstacles. This is extremely helpful in quickly covering the workspace and possibly generating a simple path from the start to the goal position.

- The SCP step computes the locally optimal trajectory in the neighborhood of the geometric path, even if the original geometric path is not close to the locally optimal trajectory. This is useful in reducing the number of samples and computation time necessary to find the optimal trajectory.

Therefore, the SE–SCP algorithms are suitable for generating optimal trajectories through cluttered environments while satisfying the vehicle dynamics constraints.

6.4 Chapter Summary

In this chapter we presented a motion planning algorithm, through geometrically-fixed uncooperative cluttered environments, that uses the vehicle dynamics and minimizes the given convex cost function. Our SE–SCP algorithm has two main steps. During the spherical expansion step, the algorithm explores the workspace using a random sampling technique to generate collision-free spheres. This step generates coarse geometric paths from the start position to the goal position. During the SCP step, locally optimal trajectories are generated along those geometric paths using SCP optimization that considers both the vehicle dynamics and collision-avoidance constraints. Therefore, as newer paths are discovered by the spherical expansion step, better trajectories are generated by the SCP step. Hence the algorithm guarantees locally optimal trajectories using a finite number of samples and a globally optimal trajectory as the number of samples tends to infinity.

We demonstrated the effectiveness of the SE–SCP algorithm using experimental results and numerical simulations. We have shown that the SE–SCP algorithms significantly outperform the the RRT* and PRM* algorithms in terms of the computation time for similar trajectory costs. This is because the spherical expansion step in the SE–SCP algorithm adapts with the density of the obstacles in the environment, hence the workspace can be quickly covered with spheres. Moreover, the SCP step generates a locally optimal trajectory, which reduces the number of samples and computational time necessary to find the optimal trajectories. Therefore, the SE–SCP algorithm is suitable for fast motion planning through geometrically-fixed uncooperative cluttered environments. Future work will focus on generating trajectories through non-stationary obstacles.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The focus of this work was on optimal guidance and control algorithms for swarm self-assembly and general robotic motion planning for robots with nonlinear dynamics. The algorithms developed were designed to minimize fuel and avoid collisions while maintaining adherence to the nonlinear dynamics. These algorithms were tested in simulation and in experiments on-board wheeled robots, quadrotors, and spacecraft simulator robots.

In Chapter 2, we discussed the design and function of the robotic spacecraft simulators used to experimentally validate the work in Chapters 4 and 5. The M-STARs are air-bearing platforms which operate in a flat floor facility. This essentially negates the friction between the robot and the floor and does not cause any undesired perturbations on the robot. With the addition of spherical air bearings and linear actuators, the M-STARs are able to achieve 5DOF frictionless dynamic motion and 1DOF kinematic motion.

In Chapter 3, two methods were presented which are capable of extending sequential convex programming for use with nonlinear dynamics. Previous implementations suffered from deviations from the true nonlinear dynamics due to the sequential linearizations and discretizations. The proposed methods, SCPn and M-SCPn, numerically integrate the control trajectory resulting from the optimization to obtain a corrected nominal trajectory, which the next iteration of the optimization then linearized the dynamics around. This way, the linearization and discretization errors do not compound in each successive iteration, keeping the obtained trajectory close to the nonlinear dynamic trajectory. To ensure the corrected solution stays feasible to the nonconvex optimization problem, an additional bound around the inequality constraints is added to the SCPn optimization. This bound allows the formation of theoretical guarantees. Both methods are shown to converge to a solution with decreasing cost, and to satisfy the KKT conditions of the original nonconvex optimization problem. These claims can also be extended to problems with hyperplane-convexified inequality constraints and problems with costs that are convex functions of both state and control. In addition to theoretical guarantees, extensive simulations were performed to establish the efficacy and robustness of the two algorithms as compared to the standard, uncorrected SCP algorithm. Simulations were performed using highly nonlinear quadrotor dynamics with a convexified collision avoidance constraint. These results show that both SCPn and M-SCPn solve the terminal constraint failure of SCP and yield trajectories which adhere to the nonlinear dynamics, but suffer a performance loss when compared to SCP due to the extra computation needed to enforce the nonlinear dynamics. M-SCPn has improved performance over SCPn in terms of computation time, convergence quality, and robustness to initialization quality. Further work remains to reduce the computational burden, implement the methods on an agile quadrotor platform, and compare results against results obtained using a nonlinear optimization solver.

