Real-Time Autonomous Model Predictive Control of Spacecraft Rendezvous and Docking with Moving Obstacles

by
Courtney Bashnick

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Aerospace Engineering

Carleton University
Ottawa, Ontario

This work is dedicated to my family.

Abstract

Autonomous rendezvous and docking, whereby two spacecraft come into close proximity and subsequently make mechanical contact, is used for on-orbit servicing missions. The safety of these missions is endangered by space debris and other hazards that pose a threat for collisions. The guidance algorithm onboard a spacecraft is responsible for planning a safe path to a target spacecraft and must actively avoid these hazards for the success of the mission.

This thesis presents a real-time optimal guidance algorithm for autonomous path-planning with moving obstacles based upon the Model Predictive Control framework. Numerical simulations are completed in two- and three-dimensions to prove the functionality of the algorithm. The current laboratory facility was upgraded to validate the real-time collision avoidance capabilities of the algorithm. The experiments are, to the best of the author's knowledge, the first to demonstrate the moving obstacle avoidance capabilities of a Model Predictive Controller for spacecraft rendezvous and docking.

Acknowledgements

To my supervisor, Dr. Steve Ulrich, I express my sincerest gratitude for your guidance and the discussions that steered me over the course of this research.

To my professors at Carleton University, I thank you for imparting the knowledge that helped with this thesis. I am also indebted to my professors at the University of Calgary who nurtured my growth as an academic. I owe my deepest thanks to many, but I would like to extend a special thank you to Dr. Johnathan Burchill for your mentorship and your encouragement without which I would not be where I am today.

To my parents, there are no words that can fully express how appreciative I am for all that you do. Thank you for supporting me in every endeavour. To my boyfriend, Ivan, I thank you for your friendship and willingness to learn alongside me during this thesis. To my friends, I thank you for cheering me on.

This project would not have been possible without the hard work of numerous individuals who designed and built the laboratory. In particular, I would like to thank Kirk Hovell for showing me the ropes and Alex Crain for getting the laboratory software up-to-date for experiments with the third platform, BLUE. I also thank Nolan Chafe for helping with the build of BLUE, and Parker Stewart for assisting during the experimental phase of this thesis.

Lastly, I acknowledge those who invested in my future. The New Technologies for Canadian Observatories (NTCO) program offered support in numerous ways through unique professional skills training and workshops, as well as the opportunity to intern at Honeywell Aerospace in fulfillment of its program requirements. This research was financially supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Alexander Graham Bell Canada Graduate Scholarship, the NSERC-CREATE NTCO program, as well as entrance scholarships and a teaching assistant position provided by the Mechanical and Aerospace Engineering Department.

Table of Contents

List of Tables

List of Figures

Nomenclature

Roman Symbols

$\mathbf{A}$	Dynamic model continuous state matrix
$\mathbf{A}_d$	Dynamic model discrete state matrix
a	Semi-major axis of orbit
a	Obstacle shape parameter along the x-axis
$\mathbf{B}$	Dynamic model continuous control matrix
$\mathbf{B}_d$	Dynamic model discrete control matrix
$\mathbf{b}$	Compact equality constraint column vector
b	Obstacle shape parameter along the y-axis
$\mathbf{C}$	Compact equality constraint matrix
c	Obstacle shape parameter along the z-axis
$\mathbf{D}$	Compact linear obstacle avoidance constraint matrix
$\mathbf{d}$	Column vector associated with the gradient of the logarithmic barrier
d	Value of the Lagrange dual problem
$\mathbf{E}$	Compact linear obstacle avoidance constraint matrix per sampling instant
$\mathbf{e}$	Error of the system state from the desired state
$\mathcal{F}$	Reference frame
$\mathbf{F}$	Constraint matrix for inequality constraints
$\vec{F}$	Gravitational force vector
$\mathbf{f}$	Constraint column matrix for inequality constraints
$f(\cdot,\cdot)$	State transition function
$f(\cdot)$	Nonlinear ellipsoid function
$f_i(\cdot)$	Inequality constraint function
$\mathbf{G}$	Compact entry cone matrix of normal vectors
G	Gravitational constant

$g(\cdot,\cdot)$	Lagrange dual function
$\mathbf{H}$	Compact cost matrix
$\mathbf{h}$	Compact inequality constraint column vector
$h_i(\cdot)$	Equality constraint function
$\vec{\mathcal{I}}$	Unit vector of the ECI frame
$\mathbf{I}$	Identity matrix
$I(\cdot)$	Indicator function
$\hat{I}(\cdot)$	Approximate indicator function
$J(\cdot)$	Performance index; objective function; cost function
$\mathbf{k}$	Compact maximum thrust limitation column vector
k	The current sampling instant, or a general instant within the horizon
$\vec{\mathcal{L}}$	Unit vector of the LVLH frame
$\mathbf{L}$	Cholesky factor
$L(\cdot,\cdot,\cdot)$	Lagrangian of the minimization problem
l	Half the sidelength of a SPOT platform
l	Number of inequality constraints
$\ell(\cdot,\cdot)$	Stage cost
M	Mass of object
m	Mass of object
m	Number of control variables
m	Entry cone hyperplane slope
N	Horizon length
$\hat{\mathbf{n}}$	Entry cone hyperplane normal vector
n	Mean motion of target spacecraft
n	Number of state variables
n_{f_i}	Number of inequality constraints, $n_{f_i} = (l_x + l_u)N$
n_{h_i}	Number of equality constraints, $n_{h_i} = nN$
n_z	Number of optimization variables, $n_z = (n+m)N$
$\mathbf{P}$	Compact inequality constraint matrix
p	Value of the cost function

$\mathbf{Q}$	Cost function symmetric weight matrix of quadratic state errors
$\overline{\mathbf{Q}}$	Block matrix in $\mathbf{\Phi}$
$\tilde{\mathbf{Q}}$	Block matrix in $\mathbf{\Phi}^{-1}$
$\mathbf{q}$	Cost function column weight vector of linear state errors
$\mathbf{R}$	Cost function symmetric weight matrix of quadratic controls
$\overline{\mathbf{R}}$	Block matrix in $\mathbf{\Phi}$
$\tilde{\mathbf{R}}$	Block matrix in $\mathbf{\Phi}^{-1}$
$\vec{R}$	Position vector relative to an inertial reference frame
$\mathbf{r}$	Cost function column weight vector of linear controls
$\mathbf{r}$	Residual
$\mathbf{r}$	Chaser position
$\mathbf{r}_0$	Expansion point on the KOZ of an obstacle
$\mathbf{r}_d$	Dual residual
$\mathbf{r}_p$	Primal residual
$\bar{\mathbf{r}}$	Position of object over the horizon
r	Magnitude of position vector
r_{cone}	Entry cone activation distance
r_{hold}	Holding radius
$r_{hold,min}$	Minimum holding radius
r_{KOZ}	Radius of the KOZ
$\vec{r}$	Position vector
$\mathbf{S}$	Cost function weight matrix of cross-term states and controls
$\mathbf{S}$	Elliptical shape matrix of obstacles
T_d	Discretization time
t	Step size
$\mathbf{U}$	Compact maximum thrust limitation matrix
$\mathbf{u}$	Control input matrix
u	Control force
u_{max}	Maximum control force
$\vec{v}$	Velocity vector

$\mathbf{W}$	Partition matrices solved via forward and backward substitution
$\mathbf{x}$	State matrix
$\mathbf{x}_t$	Desired state matrix
x	Radial position of the chaser from the target spacecraft
x_0	Center of ellipsoid along the x-axis
$\mathbf{Y}$	Schur complement of $\boldsymbol{\Phi}$
$\mathbf{Y}_{ij}$	Block matrix in $\mathbf{Y}$
y	In-track position of the chaser from the target spacecraft
y_0	Center of ellipsoid along the y-axis
$\mathbf{z}$	Column vector of optimization variables
$\mathbf{z}_t$	Column vector of desired states
z	Cross-track position of the chaser from the target spacecraft
z_0	Center of ellipsoid along the z-axis

Greek Symbols

α	Backtracking line search residual refinement parameter
$\boldsymbol{\beta}$	Right-hand side of the equation to solve for $\Delta\boldsymbol{\nu}$
β	Backtracking line search step size factor
γ	Dynamic holding radius decrement factor
$\Delta\mathbf{z}$	Primal search direction
$\Delta\boldsymbol{\nu}$	Dual search direction
$\Delta\boldsymbol{\xi}$	Primal-dual Newton step
ϵ	Residual tolerance
ζ	Dynamic holding radius attitude error
η	Dynamic holding radius positional error
θ	Attitude of spacecraft
θ_h	Entry cone half angle
$\dot{\theta}$	True anomaly rate
κ	Positive tuning parameter

$\boldsymbol{\lambda}$	Lagrange multiplier vector for λ
λ_i	Lagrange multiplier associated with the ith inequality constraint
μ	Gravitational parameter
$\boldsymbol{\nu}$	Lagrange multiplier vector for ν
ν_i	Lagrange multiplier associated with the ith equality constraint
$\boldsymbol{\xi}$	Primal-dual variable
$\vec{\rho}$	Position of chaser relative to target spacecraft
τ	Control torque
$\boldsymbol{\Phi}$	Hessian of the augmented cost function
$\phi(\cdot)$	Logarithmic barrier

Superscripts

-1	Matrix inversion
T	Matrix transpose
$(\cdot)$	Instant within the horizon
$+$	Subpartition solved via forward substitution
$\times$	Cross product matrix operator
$*$	Associated with an optimal solution

Subscripts

0	Indicates an initial value
c	Indicates the chaser spacecraft
$c1$	Indicates hyperplane 1 of the entry cone
$c2$	Indicates hyperplane 2 of the entry cone
$dock$	Indicates the docking port
f	Indicates the final state
I	Earth-Centered Inertial Frame

L	Local-Vertical Local-Horizontal Frame
obj	Indicates the obstacle
$\overline{\mathbf{Q}}_j$	Associated with the $\overline{\mathbf{Q}}^{(j)}$ block matrix
$\overline{\mathbf{R}}_j$	Associated with the $\overline{\mathbf{R}}^{(j)}$ block matrix
sc	Indicates a general spacecraft
t	Indicates the target spacecraft
tar	Indicates the target spacecraft
u	Indicates the control variables
x	Indicates the state variables
x	X-component within the respective reference frame
y	Y-component within the respective reference frame
z	Z-component within the respective reference frame
$\oplus$	Indicates a planet, namely the Earth

Other Symbols

$\mathbb{R}$	Set of real numbers
D	Derivative
∂	Partial derivative
$\nabla(\cdot)$	Gradient operator
$\nabla^2(\cdot)$	Hessian operator
$\sum(\cdot)$	Summation operator

Acronyms

AAPF	Adaptive Artificial Potential Function
APF	Artificial Potential Function
ARVD	Autonomous Rendezvous and Docking
ASAT	Anti-Satellite

COM	Center of Mass
CPU	Central Processing Unit
DOF	Degrees of Freedom
ECI	Earth-Centered Inertial
ESA	European Space Agency
FHOLOC	Finite-Horizon Open-Loop Optimal Control
GNC	Guidance, Navigation, and Control
GPIO	General Purpose Input/Output
IDVD	Inverse Dynamics in the Virtual Domain
IPOPT	Interior Point OPTimizer
ISNM	Infeasible Start Newton Method
ISS	International Space Station
KKT	Karush-Kuhn-Tucker
KOZ	Keep-Out-Zone
LED	Light-Emitting Diode
LEO	Low Earth Orbit
LQ	Linear-Quadratic
LQR	Linear Quadratic Regulator
LVLH	Local-Vertical Local-Horizontal
MATLAB	Matrix Laboratory
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MPC	Model Predictive Control
NASA	National Aeronautics and Space Administration
NMPC	Nonlinear Model Predictive Control
OLOC	Open-Loop Optimal Control
OSAM-1	On-Orbit Servicing, Assembly, and Manufacturing mission 1
PD	Proportional Derivative
POSEIDYN	Proximity Operation of Spacecraft Experimental hardware-In-the-loop DYNamic
QP	Quadratic Programming

RAM	Random Access Memory
RP3	Raspberry Pi 3
RVD	Rendezvous and Docking
SPHERES	Synchronized Position Hold Engage and Reorient Experimental Satellite
SPOT	Spacecraft Proximity Operations Testbed
SRCL	Spacecraft Robotics and Control Laboratory
SSO	Sun-Synchronous Orbit
TEAMS	Test Environment for Applications of Multiple Spacecraft

Chapter 1

Introduction

1.1 Motivation

A spacecraft rendezvous and docking (RVD) mission brings two independently orbiting spacecraft into close proximity and eventual mechanical contact through a series of orbital maneuvers with the goal of performing on-orbit servicing activities: inspection, refurbishing, refuelling, relocation, and assembly. The RVD mission can be performed with a crew, as was done to service the Hubble Space Telescope in the 1990s and early 2000s, but is now commonly performed autonomously, such as the supplying of the International Space Station (ISS) using the SpaceX Cargo Dragon spacecraft, Progress spacecraft, and Cygnus spacecraft. More planned autonomous RVD (ARVD) missions to be launched after 2025 include the European Space Agency's (ESA's) ClearSpace-1 [1] and the National Aeronautics and Space Administration's (NASA's) On-Orbit Servicing, Assembly, and Manufacturing mission 1 (OSAM-1) [2].

ARVD is a well-established technology that offers benefits over its human-in-the-loop alternative [3]. Increased autonomy is necessary for situations when ground intervention is impractical, e.g., Mars sample return missions [4], due to communication delays to the ground. Autonomy is also necessary when contingencies occur during an RVD mission whilst two spacecraft are in close proximity. It may be infeasible for ground controllers to react to the contingency within a reasonable amount of time or the real-time state information of the spacecraft may not be attainable due to a limited communication bandwidth [5]. In either case, the spacecraft must have

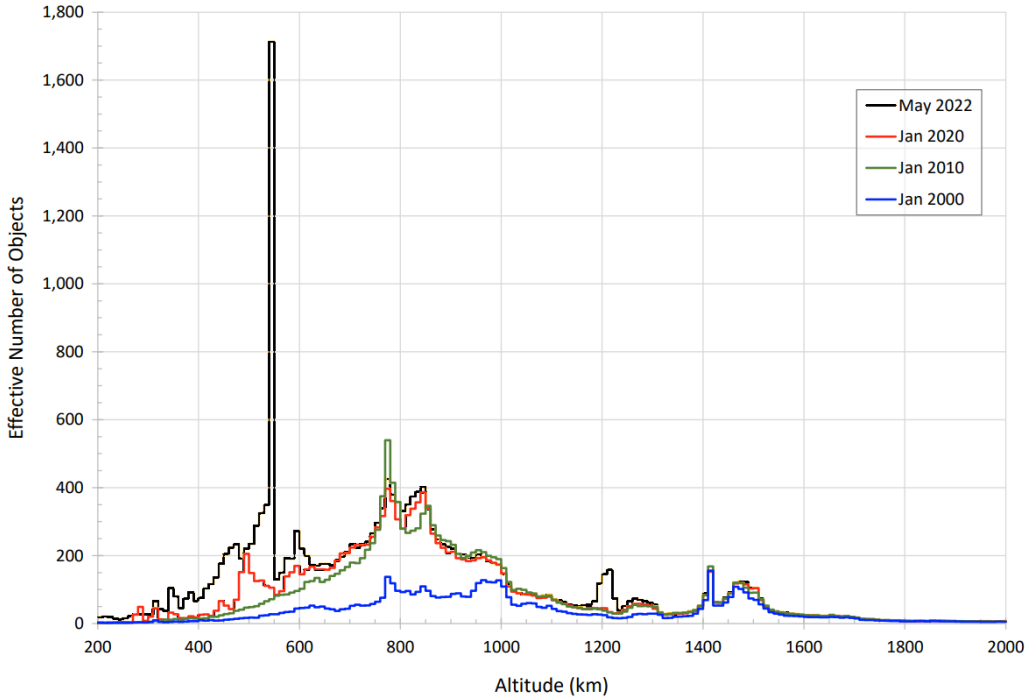


Figure 1.1: Effective numbers of objects > 10 cm per 10 km altitude bin between 200 and 2000 km altitude at four different epochs. Sourced from [6].

a reasonable level of autonomy to safely react to the situation.

One serious threat to the safety of spacecraft, particularly those with altitudes below 1000 km in low Earth orbit (LEO), is the increasing population of these orbits. As spacecraft malfunction or run out of fuel, they lose the ability to control themselves in their orbit. The result is a satellite-sized bullet racing around the Earth at approximately 27 000 km/h. Figure 1.1 shows the effective number of objects larger than 10 cm per 10 km altitude bin at four different epochs. The drivers that increase the number of objects are a combination of anti-satellite (ASAT) tests whereby a satellite is intentionally destroyed, accidental collisions between spacecraft, and the launch of large constellations including SpaceX's Starlink fleet [6]. The steep increase in objects below 600 km is attributed to Starlink satellites and CubeSats. The objects around 800 km altitude are fragments of an ASAT test and an accidental collision. The recent increase in objects around 1200 km altitude is from the launch of the OneWeb large constellation.

As LEO becomes increasingly crowded, there is a higher likelihood that defunct

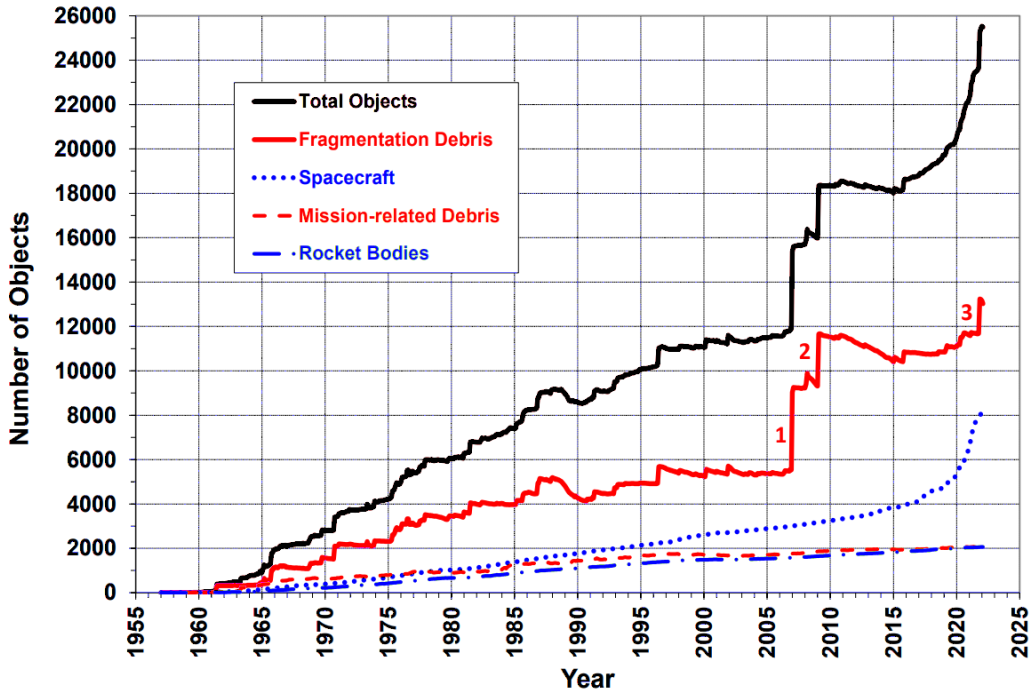


Figure 1.2: The increase of cataloged objects over time using data from March 2022. The three upward jumps in fragmentation debris correspond to (1) the ASAT test conducted by China in 2007, (2) the accidental collision between Iridium 33 and Cosmos 2251 in 2009, and (3) the ASAT test conducted by the Russian Federation in November 2021. Sourced from [7].

satellites and fragments of spacecraft, called space debris, will collide with each other and other functional spacecraft to produce even more debris. Indeed, the growing trend of space objects over time is shown in Fig. 1.2. There are three instances pointed out that contributed greatly to the current amount of space debris [7]: The first is the ASAT test conducted by China in 2007 and is considered to be the single worst contamination event of LEO with thousands of debris thrown into long-duration orbits [8]; Second is the accidental collision between Iridium 33 and Cosmos 2251 in 2009; Third is the ASAT test conducted by the Russian Federation in November 2021. The exponential increase of space debris is called *Kessler Syndrome*, named after the scientist who studied the phenomenon in 1978 [9].

For the successful execution of ARVD missions, it is important that spacecraft are able to actively avoid collisions with space debris. The system onboard the spacecraft responsible for determining its trajectory is the Guidance, Navigation,

and Control (GNC) system, with the guidance algorithm in particular that conducts collision avoidance path-planning. The objective of the guidance algorithm is to nullify the relative motion of two spacecraft whilst satisfying operational requirements, e.g., limitations to the engine thrust output, and safety requirements, e.g., avoiding space debris. Extensive research is being done in the field of guidance algorithms for ARVD, and the development of faster processors has since encouraged the advancement of *real-time* autonomous guidance algorithms [3]. These novel real-time guidance algorithms for ARVD are the motivation for this research.

1.2 Problem Statement

The need for safe, precise, and reliable spacecraft RVD operations continues to drive enabling technologies such as GNC towards autonomy. Autonomous guidance algorithms for RVD fall under two categories: analytical methods and optimization methods. Analytical methods, as evident by their name, simply solve a closed-form expression to find a trajectory. Optimization methods, on the other hand, are founded from optimal control theory and are studied to a greater extent due to their ability to optimize a specified performance index, e.g., fuel consumption or time of flight. The trade-off between analytical and optimization methods is in the form of computational efficiency versus optimality of trajectory: Analytical methods are less computationally intensive and can therefore be easily implemented in real-time on space hardware; Optimization methods are much more computationally expensive, although have a supremacy over analytical methods since their trajectory is optimized.

Next generation guidance algorithms for ARVD are classified as high priority technology by the United States National Research Council for the 2011-2021 decade and beyond [10]. Furthermore, it is recommended that key features of these algorithms include:

1. *Optimality*: The trajectory determined by the algorithm should be optimal, minimizing a parameter such as fuel consumption.

2. *Real-time implementable*: The algorithm must execute onboard the spacecraft in real-time.

Thus, the technological challenge to be addressed in this thesis is the development of an optimal RVD path-planning algorithm that has obstacle avoidance capabilities and can be implemented in real-time onboard spacecraft.

1.3 Previous Work

Research into the development and testing of spacecraft path-planning algorithms for RVD began in the 1960s and has been highly active since that time. A comprehensive literature review of the developments in the autonomous GNC field for aerospace vehicles over the past fifteen years can be found in Chai et al. [11].

The original guidance algorithms for RVD were limited due to the low computational capabilities of spacecraft at the time. A key algorithm developed was the glideslope method [12], in which the active "chaser" spacecraft moves on a straight path to the passive "target" spacecraft, and was used in the Space Shuttle programme for path-planning in real-time. Nolet [13] used the glideslope method for autonomous RVD with a cooperative tumbling target using the Synchronized Position Hold Engage and Reorient Experimental Satellite (SPHERES) units. Although the glideslope method is a light algorithm that can be computed in real-time, it does not have the ability to optimize the trajectory or employ path constraints such as obstacle avoidance or plume impingement in the problem.

More recently, Hovell and Ulrich [14] developed a guidance strategy that employs deep reinforcement learning such that complex guidance strategies can be learned indirectly through reward-based training. The guidance algorithm employed here generates reference velocity commands in real-time that are fed to a conventional controller to track. Experiments using the Spacecraft Robotics and Control Laboratory's (SRCL's) Spacecraft Proximity Operations Testbed (SPOT) successfully validated the guidance policy with a rotating target spacecraft and virtual static obstacle, as well as demonstrated its robustness to perturbations caused by manual interference [15]. Advantages of using neural networks include their approximation of optimal guidance algorithms, however, downsides include their long training period.