In Chapter 4, a distributed optimal control and guidance algorithm has been presented to allow for construction using a heterogeneous swarm of component satellites with limited communication radii. The agent types chosen can create a diverse set of final configurations which can cover the plane and build out-of-plane. This extends prior work in the field because it is both distributed and heterogeneous, can function in a complex dynamic environment, and accounts for relative attitude dynamics. The SOCA algorithm can correctly assign the heterogeneous agents for all target sets and avoid collision only where necessary. The algorithm was also made robust to uncertainty in the nominal trajectory. The handling of

nonlinear dynamics was improved to make the trajectories commanded by SOCA realistic and achievable. An advanced control allocation scheme was presented to handle the changing shape parameters and actuator availability during self-assembly. The simulation results show SOCA performing assignment and trajectory generation for 20-54 agents in two and three dimensional final configurations with realistic trajectories. The algorithm is experimentally validated on 6 omni-wheeled robots and four spacecraft simulators. The wheeled robots were not sufficient to make claims about the success of the algorithm, but the spacecraft simulators were used to successfully experimentally validate the optimal construction algorithm. The proposed scheme is useful for missions ranging from sparse aperture interferometric telescope construction to space colony or station construction.
This algorithm can also be used for a higher degree of heterogeneity without substantially altering the algorithm, as discussed in the auction algorithm section. While adding more agents would expand the possible final shapes even further, it would also increase system manufacturing cost and does not affect the design or execution of the algorithm substantially.

In Chapter 5, two propellant-free and ultra-low propellant docking systems were presented and compared against propulsive methods. For the tether-based system, the savings in fuel for the tether system vs the propulsive system using the optimal control was not considered large enough to justify the added mechanical complexity. The tether does offer a significant reduction in risk by its nature so overall is still considered favorable. A preliminary propellant-less maneuver was attempted with some degree of success, but to achieve the same ultra-soft dock would require further study.
The electromagnetic and propulsive docking systems presented in the second half and experimentally validated. The electromagnet-based docking system performed admirably and was able to complete docks from up to a 55 cm intersatellite distance. The electromagnet-based docking system suffers from angular drift, most likely due to imbalances and manufacturing issues in the reaction wheels. The thruster-based docking system was able to successfully capture with a fairly low relative velocity but to achieve such a low velocity the propellant

usage was high. Work remains to isolate the potential error sources in the electromagnetic docking system to reduce the drift, incorporate on-board navigation to remove the reliance on the motion capture system, and redesign the gripper to improve robustness to misalignment.

Finally in Chapter 6, we presented a motion planning algorithm, through geometrically-fixed uncooperative cluttered environments, that uses the nonlinear vehicle dynamics and minimizes the given convex cost function. Our SE–SCP algorithm has two main steps. During the spherical expansion step, the algorithm explores the workspace using a random sampling technique to generate collision-free spheres. This step generates coarse geometric paths from the start position to the goal position. During the SCP step, locally optimal trajectories are generated along those geometric paths using SCP optimization that considers both the vehicle dynamics and collision-avoidance constraints. Therefore, as newer paths are discovered by the spherical expansion step, better trajectories are generated by the SCP step. Hence the algorithm guarantees locally optimal trajectories using a finite number of samples and a globally optimal trajectory as the number of samples tends to infinity. We demonstrated the effectiveness of the SE–SCP algorithm using experimental results and numerical simulations. Therefore, the SE–SCP algorithm is suitable for fast motion planning through geometrically-fixed uncooperative cluttered environments.

7.2 Future Work

This dissertation has addressed several guidance and control challenges that face in-orbit self-assembly missions, but the other challenges exist for such an application that have not been addressed here. Some such challenges are:

- Swarm Navigation - This dissertation assumes the agent navigation is performed by another subsystem, but in reality issues in navigation can drastically affect the desired guidance and control of a system. Particularly in a self-assembling swarm, each agent's pose must be known very precisely as it approaches a dock. Leveraging the swarm's estimates of the other agents could help with this issue. If a metric could be determined to quantify the observability of each agent, it would be helpful to incorporate that into SOCA to generate fuel and observability optimal trajectories. Part of this work could involve detecting obstacles as they appear in the sensor field of view.

- Assembly Validation - For precision systems that are self-assembled like space telescopes or habitats, it is necessary to validate the precision of the structure that is created. This could be accomplished by one or more free-fliers circling the assembly with knowledge of the desired shape. The orbit design of such a system would be complex if the self-assembled structure were placed at a Lagrange point, depending on the distance requirements for accurate knowledge of the assembly.

- Realistic Docking Maneuvers - From the models and extensive experimental data collected during the testing in Chapter 5, we have a good idea of the close proximity motion during a docking maneuver. Incorporating this knowledge into SOCA in a meaningful way would improve the fidelity of the simulations. Additionally it would be useful in this work to accurately characterize the expected error profile of the flat floor facility.

- Theoretical Guarantees for SE–SCP - The algorithm developed in Chapter 6 performs quite well in practice and is shown to outperform some similar methods, but cementing the theoretical benefits of this approach would be beneficial.

References

[1] Attitude determination control systems. http://bluecanyontech.com/wp-content/uploads/2017/04/DataSheet_ADCS_07_F.pdf. Accessed: 2017-5-11.

[2] Magnetic capture docking system. https://technology.nasa.gov/patent/MSC-TOPS-63. Accessed: 2016-08-16.

[3] Overview of the dart mishap investigation results. https://www.nasa.gov/pdf/148072main_DART_mishap_overview.pdf. Accessed: 2016-12-11.

[4] Propulsion unit for cubesats. http://www.vacco.com/images/uploads/pdfs/11044000-01_PUC.pdf. Accessed: 2017-4-11.

[5] Pulsed plasma thrusters. http://www.busek.com/technologies__ppt.htm. Accessed: 2017-6-11.

[6] Cleanspace one. https://espace.epfl.ch/CleanSpaceOne_1, 2018. EPFL Space Engineering Center eSpace. Accessed: 2018-01-30.

[7] Rascal-1: A do-si-do in space. https://www.sluspacelab.com/slu04-rascal1/, 2018. Saint Louis University SSRL. Accessed 2018-02-15.

[8] Syed Jan Abas and A Salman. Geometric and group-theoretic methods for computer graphic studies of islamic symmetric patterns. In *Computer Graphics Forum*, volume 11, pages 43–53. Wiley Online Library, 1992.

[9] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011.

[10] Brij N. Agrawal and Richard E. Rasmussen. Air-bearing-based satellite attitude dynamics simulator for control software research and development. 4366(831):204–214, 2001.

[11] Ahmad Aljamali and Ebad Banissi. Normalization and exploration design method of islamic geometric patterns. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, pages 42–48. IEEE, 2003.

[12] Yassine Ariba, Denis Arzelier, Laura Sofia Urbina, and Christophe Louembet. V-bar and r-bar glideslope guidance algorithms for fixed-time rendezvous: A linear programming approach. *IFAC-PapersOnLine*, 49(17):385–390, 2016.

[13] Umair Ashun. *Dynamics and Control of Electromagnetic Satellite Formations*. PhD thesis, Massachusetts Institute of Technology, 2007.

[14] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*, pages 1917–1922. IEEE, 2012.

[15] Francesca Baldini, Saptarshi Bandyopadhyay, Rebecca Foust, Soon-Jo Chung, Amir Rahmani, Jean-Pierre de la Croix, Alexandra Bacula, Christian M Chilan, and Fred Hadaegh. Fast motion planning for agile space systems with multiple obstacles. In *AIAA/AAS astrodynamics specialist conference*, page 5683, 2016.

[16] Phillip Ball. Islamic tiles reveal sophisticated maths. `http://www.nature.com/news/2007/070219/full/news070219-9.html`. Accessed: 2017-2-28.

[17] 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. In *Proc. 9th International Workshop on Satellite Constellations and Formation Flying (IWSCFF), Boulder, Colorado*, June 19-21, 2017.

[18] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Nonlinear attitude control of spacecraft with a large captured object. *Journal of Guidance, Control, and Dynamics*, 39(4):754–769, 2016.

[19] Saptarshi Bandyopadhyay, Rebecca Foust, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Review of formation flying and constellation missions using nanosatellites. *Journal of Spacecraft and Rockets*, (0):567–578, 2016.

[20] Stephen P Banks and K Dinesh. Approximate optimal control and stability of nonlinear finite-and infinite-dimensional systems. *Annals of Operations Research*, 98(1-4):19–44, 2000.

[21] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[22] Marc Bodson. Evaluation of optimization methods for control allocation. *Journal of Guidance, Control, and Dynamics*, 25(4):703–711, 2002.

[23] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. Gusto: Guaranteed sequential trajectory optimization via sequential convex programming. In *2019 IEEE International Conference on Robotics and Automation (ICRA), IEEE*, 2019.

[24] Grant Bonin, Niels Roth, Scott Armitage, Josh Newman, Ben Risi, and Robert E Zee. Canx–4 and canx–5 precision formation flight: Mission accomplished! 2015.

[25] Joseph Bonometti. Boom rendezvous alternative docking approach. *Space*, pages 19–21, 2006.

[26] J. Bowen, A. Tsuda, J. Abel, and M. Villa. Cubesat proximity operations demonstration (cpod) mission update. In *2015 IEEE Aerospace Conference*, pages 1–8, March 2015.

[27] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge Univ. Press, Cambridge, U.K., 2004.

[28] C. P. Bridges, B. Taylor, N. Horri, C. I. Underwood, S. Kenyon, J. Barrera-Ars, L. Pryce, and R. Bird. Strand-2: Visual inspection, proximity operations amp;amp; nanosatellite docking. In *2013 IEEE Aerospace Conference*, pages 1–8, March 2013.

[29] Allen Chen. Propulsion system characterization for the spheres formation flight and docking testbed. master's thesis, 2002.

[30] Yufan Chen, Mark Cutler, and Jonathan P How. Decoupled multiagent path planning via incremental sequential convex programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961. IEEE, 2015.

[31] John Chilton. *Space grid structures.* Taylor & Francis, 2007.

[32] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[33] Soon-Jo Chung. *Nonlinear control and synchronization of multiple Lagrangian systems with application to tethered formation flight spacecraft.* PhD thesis, Massachusetts Institute of Technology, 2007.

[34] Soon-Jo Chung, Danielle Adams, Alvar Saenz-Otero, Edmund Kong, David W Miller, David Leisawitz, Enrico Lorenzini, and Steve Sell. Spheres tethered formation flight testbed: advancements in enabling nasa's specs mission. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 62680B–62680B. International Society for Optics and Photonics, 2006.