Model Predictive Control (MPC) is an optimization method whereby a user-defined cost function is minimized over a period of time, called the horizon, subject to a set of constraint equations. MPC has been applied successfully in various industries, but has primarily been used in systems with slow dynamics due to its computational complexity. Due to its ability to handle constraints meanwhile explicitly minimizing a specified performance index, MPC is an effective control strategy for spacecraft operations. Richards and How [16] proposed an MPC strategy that requires a mixed-integer linear program be solved at every control cycle. The introduction of binary weights to switch on or off different convex constraints makes the algorithm computationally fast, but yields a conservative trajectory. Variations on this work include using a variable horizon [17] and constraint tightening whereby the algorithm is made more robust to disturbances [18]. Simulations comparing the glideslope algorithm with fixed and variable horizon MPC with and without disturbances show that the MPC approach commands a lower control effort.

The optimal control problem can also be formulated using the linear-quadratic (LQ) MPC framework. Previous work has considered docking to both a stationary and rotating target under planar dynamics and the Clohessy-Wiltshire-Hill, or simply Hill's, equations in two- and three-dimensions with and without static obstacle avoidance capabilities [19–23]. The authors, however, did not consider scenarios with both a rotating target and moving obstacles. The obstacle constraints themselves are implemented using rotating hyperplanes, a dual hyperplane approach, and a direct linearization method [24]. A nonlinear ellipsoid approach can also be used for the obstacle avoidance constraint, as was done for moving obstacle simulations [25], static obstacle experiments [23], and rotating target experiments with a dynamic holding radius on the Naval Postgraduate School's Proximity Operation of Spacecraft Experimental hardware-In-the-loop DYNamic (POSEIDYN) simulator [26]. However, using nonlinear constraints renders the problem to a nonlinear MPC (NMPC) framework which is more computationally expensive to solve. The LQ-MPC approach can be combined with a Linear Quadratic Regulator (LQR) controller, MPC+LQR, to correct any residual errors following the controls from MPC alone. It was shown in hardware-in-loop experiments at the Test Environment for Applications of Multiple

Spacecraft (TEAMS) facility that the LQR controller increased the required propellant by 21% and achieved similar performance to standalone LQ-MPC [27].

Work has also gone into developing MPC controllers that handle disturbances. Robust MPC can be formulated using a constraint tightening approach [28], a chance-constrained model whereby probabilistic Gaussian disturbances are incorporated into the controller constraints [29, 30], or an additional feedback controller used to keep the actual state within a "tube" around the nominal trajectory [31, 32], called tube-based MPC. Here, a feasible reference trajectory is computed in inertial space using the nominal target motion and is then tracked in the target spacecraft body frame to intrinsically account for any variation in the target's motion. However, due to this variation the chaser departs from the reference trajectory in the inertial space, and the extent of this deviation is statistically estimated using a Monte-Carlo-based approach. This method guarantees convergence of tracking the reference trajectory and was successfully validated using a detailed simulation environment [33] and experiments using NASA's Astrobee robots [34]. Of note, these experiments included virtual moving obstacles attached to the target spacecraft, i.e., solar panels and an antenna, however, the MPC controller is used only for tracking purposes while the path-planning is computed through extrapolation of an offline-generated look-up table. The performance of robust tube-based MPC was compared to a classical MPC formulation on POSEIDYN to verify its superior capabilities of handling disturbances [35]. For real-time implementation of the tube-based MPC method, however, the evaluation of the constraints to determine the tubes must be done offline due to their large computational expense.

The performance of MPC has additionally been compared to different guidance algorithms, including the Artificial Potential Function (APF), Adaptive APF (AAPF), and Inverse Dynamics in the Virtual Domain (IDVD) approaches. Similar to the glideslope method, the APF approach is an analytically-based method that solves for the trajectory to a target spacecraft in real-time [36]. The APF method uses a potential in state-space and guides the spacecraft along a valley to the desired target position. By defining a repulsive potential in the relevant regions, this method can handle path constraints such as obstacles. Dong et al. [37] show that their APF-based

controller converges faster than MPC to a smooth trajectory, but has a comparatively higher control effort due to the APF method's inability to optimize the trajectory.

A similar guidance algorithm to the APF method uses Lyapunov vector fields [38] to calculate the desired velocities that guide the spacecraft on a circular trajectory to an uncooperative target in elliptical orbit. The user-defined Lyapunov function can be chosen such that the guidance law consists of simple analytical equations that consider path constraints and can be solved by a computationally-limited spacecraft. This approach, however, does not produce optimal trajectories.

APF can be combined with an LQR controller to produce a path-planning method that optimizes the trajectory as well as allows for collision avoidance [39]. This algorithm was tested using SPHERES and was deemed successful, although the docking was performed with a static virtual obstacle and virtual target due to testing limitations. The LQR/APF technique has also been adapted to use a so-called wall-following technique to overcome local minima [40]. This algorithm was successfully validated on the POSEIDYN experimental testbed.

An alternative variation on the APF method is AAPF, which uses a time-dependent form of the artificial potentials and therefore increases the method's optimality [41]. Numerical Monte Carlo simulations and experimental test results on POSEIDYN [42,43] show that the AAPF method commands less control effort with a faster convergence time as compared to the APF method. Experiments were also conducted that compared APF and AAPF to other real-time guidance methods including MPC [44] and found that overall MPC-based methods expended the least control effort per unit of maneuvering time out of all of the proposed algorithms. The experiments involving MPC used only static obstacles.

IDVD quasi-optimizes a trajectory and can be implemented in real-time due to its algebraic nature [45]. A drawback of IDVD, however, is the exponential increase in computational time with an increase in the number of nodes in the problem, limiting its ability to be scaled to larger control problems. Additionally, IDVD is sensitive to the initial guess making it difficult to implement. IDVD-based methods for minimum-energy docking have been pursued and verified to closely resemble optimal solutions [46, 47]. Comparisons of the performance of a non-convex IDVD approach

and LQ-MPC on the POSEIDYN testbed [48] show that the IDVD approach exhibits a smaller control effort than LQ-MPC; however, the drawback with IDVD is its high nonlinearity with no guarantee of convergence and high computational expense. The LQ-MPC approach, on the other hand, may have more expensive maneuvers, but also has deterministic convergence properties.

An additional advantage of LQ-MPC is its inherent robustness to unmeasured disturbances, as demonstrated through simulations and experiments using POSEIDYN [20]. Furthermore, the problem can be solved in real-time using MATLAB solvers such as `quadprog`, `fmincon` [25], and `SDPT3` [27], or CVXGEN [22, 27, 49], which generates a custom primal-dual interior point solver in C specifically designed to exploit the structure of the quadratic program family. The NMPC problem can be solved in real-time using the Interior Point OPTimizer (IPOPT) software package [23].

The viability of real-time implementation depends on the computation time to solve the optimal control problem: a more complex problem, e.g., with a longer horizon or more constraint equations, takes longer to solve. Hence, work is also being done to create fast custom solvers for MPC problems. So-called Fast MPC was proposed formally by Wang and Boyd [50], Boyd being one of the authors of the CVXGEN software. Fast MPC describes an algorithm similar to CVXGEN to create a custom solver that exploits the structure of the LQ-MPC problem to be solved for fast computation. It should be noted that similar algorithms are being implemented on field programmable gate arrays [51], known for their high computational speeds.

As demonstrated in this section, the guidance algorithm that balances optimality of the trajectory and computational speed for real-time implementation is MPC. Furthermore, to the best of the author's knowledge, MPC has not been experimentally validated with moving obstacle avoidance capabilities.

1.4 Thesis Objectives

The objective of this thesis is to design and experimentally validate a guidance algorithm enabling autonomous collision-free spacecraft RVD. More specifically, the real-time optimal path-planning algorithm is developed using the MPC framework.

The optimal control problem is solved using pre-existing solvers, e.g., `IPOPT`, `fmincon`, `quadprog`, and a custom solver based on the work of Wang and Boyd [50].

Both a linear and nonlinear optimal control problem are formulated using ellipsoidal and linearized (via direct linearization) obstacle avoidance constraints, respectively. Additional constraints to the problem are in the form of maximum thrust limitations, a dynamic holding radius, and an entry cone to promote safe docking.

Simulations are performed using the three-dimensional Hill's equations as well as two-dimensional planar equations of motion. Experimental validation is performed on the planar Spacecraft Proximity Operations Testbed at Carleton University, for which a third platform was constructed to act as the moving obstacle. Animations of the simulated and experimental test cases presented can be found in the video showcasing the results of this work: https://youtu.be/nRp4cs456P4.

This work complements the existing literature on LQ-MPC and additionally is, to the best of the author's knowledge, the first to experimentally validate the moving obstacle avoidance capabilities of MPC and the first to use the Fast MPC algorithm to solve RVD problems.

1.5 Contributions

The contributions of this work in the field of optimal guidance algorithms are as follows:

- The development of a guidance algorithm based upon the MPC framework;
- The development of a Fast MPC solver;
- The validation of the MPC algorithm through numerical simulations in two- and three-dimensions using planar dynamics and Hill's equations, respectively, with moving obstacles;
- The development of a new platform for the laboratory facility that allows for experimentation with a moving obstacle;
- The first experimental validation of the MPC algorithm with a rotating/translating target and translating obstacle using the upgraded facility;

- The comparison of the average computational speeds of different solvers (`IPOPT`, `fmincon`, `quadprog`, and a custom solver) and total impulse required by their solutions.

1.6 Thesis Organization

This thesis is divided into a number of sections and organized in such a way to guide the reader through the completed research. Chapter 2: *Orbital Mechanics* presents the equations of motion for a spacecraft in orbit and the equations of relative motion relevant to RVD missions. Further elaboration on the language and concepts used to describe RVD missions is provided in Chapter 3: *Guidance for Autonomous Rendezvous Missions*, as well as a brief introduction to the functions of the GNC system and the constituents of MPC as a guidance algorithm.

A formal mathematical foundation for optimal control problems is reviewed in Chapter 4: *Optimal Control Problems* with the formulation of the general MPC guidance problem. Chapter 5: *Solving Optimal Control Problems* introduces the proposed methods and algorithms required to augment and subsequently solve an equality constrained problem via Fast MPC.

The optimal control problem is solved for a reference scenario using Hill's equations in Chapter 6: *Three-Dimensional Simulations* whereby the relevant inequality constraints are formulated and the simulation results using two moving obstacles are presented. An overview of the SRCL's SPOT at Carleton University is provided in Chapter 7: *Experimental Facility*. Chapter 8: *Two-Dimensional Simulations* formulates the optimal control problem using planar dynamics and presents results using two moving obstacles. Finally, the MPC guidance algorithm is validated using SPOT in Chapter 9: *Experimental Validation*.

Chapter 10: *Conclusion* closes this thesis with a conclusion, summary of the completed research for this thesis, and recommendations for future work on the topic.

Chapter 2

Orbital Mechanics

This chapter introduces the mathematical model used to describe how spacecraft move in orbit relative to the Earth and relative to each other. The rendezvous and docking problem analyzed in this work consists of two spacecraft, the target and the chaser, whereby the objective of the chaser spacecraft is to find a trajectory that will bring it closer to and subsequently dock with the target spacecraft. Understanding how the control forces impact the acceleration of the chaser spacecraft with respect to the target is paramount to the success of the mission. Consequently, the relevant reference frames, equations of motion, and assumptions thereof are presented.

2.1 Reference Frames

A vector is formally defined as a mathematical object with a magnitude and a direction and is written with respect to a reference frame. A reference frame $\mathcal{F}$ is defined by its origin and a set of three orthonormal vectors. In this work, two right-handed reference frames are particularly of interest to describe the spacecraft dynamics equations and relative motion.

2.1.1 Earth-Centered Inertial (ECI) Frame

The Earth-Centered Inertial (ECI) reference frame, depicted in Fig. 2.1, denoted by $\mathcal{F}_I$, has its origin at the center of the Earth and describes a non-rotating, non-accelerating coordinate system. Here, it is assumed that the Earth is a perfect sphere,

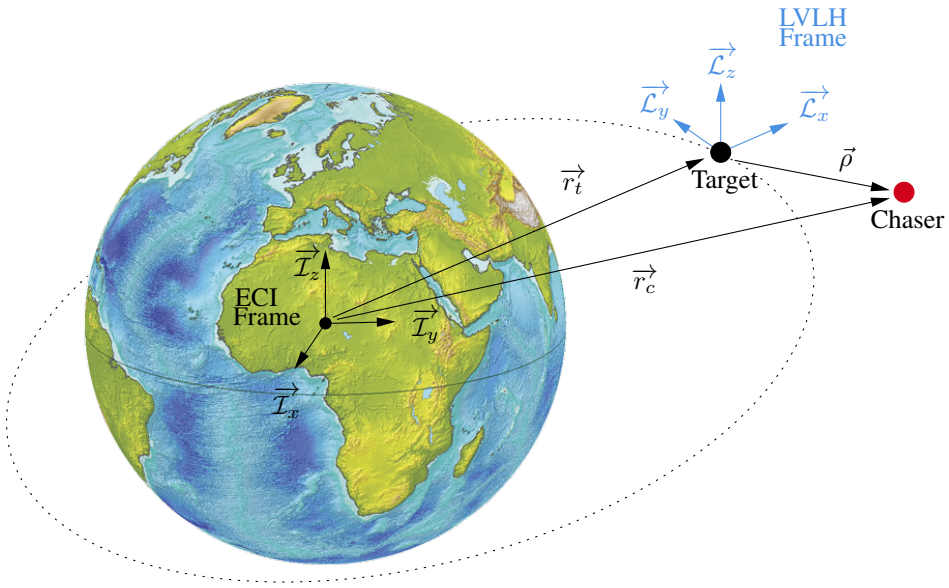


Figure 2.1: The Earth-Centered Inertial (ECI) and Local-Vertical-Local-Horizontal (LVLH) reference frames.

i.e., the geometric center of the Earth is its center of mass. The ECI reference frame is composed of three orthonormal vectors denoted by $\vec{\mathcal{I}}_x$, $\vec{\mathcal{I}}_y$, and $\vec{\mathcal{I}}_z$. The $\vec{\mathcal{I}}_z$ axis points in the direction of the Earth's spin axis, $\vec{\mathcal{I}}_x$ points toward the Vernal Equinox (when the Earth-Sun vector points toward the intersection of the ecliptic and the plane of Earth's equator), and $\vec{\mathcal{I}}_y$ completes the triad. The ECI reference frame is used to define the *absolute* position and velocity of a spacecraft.

2.1.2 Local-Vertical-Local-Horizontal (LVLH) Frame

The Local-Vertical-Local-Horizontal (LVLH) reference frame, $\mathcal{F}_L$, is defined to describe the motion of one spacecraft with respect to another in orbit, i.e., it describes the *relative motion* of spacecraft. As shown in Fig. 2.1, the origin of the LVLH reference frame is at the center of mass of the target spacecraft. Unlike the ECI frame, the LVLH frame describes a rotating, accelerating coordinate system. The radial direction $\vec{\mathcal{L}}_x$ of the frame points outward along the vector from the Earth's center to the target spacecraft, the cross-track direction $\vec{\mathcal{L}}_z$ points normal to the orbital plane (along the angular momentum vector of the target spacecraft), and the in-track direction $\vec{\mathcal{L}}_y$ completes the triad. For circular orbits, the in-track axis points in the

direction of the target spacecraft's velocity vector. Thus, given the position vector $\vec{r}_t$ and velocity vector $\vec{v}_t$ of the target spacecraft in the ECI frame, the orthonormal vectors of the LVLH frame are calculated as:

$$\vec{\mathcal{L}}_x = \frac{\vec{r}_t}{|\vec{r}_t|} \qquad \vec{\mathcal{L}}_y = \vec{\mathcal{L}}_z \times \vec{\mathcal{L}}_x \qquad \vec{\mathcal{L}}_z = \frac{\vec{r}_t \times \vec{v}_t}{|\vec{r}_t \times \vec{v}_t|} \tag{2.1}$$

2.2 Spacecraft Dynamics Equations

A reasonable model that describes the orbital motion of spacecraft can be derived from Newtonian mechanics. Consider Newton's law of gravitation which gives the force that an object with mass M exerts on another object with mass m,

$$\vec{F} = -\frac{GMm}{|\vec{r}|^3}\vec{r}, \tag{2.2}$$

where $\vec{r}$ is the vector that points from object M to object m and G is the gravitational constant. For a spacecraft with mass m_{sc} orbiting a massive planetary body with mass $M_\oplus$, there are some reasonable assumptions that can be made to generate the mathematical model for the spacecraft's motion.

Assumption 1

The only force acting on each object is the gravitational force of the other. It neglects any perturbing forces including those caused by third bodies (i.e., other celestial bodies, notably the sun and moon), solar radiation pressure, and atmospheric drag.

Assumption 2

The objects are homogeneous and spherical such that, as a consequence of Newton's shell theorem, the gravitational field of the object is the same as if its entire mass was concentrated at a point at its center.

Assumption 3

The mass of the spacecraft is negligible compared to the mass of the planetary body, $M_\oplus \gg m_{sc}$.

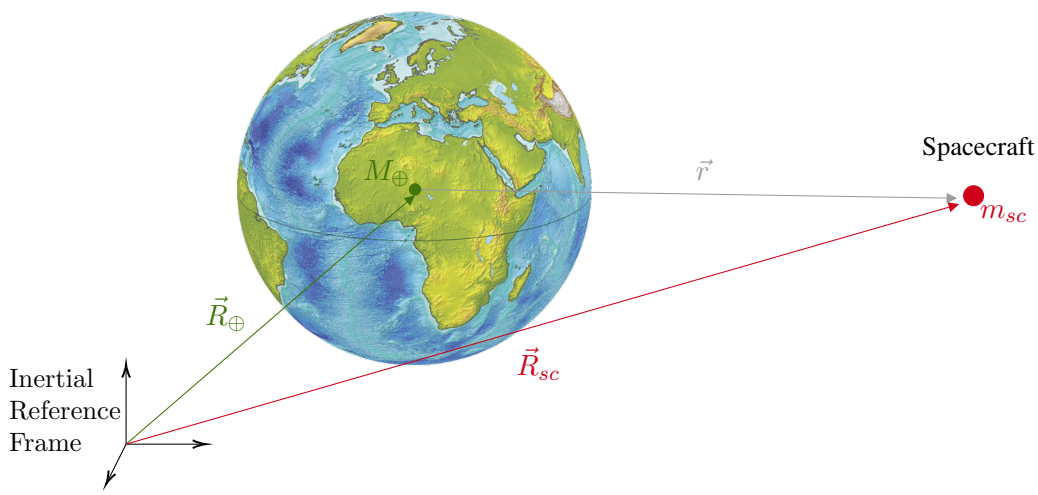


Figure 2.2: The positions of a planetary body and a spacecraft as seen in an inertial reference frame.

The position of the spacecraft relative to the planetary body as shown in Fig. 2.2 is given by

$$\vec{r} = \vec{R}_{sc} - \vec{R}_\oplus, \tag{2.3}$$

where $\vec{R}_{sc}$ and $\vec{R}_\oplus$ are the inertial positions of the spacecraft and planetary body, respectively. The parameter of interest is the dynamical motion of the spacecraft with respect to the planetary body, $\ddot{\vec{r}}$, given by the second time-derivative of Eq. (2.3),

$$\ddot{\vec{r}} = \ddot{\vec{R}}_{sc} - \ddot{\vec{R}}_\oplus. \tag{2.4}$$

Assumptions 1 and 2 are used in conjunction with Newton's second law and law of gravitation to give the equations of motion of the spacecraft and planetary body as seen in the inertial reference frame,

$$m_{sc}\ddot{\vec{R}}_{sc} = -\frac{GM_\oplus m_{sc}}{|\vec{r}|^3}\vec{r} \tag{2.5}$$

and

$$M_{\oplus}\ddot{\vec{R}}_{\oplus} = \frac{GM_{\oplus}m_{sc}}{|\vec{r}|^3}\vec{r}, \tag{2.6}$$

respectively. Substituting these accelerations into Eq. (2.4), the equations of motion of the spacecraft relative to the planetary body are

$$\ddot{\vec{r}} = -\frac{G\left(M_{\oplus} + m_{sc}\right)}{|\vec{r}|^3}\vec{r}. \tag{2.7}$$

Using Assumption 3 whereby $M_{\oplus} \gg m_{sc}$, the *gravitational parameter* of the two-body system, $\mu \triangleq G\left(M_{\oplus} + m_{sc}\right) \approx GM_{\oplus}$, replaces the numerator to give the well-known two-body equations of motion,

$$\ddot{\vec{r}} = -\frac{\mu}{|\vec{r}|^3}\vec{r}. \tag{2.8}$$

Although derived using an external inertial reference frame, this result can be directly found from Eq. (2.5) under the assumption that the force exerted on the planet from the spacecraft is negligible, i.e., $\ddot{\vec{R}}_{\oplus} \approx 0$ such that Eq. (2.4) reads $\ddot{\vec{r}} = \ddot{\vec{R}}_{sc}$.

2.3 Spacecraft Relative Motion

For spacecraft rendezvous and docking whereby the chaser spacecraft is directed to approach a target spacecraft, it is more convenient to write the spacecraft equations of motion in the LVLH frame centered at the target spacecraft.

The players of the rendezvous scenario are shown in Fig. 2.3. Of particular interest is the relative separation of the target and chaser spacecraft, $\vec{\rho}$, written in the LVLH frame as

$$\vec{\rho} \triangleq x\vec{\mathcal{L}}_x + y\vec{\mathcal{L}}_y + z\vec{\mathcal{L}}_z = \begin{bmatrix}\vec{\mathcal{L}}_x & \vec{\mathcal{L}}_y & \vec{\mathcal{L}}_z\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix}, \tag{2.9}$$

where x, y, and z denote the radial, in-track, and cross-track separations, respectively.

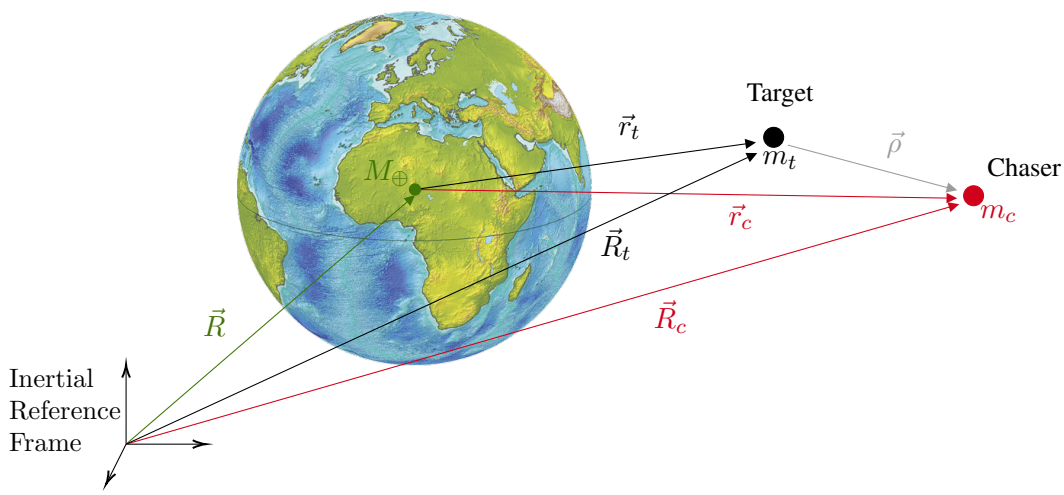


Figure 2.3: The positions of a planetary body, chaser spacecraft, and target spacecraft as seen in an inertial reference frame.