[35] Soon-Jo Chung and David W Miller. Propellant-free control of tethered formation flight, part 1: Linear control and experimentation. *Journal of guidance, control, and dynamics*, 31(3):571–584, 2008.

[36] Soon-Jo Chung, David W Miller, and Olivier L de Weck. Argos testbed: study of multidisciplinary challenges of future spaceborne interferometric arrays. *Optical Engineering*, 43(9):2156–2168, 2004.

[37] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[38] Soon-Jo Chung, Jean-Jacques E Slotine, and David W Miller. Nonlinear model reduction and decentralized control of tethered formation flight. *Journal of Guidance, Control, and Dynamics*, 30(2):390–400, 2007.

[39] Soon-Jo Chung, Jean-Jacques E Slotine, and David W Miller. Propellant-free control of tethered formation flight, part 2: Nonlinear underactuated control. *Journal of guidance, control, and dynamics*, 31(5):1437–1446, 2008.

[40] Marco Ciarcia, Roberto Cristi, and Marcello Romano. Experimental emulation of the scaled clohessy-wiltshire dynamics on a flat air-bearing testbed. In *AIAA Guidance, Navigation, and Control Conference*, page 1047, 2017.

[41] Robert Coolman. Tessellation: The geometry of tiles, honeycombs and m.c. escher. `http://www.livescience.com/50027-tessellation-tiling.html`. Accessed: 2017-3-17.

[42] R. Cowen. The wheels come off kepler. *Nature*, 497(7450):417–418, 2013.

[43] Howard D Curtis. *Orbital mechanics for engineering students*. Butterworth-Heinemann, 2013.

[44] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 42–49. IEEE, 2015.

[45] Quoc Tran Dinh and Moritz Diehl. Local convergence of sequential convex programming for nonconvex optimization. In *Recent Advances in Optimization and its Applications in Engineering*, pages 93–102. Springer, 2010.

[46] Steven Dutch. Structure of beryl and cordierite. `https://www.uwgb.edu/dutchs/Petrology/Beryl-CordStruc.htm`. Accessed: 2017-2-28.

[47] Matteo Duzzi, Lorenzo Olivieri, and Alessandro Francesconi. Tether-aided spacecraft docking procedure. 4S Symposium, 2016.

[48] Youngho Eun, Chandeok Park, and Sang-Young Park. Design and Development of Ground-Based 5-DOF Spacecraft Formation Flying Testbed. *AIAA Modeling and Simulation Technologies Conference*, pages 1–7, January 2016.

[49] Jacob Everist, Kasra Mogharei, Harshit Suri, Nadeesha Ranasinghe, Berok Khoshnevis, Peter Will, and Wei-Min Shen. A system for in-space assembly. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2356–2361. IEEE, 2004.

[50] Fariba Fahroo and I Michael Ross. Advances in pseudospectral methods for optimal control. In *AIAA guidance, navigation and control conference and exhibit*, page 7309, 2008.

[51] Wigbert Fehse. *Automated rendezvous and docking of spacecraft*, volume 16. Cambridge university press, 2003.

[52] Rebecca Foust, Soon-Jo Chung, and Fred Hadaegh. Autonomous in-orbit satellite assembly from a modular heterogeneous swarm using sequential convex programming. In *AIAA/AAS Astrodynamics Specialist Conference, AIAA SPACE and Astronautics Forum and Exposition, Long Beach, CA*, Sept. 13-16, 2016.

[53] Rebecca Foust, Soon-Jo Chung, and Fred Y Hadaegh. Solving optimal control with nonlinear dynamics using sequential convex programming. In *AIAA Scitech 2019 Forum*, page 0652, 2019.

[54] Rebecca C Foust, Soon-Jo Chung, and Fred Y Hadaegh. Real-time optimal control and target assignment for autonomous in-orbit satellite assembly from a modular heterogeneous swarm. In *AIAA/AAS Space Flight Mechanics Meeting*, 2016.

[55] Rebecca C Foust, Elena Sorina Lupu, Yashwanth Kumar Nakka, Soon-Jo Chung, and Fred Y Hadaegh. Ultra-soft electromagnetic docking with applications to in-orbit assembly. In *International Astronautical Congress*, 2018.

[56] Rebecca C Foust, Yashwanth K Nakka, Ayush Saxena, Soon-Jo Chung, and Fred Y Hadaegh. Automated rendezvous and docking using tethered formation flight. In *9th International Workshop on Satellite Constellations and Formation Flying*, 2017.

[57] Rebecca C Foust, Michelle Zhao, Suzanne Oliver, Soon-Jo Chung, and Fred Y Hadaegh. Distributed control of an evolving satellite assembly during in-orbit construction. In *International Astronautical Congress*, 2017.

[58] D Gallardo and R Bevilacqua. Six Degrees of Freedom Experimental Platform for Testing Autonomous Satellites Operations. *Proceedings of the 8th International Conference on Guidance, Navigation and Control*, 2011.

[59] Daniele Gallardo, Riccardo Bevilacqua, and Richard Rasmussen. Advances on a 6 degrees of freedom testbed for autonomous satellites operations. In *AIAA Guidance, Navigation, and Control Conference*, page 6591, 2011.

[60] Divya Garg, Michael Patterson, William W Hager, Anil V Rao, David A Benson, and Geoffrey T Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851, 2010.

[61] Mathieu Geisert and Nicolas Mansard. Trajectory generation for quadrotor based systems using numerical optimal control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 2958–2964. IEEE, 2016.

[62] Qi Gong, Fariba Fahroo, and I Michael Ross. Spectral algorithm for pseudospectral methods in optimal control. *Journal of Guidance, Control, and Dynamics*, 31(3):460–471, 2008.

[63] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming (version 1.22), May 2012. http://cvxr.com/cvx/.

[64] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[65] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[66] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[67] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, March 2014.

[68] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[69] Hari B Hablani, Myron L Tapper, and David J Dana-Bashian. Guidance and relative navigation for autonomous rendezvous in a circular orbit. *Journal of Guidance, Control, and Dynamics*, 25(3):553–562, 2002.

[70] Fred Y Hadaegh, Soon-Jo Chung, and Harish M Manohara. On development of 100-gram-class spacecraft for swarm applications. *IEEE Systems Journal*, 10(2):673–684, 2016.

[71] Markus Hehn and Raffaello D'Andrea. Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*, 31(4):877–892, 2015.

[72] A Scott Howe and Ian Gibson. Trigon robotic pairs. In *AIAA Space*, 2006.

[73] Lucas Janson, Brian Ichter, and Marco Pavone. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *The International Journal of Robotics Research*, pages 1–16, 2017.

[74] Christopher Michael Jewison. *Reconfigurable thruster selection algorithms for aggregative spacecraft systems.* PhD thesis, Massachusetts Institute of Technology, 2014.

[75] Tor A Johansen and Thor I Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013.

[76] Laura Jones and Mason A Peck. Stability and Control of a Flux-Pinned Docking Interface for Spacecraft. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–12, 2010.

[77] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[78] Hassan Khalil. *Nonlinear Systems.* Prentice Hall, 3rd edition, 2002.

[79] Edmund Mun-Choong Kong. *Spacecraft Formation Flight Exploiting Potential Fiels*. PhD thesis, Massachusetts Institute of Technology, 2002.

[80] Daniel W. Kwon. *Cryogenic Heat Pipe for Cooling High Temperature Superconductors with Application to Electromagnetic Formation Flight Satellites*. PhD thesis, Massachusetts Institute of Technology, 2009.

[81] Xinfu Liu. Fuel-optimal rocket landing with aerodynamic controls. *Journal of Guidance, Control, and Dynamics*, 42(1):65–77, 2018.

[82] Xinfu Liu and Ping Lu. Solving nonconvex optimal control problems by convex optimization. *Journal of Guidance, Control, and Dynamics*, 37(3):750–765, 2014.

[83] Xinfu Liu, Ping Lu, and Binfeng Pan. Survey of convex optimization for aerospace applications. *Astrodynamics*, 1(1):23–40, 2017.

[84] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2015.

[85] Peter J Lu and Paul J Steinhardt. Decagonal and quasi-crystalline tilings in medieval islamic architecture. *science*, 315(5815):1106–1110, 2007.

[86] Ping Lu and Xinfu Liu. Autonomous trajectory planning for rendezvous and proximity operations by conic optimization. *Journal of Guidance, Control, and Dynamics*, 36(2):375–389, 2013.

[87] Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems with state constraints. *IFAC-PapersOnLine*, 50(1):4063–4069, 2017.

[88] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3636–3641. IEEE, 2016.

[89] F Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*, volume 33. Springer, 2014.

[90] F Landis Markley, Reid G Reynolds, Frank X Liu, and Kenneth L Lebsock. Maximum torque and momentum envelopes for reaction wheel arrays. *Journal of Guidance, Control, and Dynamics*, 33(5):1606–1614, 2010.

[91] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE, 2011.

[92] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[93] David W Miller, A Saenz-Otero, J Wertz, A Chen, G Berkowski, C Brodel, S Carlson, D Carpenter, S Chen, S Cheng, et al. Spheres: a testbed for long duration satellite formation flying in micro-gravity conditions. In *Proceedings of the AAS/AIAA space flight mechanics meeting*, pages 167–179. Clearwater, Florida, January, 2000.

[94] Swati Mohan. Tools for reconfigurable control system comparisons for autonomous assembly applications. In *International Astronautical Congress, Daejon, South Korea, IAC-09 C*, volume 1, pages 12–16, 2009.

[95] Swati Mohan. *Quantative selection and design of model generation architectures for on-orbit autonomous assembly.* PhD thesis, Massachusetts Institute of Technology, 2010.

[96] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh. Swarm-keeping strategies for spacecraft under J2 and atmospheric drag perturbations. *J. Guid. Control Dyn.*, 35(5):1492 – 1506, 2012.