From Fig. 2.3 the relative separation can also be written as

$$\vec{\rho} = \vec{r}_c - \vec{r}_t. \tag{2.10}$$

By taking the second time derivative of Eq. (2.10), the equations of motion of the chaser spacecraft with respect to the target spacecraft in the radial, in-track, and cross-track directions can be derived,

$$\ddot{\vec{\rho}} = \ddot{\vec{r}}_c - \ddot{\vec{r}}_t, \tag{2.11}$$

where the spacecraft obey the two-body problem from §2.2 and thus the equations of motion for the target and chaser spacecraft, respectively, are:

$$\ddot{\vec{r}}_t = -\frac{\mu}{|\vec{r}_t|^3}\vec{r}_t \qquad \ddot{\vec{r}}_c = -\frac{\mu}{|\vec{r}_c|^3}\vec{r}_c \tag{2.12}$$

Further knowledge that the LVLH frame is non-inertial allows the equations of nonlinear relative motion to be derived. This derivation is presented in full in Appendix A.2. The resulting unforced nonlinear equations of motion of the chaser spacecraft

with respect to the target spacecraft in the LVLH frame are

$$\begin{cases} \ddot{x} + 2\frac{\dot{r}_t}{r_t}\dot{\theta}y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 x + \frac{\mu}{r_c^3}(r_t + x) - \frac{\mu}{r_t^2} & = 0 \\ \ddot{y} - 2\frac{\dot{r}_t}{r_t}\dot{\theta}x + 2\dot{\theta}\dot{x} - \dot{\theta}^2 y + \frac{\mu}{r_c^3}y & = 0 \\ \ddot{z} + \frac{\mu}{r_c^3}z & = 0 \end{cases} \tag{2.13}$$

where $\dot{\theta}$ is the true anomaly rate of change of the target spacecraft and $\dot{r}_t$ is the rate of change of the target's orbital radius.

For control theory, it is common to use the linearized equations of relative motion. Along with the assumptions already made to derive the two-body equations of motion, two more assumptions are made to linearize the nonlinear equations.

Assumption 4

The distance from the target to the chaser spacecraft is much smaller than the target's orbital radius, i.e., $r_t \gg |\vec{\rho}| = \sqrt{x^2 + y^2 + z^2}$.

Assumption 5

The orbit of the target spacecraft is circular such that the orbital radius and true anomaly rate, $\dot{r}_t$ and $\dot{\theta}$, are constant. Consequently, the true anomaly rate is equal to the mean motion, $\dot{\theta} = n = \sqrt{\mu/r_t^3}$.

The steps that go through linearizing Eq. (2.13) are shown in Appendix A.3. The resulting unforced linear equations of relative motion, also known as *Hill's equations*, are given by:

$$\begin{cases} \ddot{x} - 3n^2 x - 2n\dot{y} & = 0 \\ \ddot{y} + 2n\dot{x} & = 0 \\ \ddot{z} + n^2 z & = 0 \end{cases} \tag{2.14}$$

2.4 Spacecraft Control

To control the motion of a spacecraft using Hill's equations, the thrust accelerations, denoted by u_x, u_y, and u_z, for the thrusters onboard the spacecraft are included in

the linear equations,

$$\begin{cases} \ddot{x} - 3n^2 x - 2n\dot{y} & = \frac{u_x}{m} \\ \ddot{y} + 2n\dot{x} & = \frac{u_y}{m} \\ \ddot{z} + n^2 z & = \frac{u_z}{m} \end{cases} \tag{2.15}$$

where m denotes the chaser spacecraft mass. By defining the state matrix $\mathbf{x} \triangleq \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$ and the control input matrix $\mathbf{u} \triangleq \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$, the linearized equations of motion from Eq. (2.15) can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}, \tag{2.16}$$

which is in the form of the familiar continuous-time state-space equation,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{2.17}$$

2.5 Limits to Applications

To derive the equations of relative motion, five assumptions were made that limit the applicability of the equations to the general motion of spacecraft and consequently decrease its fidelity to model real-world dynamics.

Assumption 1 states that the only external force acting on the spacecraft is the gravitational force of the planetary body. This assumption neglects a number of significant natural perturbations seen to impact satellite orbits: the gravitational forces of the sun, moon, and other objects; solar radiation pressure; atmospheric drag; albedo; and relativistic effects. This also does not consider so-called artificial perturbations:

leaking propellant, thruster misfiring, and plume impingement between spacecraft.

Assumption 2 states that the planetary body and the spacecraft are homogeneous spheres. This assumption neglects the significant perturbations to spacecraft orbits due to the oblateness of the Earth, otherwise known as J2 and higher orders.

Assumption 3 states that the mass of the spacecraft is negligible compared to the mass of the planetary body, the Earth. This assumption represents real scenarios and is naturally applicable.

Assumption 4 states that the separation of the chaser and target is small compared to the orbital radius. This assumption holds for small separations (a few tens of kilometers when in LEO [52]), but the accuracy of Hill's equations decreases with increasing separation of the spacecraft.

Assumption 5 states that the target spacecraft is in a circular orbit. This assumption only holds for near-circular orbits and would not apply for spacecraft in elliptical orbits.

Consequently, the scope of this thesis is limited to RVD operations in circular orbit with relatively small spacecraft separations.

Chapter 3

Guidance for Autonomous Rendezvous Missions

This chapter provides a brief introduction to the language used in discussing rendezvous missions, starting with the various phases of a mission and their objectives. The question of *how* the spacecraft actually accomplishes the maneuvers required for a successful autonomous rendezvous mission leads to an introduction of the Guidance, Navigation, and Control subsystem, with a particular emphasis on guidance algorithms and a brief description of the specific guidance algorithm used in this work.

3.1 Definitions

The following definitions are based on those used by the International Organization for Standardization [53], NASA's *Rendezvous and Proximity Operations Handbook* [54], and Fehse's *Automated Rendezvous and Docking of Spacecraft* [52].

3.1.1 Rendezvous

A *rendezvous* is the process wherein two independently orbiting spacecraft are intentionally brought close together through a series of orbit maneuvers designed to achieve a gentle meeting at a planned time and point in orbit. In general, only one of the spacecraft, the "chaser," is actively performing the orbital maneuvers, meanwhile

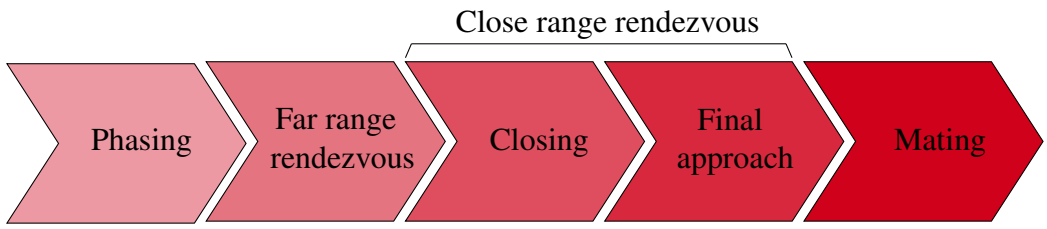


Figure 3.1: The main phases of a rendezvous mission.

the second spacecraft, the "target," is passively moving along its orbit. Note that the target need not be a spacecraft for a rendezvous mission, but could be any space object including natural satellites or space debris.

A rendezvous mission has a series of phases, as shown in Fig. 3.1. Before initiating the rendezvous operation, the chaser spacecraft must maneuver into the orbital plane of the target vehicle, i.e., their orbits must be coplanar. A *phasing* maneuver is then performed to bring the chaser spacecraft closer to the target. Phasing maneuvers take advantage of the fact that a lower orbit has a shorter orbital period, and vice versa for higher orbits, thus by raising or lowering the altitude of the chaser spacecraft it will be able to 'wait for' or 'catch up to' the target spacecraft, respectively. It is more common to lower the altitude of the chaser spacecraft for phasing maneuvers due to the lower fuel cost [52]. In general, all phasing maneuvers are controlled from the ground.

The *far range rendezvous* phase begins when relative navigation between the chaser and target becomes available and so the equations of relative motion of the chaser spacecraft (as discussed in §2.3) can be used to determine the trajectory of the chaser spacecraft with respect to the target. During this phase, which begins around a few tens of kilometers from the target, the differences in position, velocity, and angular rate of the spacecraft are reduced.

The *close range rendezvous* phase begins when the chaser is a few kilometers away from the target spacecraft and can be separated into two main subphases. During the *closing* phase, the chaser continues to reduce its relative position and velocity with respect to the target and begins to align itself along the axis of the docking mechanism on the target spacecraft. Only during the *final approach* phase does the chaser spacecraft achieve the necessary position, velocity, attitude, and angular

(a) Docking

(b) Berthing

Figure 3.2: The mating phase as seen from the ISS. Panel (a) shows the Soyuz MS-17 spacecraft, with Expedition 64 crew members, a few meters from docking to the Rassvet module on October 14, 2020. Panel (b) shows Northrop Grumman's Cygnus space freighter awaiting capture with Canadarm2 on August 12, 2021. Credit: NASA.

rate for mating. During final approach, which starts around 100 to 500 m from the target, an imaginary cone-shaped approach corridor extending from the target docking mechanism is usually defined within which the chaser trajectory must remain for safety and visibility reasons. At the end of this phase, the chaser mating interface is within reception range of the target docking mechanism.

The *mating* phase is when the chaser spacecraft makes mechanical contact with the target spacecraft and attenuates any residual relative motion between the spacecraft. The capture of the target can be accomplished through *docking*, whereby the body of the chaser is maneuvered to align with and interface the corresponding interface on the target vehicle, as shown in Fig. 3.2a between the ISS and the Soyuz MS-17 spacecraft; or through *berthing*, whereby a manipulator arm with a grapple mechanism on either the chaser or target is used to capture a fixture on the other spacecraft. In the case of the Cygnus space freighter, shown in Fig. 3.2b, it is the ISS that has the manipulator arm, the Canadarm2, to perform the capture.

3.1.2 Proximity Operations

Proximity operations are those that occur when two spacecraft are relatively close together. Specifically, it is when the spacecraft have a sufficiently small relative position

and velocity, about 1 km separation and 0.3 m/s, such that they can easily restore proximity without needing to rendezvous once more [54]. During proximity operations, generally only one of the two spacecraft is actively maneuvering to accomplish mission objectives, including stationkeeping and flyarounds.

3.1.3 Classifications of Target Cooperativity

A rendezvous mission is not one-size-fits-all since each mission is unique in its players and objectives. One factor that plays an important role in the mission design and execution is the degree to which the target spacecraft participates in the final approach operations. The participation of the target can be divided into three categories [52]:

Cooperative

Whereby the target vehicle actively performs attitude maneuvers and aids in berthing operations.

Passively cooperative

Whereby the target vehicle has a fixed attitude and sensor interfaces, but does not perform attitude maneuvers or berthing operations.

Uncooperative

Whereby the target vehicle is uncontrolled and likely tumbling. This is the scenario seen for depleted spacecraft and orbital debris.

It is clear that RVD to an uncooperative target is the most complex task. The work in this thesis focuses on RVD with uncooperative targets.

3.2 Guidance, Navigation, and Control

A spacecraft has a variety of subsystems that must work cohesively for a successful mission: propulsion; attitude determination and control system; guidance, navigation, and control (GNC); power; telemetry, tracking, and command; structures and mechanisms; and thermal control. The GNC subsystem provides rendezvous directions to the spacecraft.

The typical control architecture for maneuvering a spacecraft is shown in Fig. 3.3. The sensors onboard measure the actual state of the spacecraft, and this information

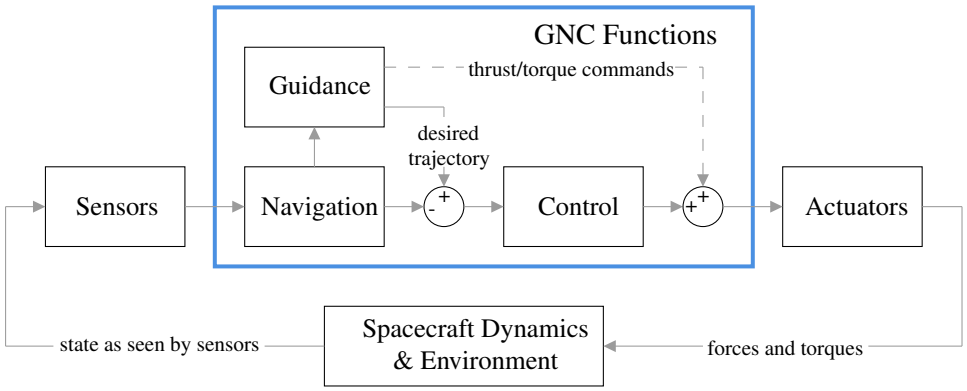


Figure 3.3: A typical control architecture for spacecraft maneuvering, based on [52].

is fed into the navigation function which combines and processes the data to produce an estimate of the current state. The guidance function uses the current state information to generate the desired trajectory of the spacecraft as well as the control forces to produce said trajectory. Any errors between the current state of the spacecraft and the desired state as generated by the guidance function are handled by the control function which produces thrust and torque commands to nullify the errors. The thrust commands are realized by the spacecraft actuators and the state of the spacecraft will change according to its dynamics and environment disturbances.

3.2.1 Navigation

The navigation function is responsible for processing the inputs of the various sensors onboard regarding the present state of the vehicle to determine an estimation of the state vector with reduced noise errors. Often, a digital filter is used for this task seeing as it predicts the current state variables and compares this estimate to the measured state whereby a better estimate may be made for the next sampling instant through updating a weighting matrix. The navigation filter is necessary when the sensor data is only intermittently available. A Kalman filter is a typical choice for the navigation filter. In this thesis, it is assumed that the sensor data is readily available with negligible uncertainty such that a sophisticated navigation function is unnecessary.

3.2.2 Guidance

The guidance function is responsible for generating the trajectory of the spacecraft to the desired location and attitude using as input the current state information provided by the navigation function as well as a relative motion model of the spacecraft dynamics. The output of the guidance function is the desired trajectory and, in the case of MPC-based guidance laws, the set of control forces necessary to execute it.

3.2.3 Control

The control function is responsible for providing the force and torque commands to compensate for errors in the actual state (provided by the navigation function) from the desired state (provided by the guidance function). Discrepancies between the two states could be a result of unmodelled disturbances, inaccurate model assumptions, and sensor errors.

3.3 Model Predictive Control

This work focuses on the guidance function otherwise referred to as the guidance or path-planning algorithm. As discussed in §1.3, MPC provides the framework for an optimization-based guidance algorithm and is actively being researched for spacecraft RVD operations.

3.3.1 Philosophy

MPC, also called receding horizon control, is an advanced control framework that may best be described as a repetitive decision-making process that forecasts the future states of a system and minimizes a user-defined performance index, otherwise known as the cost, subject to constraints. The repetitiveness comes from recomputing the solution every time the current state is updated. In this way, MPC generates a set of control actions that continuously compensate for uncertainties and disturbances, thereby negating the need for a control function in the GNC framework (refer to Fig. 3.3). This result was confirmed by [27] whereby an MPC+LQR controller was found

to achieve inferior performance and require 21% more propellant as compared to an MPC controller.

Theoretically, an ideal approach to MPC employs infinite-horizon open-loop optimal control (OLOC) whereby an infinite sum of stage costs is minimized and thus the resulting solution is optimal over the entire trajectory. Practically, this approach is not applicable due to its heavy computational burden.

An alternative approach truncates the infinite horizon and instead uses a finite-horizon of length N such that the formulation is called finite-horizon OLOC (FHOLOC). When all states and control actions beyond the horizon length are neglected in the cost and constraint functions, this is called the lower approximate FHOLOC formulation. When the states and control actions beyond the horizon length are accounted for using terminal constraints and a terminal cost, this is called the upper approximate FHOLOC formulation. In general, the lower approximate FHOLOC can lead to destabilization or instability if the horizon length is too short. The upper approximate FHOLOC, on the other hand, has guarantees for stability due to the inclusion of the terminal constraints [55].

A visualization of the MPC process is shown in Fig. 3.4. At the sampling instant k, a FHOLOC problem is solved using the current state of the plant as the initial state. An optimal control sequence over the prediction horizon is determined through

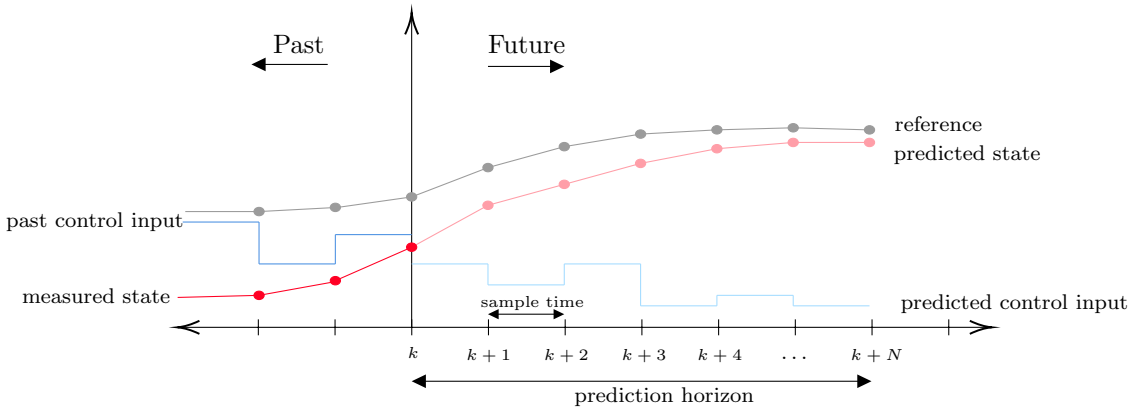


Figure 3.4: A visualization of model predictive control. At the current sample time k, a sequence of control inputs are calculated that minimize the error between the predicted state and reference trajectory over the prediction horizon N. This process is repeated every sampling instant whereby the prediction horizon continuously recedes by one step.

this process, and only the first control of this sequence is applied. At this point, the sampling indices shift by one, i.e., $\{k+1, k+2, k+3, \dots\}$ becomes $\{k, k+1, k+2, \dots\}$. The process repeats whereby the optimal control problem is re-solved using the new current state of the plant as the initial condition. By implementing this discretized scheme, the MPC problem is based on a discrete-time system representation.

3.3.2 Formulation

The FHOLOC problem for MPC is formulated from three main constituents [56].

Model

The dynamics of the system being controlled must be known. For spacecraft rendezvous, the equations discussed in §2.4 apply, but must be discretized. The general form of a discrete-time control system is given by

$$\mathbf{x}^{(k+1)} = f(\mathbf{x}^{(k)}, \mathbf{u}^{(k)}), \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathbb{R}^m$ is the control force, the dynamics for the system are represented by the state transition function $f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and the superscript $(\cdot)$ is used throughout this thesis to denote the sampling instant. The discrete form of the continuous-time state-space equation from Eq. (2.17) is written as

$$\mathbf{x}^{(k+1)} = \mathbf{A_d}\mathbf{x}^{(k)} + \mathbf{B_d}\mathbf{u}^{(k)}. \tag{3.2}$$

The discretized state and control matrices, $\mathbf{A_d} \in \mathbb{R}^{n\times n}$ and $\mathbf{B_d} \in \mathbb{R}^{n\times m}$, respectively, can be obtained from their continuous-time counterparts, $\mathbf{A}$ and $\mathbf{B}$, using a zero-order hold where T_d is the discretization or sample time:

$$\begin{cases} \mathbf{A_d} &= e^{\mathbf{A}T_d} \\ \mathbf{B_d} &= \left(\int_0^{T_d} e^{\mathbf{A}t} dt\right) \mathbf{B} \end{cases} \tag{3.3}$$

Cost Function

The performance index, also called the cost function or objective function, is a quantity defined in terms of the state and control variables over the horizon. This is the value to be minimized. A general cost function J is given in the form of

$$J \triangleq \ell_f(\mathbf{x}^{(k+N)}) + \sum_{\tau=k}^{k+N-1} \ell(\mathbf{x}^{(\tau)}, \mathbf{u}^{(\tau)}), \tag{3.4}$$

where $\ell_f(\cdot) : \mathbb{R}^n \to \mathbb{R}_+$ is the *terminal cost* and $\ell(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ is the *stage cost*.

Constraints

The constraint equations give hard limits to the permissible values for the state $\mathbf{x}$ and control $\mathbf{u}$ variables over the horizon. Constraints arise due to, for example, maximum thrust limitations, obstacle avoidance, safety requirements, and/or desired operation modes.

Chapter 4

Optimal Control Problems

This chapter reviews the formal mathematical foundation for optimal control problems based upon Boyd and Vandenberghe's *Convex Optimization* [57, Ch. 4-5]. The basic terminology used in the field of optimization is defined and the constituents of an optimal control problem introduced last chapter are elaborated further to address the specific problem of spacecraft rendezvous.

4.1 Optimization Problems

Optimization problems are made up of different elements, as briefly described in §3.3.2. The *optimization variable* $\mathbf{z} \in \mathbb{R}^{n_z}$ is the vector of parameters of length n_z to be varied, the *objective* or *cost function* $J(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is the quantity that is to be minimized, and the *inequality constraint functions* $f_i(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ and *equality constraint functions* $h_i(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ limit the available values for $\mathbf{z}$.

An optimization problem in standard form is written as

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{z}) \\ \text{subject to} \quad & f_i(\mathbf{z}) \leq 0, \quad i = 1, \ldots, n_{f_i} \\ & h_i(\mathbf{z}) = 0, \quad i = 1, \ldots, n_{h_i} \end{aligned} \tag{4.1}$$

where n_{f_i} and n_{h_i} define the number of inequality and equality constraints, respectively. The optimization problem is said to be *feasible* if there exists a point $\mathbf{z}$ for

which all inequality and equality constraints are satisfied. The *optimal point* $\mathbf{z}^*$ is a feasible point that additionally minimizes the cost function J. The *optimal value* p^* of the problem is defined as the lowest value of $J(\mathbf{z})$ that satisfies all constraints,

$$p^* \triangleq \inf \{ J(\mathbf{z}) \mid f_i(\mathbf{z}) \leq 0,\ i = 1, \ldots, n_{f_i},\ h_i(\mathbf{z}) = 0,\ i = 1, \ldots, n_{h_i} \}, \tag{4.2}$$

or written more intuitively as

$$p^* = J(\mathbf{z}^*). \tag{4.3}$$

4.2 Lagrangian Formulation

The optimization problem in Eq. (4.1) can be augmented to directly account for the constraint equations in the objective function. This is done for the purpose of solving the problem and will be discussed further in Chapter 5, which presents methods of solving optimization problems.

The Lagrangian $L : \mathbb{R}^{n_z} \times \mathbb{R}^{n_{f_i}} \times \mathbb{R}^{n_{h_i}} \to \mathbb{R}$ associated with the optimization problem in Eq. (4.1) is written as

$$L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \triangleq J(\mathbf{z}) + \sum_{i=1}^{n_{f_i}} \lambda_i f_i(\mathbf{z}) + \sum_{i=1}^{n_{h_i}} \nu_i h_i(\mathbf{z}), \tag{4.4}$$

where λ_i is called the Lagrange multiplier associated with the ith inequality constraint $f_i(\mathbf{z}) \leq 0$ and ν_i is similarly called the Lagrange multiplier associated with the ith equality constraint $h_i(\mathbf{z}) = 0$. The vectors $\boldsymbol{\lambda} \in \mathbb{R}^{n_{f_i}}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n_{h_i}}$ are called the dual variables or Lagrange multiplier vectors.

From the Lagrangian associated with an optimization problem, a so-called *dual problem* can be formulated whereby the Lagrange dual function given by

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \triangleq \inf L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{4.5}$$

is to be maximized subject to $\boldsymbol{\lambda} \succeq 0$. When discussing dual problems, the original optimization problem in Eq. (4.1) is called the *primal problem*. The dual problem itself is not important for this work, but it is suffice to know that the solutions to

the primal and dual problems are related: the optimal value of the Lagrange dual problem, denoted by d^*, is the best lower bound on p^* that can be obtained by the Lagrange dual function. The difference between the optimal values of the primal and dual problem, $p^* - d^*$, is called the optimal duality gap. If the duality gap is zero, then it is said that strong duality holds. Convex optimization problems whereby Slater's constraint qualification holds, i.e., there exists a *strictly feasible* point $\mathbf{z}$, have strong duality.

Now assume that an optimization problem with strong duality has differentiable cost and constraint equations. Let the primal and dual optimal points be denoted as $\mathbf{z}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$, respectively. Since the associated Lagrangian is minimized at the optimal point, this is equivalent to saying that its gradient with respect to $\mathbf{z}$ vanishes at $\mathbf{z}^*$,

$$\nabla_{\mathbf{z}} L(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = \nabla_{\mathbf{z}} J(\mathbf{z}^*) + \sum_{i=1}^{n_{f_i}} \lambda_i^* \nabla_{\mathbf{z}} f_i(\mathbf{z}^*) + \sum_{i=1}^{n_{h_i}} \nu_i^* \nabla_{\mathbf{z}} h_i(\mathbf{z}^*) = 0. \tag{4.6}$$

Thus the necessary conditions for a solution $\mathbf{z}^*$ to be optimal is to satisfy:

$$f_i(\mathbf{z}^*) \leq 0, \quad i = 1, \ldots, n_{f_i} \tag{4.7}$$

$$h_i(\mathbf{z}^*) = 0, \quad i = 1, \ldots, n_{h_i} \tag{4.8}$$

$$\lambda_i^* \geq 0, \quad i = 1, \ldots, n_{f_i} \tag{4.9}$$

$$\lambda_i^* f_i(\mathbf{z}^*) = 0, \quad i = 1, \ldots, n_{f_i} \tag{4.10}$$

$$\nabla_{\mathbf{z}} J(\mathbf{z}^*) + \sum_{i=1}^{n_{f_i}} \lambda_i^* \nabla_{\mathbf{z}} f_i(\mathbf{z}^*) + \sum_{i=1}^{n_{h_i}} \nu_i^* \nabla_{\mathbf{z}} h_i(\mathbf{z}^*) = 0 \tag{4.11}$$

The conditions in Eq. (4.7) and Eq. (4.8) are simply the inequality and equality constraints of the primal problem. The condition in Eq. (4.9) was defined for the Lagrangian multiplier. The condition in Eq. (4.11) is simply repeating Eq. (4.6). The condition in Eq. (4.10) is known as complementary slackness and follows from strong duality.

Equations (4.7)-(4.11) are well-known and are called the *Karush-Kuhn-Tucker* (KKT) conditions. The algorithms used to solve convex optimization problems are

solving these KKT conditions, whether directly or indirectly.

4.3 Optimal Control Problems

An *optimal control problem* is a specific type of optimization problem, Eq. (4.1), for which the control input $\mathbf{u} \in \mathbb{R}^m$ is determined to transfer a system $\mathbf{x} \in \mathbb{R}^n$ to a desired state $\mathbf{x}_t \in \mathbb{R}^n$ over a period of time. In this case, solutions for both the system state variables and control variables must be computed over the entire period of time, called the planning or time horizon N. The constituents of the optimal control problem as introduced in §3.3.2 are elaborated upon below.

Cost Function

Typically the cost function for an optimal control problem is written as

$$J = \ell_f(\mathbf{e}^{(k+N)}) + \sum_{\tau=k}^{k+N-1} \ell(\mathbf{e}^{(\tau)}, \mathbf{u}^{(\tau)}), \tag{4.12}$$

where $\mathbf{e}^{(k)} \triangleq \mathbf{x}^{(k)} - \mathbf{x}_t^{(k)} \in \mathbb{R}^n$ is the error of the system's state from the desired state, k is the current sampling instant, and $\ell(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $\ell_f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ are the stage cost function and terminal cost function, respectively. The stage costs seen in literature are commonly quadratic (convex) with a general form

$$\ell(\mathbf{e}^{(k)}, \mathbf{u}^{(k)}) = \begin{bmatrix} \mathbf{e}^{(k)} \\ \mathbf{u}^{(k)} \end{bmatrix}^T \begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{e}^{(k)} \\ \mathbf{u}^{(k)} \end{bmatrix} + \mathbf{q}^T \mathbf{e}^{(k)} + \mathbf{r}^T \mathbf{u}^{(k)} \tag{4.13}$$

with parameters $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{n\times n}$, $\mathbf{S} \in \mathbb{R}^{n\times m}$, $\mathbf{R} = \mathbf{R}^T \in \mathbb{R}^{m\times m}$, $\mathbf{q} \in \mathbb{R}^n$, and $\mathbf{r} \in \mathbb{R}^m$ which are used to specify the weighting of the state errors and controls in the stage cost function. The terminal cost function has a similar form to Eq. (4.13), but only has state error-dependence,

$$\ell_f(\mathbf{e}^{(k+N)}) = \left(\mathbf{e}^{(k+N)}\right)^T \mathbf{Q}_f \mathbf{e}^{(k+N)} + \mathbf{q}_f^T \mathbf{e}^{(k+N)}. \tag{4.14}$$

It has been shown that setting $\mathbf{Q}_f$ as the solution to the discrete algebraic Riccati equation for the unconstrained infinite horizon variant of the optimal control problem offers local stability [55],

$$\mathbf{Q}_f = \mathbf{A}^T\mathbf{Q}_f\mathbf{A} - \mathbf{A}\mathbf{Q}_f\mathbf{B}\left(\mathbf{B}^T\mathbf{Q}_f\mathbf{B} + \mathbf{R}\right)^{-1}\mathbf{B}^T\mathbf{Q}_f\mathbf{A}, \tag{4.15}$$

and has been used in many publications for spacecraft rendezvous guidance algorithms [19–24, 26, 31, 35, 43, 48, 58].

When considering state control separable problems, the cross-term weighting matrix $\mathbf{S}$ is equal to zero. When $\mathbf{q} = \mathbf{r} = 0$, the stage cost becomes

$$\begin{aligned}
\ell(\mathbf{e}^{(k)}, \mathbf{u}^{(k)}) &= \begin{bmatrix}\mathbf{e}^{(k)}\\ \mathbf{u}^{(k)}\end{bmatrix}^T \begin{bmatrix}\mathbf{Q} & 0\\ 0 & \mathbf{R}\end{bmatrix} \begin{bmatrix}\mathbf{e}^{(k)}\\ \mathbf{u}^{(k)}\end{bmatrix} && (4.16)\\
&= \left(\mathbf{e}^{(k)}\right)^T \mathbf{Q}\mathbf{e}^{(k)} + \left(\mathbf{u}^{(k)}\right)^T \mathbf{R}\mathbf{u}^{(k)} && (4.17)\\
&= \left(\mathbf{x}^{(k)} - \mathbf{x}_t^{(k)}\right)^T \mathbf{Q}\left(\mathbf{x}^{(k)} - \mathbf{x}_t^{(k)}\right) + \left(\mathbf{u}^{(k)}\right)^T \mathbf{R}\mathbf{u}^{(k)} && (4.18)
\end{aligned}$$

and the total cost as per Eq. (4.12) can be written as

$$\begin{aligned}
J = &\left(\mathbf{x}^{(k+N)} - \mathbf{x}_t^{(k+N)}\right)^T \mathbf{Q}_f \left(\mathbf{x}^{(k+N)} - \mathbf{x}_t^{(k+N)}\right) + \\
&\sum_{\tau=k}^{k+N-1} \left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right)^T \mathbf{Q}\left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right) + \left(\mathbf{u}^{(\tau)}\right)^T \mathbf{R}\mathbf{u}^{(\tau)}.
\end{aligned} \tag{4.19}$$

For a spacecraft rendezvous problem, the indices in the summation of Eq. (4.19) must be modified slightly since the current state of the spacecraft $\mathbf{x}^{(k)}$ will not be optimized, but the current control force $\mathbf{u}^{(k)}$ will. The modified version of the cost function that considers the correct optimization variables is given as

$$\begin{aligned}
J = &\left(\mathbf{x}^{(k+N)} - \mathbf{x}_t^{(k+N)}\right)^T \mathbf{Q}_f \left(\mathbf{x}^{(k+N)} - \mathbf{x}_t^{(k+N)}\right) + \left(\mathbf{u}^{(k)}\right)^T \mathbf{R}\mathbf{u}^{(k)} + \\
&\sum_{\tau=k+1}^{k+N-1} \left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right)^T \mathbf{Q}\left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right) + \left(\mathbf{u}^{(\tau)}\right)^T \mathbf{R}\mathbf{u}^{(\tau)}.
\end{aligned} \tag{4.20}$$

Equality Constraints

The equality constraint functions h_i from Eq. (4.1) come from the dynamics of the system, which were developed in Chapter 2. Adhering to RVD problems that satisfy the assumptions of the dynamic model derivation, i.e., short separations in circular orbits, Hill's equations are used. From §2.3 this is

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}}_{\mathbf{u}} \tag{4.21}$$

or more compactly written as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{4.22}$$

which is the well-known continuous-time state-space equation. As discussed in §3.3, the MPC scheme requires a discretized system representation following the form

$$\mathbf{x}^{(k+1)} = \mathbf{A_d}\mathbf{x}^{(k)} + \mathbf{B_d}\mathbf{u}^{(k)}, \tag{4.23}$$

which can be achieved using a zero-order hold as given in Eq. (3.3).

Inequality Constraints

The inequality constraints for an RVD problem are used to control the behaviour of the trajectory and control forces. These constraints could be due to maximum thrust limitations, obstacle avoidance, limiting plume impingement, safety corridors, etc. Specific mathematical descriptions for the scenarios considered in this work will be discussed in Chapters 6 and 8. For now, they can be treated as the previously defined equation, $f_i(\mathbf{x}, \mathbf{u}) \leq 0$.

Standard Form

Using these different constituents, the optimal control problem from Eq. (4.1) can be posed as follows, noting that the index for the current sampling instant k has been set to a value of 1 in this notation for brevity:

$$\begin{aligned}
\text{minimize} \quad & \left(\mathbf{x}^{(N+1)} - \mathbf{x}_t^{(N+1)}\right)^T \mathbf{Q}_f \left(\mathbf{x}^{(N+1)} - \mathbf{x}_t^{(N+1)}\right) + \left(\mathbf{u}^{(1)}\right)^T \mathbf{R}\mathbf{u}^{(1)} + \\
& \sum_{\tau=2}^{N} \left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right)^T \mathbf{Q} \left(\mathbf{x}^{(\tau)} - \mathbf{x}_t^{(\tau)}\right) + \left(\mathbf{u}^{(\tau)}\right)^T \mathbf{R}\mathbf{u}^{(\tau)} \\
\text{subject to} \quad & \mathbf{x}^{(i+1)} - \mathbf{A_d}\mathbf{x}^{(i)} - \mathbf{B_d}\mathbf{u}^{(i)} = 0, \quad i = 1, \ldots, N \\
& f_i(\mathbf{x}, \mathbf{u}) \leq 0, \quad i = 1, \ldots, n_{f_i}
\end{aligned} \tag{4.24}$$

4.4 Compact Notation

For the remainder of this thesis, unless otherwise stated, the defined vectors are column vectors; it is implied that the components making up these column vectors are oriented in the correct, stackable orientation seeing as the explicit transpose notation on each element has been dropped for the sake of brevity.

It is convenient to write the optimal control problem in Eq. (4.24) in a compact form [50]. To do so, the overall optimization column vector $\mathbf{z}$ is defined to contain the states and controls $\mathbf{x}$ and $\mathbf{u}$ of the system at each sampling instant over the horizon,

$$\mathbf{z} \triangleq \begin{bmatrix} \mathbf{u}^{(k)} & \mathbf{x}^{(k+1)} & \mathbf{u}^{(k+1)} & \ldots & \mathbf{x}^{(k+N-1)} & \mathbf{u}^{(k+N-1)} & \mathbf{x}^{(k+N)} \end{bmatrix}^T \in \mathbb{R}^{n_z}, \tag{4.25}$$

noting that the total number of optimization variables is $n_z = (n+m)N$. The target vector, $\mathbf{z}_t \in \mathbb{R}^{n_z}$, is also defined to be the same size as $\mathbf{z}$ and contain information about the desired states of the system over the horizon,

$$\mathbf{z}_t \triangleq \begin{bmatrix} 0 & \mathbf{x}_t^{(k+1)} & 0 & \ldots & \mathbf{x}_t^{(k+N-1)} & 0 & \mathbf{x}_t^{(k+N)} \end{bmatrix}^T, \tag{4.26}$$

where the 0 entries are zero-vectors the same size as $\mathbf{u}$.

Cost Function

To compactify the cost function in Eq. (4.24), a diagonal block matrix $\mathbf{H} \in \mathbb{R}^{n_z \times n_z}$ that contains the $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q}_f$ matrices is defined,

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{R} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \mathbf{Q} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{R} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{Q} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{R} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \mathbf{Q} & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \mathbf{R} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \mathbf{Q}_f \end{bmatrix}. \tag{4.27}$$

The cost function from Eq. (4.24) can be written in the compact notation,

$$J(\mathbf{z}) = (\mathbf{z} - \mathbf{z}_t)^T \mathbf{H} (\mathbf{z} - \mathbf{z}_t). \tag{4.28}$$

Equality Constraints

Similar compactness can be achieved for the equality constraints. Defining the matrix $\mathbf{C} \in \mathbb{R}^{n_{h_i} \times n_z}$, where $n_{h_i} = nN$, to contain the discrete state and control matrices,

$$\mathbf{C} \triangleq \begin{bmatrix} -\mathbf{B}_d & \mathbf{I}_n & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I}_n & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -\mathbf{A}_d & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{I}_n & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I}_n \end{bmatrix} \tag{4.29}$$

and the column vector $\mathbf{b} \in \mathbb{R}^{n_{h_i}}$ to be

$$\mathbf{b} \triangleq \begin{bmatrix} \mathbf{A}_d \mathbf{z}^{(k)} & 0 & 0 & \dots & 0 & 0 \end{bmatrix}^T, \tag{4.30}$$

where $\mathbf{z}^{(k)}$ is the current state of the system, then the equality constraints from Eq. (4.24) can be reduced to the expression

$$\mathbf{C}\mathbf{z} = \mathbf{b}. \tag{4.31}$$

Inequality Constraints

Only linear equations can be written in compact form. A general state control separable linear inequality constraint at any given step within the horizon can be written as

$$\begin{cases} \mathbf{F}_x\mathbf{x} \leq \mathbf{f}_x \\ \mathbf{F}_u\mathbf{u} \leq \mathbf{f}_u \end{cases} \tag{4.32}$$

where $\mathbf{F}_x \in \mathbb{R}^{l_x \times n}$ and $\mathbf{F}_u \in \mathbb{R}^{l_u \times m}$ are matrices holding information about the constraint equations, l_x and l_u are the number of inequality constraints on the variables $\mathbf{x}$ and $\mathbf{u}$, respectively, and the column vectors $\mathbf{f}_x \in \mathbb{R}^{l_x}$ and $\mathbf{f}_u \in \mathbb{R}^{l_u}$ are determined from the constraint equations. Note that the total number of inequality constraints n_{f_i} is related to l_x and l_u via $n_{f_i} = (l_x + l_u)N$.

Defining the matrix $\mathbf{P} \in \mathbb{R}^{n_{f_i} \times n_z}$ to contain the constraint equation matrices,

$$\mathbf{P} \triangleq \begin{bmatrix} \mathbf{F}_u^{(k)} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \mathbf{F}_x^{(k+1)} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{F}_u^{(k+1)} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{F}_x^{(k+2)} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{F}_u^{(k+N-2)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \mathbf{F}_x^{(k+N-1)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \mathbf{F}_u^{(k+N-1)} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \mathbf{F}_x^{(k+N)} \end{bmatrix} \tag{4.33}$$

and the column vector $\mathbf{h} \in \mathbb{R}^{n_{f_i}}$ to be

$$\mathbf{h} \triangleq \begin{bmatrix} \mathbf{f}_u^{(k)} & \mathbf{f}_x^{(k+1)} & \mathbf{f}_u^{(k+1)} & \dots & \mathbf{f}_x^{(k+N-1)} & \mathbf{f}_u^{(k+N-1)} & \mathbf{f}_x^{(k+N)} \end{bmatrix}^T, \tag{4.34}$$

then the linear inequality constraints can be written compactly as

$$\mathbf{P}\mathbf{z} \leq \mathbf{h}. \tag{4.35}$$

Compact Formulation

Using the compact cost in Eq. (4.28), equality constraints in Eq. (4.31), and inequality constraints in Eq. (4.35), the optimal control problem with quadratic cost and linear constraints, i.e., a quadratic programming (QP) problem, can be written in compact form as:

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{z}) = (\mathbf{z} - \mathbf{z}_t)^T \mathbf{H} (\mathbf{z} - \mathbf{z}_t) \\ \text{subject to} \quad & \mathbf{C}\mathbf{z} = \mathbf{b} \\ & \mathbf{P}\mathbf{z} \leq \mathbf{h} \end{aligned} \tag{4.36}$$

One important property to note about the matrices $\mathbf{H}$, $\mathbf{C}$, and $\mathbf{P}$ is that they are structured along the diagonal. The numerical method discussed in §5.4 exploits this special block diagonal structure of the problem for fast and efficient computation [50].

4.5 Equivalent Cost

It is not convenient to have the quadratic cost of the state variable paired with the target state variable, i.e., $(\mathbf{z} - \mathbf{z}_t)$, especially when this format of the cost for an optimal control problem is not supported by black box solvers like `quadprog`. Fortunately, the cost function can be altered to give an *equivalent* problem, i.e., the solution from one problem can be used to find the solution of the other.

The compact cost function in Eq. (4.36) can be expanded, noting that the quantity $\mathbf{z}_t^T \mathbf{H}^T \mathbf{z}$ is a scalar so $\left(\mathbf{z}_t^T \mathbf{H}^T \mathbf{z}\right)^T = \mathbf{z}_t^T \mathbf{H}^T \mathbf{z}$, and since $\mathbf{H}$ is symmetric, $\mathbf{H} = \mathbf{H}^T$, it can

be noted that $\mathbf{z}_t^T \mathbf{H}^T \mathbf{z} = \mathbf{z}_t^T \mathbf{H} \mathbf{z}$.

$$
\begin{aligned}
J &= (\mathbf{z} - \mathbf{z}_t)^T \, \mathbf{H} \, (\mathbf{z} - \mathbf{z}_t) \\
J &= \left(\mathbf{z}^T - \mathbf{z}_t^T\right) (\mathbf{H}\mathbf{z} - \mathbf{H}\mathbf{z}_t) \\
J &= \mathbf{z}^T \mathbf{H} \mathbf{z} - \mathbf{z}^T \mathbf{H} \mathbf{z}_t - \mathbf{z}_t^T \mathbf{H} \mathbf{z} + \mathbf{z}_t^T \mathbf{H} \mathbf{z}_t \\
J &= \mathbf{z}^T \mathbf{H} \mathbf{z} - \left(\mathbf{z}_t^T \mathbf{H}^T \mathbf{z}\right)^T - \mathbf{z}_t^T \mathbf{H} \mathbf{z} + \mathbf{z}_t^T \mathbf{H} \mathbf{z}_t \\
J &= \mathbf{z}^T \mathbf{H} \mathbf{z} - 2\mathbf{z}_t^T \mathbf{H} \mathbf{z} + \mathbf{z}_t^T \mathbf{H} \mathbf{z}_t
\end{aligned}
$$

In the optimal control problem the vector $\mathbf{z}_t$ is a constant for each iteration, thus the contribution of the last term in this cost function does not impact the values chosen for $\mathbf{z}$. The cost function to an equivalent optimal control problem is given by the first two terms,

$$
J = \mathbf{z}^T \mathbf{H} \mathbf{z} - 2\mathbf{z}_t^T \mathbf{H} \mathbf{z}. \tag{4.37}
$$

Chapter 5

Solving Optimal Control Problems

In the last chapter an optimal control problem was formulated. This chapter discusses methods of solving optimization problems and presents a specific gradient-based method called the infeasible start Newton method based on the work of Wang and Boyd's *Fast Model Predictive Control Using Online Optimization* [50]. Lastly, the black box solvers that are relevant to this work are briefly discussed.

The information is this chapter is based on Boyd and Vandenberghe's *Convex Optimization* [57] as well as Wang and Boyd's *Fast Model Predictive Control Using Online Optimization* [50].

5.1 Gradient-Based Methods

Numerical methods used to solve nonlinear programming problems can be categorized into two classes [59]: *gradient-based* methods, which are deterministic and give locally optimal solutions; and *heuristic* methods, which are stochastic and give globally optimal solutions. Examples of heuristic methods include genetic algorithms, simulated annealing, and particle swarm optimization, and are considered for complex problems when no other efficient computation method is available, e.g., when the cost function is non-differentiable.

A gradient-based method, as per its name, uses the gradient of the cost function to determine the search direction $\Delta\mathbf{z}$ that the solution should follow from an input

initial guess $\mathbf{z}$. The updated solution takes the form

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + t^{(k)} \Delta \mathbf{z}^{(k)}, \tag{5.1}$$

where $t > 0$ is the step size. The entire methodology of gradient-based methods is captured by Eq. (5.1): calculate the search direction, calculate the step size, update the solution, repeat. This process is written in Algorithm 1.

The search direction is chosen to decrease the cost function, i.e.,

$$J(\mathbf{z}^{(k+1)}) < J(\mathbf{z}^{(k)}) \tag{5.2}$$

except when $\mathbf{z}^{(k)}$ is optimal. Many methods exist for determining the search direction, among them gradient descent, steepest descent, and Newton's method. Newton's method is particularly of interest since it has rapid convergence near the optimal solution and scales well with problem size. A consequence of this rapid convergence, however, is the computational cost of forming and storing the Hessian of the cost function [57, Ch. 9.5].

In order to use a gradient-based method to solve an optimal control problem in the form of Eq. (4.36), the problem must be reformulated. This topic is dealt with in the next section.

Algorithm 1: General Gradient-Based Method

Data: initial guess $\mathbf{z}$

repeat

1. *Compute search direction,* $\Delta \mathbf{z}$.
2. *Compute step size,* t.
3. *Update* $\mathbf{z} := \mathbf{z} + t\Delta \mathbf{z}$.

until *stopping criterion is satisfied*

5.2 Augmented Equality Constrained Problem

Consider the (convex) quadratic programming problem given by Eq. (4.36) with equivalent cost in Eq. (4.37) from Chapter 4, reproduced below:

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T\mathbf{H}\mathbf{z} - 2\mathbf{z}_t^T\mathbf{H}\mathbf{z} \\ \text{subject to} \quad & \mathbf{C}\mathbf{z} = \mathbf{b} \\ & \mathbf{P}\mathbf{z} \leq \mathbf{h} \end{aligned} \tag{5.3}$$

Interior-point methods solve a convex optimization problem with linear equality and inequality constraints, such as those given by Eq. (5.3), by reducing it to a sequence of linear equality constrained problems and applying Newton's method.