[97] D. Morgan, S.-J. Chung, and F. Y. Hadaegh. Spacecraft swarm guidance using a sequence of decentralized convex optimizations. In *AIAA/AAS Astrodynamics Specialist Conference*, Minneapolis, MN, August 2012. AIAA 2012-4583.

[98] D. Morgan, S.-J. Chung, and F. Y. Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *J. Guid. Control Dyn.*, 37(6):1725–1740, 2014.

[99] Daniel Morgan, Soon-Jo Chung, Lars Blackmore, Behcet Acikmese, David Bayard, and Fred Y Hadaegh. Swarm-keeping strategies for spacecraft under j_2 and atmospheric drag perturbations. *Journal of Guidance, Control, and Dynamics*, 35(5):1492–1506, 2012.

[100] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, 2014.

[101] Daniel Morgan, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, 2016.

[102] Osamu Mori and Saburo Matunaga. Formation and attitude control for rotational tethered satellite clusters. *Journal of Spacecraft and Rockets*, 44(1):211–220, 2007.

[103] ApS Mosek. The mosek optimization toolbox for matlab manual, 2015.

[104] Zoltán Nagy, Jake J Abbott, and Bradley J Nelson. The magnetic self-aligning hermaphroditic connector a scalable approach for modular microrobots. In *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, pages 1–6. IEEE, 2007.

[105] Yashwanth Kumar Nakka, Rebecca C Foust, Elena Sorina Lupu, David B Elliott, Irene S Crowell, Soon-Jo Chung, and Fred Y Hadaegh. Six degree-of-freedom spacecraft dynamics simulator for formation control research. In *2018 AAS/AIAA Astrodynamics Specialist Conference*, 2018.

[106] M. Nohmi. Mission design of a tethered robot satellite "stars" for orbital experiment. In *2009 IEEE Control Applications, (CCA) Intelligent Control, (ISIC)*, pages 1075–1080, July 2009.

[107] M Nohmi. Initial orbital performance result of nano-satellite stars-ii. In *Proceedings of the 2014 International Symposium on Artificial Intelligence, Robots and Automation in Space*, 2014.

[108] Simon Nolet and David W Miller. Autonomous docking experiments using the spheres testbed inside the iss. In *Sensors and Systems for Space Applications in Defense and Security Symposium*, volume 6555. SPIE, 2007.

[109] Mohamed Okasha, Chandeok Park, and Sang-Young Park. Guidance and control for satellite in-orbit-self-assembly proximity operations. *Aerospace Science and Technology*, 41:289–302, 2015.

[110] Lorenzo Olivieri, Riccardo Mantellato, Francesco Branz, Francesco Sansone, Alessandro Cavinato, Marco Gaino, Davide Petrillo, Alessandro Francesconi, and Enrico C Lorenzini. Cubesat mission concept for tethered electromagnetic docking demonstration. In *Tartu Conference on Space Science and Technology*, 2014.

[111] Dimitra Panagou, Matt Turpin, and Vipin Kumar. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6757–6762. IEEE, 2014.

[112] Diego Pardo, Lukas Möller, Michael Neunert, Alexander W Winkler, and Jonas Buchli. Evaluating direct transcription and nonlinear optimization methods for robot motion planning. *IEEE Robotics and Automation Letters*, 1(2):946–953, 2016.

[113] Jing Pei, Luke Murchison, Victor Stewart, James Rosenthal, Drew Sellers, Mark Banchy, Adam BenShabat, Ryan Elandt, David Elliott, and Adam K Weber. Autonomous rendezvous and docking of two 3u cubesats using a novel permanent-magnet docking mechanism. In *54th AIAA Aerospace Sciences Meeting*, 2016.

[114] Roger Penrose. Pentaplexity. *Eureka*, 39:16–32, 1978.

[115] Davide Petrillo, M Buonomo, A Cavinato, F Chiariotti, M Gaino, F Branz, R Mantellato, L Olivieri, F Sansone, A Francesconi, et al. Flexible electromagnetic leash docking system (felds) experiment from design to microgravity testing. In *66Th International Astronautical Congress, IAC-15 E*, volume 2, 2015.

[116] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[117] Martin W Regehr, Ahmet B Acikmese, Asif Ahmed, M Aung, KC Clark, P MacNeal, J Shields, G Singh, R Bailey, C Bushnell, et al. The formation control testbed. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 1, pages 557–564. IEEE, 2004.

[118] Arthur Richards, Tom Schouwenaars, Jonathan P How, and Eric Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

[119] John W Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295. IEEE, 2013.

[120] I. M. Ross and F. Fahroo. Legendre pseudospectral approximations of optimal control problems. *Lecture Notes in Control and Information Systems*, 295:327–342, 2003.

[121] K. Saulnier, D. Pérez, R. C. Huang, D. Gallardo, G. Tilton, and R. Bevilacqua. A six-degree-of-freedom hardware-in-the-loop simulator for small spacecraft. *Acta Astronautica*, 105(2):444–462, 2014.