Consider the matter of reformulating the optimal control problem as an equality constrained problem. In other words, the inequality constraints $\mathbf{P}\mathbf{z} \leq \mathbf{h}$ must be included *implicitly* in the objective function. This can be done through, what is typically called, an indicator function. The basic idea of an indicator function is to penalize the cost when an inequality constraint is violated; thus, the indicator function would equal zero when $\mathbf{P}\mathbf{z} \leq \mathbf{h}$ and would equal infinity when $\mathbf{P}\mathbf{z} > \mathbf{h}$, i.e.,

$$I(x) \triangleq \begin{cases} 0, & x \leq 0 \\ \infty, & x > 0 \end{cases} \tag{5.4}$$

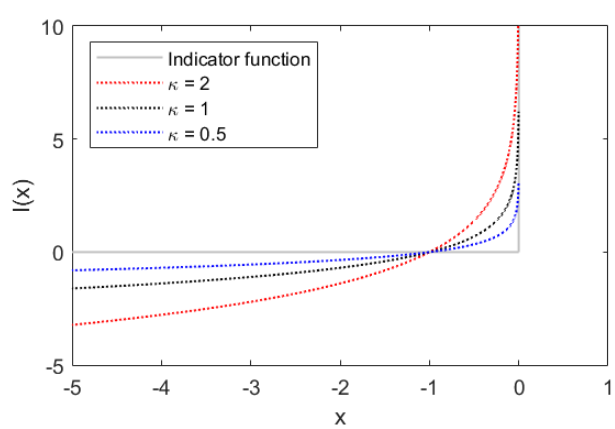


Figure 5.1: Indicator function, based on the figure in [57].

as shown by the gray line in Fig. 5.1. The indicator function is clearly nondifferentiable and this becomes problematic when determining parameter search directions since a derivative of the cost function is needed. A common workaround is to approximate the indicator function using

$$\hat{I}(x) \triangleq -\kappa \log(-x), \tag{5.5}$$

where κ is a positive tuning parameter. This is a smooth, convex function, and as κ approaches zero the approximation approaches the indicator function, as shown in Fig. 5.1. The *logarithmic barrier* for a problem with inequality constraints $\mathbf{Pz} \leq \mathbf{h}$ is defined as the summation of the logarithm of each constraint value in the vector $\mathbf{h} - \mathbf{Pz} \in \mathbb{R}^{n_{f_i}}$,

$$\phi(\mathbf{z}) \triangleq -\sum_{i=1}^{n_{f_i}} \log([\mathbf{h} - \mathbf{Pz}]_i), \tag{5.6}$$

such that the augmented objective function for the equality constrained problem is

$$J(\mathbf{z}) = \mathbf{z}^T\mathbf{Hz} - 2\mathbf{z}_t^T\mathbf{Hz} - \kappa \sum_{i=1}^{n_{f_i}} \log([\mathbf{h} - \mathbf{Pz}]_i). \tag{5.7}$$

The equality constrained problem can then be posed as:

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T\mathbf{Hz} - 2\mathbf{z}_t^T\mathbf{Hz} - \kappa \sum_{i=1}^{n_{f_i}} \log([\mathbf{h} - \mathbf{Pz}]_i) \\ \text{subject to} \quad & \mathbf{Cz} = \mathbf{b} \end{aligned} \tag{5.8}$$

5.3 Determining the Search Direction

Solving an equality constrained problem can be done using various algorithms, e.g., primal-dual interior point methods and barrier methods, but ultimately the problem reduces to solving the KKT conditions as previously developed in §4.2. This work focuses on a variation of the barrier method called the *infeasible start Newton method* (ISNM) that, unlike the barrier method, does not require the initial guess to be

feasible in the equality constraints. It is important, however, to begin with an initial guess for $\mathbf{z}$ that strictly satisfies the inequality constraints, i.e., $\mathbf{Pz} < \mathbf{h}$, since the logarithm and its gradient are undefined at zero.

5.3.1 Gradient of the Cost

To begin, consider the augmented problem in Eq. (5.8). Using the definition of the Lagrangian from Eq. (4.4), the Lagrangian associated with this problem is

$$L(\mathbf{z}, \boldsymbol{\nu}) = \mathbf{z}^T\mathbf{Hz} - 2\mathbf{z}_t^T\mathbf{Hz} - \kappa \sum_{i=1}^{n_{f_i}} \log([\mathbf{h} - \mathbf{Pz}]_i) + \boldsymbol{\nu}^T (\mathbf{Cz} - \mathbf{b}) . \tag{5.9}$$

From §4.2, the optimal solutions for $\mathbf{z}^*$ and $\boldsymbol{\nu}^*$ are found by setting the gradient of the Lagrangian with respect to $\mathbf{z}$ equal to zero.

$$\begin{aligned}
\nabla_{\mathbf{z}} L(\mathbf{z}^*, \boldsymbol{\nu}^*) &= 0 \\
\left[\mathbf{z}^T\mathbf{H} + \mathbf{z}^T\mathbf{H}^T - 2\mathbf{z}_t^T\mathbf{H} - \kappa \sum_{i=1}^{n_{f_i}} \left\{ \frac{\partial \log([\mathbf{h} - \mathbf{Pz}]_i)}{\partial \mathbf{z}} \right\} + \boldsymbol{\nu}^T\mathbf{C} \right] \Bigg|_{\substack{\mathbf{z} = \mathbf{z}^* \\ \boldsymbol{\nu} = \boldsymbol{\nu}^*}} &= 0 \\
\left[2\mathbf{z}^T\mathbf{H} - 2\mathbf{z}_t^T\mathbf{H} + \kappa\mathbf{d}^T\mathbf{P} + \boldsymbol{\nu}^T\mathbf{C}\right] \Big|_{\substack{\mathbf{z} = \mathbf{z}^* \\ \boldsymbol{\nu} = \boldsymbol{\nu}^*}} &= 0 \\
2\left(\mathbf{z}^*\right)^T \mathbf{H} - 2\mathbf{z}_t^T\mathbf{H} + \kappa\mathbf{d}^T\mathbf{P} + \left(\boldsymbol{\nu}^*\right)^T \mathbf{C} &= 0
\end{aligned} \tag{5.10}$$

where it was used that $\mathbf{H}$ is symmetric and the vector $\mathbf{d} \in \mathbb{R}^{n_{f_i}}$ was introduced and defined to be

$$\mathbf{d} \triangleq \begin{bmatrix} 1/\left(h_1 - \mathbf{P}_1\mathbf{z}\right) \\ 1/\left(h_2 - \mathbf{P}_2\mathbf{z}\right) \\ \vdots \\ 1/\left(h_{n_{f_i}} - \mathbf{P}_{n_{f_i}}\mathbf{z}\right) \end{bmatrix}, \tag{5.11}$$

where h_i and $\mathbf{P}_i$ are the rows of $\mathbf{h}$ and $\mathbf{P}$, respectively.

The gradient of the Lagrangian in Eq. (5.10) can be written as a column vector

by transposing the terms,

$$2\mathbf{H}\mathbf{z}^* - 2\mathbf{H}\mathbf{z}_t + \kappa\mathbf{P}^T\mathbf{d} + \mathbf{C}^T\boldsymbol{\nu}^* = 0, \tag{5.12}$$

with the first three terms in this expression coming from the gradient of the augmented cost ∇J.

5.3.2 Residuals

The dual residual $\mathbf{r}_d \in \mathbb{R}^{n_z}$ and primal residual $\mathbf{r}_p \in \mathbb{R}^{n_{h_i}}$ are defined as the gradient of the Lagrangian and the equality constraints, respectively,

$$\begin{cases} \mathbf{r}_d & \triangleq 2\mathbf{H}\mathbf{z} - 2\mathbf{H}\mathbf{z}_t + \kappa\mathbf{P}^T\mathbf{d} + \mathbf{C}^T\boldsymbol{\nu} \\ \mathbf{r}_p & \triangleq \mathbf{C}\mathbf{z} - \mathbf{b} \end{cases} \tag{5.13}$$

and should equal zero at the optimal solution of the augmented problem as per Eq. (5.12). The residual is defined as a stacked variable of the dual and primal residuals, $\mathbf{r} \triangleq [\mathbf{r}_d \;\; \mathbf{r}_p]^T$, and is used within the algorithm to check the state of optimality of the current solution in both $\mathbf{z}$ and $\boldsymbol{\nu}$, herein denoted by the stacked variable $\boldsymbol{\xi} \triangleq [\mathbf{z} \;\; \boldsymbol{\nu}]^T$.

The residual is used to find the search direction for $\boldsymbol{\xi}$ needed to reduce the augmented cost further. This is done by calculating a step $\Delta\boldsymbol{\xi} = [\Delta\mathbf{z} \;\; \Delta\boldsymbol{\nu}]^T$ such that the next state $\boldsymbol{\xi} + \Delta\boldsymbol{\xi}$ approximately satisfies the optimality conditions, $\mathbf{r}(\mathbf{z}^*, \boldsymbol{\nu}^*) = 0$. The first order approximation of the residual expanded at the next state is

$$\mathbf{r}(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}) \approx \mathbf{r}(\boldsymbol{\xi}) + D\mathbf{r}(\boldsymbol{\xi})\Delta\boldsymbol{\xi}. \tag{5.14}$$

The *primal-dual Newton step* is defined as the step $\Delta\boldsymbol{\xi}$ for which the Taylor approximation vanishes, i.e.,

$$D\mathbf{r}(\boldsymbol{\xi})\Delta\boldsymbol{\xi} = -\mathbf{r}(\boldsymbol{\xi}) \tag{5.15}$$

Using the definition of the residual and its relationship to the problem data from Eq.

(5.13),

$$\mathbf{r} \triangleq \begin{bmatrix} \mathbf{r}_d \\ \mathbf{r}_p \end{bmatrix} = \begin{bmatrix} \nabla J + \mathbf{C}^T \boldsymbol{\nu} \\ \mathbf{C}\mathbf{z} - \mathbf{b} \end{bmatrix} \tag{5.16}$$

the derivative of the residual is given by

$$D\mathbf{r} = \begin{bmatrix} \nabla^2 J & \mathbf{C}^T \\ \mathbf{C} & 0 \end{bmatrix}. \tag{5.17}$$

Substituting Eq. (5.17) into Eq. (5.15), the primal-dual Newton step can be written as

$$\begin{bmatrix} \nabla^2 J & \mathbf{C}^T \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z} \\ \Delta\boldsymbol{\nu} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_d \\ \mathbf{r}_p \end{bmatrix}. \tag{5.18}$$

The Hessian of the augmented cost function from the optimization problem in Eq. (5.8) is given by

$$\boldsymbol{\Phi} \triangleq \nabla^2 J = 2\mathbf{H} + \kappa \mathbf{P}^T \operatorname{diag}(\mathbf{d})^2 \mathbf{P}. \tag{5.19}$$

Substituting Eq. (5.19) into Eq. (5.18) gives

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{C}^T \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z} \\ \Delta\boldsymbol{\nu} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_d \\ \mathbf{r}_p \end{bmatrix}. \tag{5.20}$$

5.3.3 Solve using Block Elimination

The search directions $\Delta\boldsymbol{\nu}$ and $\Delta\mathbf{z}$ can be calculated from the system of linear equations in Eq. (5.20) using block elimination, whereby a subset of variables is eliminated and then the smaller system is solved. In this case, the dependence of $\Delta\boldsymbol{\nu}$ on $\Delta\mathbf{z}$ will be eliminated.

From the first equation of Eq. (5.20), isolate for $\Delta\mathbf{z}$ as

$$\Delta\mathbf{z} = -\boldsymbol{\Phi}^{-1} \left(\mathbf{r}_d + \mathbf{C}^T \Delta\boldsymbol{\nu} \right), \tag{5.21}$$

and substitute this expression into the second equation of Eq. (5.20) to get the reduced equation,

$$\mathbf{C}\boldsymbol{\Phi}^{-1}\left(\mathbf{r}_d + \mathbf{C}^T\Delta\boldsymbol{\nu}\right) = \mathbf{r}_p. \tag{5.22}$$

The reduced equation, Eq. (5.22), can be rearranged to give

$$\underbrace{\mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{C}^T}_{\mathbf{Y}}\Delta\boldsymbol{\nu} = \underbrace{\mathbf{r}_p - \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{r}_d}_{-\boldsymbol{\beta}}. \tag{5.23}$$

The matrix on the left is called the *Schur complement of* $\boldsymbol{\Phi}$, denoted by $\mathbf{Y} \in \mathbb{R}^{n_{h_i} \times n_{h_i}}$, and the right-hand side is denoted by the vector $-\boldsymbol{\beta} \in \mathbb{R}^{n_{h_i}}$:

$$\mathbf{Y} \triangleq \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{C}^T \tag{5.24}$$

$$\boldsymbol{\beta} \triangleq -\mathbf{r}_p + \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{r}_d \tag{5.25}$$

Thus $\Delta\boldsymbol{\nu}$ can be determined by solving the reduced equation

$$\mathbf{Y}\Delta\boldsymbol{\nu} = -\boldsymbol{\beta}, \tag{5.26}$$

and $\Delta\mathbf{z}$ can be determined from the first equation of Eq. (5.20),

$$\boldsymbol{\Phi}\Delta\mathbf{z} = -\mathbf{r}_d - \mathbf{C}^T\Delta\boldsymbol{\nu}. \tag{5.27}$$

The author notes a mistake in [50] where this equation has written $\boldsymbol{\nu}$ instead of $\Delta\boldsymbol{\nu}$.

5.3.4 Summary of Section

The process of solving for the search direction of the ISNM based on the work of Wang and Boyd [50] is summarized in Algorithm 2. Algorithm 2 is a sub-algorithm of Algorithm 1 that completes *Step 1*.

Algorithm 2: Compute Primal and Dual Newton Steps

1. Form the Schur complement $\mathbf{Y} = \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{C}^T$ and $\boldsymbol{\beta} = -\mathbf{r}_p + \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{r}_d$
2. Determine $\Delta\boldsymbol{\nu}$ by solving $\mathbf{Y}\Delta\boldsymbol{\nu} = -\boldsymbol{\beta}$
3. Determine $\Delta\mathbf{z}$ by solving $\boldsymbol{\Phi}\Delta\mathbf{z} = -\mathbf{r}_d - \mathbf{C}^T\Delta\boldsymbol{\nu}$

5.4 Fast MPC

To perform Algorithm 2, the inverse of the large matrices $\boldsymbol{\Phi}$ and $\mathbf{Y}$ must be computed. In MATLAB, built-in functions are provided so that the user can easily invert matrices using a command like `inv()` or through the backslash notation \. Generally speaking, however, inverting a matrix is a slow operation.

Through exploiting the structure in the compact problem, discussed in §4.4, the need for inversion can be avoided. This is done through computing the Cholesky factorization and using forward and backward substitution [50, 57]. The equations used for these three algorithms are provided in Appendix B.

The steps of Algorithm 2 are broken down in the remainder of this section and the process of performing "fast" computations is presented. This is based on Wang and Boyd's *Fast Model Predictive Control Using Online Optimization* [50].

5.4.1 Step 1: Forming the Schur complement, Y

The definition of the Schur complement is $\mathbf{Y} \triangleq \mathbf{C}\boldsymbol{\Phi}^{-1}\mathbf{C}^T$, which has dependence on the Hessian of the cost function, $\boldsymbol{\Phi}$,

$$\boldsymbol{\Phi} \triangleq \begin{bmatrix} \overline{\mathbf{R}}^{(0)} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \overline{\mathbf{Q}}^{(1)} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \overline{\mathbf{R}}^{(1)} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \overline{\mathbf{Q}}^{(N-1)} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & \overline{\mathbf{R}}^{(N-1)} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \overline{\mathbf{Q}}_f \end{bmatrix} \tag{5.28}$$

or rather, its inverse, $\mathbf{\Phi}^{-1}$,

$$\mathbf{\Phi}^{-1} = \begin{bmatrix} \tilde{\mathbf{R}}^{(0)} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \tilde{\mathbf{Q}}^{(1)} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \tilde{\mathbf{R}}^{(1)} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \tilde{\mathbf{Q}}^{(N-1)} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & \tilde{\mathbf{R}}^{(N-1)} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \tilde{\mathbf{Q}}_f \end{bmatrix}. \tag{5.29}$$

Note the difference in the elements of these two matrices: the block matrices in the original matrix $\mathbf{\Phi}$ are written with overlines, whereas the block matrices of the inverted matrix $\mathbf{\Phi}^{-1}$ are written with tildes.

Each block matrix in $\mathbf{\Phi}$ is the inverse of the corresponding block matrix in $\mathbf{\Phi}^{-1}$:

$$\begin{cases} \tilde{\mathbf{R}}^{(j)} & = \left(\overline{\mathbf{R}}^{(j)}\right)^{-1}, \quad j = 0, ..., N-1 \\ \tilde{\mathbf{Q}}^{(j)} & = \left(\overline{\mathbf{Q}}^{(j)}\right)^{-1}, \quad j = 1, ..., N-1 \\ \tilde{\mathbf{Q}}_f & = \left(\overline{\mathbf{Q}}_f\right)^{-1} \end{cases} \tag{5.30}$$

Since each of these blocks is symmetric and positive definite, they can be factored via Cholesky factorization. The Cholesky factor $\mathbf{L}$, a lower triangular matrix, of a symmetric and positive definite matrix $\mathbf{A}$ is defined as $\mathbf{A} \triangleq \mathbf{L}\mathbf{L}^T$. Applied to the blocks of $\mathbf{\Phi}$ this is:

$$\begin{cases} \overline{\mathbf{R}}^{(j)} & = \mathbf{L}_{\overline{\mathbf{R}}_j}\mathbf{L}^T_{\overline{\mathbf{R}}_j}, \quad j = 0, ..., N-1 \\ \overline{\mathbf{Q}}^{(j)} & = \mathbf{L}_{\overline{\mathbf{Q}}_j}\mathbf{L}^T_{\overline{\mathbf{Q}}_j}, \quad j = 1, ..., N-1 \\ \overline{\mathbf{Q}}_f & = \mathbf{L}_{\overline{\mathbf{Q}}_f}\mathbf{L}^T_{\overline{\mathbf{Q}}_f} \end{cases} \tag{5.31}$$

To use the relations in Eq. (5.31), the block matrices making up the Schur complement must be defined. To determine these blocks, the definition of the Schur complement, given by Eq. (5.24), in matrix form is used:

$$
\mathbf{Y} = \begin{bmatrix} -\mathbf{B}_d & \mathbf{I}_n & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I}_n & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -\mathbf{A}_d & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{I}_n & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{R}}^{(0)} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \tilde{\mathbf{Q}}^{(1)} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \tilde{\mathbf{R}}^{(1)} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \tilde{\mathbf{Q}}^{(N-1)} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & \tilde{\mathbf{R}}^{(N-1)} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \tilde{\mathbf{Q}}_f \end{bmatrix} \begin{bmatrix} -\mathbf{B}_d^T & 0 & 0 & \dots & 0 & 0 \\ \mathbf{I}_n & -\mathbf{A}_d^T & 0 & \dots & 0 & 0 \\ 0 & -\mathbf{B}_d^T & 0 & \dots & 0 & 0 \\ 0 & \mathbf{I}_n & -\mathbf{A}_d^T & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \mathbf{I}_n & -\mathbf{A}_d^T \\ 0 & 0 & 0 & \dots & 0 & -\mathbf{B}_d^T \\ 0 & 0 & 0 & \dots & 0 & \mathbf{I}_n \end{bmatrix} \tag{5.32}
$$

The expansion of this equation leads to:

$$
\mathbf{Y} = \begin{bmatrix} \mathbf{B}_d\tilde{\mathbf{R}}^{(0)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(1)} & -\tilde{\mathbf{Q}}^{(1)}\mathbf{A}_d^T & 0 & \dots & 0 \\ -\mathbf{A}_d\tilde{\mathbf{Q}}^{(1)} & \mathbf{A}_d\tilde{\mathbf{Q}}^{(1)}\mathbf{A}_d^T + \mathbf{B}_d\tilde{\mathbf{R}}^{(1)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(2)} & -\tilde{\mathbf{Q}}^{(2)}\mathbf{A}_d^T & \dots & 0 \\ 0 & -\mathbf{A}_d\tilde{\mathbf{Q}}^{(2)} & \mathbf{A}_d\tilde{\mathbf{Q}}^{(2)}\mathbf{A}_d^T + \mathbf{B}_d\tilde{\mathbf{R}}^{(2)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(3)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -\tilde{\mathbf{Q}}^{(N-1)}\mathbf{A}_d^T \\ 0 & 0 & 0 & \dots & \mathbf{A}_d\tilde{\mathbf{Q}}^{(N-1)}\mathbf{A}_d^T + \mathbf{B}_d\tilde{\mathbf{R}}^{(N-1)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}_f \end{bmatrix} \tag{5.33}
$$

From Eq. (5.33) the general rules for calculating each block matrix $\mathbf{Y}_{ij} \in \mathbb{R}^{n\times n}$ can be determined:

$$\begin{cases} \mathbf{Y}_{11} & = \mathbf{B}_d\tilde{\mathbf{R}}^{(0)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(1)} \\ \mathbf{Y}_{ii} & = \mathbf{A}_d\tilde{\mathbf{Q}}^{(i-1)}\mathbf{A}_d^T + \mathbf{B}_d\tilde{\mathbf{R}}^{(i-1)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(i)} \\ \mathbf{Y}_{i,i+1} & = \mathbf{Y}_{i+1,i}^T = -\tilde{\mathbf{Q}}^{(i)}\mathbf{A}_d^T \end{cases} \tag{5.34}$$

Using these rules and the relations in Eq. (5.30) and Eq. (5.31), the Schur complement can be formed. Consider, for example, the second rule of Eq. (5.34). This rule can be re-written as:

$$\begin{aligned} \mathbf{Y}_{ii} &= \mathbf{A}_d\tilde{\mathbf{Q}}^{(i-1)}\mathbf{A}_d^T + \mathbf{B}_d\tilde{\mathbf{R}}^{(i-1)}\mathbf{B}_d^T + \tilde{\mathbf{Q}}^{(i)} \\ &= \mathbf{A}_d\left(\overline{\mathbf{Q}}^{(i-1)}\right)^{-1}\mathbf{A}_d^T + \mathbf{B}_d\left(\overline{\mathbf{R}}^{(i-1)}\right)^{-1}\mathbf{B}_d^T + \left(\overline{\mathbf{Q}}^{(i)}\right)^{-1} \\ &= \mathbf{A}_d\left(\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T\right)^{-1}\mathbf{A}_d^T + \mathbf{B}_d\left(\mathbf{L}_{\overline{\mathbf{R}}_{i-1}}\mathbf{L}_{\overline{\mathbf{R}}_{i-1}}^T\right)^{-1}\mathbf{B}_d^T + \left(\mathbf{L}_{\overline{\mathbf{Q}}_i}\mathbf{L}_{\overline{\mathbf{Q}}_i}^T\right)^{-1} \end{aligned}$$

Through clever partitioning of this expression,

$$\mathbf{Y}_{ii} = \mathbf{A}_d\underbrace{\left(\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T\right)^{-1}\mathbf{A}_d^T}_{\mathbf{W}_1} + \mathbf{B}_d\underbrace{\left(\mathbf{L}_{\overline{\mathbf{R}}_{i-1}}\mathbf{L}_{\overline{\mathbf{R}}_{i-1}}^T\right)^{-1}\mathbf{B}_d^T}_{\mathbf{W}_2} + \underbrace{\left(\mathbf{L}_{\overline{\mathbf{Q}}_i}\mathbf{L}_{\overline{\mathbf{Q}}_i}^T\right)^{-1}\mathbf{I}}_{\mathbf{W}_3}, \tag{5.35}$$

forward and backward substitution can be used on each partition, and then a simple multiplication and summation gives the block matrix $\mathbf{Y}_{ii}$. To show how this is done, consider the first partition,

$$\mathbf{W}_1 \triangleq \left(\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T\right)^{-1}\mathbf{A}_d^T \implies \mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}\underbrace{\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T\mathbf{W}_1}_{\mathbf{W}_1^+} = \mathbf{A}_d^T \tag{5.36}$$

Through defining the subpartition $\mathbf{W}_1^+ \triangleq \mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T\mathbf{W}_1$, forward substitution (see Appendix B.1) can be used to solve for $\mathbf{W}_1^+$ from

$$\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}\mathbf{W}_1^+ = \mathbf{A}_d^T \tag{5.37}$$

and backward substitution (see Appendix B.2) can be used to solve for $\mathbf{W}_1$, which is the original partition,

$$\mathbf{L}_{\overline{\mathbf{Q}}_{i-1}}^T \mathbf{W}_1 = \mathbf{W}_1^+. \tag{5.38}$$

This process is repeated to calculate the partitions $\mathbf{W}_2$ and $\mathbf{W}_3$, and this technique is used to calculate the other blocks of $\mathbf{Y}$ as defined by Eq. (5.34). The Schur complement can then be built as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} & 0 & \dots & 0 & 0 \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} & \mathbf{Y}_{23} & \dots & 0 & 0 \\ 0 & \mathbf{Y}_{32} & \mathbf{Y}_{33} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \mathbf{Y}_{N-1,N-1} & \mathbf{Y}_{N-1,N} \\ 0 & 0 & 0 & \dots & \mathbf{Y}_{N,N-1} & \mathbf{Y}_{NN} \end{bmatrix}. \tag{5.39}$$

5.4.2 Step 2: Solving for $\Delta\nu$

Once $\mathbf{Y}$ has been formed, it must be factored via Cholesky factorization, i.e., $\mathbf{Y} = \mathbf{L}\mathbf{L}^T$. The Cholesky factorization of a block diagonal matrix like $\mathbf{Y}$ can be computed quite efficiently, and the algorithm to do so is given in Appendix B.4.

Then, the dual step $\Delta\boldsymbol{\nu}$ can be calculated from Eq. (5.26) via forward and backward substitution:

$$\mathbf{Y}\Delta\nu = \mathbf{L}\underbrace{\mathbf{L}^T\Delta\nu}_{\mathbf{W}} = -\boldsymbol{\beta} \tag{5.40}$$

First, define the matrix $\mathbf{W} \triangleq \mathbf{L}^T\Delta\boldsymbol{\nu}$. Then, solve for $\mathbf{W}$ by forward substitution,

$$\mathbf{L}\mathbf{W} = -\boldsymbol{\beta}, \tag{5.41}$$

and solve for $\Delta\boldsymbol{\nu}$ via backward substitution,

$$\mathbf{L}^T\Delta\boldsymbol{\nu} = \mathbf{W}. \tag{5.42}$$

5.4.3 Step 3: Solving for $\Delta\mathbf{z}$

During the construction of the Schur complement in §5.4.1, the blocks of $\tilde{\mathbf{Q}}^{(j)}$, $\tilde{\mathbf{R}}^{(j)}$, and $\tilde{\mathbf{Q}}_f$ are computed through the process of solving the partitions via forward and backward substitution. As a result, the inverse matrix $\mathbf{\Phi}^{-1}$ can be built from these blocks via Eq. (5.29). The Newton step Δz can be directly calculated via Eq. (5.27).

5.5 Determining a Step Size

The next step of Algorithm 1, *Step 2*, is concerned with determining the step size t that the next solution iteration will take along the search direction.

The *backtracking line search* is an effective method that is commonly used. The step size t is initialized to a value of unity and repeatedly decreased by a value of $\beta \in (0, 1)$ if the inequality constraints are violated at the next solution iteration. The step size is then refined further to decrease the residual of the current solution scaled by a factor of $\alpha \in (0, 1/2)$ smaller. This process is summarized in Algorithm 3.

Algorithm 3: Backtracking Line Search

Data: $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$

$t := 1$

repeat

  $t := \beta t$ `// Ensure feasibility of initial step size`

until all $\mathbf{h} - \mathbf{P}(\mathbf{z} + t\Delta\mathbf{z}) > 0$

while $||\mathbf{r}(\mathbf{z} + t\Delta\mathbf{z}, \boldsymbol{\nu} + t\Delta\boldsymbol{\nu})||_2 > (1 - \alpha t)||\mathbf{r}(\mathbf{z}, \boldsymbol{\nu})||_2$ **do**

  $t := \beta t$ `// Line search`

end

5.6 Summary of Algorithm

Algorithm 1, with sub-Algorithms 2 and 3, is repeated until the residual, which equals zero for an optimal solution, is below a user-defined tolerance ϵ. The entire process is repeated for decreasing values of κ, which are used to weigh the inequality constraints in the augmented cost function of Eq. (5.8). As $\kappa \to 0$, the problem approaches the original cost function of Eq. (5.3) and so the solution approaches that of the original optimization problem that was posed.

The entirety of the ISNM is given in Algorithm 4.

Algorithm 4: Infeasible Start Newton Method

for $\kappa = \{\kappa_1, \kappa_2, \kappa_3, \dots\}$ **do**

  repeat

    1. *Compute primal and dual Newton steps*, $\Delta\mathbf{z}$ and $\Delta\boldsymbol{\nu}$
    2. *Backtracking line search* on $||\mathbf{r}||_2$
    3. *Update* $\mathbf{z} := \mathbf{z} + t\Delta\mathbf{z}$ and $\boldsymbol{\nu} := \boldsymbol{\nu} + t\Delta\boldsymbol{\nu}$

  until $(||\mathbf{r}(\mathbf{z}, \boldsymbol{\nu})||_2 \leq \epsilon$ and $\mathbf{C}\mathbf{z} = \mathbf{b})$ or *maximum iterations exceeded*

end

Algorithm 4 can be computed much faster through two simple variations. The first variation is to decrease the number of loops for κ. It was found that using a fixed κ value, where κ was not too small, gave a good quality solution as compared to solving the full problem [50]. By fixing κ, the number of Newton iterations to solve the problem decreases significantly. Additionally, by fixing κ the warm start from the previous solution gives a good initial guess for the next solution. In terms of choosing a κ value, the authors of [50] suggest decreasing the current κ by factors of 10 until a drop in quality of the solution is shown in simulation.

The second variation to speed up Algorithm 4 is to decrease the number of maximum iterations to a value between 3 and 10. As a result, the solution may not be primal feasible, but the inequality constraints are satisfied and the control is still of high quality [50]. The argument for this approach is that, other than the control

force for the first sampling instant within the horizon, the rest of the plan is not used. Thus, it is not imperative that the plan be perfectly computed. The concept of planning is only to ensure the current control force does not have a negative effect on the future behaviour of the system [50].

In this thesis, Algorithm 4 is used with a fixed κ value and a maximum iteration count of ten. There are two variations on this solver:

Custom Solver 1: `ISNM`

The first, referred to as `ISNM` for the remainder of the thesis, computes the search directions directly from Eq. (5.26) and Eq. (5.27) using the MATLAB backslash notation, \, to invert the matrices.

Custom Solver 2: `Fast MPC`

The second, referred to as `Fast MPC` for the remainder of the thesis, computes the search directions following the approach outlined in §5.4 whereby the inversion of matrices is avoided through use of Cholesky factorization and forward and backward substitution.

5.7 Black Box Solvers

The performance of `ISNM` and `Fast MPC` is compared to so-called black box solvers that are open-source, e.g., `IPOPT`, or available through MATLAB, e.g., `fmincon` and `quadprog`. A brief description of each solver is provided in this section. For a more thorough description the interested reader is referred to the relevant works cited.

Black Box Solver 1: `IPOPT`

`IPOPT` is an open-source nonlinear optimization solver that uses a primal-dual interior point method similar to Algorithm 4 [60]: the optimization problem is converted into an equality constrained problem using the logarithmic barrier, the primal-dual Newton steps are calculated using Cholesky factorization of the Hessian matrix, and the step size is calculated using a backtracking line search.

The `IPOPT` solver differs, however, in that it has a sophisticated approach to reducing the κ parameter used to weigh the constraint violations in the cost function. Additionally, it uses a filter to ensure that the backtracking line search sufficiently decreases the cost or constraint violations, and also implements a second-order correction for this filter. Further information can be found in [61].

For this thesis, `IPOPT` is implemented in MATLAB via OPTI Toolbox[1] and is used to solve both a nonlinear and linear optimal control problem.

Black Box Solver 2: `fmincon`

The algorithm used by MATLAB's `fmincon` is a sequential quadratic programming approach that implements an active-set strategy[2] whereby the algorithm keeps track of which constraints are active, i.e., they hold with equality at the solution. A linear programming problem is solved to find a feasible solution for the initial guess and then the Hessian of the Lagrangian is approximated to form a QP subproblem. The set of active constraints is updated every iteration and used to calculate the search direction and step size [62].

In this thesis, `fmincon` is used to solve a linear optimal control problem.

Black Box Solver 3: `quadprog`

MATLAB's `quadprog` is a QP problem solver that, similar to `fmincon`, uses an active-set method. This method is the only algorithm of `quadprog` that is compatible with Simulink and code generation necessary for experimental validation with SPOT. For this algorithm[3], the active constraints are treated temporarily as equalities in the subproblem. The subproblem is solved in the primal-dual way using Cholesky factorization [63].

In this thesis, `quadprog` is used to solve a linear optimal control problem.

[1] Industrial Information & Control Centre, "OPTI Toolbox", n.d. (accessed May 18, 2021) http://www.i2c2.aut.ac.nz/Resources/Software/OptiToolbox_Original.html

[2] MathWorks, "Constrained Nonlinear Optimization Algorithms", n.d. (accessed July 20, 2022) https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[3] MathWorks, "Quadratic Programming Algorithms", n.d. (accessed July 20, 2022) https://www.mathworks.com/help/optim/ug/quadratic-programming-algorithms.html

Chapter 6

Three-Dimensional Simulations

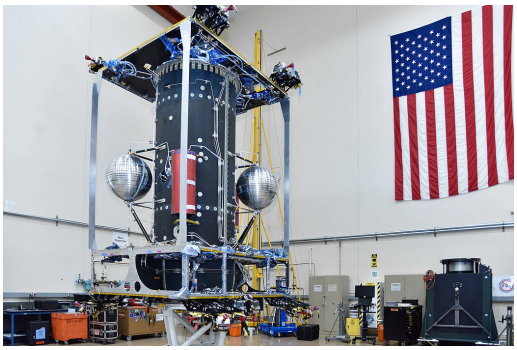
This chapter poses an optimal control problem specific to a three-dimensional rendezvous mission formulated using Hill's equations. The reference scenario used is NASA's OSAM-1 mission that will rendezvous with Landsat 7, an Earth observation satellite. The inequality constraints specific for this scenario are developed mathematically and the guidance problem is subsequently solved using the custom and black box solvers discussed in Chapter 5. Simulation results for two different test cases are presented.

6.1 Reference Scenario

NASA's OSAM-1 mission, to be launched sometime after 2025, will demonstrate never-before tested technologies including refuelling a satellite in space[1,2]. The OSAM-1 spacecraft bus is being supplied by Maxar Technologies, Fig. 6.1a, and will be delivered to NASA in late 2022 for integration and testing. OSAM-1 will be equipped with three robotic arms, one of which will be used to grapple the target spacecraft, Landsat 7, during the refuelling demonstration. The testing of OSAM-1's robotic

[1]NASA, "NASA's On-orbit Servicing, Assembly, and Manufacturing 1 Mission Ready for Spacecraft Build," May 2021 (accessed July 15, 2022) https://www.nasa.gov/image-feature/goddard/2021/nasa-s-on-orbit-servicing-assembly-and-manufacturing-1-mission-ready-for-spacecraft

[2]NASA, "NASA's Robotic OSAM-1 Mission Completes its Critical Design Review," March 2022 (accessed July 15, 2022) https://www.nasa.gov/feature/goddard/2022/nasa-s-robotic-osam-1-mission-completes-its-critical-design-review



(a) The OSAM-1 propulsion system. Credit: Maxar Technologies.

(b) A Landsat programme satellite. Credit: NASA.

Figure 6.1: The players of OSAM-1's refueling demonstration: (a) the propulsion system of OSAM-1 as of February 2021, and (b) an artist's rendition of a Landsat programme satellite.

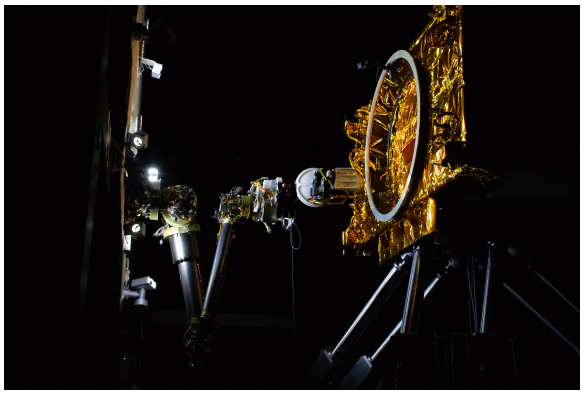


Figure 6.2: Grapple testing of OSAM-1's robotic servicing arm (left) to a partial model of the target satellite (right) in the Robotic Operations Center at NASA's Goddard Space Flight Center. Credit: NASA.

arm grappling a partial model of the Landsat 7 satellite is shown in Fig. 6.2. Landsat 7 is an Earth observation satellite that was launched in 1999 and orbits at an altitude of 705 km in a sun-synchronous orbit (SSO) where it monitors the Earth's land and coastal areas[3]. An artist's rendition of a Landsat satellite is shown in Fig. 6.1b.

[3]ESA, "Landsat-7", n.d. (accessed July 15, 2022) https://earth.esa.int/eogateway/missions/landsat-7

6.2 Problem Formulation

This scenario models the *closing* phase of the rendezvous problem (see §3.1.1), whereby the chaser spacecraft is within sufficient proximity of the target spacecraft for Hill's equations to be used for its relative motion model. Only the translational motion of the chaser spacecraft is considered.

6.2.1 Cost Function

From the definition of the LVLH frame in §2.1.2, the target is located at the origin of the coordinate system and it is the objective of the chaser spacecraft to close the distance between itself and the target whilst minimizing the control effort associated with the maneuver. The compact cost function as described in §4.4 applies where the desired position of the chaser, $\mathbf{z}_t$, is the origin:

$$J(\mathbf{z}) = \mathbf{z}^T \mathbf{H} \mathbf{z} \tag{6.1}$$

Since OSAM-1 will utilize a robotic arm for mating, the chaser will need to end its rendezvous maneuver an established distance away from the target, called the holding position, as shown in Fig. 6.3. The holding distance r_{hold} is set to the length of one of the robotic arms of the OSAM-1 spacecraft, 2.3 m [64].

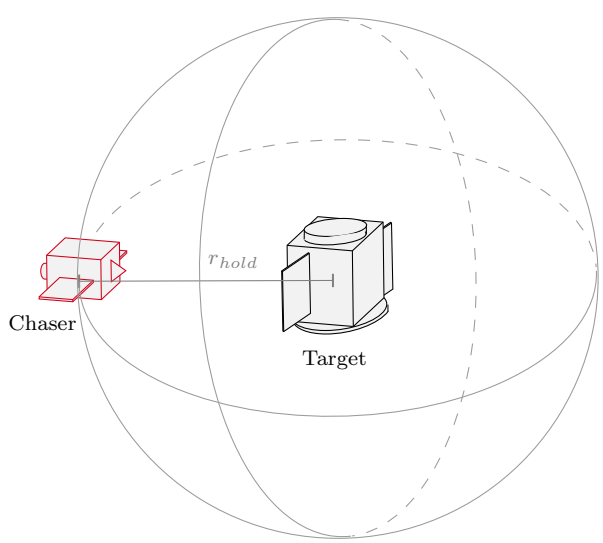


Figure 6.3: A visualization of the holding position of the chaser.

6.2.2 Equality Constraints

As previously mentioned, Hill's equations are used for the relative motion model,

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}}_{\mathbf{u}}, \tag{6.2}$$

discretized using a zero-order hold as in Eq. (3.3). As apparent by Eq. (6.2), it is necessary to define the mass of the chaser spacecraft m and the mean motion $n = \sqrt{\mu_\oplus / a^3}$ of the target spacecraft, where $\mu_\oplus = 3.986 \times 10^5$ km^3/s^2 is the gravitational parameter of the Earth and a is the semi-major axis of the target's orbit. These parameters are easily defined since Landsat 7 orbits at an altitude of 705 km, and the literature suggests that the planned wet mass of OSAM-1 at launch will be ∼6500 kg [65]. With these values and a discretization time of 15 s, the discrete-time state and control matrices $\mathbf{A}_d$ and $\mathbf{B}_d$ are

$$\mathbf{A}_d = \begin{bmatrix} 1.0004 & 0 & 0 & 14.9994 & 0.2383 & 0 \\ -4.0096 \times 10^{-6} & 1 & 0 & -0.2383 & 14.9975 & 0 \\ 0 & 0 & 0.9999 & 0 & 0 & 14.9994 \\ 5.0475 \times 10^{-5} & 0 & 0 & 0.9999 & 0.0318 & 0 \\ -8.0191 \times 10^{-7} & 0 & 0 & -0.0318 & 0.9995 & 0 \\ 0 & 0 & -1.6825 \times 10^{-5} & 0 & 0 & 0.9999 \end{bmatrix} \tag{6.3}$$

and

$$\mathbf{B}_d = \begin{bmatrix} 0.0173 & 1.8331 \times 10^{-4} & 0 \\ -1.8331 \times 10^{-4} & 0.0173 & 0 \\ 0 & 0 & 0.0173 \\ 0.0023 & 3.6661 \times 10^{-5} & 0 \\ -3.6661 \times 10^{-5} & 0.0023 & 0 \\ 0 & 0 & 0.0023 \end{bmatrix}. \tag{6.4}$$

6.2.3 Inequality Constraints

This rendezvous scenario considers limitations to the state and control variables in the form of maximum thrust limits and obstacle avoidance constraints, as will be mathematically formulated in this section.

Thruster Limitations

The thrusters onboard the chaser spacecraft will be limited in the amount of thrust they can provide. At each step within the horizon, there will be a maximum thrust u_{max} that applies to the radial x, in-track y, and cross-track z directions:

$$\begin{cases} |u_x^{(k)}| \leq u_{max} \\ |u_y^{(k)}| \leq u_{max} \\ |u_z^{(k)}| \leq u_{max} \end{cases} \tag{6.5}$$

These thrust constraints are linear and can be written in a more compact form using the optimization vector $\mathbf{z}$ defined in §4.4, reproduced below for convenience:

$$\mathbf{z} \triangleq \begin{bmatrix} \mathbf{u}^{(k)} & \mathbf{x}^{(k+1)} & \mathbf{u}^{(k+1)} & \dots & \mathbf{x}^{(k+N-1)} & \mathbf{u}^{(k+N-1)} & \mathbf{x}^{(k+N)} \end{bmatrix}^T \in \mathbb{R}^{n_z}$$

Defining the matrix $\mathbf{U} \in \mathbb{R}^{mN \times n_z}$ to be

$$\mathbf{U} \triangleq \begin{bmatrix} \mathbf{I}_m & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_m & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{I}_m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \mathbf{I}_m & 0 \end{bmatrix}, \tag{6.6}$$

where $\mathbf{I}_m \in \mathbb{R}^{m \times m}$ is the identity matrix, and the vector $\mathbf{k} \in \mathbb{R}^{mN}$ as

$$\mathbf{k} \triangleq \begin{bmatrix} u_{max} & u_{max} & u_{max} & \dots & u_{max} & u_{max} & u_{max} \end{bmatrix}^T, \tag{6.7}$$

the thrust constraints can be written as

$$\pm \mathbf{U}\mathbf{z} \leq \mathbf{k}. \tag{6.8}$$

Maxar Technologies has based the OSAM-1 bus on their 1300-Class spacecraft platform[4], which generally uses chemical bipropellant liquid rocket engines[5]. Commonly seen on spacecraft are the 10 N bipropellant thrusters[6], which have been flown on notable missions: Galileo (1989) to Jupiter, Mars Express (2003) to Mars, and LISA Pathfinder (2015) which was a proof-of-concept mission that tested the feasibility of an ESA orbiting laser observatory for gravitational wave detection. Assuming that there are two 10 N bipropellant thrusters on each side of the chaser spacecraft, the maximum thrust limit is $u_{max} = 20\,\mathrm{N}$.

[4]Maxar Technologies, "OSAM-1 Passes Accommodation CDR, Propulsion System Module Complete," May 2021 (accessed July 15, 2022) https://blog.maxar.com/space-infrastructure/2021/osam-1-passes-accommodation-cdr-propulsion-system-module-complete

[5]Maxar Technologies, "Maxar 1300-Class Spacecraft Bus for RSDO Applications," n.d. (accessed July 15, 2022) https://rsdo.gsfc.nasa.gov/images/catalog-rapidIV/Maxar_1300_Data_sheet-Rapid_IV.pdf

[6]ArianeGroup's Orbital Propulsion Centre, "10N Bi-Propellant Thruster," n.d. (accessed July 15, 2022) https://www.space-propulsion.com/spacecraft-propulsion/bipropellant-thrusters/10-bipropellant-thrusters.html

Obstacle Avoidance

As more spacecraft are launched into orbit the near-Earth environment is becoming increasingly crowded, as shown in Fig. 1.1 that helped motivate this research. Landsat 7, in particular, is orbiting at a 705 km altitude which, according to Fig. 1.1, has roughly 200 objects larger than 10 cm it must actively avoid. Obstacle avoidance constraints can be written in the form of hyperplanes, i.e., rotating hyperplanes [20, 21, 66] or dual hyperplanes [23], or treated with an elliptical exclusion zone [25].

The elliptical exclusion zone offers the least conservative method of dealing with obstacles thereby promoting the most optimal trajectory. As shown in Fig. 6.4, an ellipsoidal keep-out-zone (KOZ) is drawn around the obstacle that identifies the region the chaser spacecraft must remain outside of.

The inequality constraint associated with this KOZ is directly related to the general equation of an ellipsoid centered at (x_0, y_0, z_0) with axes a, b, and c, as shown in Fig. 6.5,

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2} = 1. \tag{6.9}$$

The factors a^{-2}, b^{-2}, and c^{-2}, which give the shape of the ellipsoid, can be wrapped into a shape matrix, $\mathbf{S}$,

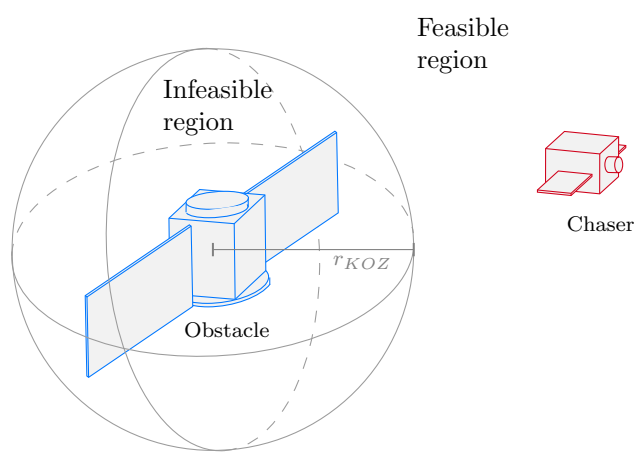


Figure 6.4: An ellipsoidal keep-out-zone around the obstacle defines an infeasible region for the spacecraft trajectory.