[122] Daniel P. Scharf, Jason A. Keim, and Fred Y. Hadaegh. Flight-like ground demonstrations of precision maneuvers for spacecraft formations - Part II. *IEEE Systems Journal*, 4(1):96–106, 2010.

[123] Markus Schlotterer, Eviatar Edlerman, Federico Fumenti, Pini Gurfil, Stephan Theil, and Hao Zhang. On-ground testing of autonomous guidance for multiple satellites in a cluster. In *Proceedings of the 8th International Workshop on Satellite Constellations and Formation Flying*, 06 2015.

[124] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[125] Jana L. Schwartz, Mason A. Peck, and Christopher D. Hall. Historical Review of Air-Bearing Spacecraft Simulators. *Journal of Guidance, Control, and Dynamics*, 26(4):513–522, 2003.

[126] Jana Lyn Schwartz. *The distributed spacecraft attitude control system simulator: from design concept to decentralized control.* PhD thesis, Virginia Tech, 2004.

[127] Samuel Schweighart. *Electromagnetic Formation Flight Dipole Solution Planning.* PhD thesis, Massachusetts Institute of Technology, 2005.

[128] Jungwon Seo, Mark Yim, and Vipin Kumar. Assembly planning for planar structures of a brick wall pattern with rectangular modular robots. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1016–1021. IEEE, 2013.

[129] Wei-Min Shen, Peter Will, and Berok Khoshnevis. Self-assembly in space via self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation, 2003.*, volume 2, pages 2516–2521. IEEE, 2003.

[130] Malcolm D Shuster. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[131] David C Sternberg, Christopher Pong, Nuno Filipe, Swati Mohan, Shawn Johnson, and Laura Jones-Wilson. Jet propulsion laboratory small satellite dynamics testbed simulation: On-orbit performance model validation. *Journal of Spacecraft and Rockets*, 55(2):322–334, 2017.

[132] Giri P Subramanian, Rebecca Foust, Derek Chen, Stanley Chan, Younes Taleb, Dayne L Rogers, Jobin Kokkat, Saptarshi Bandyopadhyay, Daniel Morgan, Soon-Jo Chung, et al. Information-driven systems engineering study of a formation flying demonstration mission using six cubesats. In *53rd AIAA Aerospace Sciences Meeting*, page 2043, 2015.

[133] Nicholas Swain and Shadhanan Manickavasagar. A combined fault detection, identification and reconfiguration system based around optimal control allocation. In *Fault Tolerant Flight Control*, pages 399–422. Springer, 2010.

[134] Michael Szmuk, Utku Eren, and Behcet Acikmese. Successive convexification for mars 6-dof powered descent landing guidance. In *AIAA Guidance, Navigation, and Control Conference*, page 1500, 2017.

[135] Panagiotis Tsiotras. Astros: A 5dof experimental facility for research in space proximity operations. *Advances in the Astronautical Sciences*, 151:717–730, 01 2014.

[136] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic Foundations of Robotics X*, pages 175–190. Springer, 2013.

[137] Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.

[138] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009.

[139] Sasi Prabhakaran Viswanathan, Amit Sanyal, and Lee Holguin. Dynamics and control of a six degrees of freedom ground simulator for autonomous rendezvous and proximity operation of spacecraft. In *AIAA Guidance, Navigation, and Control Conference*, page 4926, 2012.

[140] Eric W. Weisstein. Space-filling polyhedra. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/Space-FillingPolyhedron.html`. Accessed: 2016-12-16.

[141] Markus Wilde, Brian Kaplinger, Tiauw Go, Hector Gutierrez, and Daniel Kirk. Orion: A simulation environment for spacecraft formation flight, capture, and orbital robotics. In *Aerospace Conference, 2016 IEEE*, pages 1–14. IEEE, 2016.

[142] Mark Yim, Kimon Roufas, David Duff, Ying Zhang, Craig Eldershaw, and Sam Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

[143] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

[144] Jingjin Yu, Soon-Jo Chung, and Petros G Voulgaris. Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies. *IEEE Transactions on Automatic Control*, 60(2):327–341, 2015.

[145] Richard Zappulla, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. *Journal of Spacecraft and Rockets*, 54(4):825–839, 2017.

[146] Victor Zykov, Andrew Chan, and Hod Lipson. Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, pages 3–6, 2007.

Appendix A

Tether Based Docking Dynamics

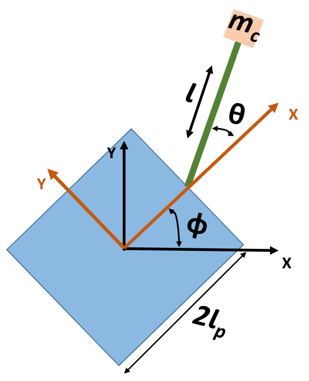


Figure A.1: Coordinate frames of Parent spacecraft with a tethered child.