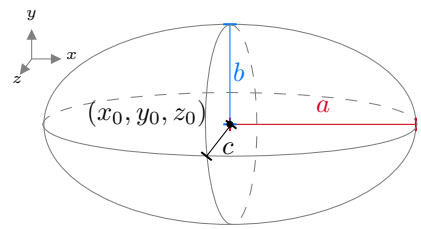


Figure 6.5: An ellipsoid with axes a, b, and c.

$$\mathbf{S} = \begin{bmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{bmatrix}. \tag{6.10}$$

The permissible values for the chaser spacecraft are outside or on the ellipsoid, so the obstacle avoidance constraint can be written from Eq. (6.9) where the left-hand side must be greater than or equal to the right-hand side. Defining the obstacle position as the center of the ellipsoid, $\mathbf{r}_{obj}^{(k)} \triangleq \begin{bmatrix} x_0^{(k)} & y_0^{(k)} & z_0^{(k)} \end{bmatrix}^T$, and the free parameters $\mathbf{r}^{(k)} \triangleq \begin{bmatrix} x^{(k)} & y^{(k)} & z^{(k)} \end{bmatrix}^T$ as the position of the chaser at sampling instant k, the obstacle avoidance constraint at each step in the horizon is given as

$$1 \leq (\mathbf{r}^{(k)} - \mathbf{r}_{obj}^{(k)})^T \mathbf{S} (\mathbf{r}^{(k)} - \mathbf{r}_{obj}^{(k)}). \tag{6.11}$$

This obstacle avoidance constraint is nonlinear, which makes solving the optimal control problem difficult (NP-hard) [24]. Alternative methods using hyperplanes produce linear constraints, with direct linearization being of particular interest. It has been shown that trajectories produced using the direct linearization obstacle avoidance constraints stay closer to the optimal path as compared to trajectories found using a dual hyperplane or rotating hyperplane approach, with all methods requiring a similar computation time [24]. Let the right-hand side of Eq. (6.11) define the function $f : \mathbb{R}^3 \to \mathbb{R}$, such that (dropping the k notation for now)

$$f(\mathbf{r}) = (\mathbf{r} - \mathbf{r}_{obj})^T \mathbf{S} (\mathbf{r} - \mathbf{r}_{obj}), \tag{6.12}$$

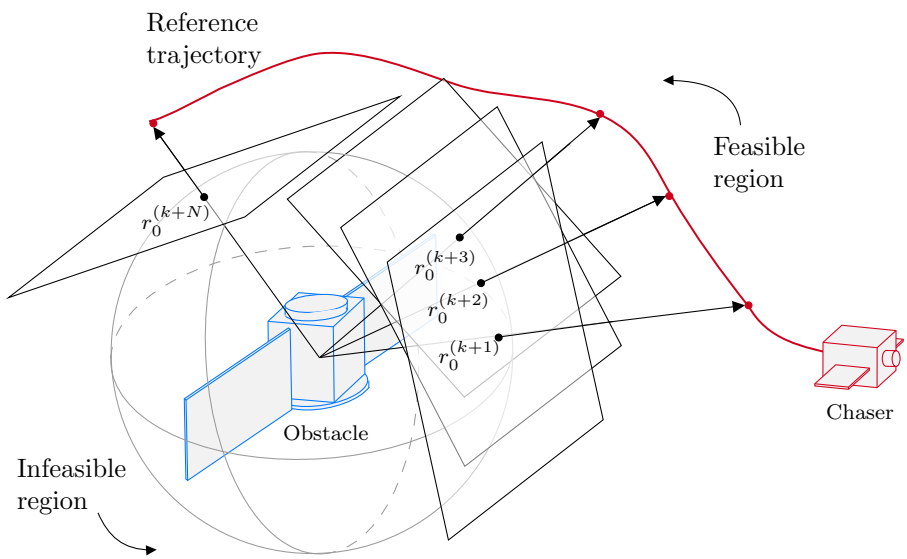


Figure 6.6: A visualization of the resulting hyperplanes from the direct linearization method.

and the nonlinear obstacle constraint in Eq. (6.11) becomes

$$1 \leq f(\mathbf{r}). \tag{6.13}$$

Direct linearization takes the first order Taylor series expansion of the nonlinear constraint term f in Eq. (6.13). The Taylor series expansion is done around the point on the boundary of the KOZ between the obstacle's center and the position of the chaser, denoted by $\mathbf{r}_0^{(k)}$ for each sampling instant k in the horizon, as shown in Fig. 6.6. Here, the reference trajectory of the chaser from the previous iteration is used to determine the expansion points, $\mathbf{r}_0^{(k)}$, over the horizon. In this way, the hyperplane that serves as the obstacle constraint rotates about the obstacle at the same rate as the chaser passes around it. This is similar to the rotating hyperplane method, however, the difference is that the rotation rate of the hyperplane is no longer fixed, but calculated. The linear expansion of f gives the linearized version of the obstacle constraint,

$$1 \leq f(\mathbf{r}_0) + \left.\frac{\partial f}{\partial \mathbf{r}}\right|_{\mathbf{r}_0} (\mathbf{r} - \mathbf{r}_0). \tag{6.14}$$

The author notes the mistake in this formula in [24] where the last term is written with $\mathbf{r}_{obj}$ instead of $\mathbf{r}_0$. The function f from Eq. (6.12) can be differentiated with respect to the vector $\mathbf{r}$ as

$$\frac{\partial f}{\partial \mathbf{r}} = (\mathbf{r} - \mathbf{r}_{obj})^T \left[\frac{\partial \mathbf{S}(\mathbf{r} - \mathbf{r}_{obj})}{\partial \mathbf{r}}\right] + [\mathbf{S}(\mathbf{r} - \mathbf{r}_{obj})]^T \left[\frac{\partial (\mathbf{r} - \mathbf{r}_{obj})}{\partial \mathbf{r}}\right] \tag{6.15}$$

and simplified as

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{r}} &= (\mathbf{r} - \mathbf{r}_{obj})^T [\mathbf{S}] + [\mathbf{S}(\mathbf{r} - \mathbf{r}_{obj})]^T [\mathbf{I}] \\
&= (\mathbf{r} - \mathbf{r}_{obj})^T \mathbf{S} + (\mathbf{r} - \mathbf{r}_{obj})^T \mathbf{S}^T \\
&= (\mathbf{r} - \mathbf{r}_{obj})^T \left[\mathbf{S} + \mathbf{S}^T\right] \\
&= 2(\mathbf{r} - \mathbf{r}_{obj})^T \mathbf{S}
\end{aligned} \tag{6.16}$$

since $\mathbf{S}$ is symmetric. Evaluating Eq. (6.16) at $\mathbf{r}_0$ gives

$$\left.\frac{\partial f}{\partial \mathbf{r}}\right|_{\mathbf{r}_0} = 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S} \tag{6.17}$$

Substituting Eq. (6.12) and Eq. (6.17) into the linearized obstacle constraint in Eq. (6.14) gives

$$\begin{aligned}
1 &\leq (\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) + 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r} - \mathbf{r}_0) \\
1 &\leq (\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) + 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r} - \mathbf{r}_0 + \mathbf{r}_{obj} - \mathbf{r}_{obj}) \\
1 &\leq (\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) + 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}\left[(\mathbf{r} - \mathbf{r}_{obj}) - (\mathbf{r}_0 - \mathbf{r}_{obj})\right] \\
1 &\leq (\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) + 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r} - \mathbf{r}_{obj}) - 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) \\
1 &\leq 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r} - \mathbf{r}_{obj}) - (\mathbf{r}_0 - \mathbf{r}_{obj})^T \mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj})
\end{aligned} \tag{6.18}$$

Equation (6.18) is the linearized obstacle constraint equation. As with the other linear equations, it can be written in compact notation. First rearrange Eq. (6.18)

such that the $\mathbf{r}$ term is on the left-hand side:

$$
\begin{aligned}
1 &\leq 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}(\mathbf{r} - \mathbf{r}_{obj}) - (\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) \\
1 &\leq 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}\mathbf{r} - 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}\mathbf{r}_{obj} - (\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}(\mathbf{r}_0 - \mathbf{r}_{obj}) \\
1 &\leq 2(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}\mathbf{r} - (\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}(\mathbf{r}_0 + \mathbf{r}_{obj}) \\
-2\underbrace{(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}}_{\mathbf{E}}\mathbf{r} &\leq -1 - \underbrace{(\mathbf{r}_0 - \mathbf{r}_{obj})^T\mathbf{S}}_{\mathbf{E}}(\mathbf{r}_0 + \mathbf{r}_{obj})
\end{aligned}
\tag{6.19}
$$

Defining $\mathbf{E}^{(k)} \triangleq (\mathbf{r}_0^{(k)} - \mathbf{r}_{obj}^{(k)})^T\mathbf{S} \in \mathbb{R}^{1\times 3}$, the expression in Eq. (6.19) can be written as

$$
-2\mathbf{E}^{(k)}\mathbf{r}^{(k)} \leq -1 - \mathbf{E}^{(k)}(\mathbf{r}_0^{(k)} + \mathbf{r}_{obj}^{(k)}) \tag{6.20}
$$

A matrix $\mathbf{D} \in \mathbb{R}^{N\times n_z}$ can be built from the blocks of $\mathbf{E}^{(k)}$ to apply this constraint to each step in the horizon:

$$
\mathbf{D} \triangleq \begin{bmatrix}
0 & \mathbf{E}^{(k+1)} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{E}^{(k+2)} & 0 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & \mathbf{E}^{(k+N)} & 0
\end{bmatrix} \tag{6.21}
$$

Similarly, column vectors $\bar{\mathbf{r}}_0 \in \mathbb{R}^{n_z}$ and $\bar{\mathbf{r}}_{obj} \in \mathbb{R}^{n_z}$ can be built that contain the expansion points $\mathbf{r}_0^{(k)}$ and positions of the obstacle $\mathbf{r}_{obj}^{(k)}$ over the horizon,

$$
\bar{\mathbf{r}}_0 \triangleq [0 \quad \mathbf{r}_0^{(k+1)} \quad 0 \quad \mathbf{r}_0^{(k+2)} \quad 0 \quad \dots \quad 0 \quad \mathbf{r}_0^{(k+N)} \quad 0]^T \tag{6.22}
$$

and

$$
\bar{\mathbf{r}}_{obj} \triangleq [0 \quad \mathbf{r}_{obj}^{(k+1)} \quad 0 \quad \mathbf{r}_{obj}^{(k+2)} \quad 0 \quad \dots \quad 0 \quad \mathbf{r}_{obj}^{(k+N)} \quad 0]^T, \tag{6.23}
$$

respectively. Using these matrices, the compact linearized obstacle avoidance constraint can be written as

$$
-2\mathbf{D}\mathbf{z} \leq -\mathbf{1}_N - \mathbf{D}(\bar{\mathbf{r}}_0 + \bar{\mathbf{r}}_{obj}). \tag{6.24}
$$

where $\mathbf{1}_N$ is a column vector of ones with length N.

The Chinese ASAT in 2007 that targeted the Fengyun-1C satellite, which was originally in an 845 km by 865 km SSO, produced thousands of pieces of debris whose orbits extend from 200 km to 4000 km [8]. The Landsat 7, in a 705 km SSO, is within this dangerous zone and so a large obstacle that represents the debris cloud is modelled. Obstacle sizes ranging from 50 m to 1500 m have been used in the literature [21, 24, 67]. To ensure that the thrust-limited chaser spacecraft is able to accelerate appropriately around the obstacles, only relatively slow moving obstacles are considered.

6.2.4 Formulation

Using the cost function in Eq. (6.1), equality constraints, and inequality constraints as defined in the last section, the optimal control problem can be formulated. Both a nonlinear and linear optimal control problem are written where the difference comes from the obstacle avoidance constraint used: the nonlinear problem uses the nonlinear obstacle avoidance constraint in Eq. (6.11) whereas the linear problem uses the linearized obstacle avoidance constraint in Eq. (6.24) for each obstacle j. Both problems use the linear maximum thrust constraint as given by Eq. (6.8).

With proper reordering of the rows of the inequality constraints as per §4.4, the inequality constraints of the linear problem can be written compactly as $\mathbf{Pz} \leq \mathbf{h}$.

Nonlinear Optimal Control Problem

$$\begin{aligned}
\text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T\mathbf{H}\mathbf{z} \\
\text{subject to} \quad & \mathbf{Cz} = \mathbf{b} \\
& \pm\mathbf{Uz} \leq \mathbf{k} \\
& 1 \leq (\mathbf{r}^{(k)} - \mathbf{r}^{(k)}_{obj,j})^T\mathbf{S}_j(\mathbf{r}^{(k)} - \mathbf{r}^{(k)}_{obj,j}), \quad k = 2, \ldots, N+1, \quad j = 1, 2, \ldots
\end{aligned} \tag{6.25}$$

Linear Optimal Control Problem

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T \mathbf{H} \mathbf{z} \\ \text{subject to} \quad & \mathbf{C}\mathbf{z} = \mathbf{b} \\ & \mathbf{P}\mathbf{z} \leq \mathbf{h} \end{aligned} \tag{6.26}$$

The trajectory of the chaser spacecraft is likely to be less fuel optimal for the linear problem as compared to the nonlinear problem since it is subject to a more conservative constraint. Furthermore, the trajectory taken by the linear problem is strongly influenced by the initial guess since the reference trajectory determines the feasible regions of the problem, as shown in Fig. 6.6. Nevertheless, the linear problem can be solved much more efficiently than the nonlinear problem.

6.3 Numerical Simulations

This section presents the simulations performed, whereby the nonlinear optimal control problem is solved using `IPOPT` and the linear optimal control problem is solved using `IPOPT`, `fmincon`, `quadprog`, `ISNM`, and `Fast MPC`. Simulations were executed on a Quad Core 1.99 GHz Intel Core i7 64bit CPU with 16 GB RAM.

Two test cases are presented with different chaser and obstacle initial conditions given in Table 6.1. The constants used in the simulation are given in Table 6.2, including the discretization time T_d, horizon length N, weights in the cost function $\mathbf{Q}$ and $\mathbf{R}$, and the parameters used in the custom solvers. In the analysis, the trajectories, average computation times, and total impulses for the test cases are compared

Table 6.1: Initial states of the chaser spacecraft and two obstacles for Test Cases A and B given in the LVLH frame. States are given in the form $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ with units of m and m/s.

Parameter	Value	
	Test Case A	Test Case B
Chaser Initial State	(50,-1000,10,0,0,0)	(0,-500,10,0.2828,0,0)
Obstacle 1 Initial State	(-30,-800,10,-0.075,-0.005,0.015)	(-20,-400,25,0.1,0.4,0)
Obstacle 2 Initial State	(-100,0,-40,0.07,-0.16,0.01)	(-5,-100,0,-0.035,0,0.01)

Table 6.2: The optimization problem parameters for the simulations.

Parameter		Value
Mission Variables	Altitude, km	705
	Chaser mass m, kg	6500
	Rendezvous final distance, m	2.3
Problem Variables	Horizon N	30
	Discretization time T_d, s	15
	$\mathbf{Q}$	diag(1,1,1,100,100,100)
	$\mathbf{R}$	diag(100,100,100)
	Max iterations	10
	Initial guess, N	(0,15,0)
Thruster Constraint	Maximum thrust u_{max}, N	20
Obstacle Constraint	Obstacle KOZ r_{KOZ}, m	A: (20,50) B: (70,25)
Custom Solver	Fixed κ	0.05
	Backtracking algorithm: α	0.01
	Backtracking algorithm: β	0.95
	Residual tolerance ϵ	10^{-9}

when obstacles are present and absent.

6.3.1 Test Case A

The trajectory of the chaser for Test Case A without obstacles is shown in Fig. 6.7 for each of the solvers. The chaser spacecraft is represented by the red circle and the target is represented by the black circle at the origin. The progress of the solution is depicted at four different snapshots, shown at $k = 1$, 24, 57, and 143. The solutions of each solver for this scenario are quite comparable.

The corresponding trajectory for Test Case A with obstacles is shown in Fig. 6.8. Here, the two obstacles are additionally shown as blue spheres. The paths travelled by the obstacles throughout the simulation are depicted by the blue lines. Comparing the trajectories between Figs. 6.7 and 6.8, it is clear that the presence of the obstacles has changed the nominal trajectory. Figure 6.8 also shows that the solutions of the solvers diverge quite early in the simulation causing the chaser to find different paths around the first-encountered obstacle. The solutions of the solvers converge once more upon approaching the target spacecraft.

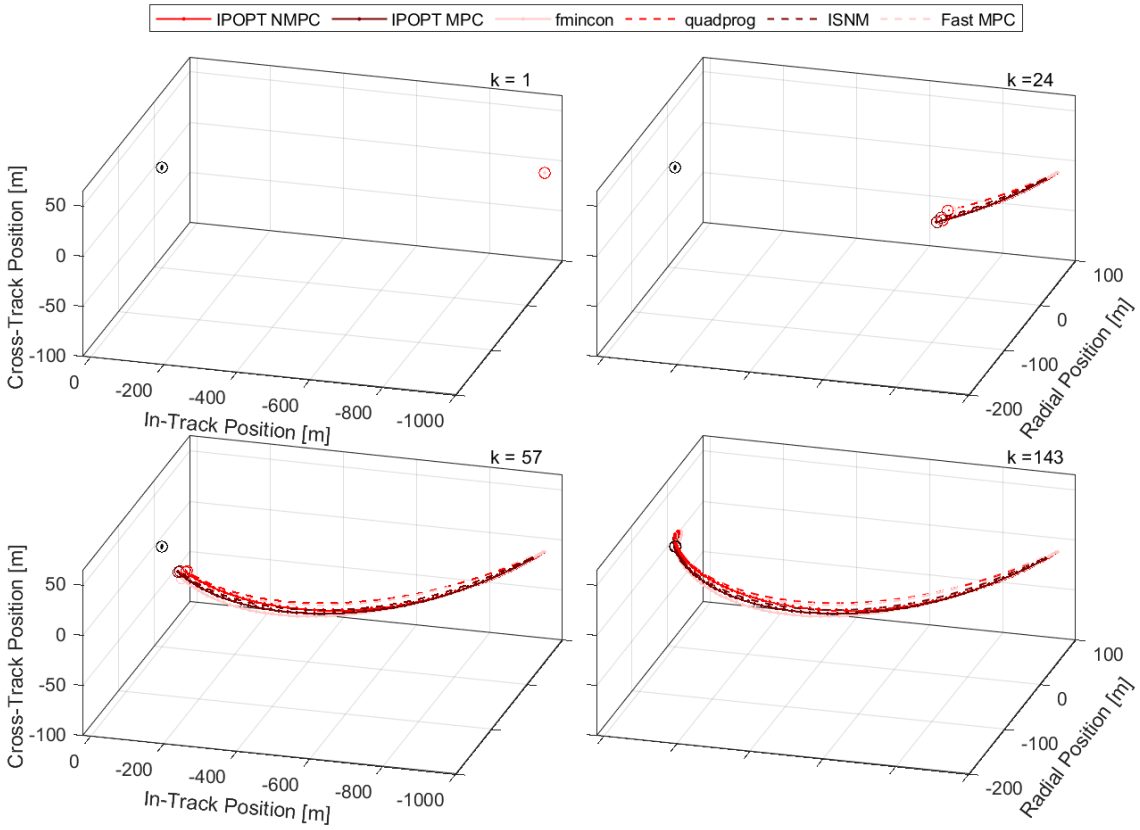


Figure 6.7: Snapshots of the chaser trajectory from each solver for Test Case A without obstacles.

The final approach of the chaser spacecraft to the target is magnified in Fig. 6.9 for the scenario without (left) and with (right) obstacles. The 2.3 m holding distance is represented by the black sphere that encompasses the target spacecraft at the origin. Similar to previous figures, the trajectory of each solver is depicted with a red line and is additionally projected onto each plane, shown in gray. From Fig. 6.9 it is clear that the trajectory of each solver differs upon approaching the target, but their solutions converge to approximately the same point on the holding radius sphere.

It is useful to compare the performance of the solvers in terms of their computation times to solve the problem and the control effort of the resulting solution. The computation time and control effort per iteration of the solution is plotted for each solver in Fig. 6.10. Figures 6.10a and 6.10b show the computation time and control effort for the scenario without obstacles, and Figs. 6.10c and 6.10d show the same plots for the scenario including obstacles.

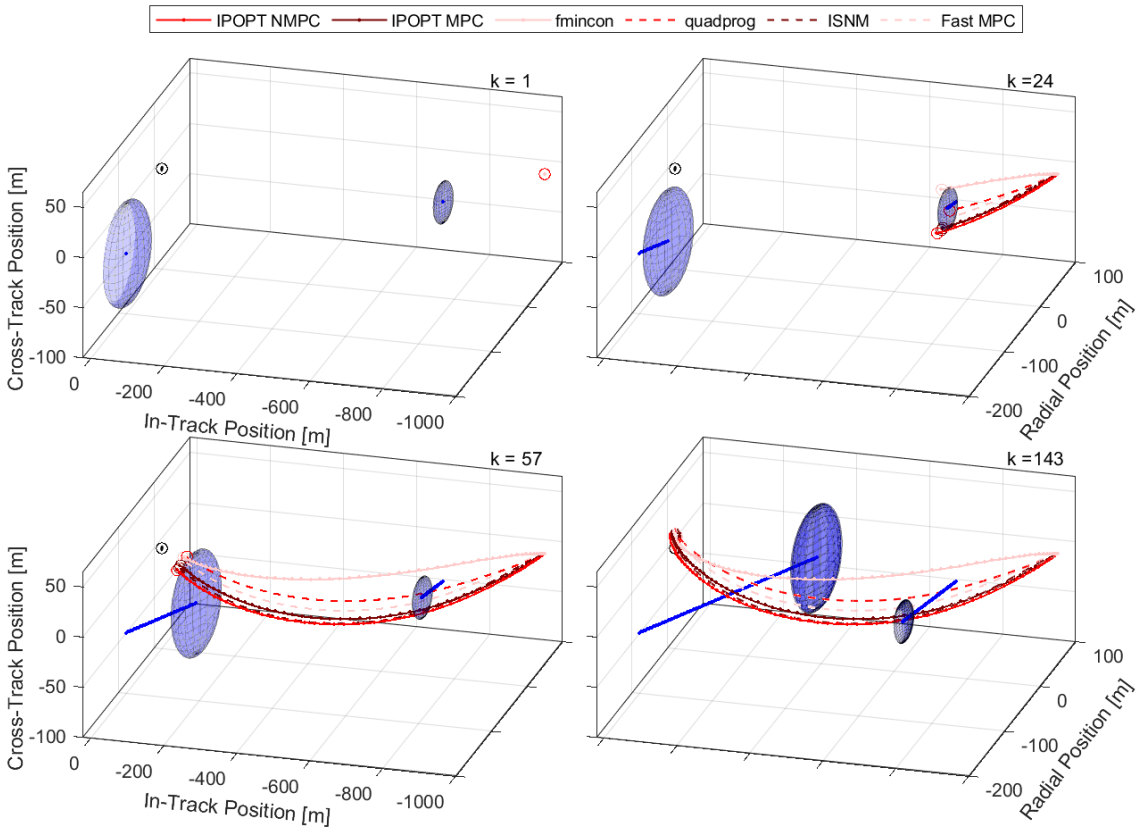


Figure 6.8: Snapshots of the chaser trajectory from each solver for Test Case A including obstacles.