$$M(\theta,\ell)\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\ell}\end{bmatrix}dx + C(\dot{\phi},\dot{\theta},\dot{\ell},\phi,\theta,\ell)\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\ell}\end{bmatrix}dx + H(\dot{\theta},\dot{\xi},\dot{\ell}) = \tau \tag{A.1}$$

$$M(\theta,\ell) = \begin{bmatrix} m_c\ell^2 + 2m_c\ell\ell_p\cos\theta + m_c\ell_p^2 + I_z & m_c\ell\left(\ell + \ell_p\cos\theta\right) & m_c\ell_p\sin\theta \\ m_c\ell\left(\ell + \ell_p\cos\theta\right) & m_c\ell^2 & 0 \\ m_c\ell_p\sin\theta & 0 & m_c \end{bmatrix} \tag{A.2a}$$

$$C(\dot\phi,\dot\theta,\dot\ell,\theta,\ell) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & 0 \end{bmatrix} \tag{A.2b}$$

$$H(\dot\phi,\dot\theta,\dot\ell,\theta,\ell) = \begin{bmatrix} -m_c\left(2\dot\ell(\ell+\ell_p\cos\theta)(\dot\phi+\dot\theta) - \dot\theta^2\ell\ell_p\sin\theta - 2\dot\phi\dot\theta\ell\ell_p\sin\theta\right) \\ -m_c(2\dot\ell\dot\phi\ell + 2\dot\ell\dot\theta\ell + \dot\ell\dot\phi\ell_p\cos\theta - \dot\phi\dot\theta\ell\ell_p\sin\theta) \\ -\dot\phi\dot\theta\ell_p m_c\cos\theta \end{bmatrix} \tag{A.2c}$$

$$\tau = \begin{bmatrix} \tau_Z & 0 & u_L \end{bmatrix}^T \tag{A.2d}$$

$$\begin{aligned}
c_{11} &= \dot\ell(\ell m_c + \ell_p m_c\cos\theta) - \dot\theta\ell\ell_p m_c\sin\theta; \\
c_{12} &= \dot\ell(\ell m_c + \ell_p m_c\cos\theta) - \dot\phi\ell\ell_p m_c\sin\theta - \dot\theta\ell\ell_p m_c\sin\theta; \\
c_{13} &= \dot\phi(\ell m_c + \ell_p m_c\cos\theta) + \dot\theta(\ell m_c + \ell_p m_c\cos\theta); \\
c_{21} &= \dot\ell((\ell m_c)/2 + (m_c(\ell+\ell_p\cos\theta))/2 - (\ell_p m_c\cos\theta)/2) + \dot\phi\ell\ell_p m_c\sin\theta; \\
c_{22} &= \dot\ell\ell m_c; \\
c_{23} &= \dot\phi((\ell m_c)/2 + (m_c(\ell+\ell_p\cos\theta))/2 - (\ell_p m_c\cos\theta)/2) + \dot\theta\ell m_c; \\
c_{31} &= -\dot\phi(\ell m_c + \ell_p m_c\cos\theta) - \dot\theta\ell m_c; \\
c_{32} &= -\dot\phi((\ell m_c)/2 + (m_c(\ell+\ell_p\cos\theta))/2 - (\ell_p m_c\cos\theta)/2) - \dot\theta\ell m_c;
\end{aligned} \tag{A.3}$$

Appendix B

Docking Experiment Analysis Tables

Test	Initial Separation Distance [m]	Final Relative Velocity [mm/sec]	Docking Time [sec]	Total Firing Time [sec]
1	1.99	0.56	61.00	31.79
2	4.00	-0.34	92.00	52.61
3	4.49	2.00	97.50	52.12
4	5.50	0.88	107.50	60.41
5	5.46	0.22	107.50	46.66
6	5.42	0.17	107.50	68.17
7	5.38	0.63	78.00	43.20
8	5.38	0.63	78.00	43.20
9	4.81	0.34	101.00	86.59
10	2.17	0.24	65.50	44.41
11	3.25	-1.92	81.50	70.86
12	4.50	0.66	96.50	67.39
13	4.50	0.66	96.50	67.39
14	1.06	-0.95	39.50	23.25

Table B.1: Thruster-based Docking - Test Results

Test	Initial Separation Distance [m]	Final Relative Velocity [mm/sec]	Docking Time [sec]	Energy [Wh]
1	0.39	0.689	73.50	0.97
2	0.40	4.349	73.60	0.98
3	0.39	-1.911	73.20	0.95
4	0.39	7.093	73.50	0.97
5	0.38	0.000	72.60	0.92
6	0.38	6.867	72.29	0.95
7	0.40	4.103	73.90	1.00
8	0.40	-0.226	73.90	0.97
9	0.47	0.016	80.40	1.13
10	0.39	-0.651	72.80	0.95
11	0.39	0.689	73.50	0.85
12	0.40	0.065	74.10	0.90
13	0.41	0.003	74.90	0.86
14	0.55	1.019	87.10	1.34

Table B.2: Electromagnet-based Docking - Test Results

Appendix C

Experiment Videos

Electromagnet-based Docking:

`https://youtu.be/q2t74AjeQE8`

Thruster-based Docking:

`https://www.youtube.com/watch?v=-k0IJelQVjk`

SOCA Validation:

`https://www.youtube.com/watch?v=62cngDR1k-E`