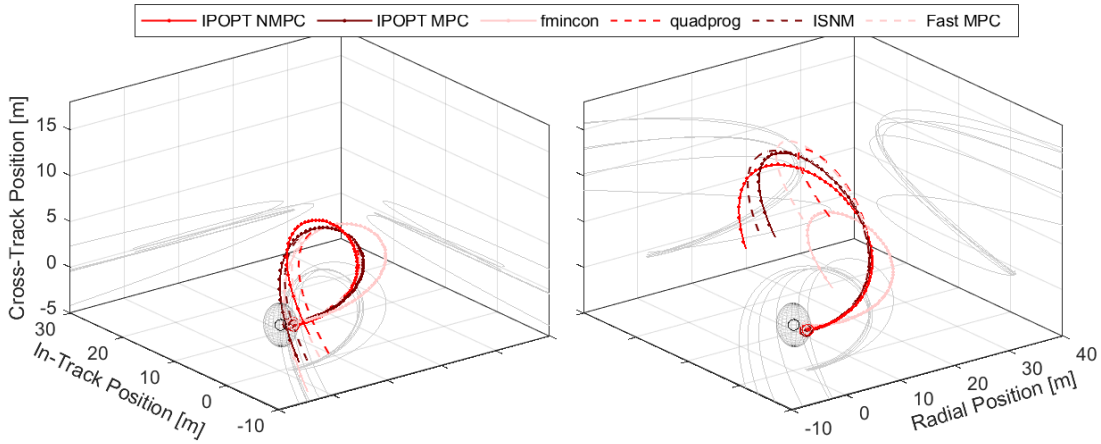
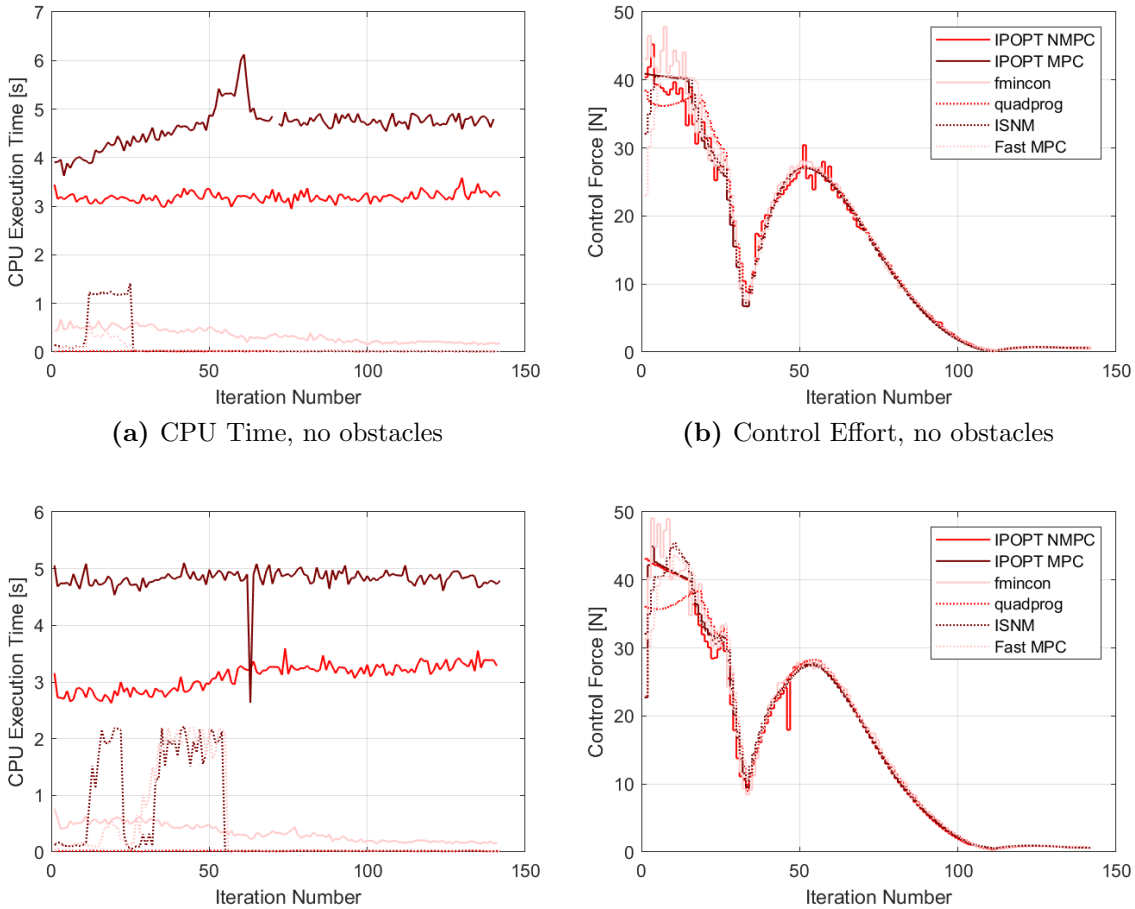


Figure 6.9: Final trajectories around the target for Test Case A without (left) and with (right) obstacles.



(a) CPU Time, no obstacles

(b) Control Effort, no obstacles

(c) CPU Time, with obstacles

(d) Control Effort, with obstacles

Figure 6.10: The CPU time of each solver and control effort of the resulting solution at each iteration for Test Case A with and without obstacles.

It is apparent from Figs. 6.10a and 6.10c that the linear `IPOPT MPC` solver has the slowest performance of all the algorithms, followed next by the nonlinear `IPOPT NMPC` solver. This result is surprising since it was expected that `IPOPT` would have a faster computational speed solving the linear problem as opposed to the nonlinear problem. It is possible that `IPOPT` is specifically tuned to solve nonlinear problems, however, user error cannot be ruled out as the cause of this discrepancy. The MATLAB wrapper for `IPOPT` provides many options for input arguments, and it is possible that the current implementation of the solver is not the most optimal.

Further analysis of Figs. 6.10a and 6.10c show that the `fmincon` solver is considerably faster than `IPOPT`, however, much slower than `quadprog` and the custom solvers.

It is also clear from Figs. 6.10a and 6.10c that the computation time for the custom solvers was not consistent over the simulation. In particular, the `ISNM` solver has a significantly larger computation time than the `Fast MPC` solver around $k = 13$ to 22, but a comparable computation time from $k = 35$ to 55. It can be noted that these large increases in Fig. 6.10c may be attributed to avoiding the moving obstacles, which occurs around these sampling instants; however, the large increase in computation time in Fig. 6.10a would not be attributed to obstacles. The cause could be attributed to the planning of maneuvers that have a significant change in the control effort as seen around $k = 35$ and $k = 50$ in Fig. 6.10b.

Figures 6.10b and 6.10d show that there is no significant change in the control effort between the nominal case without obstacles and the case with obstacles. This result is positive in nature since it means that the solvers were able to plan far enough in advance of encountering the obstacles such that a large acceleration was never necessary.

The control effort calculated by the `IPOPT NMPC` solver is not as smooth as the other solvers, as seen in Fig. 6.10b around $k = 55$ and in Fig. 6.10d around $k = 45$. This oscillatory nature may be attributed to the low maximum iteration limit allowed for the solver causing the solver to stop before an optimal solution could ever be found. Even though the same maximum iteration limit was used for all solvers, it may be particularly detrimental to the nonlinear solver which requires more iterations to reach a solution. Overall, however, the solutions of all the solvers are very comparable.

The average computation times of each solver and corresponding total impulse of their solutions are summarized in Table 6.3. The results show that `quadprog` has the fastest average computation time ($\sim$0.018 s); although, it is worth noting that `ISNM` and `Fast MPC` are very comparable to `quadprog` outside of the periods of high computation time with averages of 0.019 s and 0.021 s, respectively. The total impulse of the trajectories for each solver are very comparable, around 32.2 kN s and 33.6 kN s for the scenarios without and with obstacles, respectively. It is notable that `fmincon` provided the solution with the largest total impulse in both scenarios for Test Case A. A second noteworthy observation is that the total impulse for the solution using `ISNM` was less than that of the `Fast MPC` solver. The difference in the solutions of these

Table 6.3: Summary of mean solver computation times and total maneuver impulse for each solver.

Case	Algorithm	Average CPU Time, s		Total Impulse, kN s	
		no obstacle	with obstacle	no obstacle	with obstacle
A	`IPOPT NMPC`	3.1913	3.0981	32.13	32.93
	`IPOPT MPC`	4.6567	4.8219	31.49	33.49
	`fmincon`	0.3388	0.3398	33.09	34.33
	`quadprog`	0.0166	0.0182	32.32	33.37
	`ISNM`	0.1478	0.4609	31.73	33.80
	`Fast MPC`	0.0565	0.4052	32.45	33.98
B	`IPOPT NMPC`	3.5330	3.4538	20.43	20.77
	`IPOPT MPC`	4.7097	4.6951	20.15	20.85
	`fmincon`	0.3030	0.3391	20.87	21.00
	`quadprog`	0.0196	0.0240	20.16	20.77
	`ISNM`	0.3071	0.8570	20.22	20.82
	`Fast MPC`	0.0499	0.6742	20.33	21.09

solvers can only be attributed to using direct inversion versus Cholesky factorization of the matrices that give the search directions.

6.3.2 Test Case B

Test Case B uses the same modelling parameters as Test Case A, found in Table 6.2, except for the sizes of the obstacles. The trajectory of the chaser for Test Case B without obstacles is shown in Fig. 6.11 for each of the solvers. The progress of the solution is depicted at iterations $k = 1$, 22, 52, and 130. The trajectories of the solutions seem to have bifurcated whereby the `IPOPT NMPC` and `fmincon` solvers found a similar solution, and the `IPOPT MPC`, `quadprog`, `ISNM`, and `Fast MPC` solvers found a separate similar solution.

The corresponding trajectory for Test Case B with obstacles is shown in Fig. 6.12. At $k = 23$ the chaser is shown going around the first large obstacle, and at $k = 56$ the chaser is shown going underneath the obstacles to reach the target. Due to the lack of variability in the trajectories as calculated by the solvers, it can be assumed that the solvers all agree on the same optimal solution for this scenario.

Indeed, by zooming in to the final approach of the chaser spacecraft to the target, given in the right panel of Fig. 6.13 for the scenario with obstacles, it is clear that

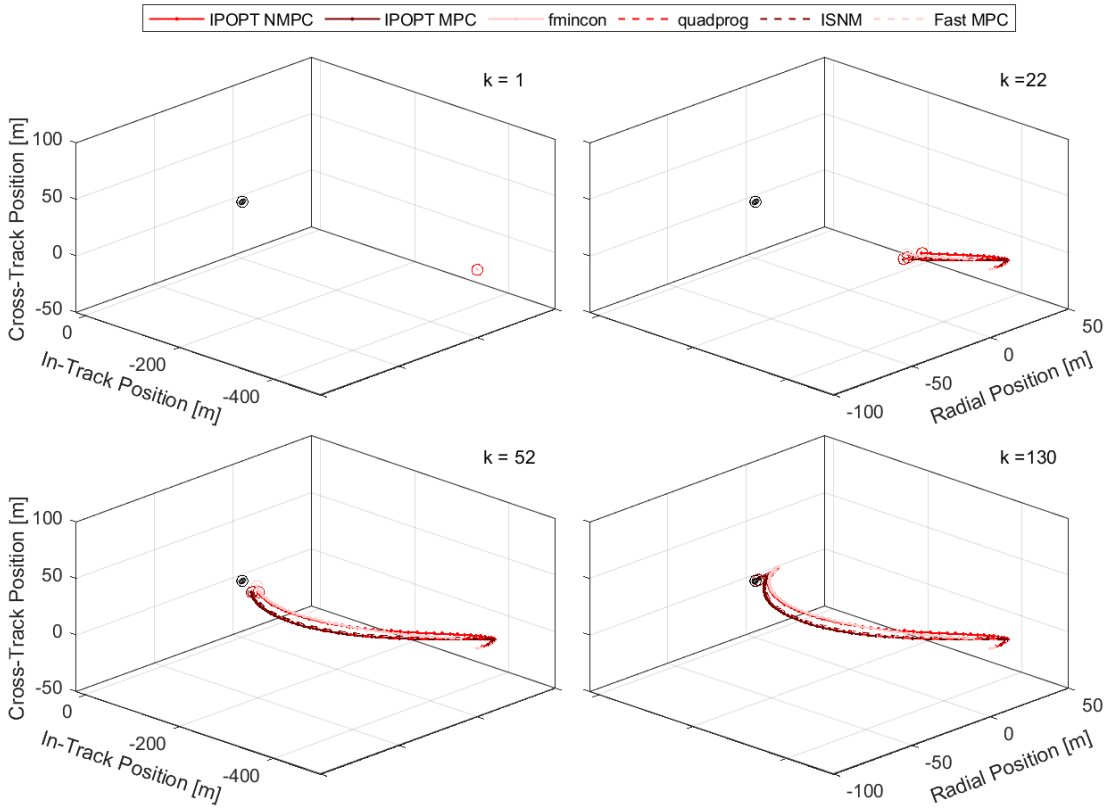


Figure 6.11: Snapshots of the chaser trajectory from each solver for Test Case B without obstacles.

the trajectories of the solvers agree. It can be noted, however, that the solution from `fmincon` has a minor divergence from the nominal trajectory taken by the other solvers.

By comparing the necessary control effort required by each solver over the horizon, shown in Fig. 6.14d, it is clear that there is very little variability between the solvers. For the scenario without the obstacles, Fig. 6.14b shows that the control effort required by the solutions of each solver is comparable; however, discrepancies for `IPOPT NMPC` and `fmincon` align with the fact that the solutions of these two solvers split from the trajectory of the other four solvers.

The behaviour of each solver in terms of their computational speed, shown in Figs. 6.14a and 6.14c, agrees with the observations made for Test Case A. Table 6.3 summarizes the average computation time and total impulse of the solution of each solver for Test Case B, showing similar trends to what was observed for Test Case

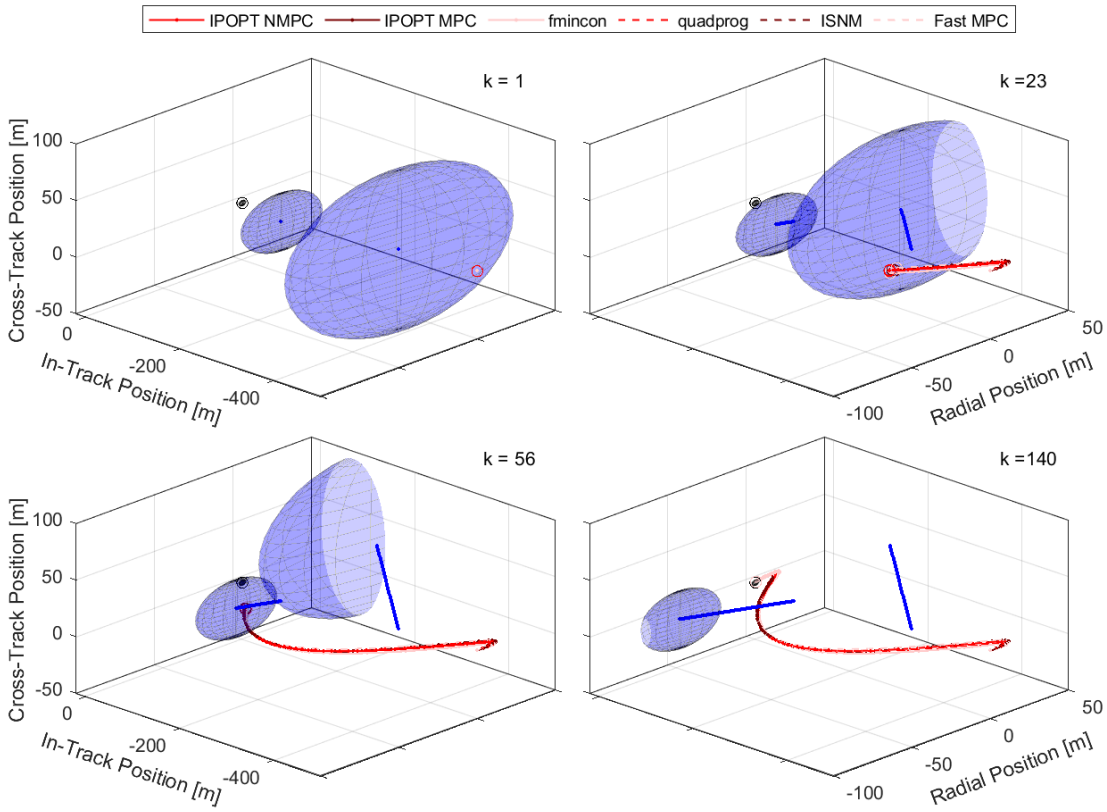


Figure 6.12: Snapshots of the chaser trajectory from each solver for Test Case B including obstacles.

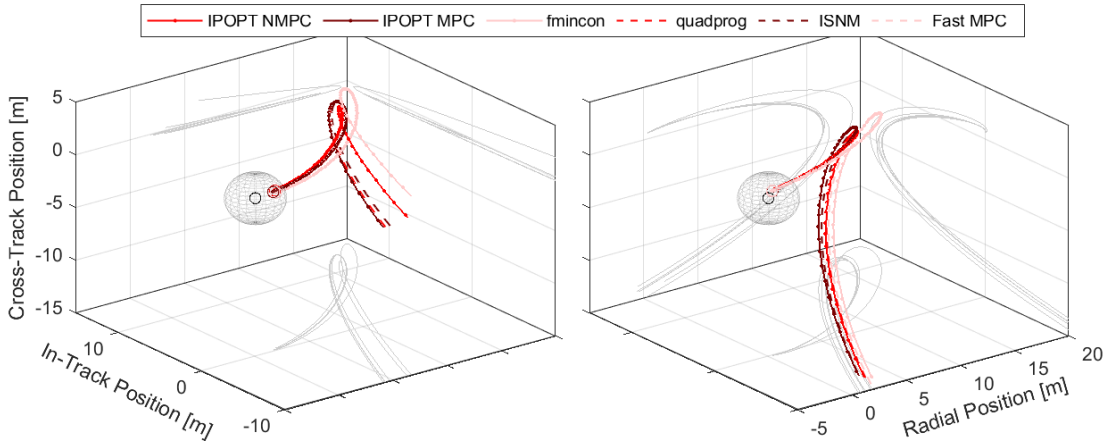
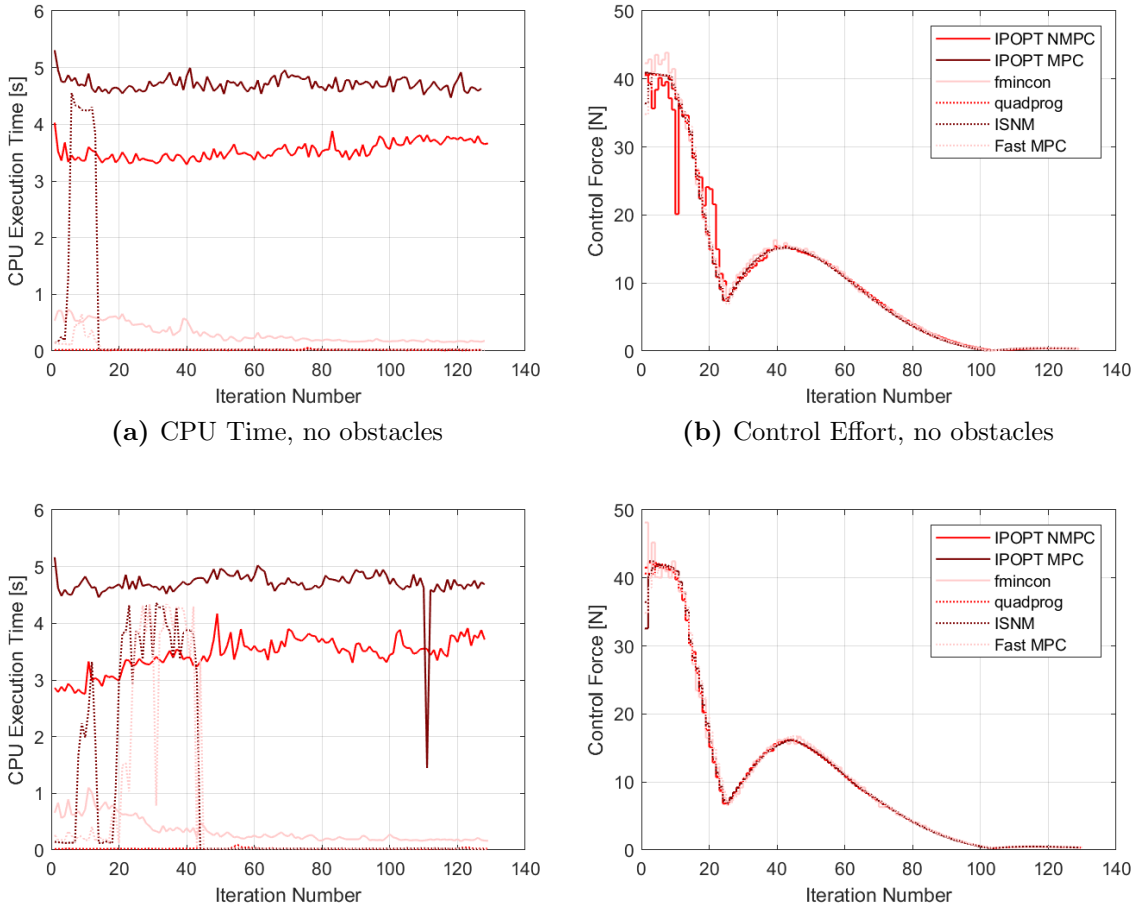


Figure 6.13: Final trajectories around the target for Test Case B without (left) and with (right) obstacles.



(a) CPU Time, no obstacles

(b) Control Effort, no obstacles

(c) CPU Time, with obstacles

(d) Control Effort, with obstacles

Figure 6.14: The CPU time of each solver and control effort of the resulting solution at each iteration for Test Case B with and without obstacles.

A. The `quadprog` solver has the fastest average computation time (∼0.022 s), but the `ISNM` and `Fast MPC` solvers have comparable performance when the periods of high computation times are excluded from the mean (∼0.025 s). The total impulse as required by each solver is very similar.

It can be noted that the maximum computational time for the slowest algorithm, the `IPOPT MPC` solver, was around ∼6 s, with an average computational time of ∼4.7 s. Compared to the discretization time of 15 s, it is still possible for even the slowest solver to provide meaningful solutions in real-time. The fastest solver, `quadprog`, with an average computational time of ∼0.02 s is two orders of magnitude faster than `IPOPT` and provides very comparable solutions in terms of total impulse.

Chapter 7

Experimental Facility

The remainder of the thesis focuses on validating the performance of the guidance algorithm in experiment. This chapter serves to introduce the Spacecraft Proximity Operations Testbed at Carleton University's Spacecraft Robotics and Control Laboratory. A major contribution was made to the facility for this work whereby a third platform, BLUE, was constructed.

7.1 Overview of Facility

The Spacecraft Proximity Operations Testbed (SPOT) at Carleton University's Spacecraft Robotics and Control Laboratory is used to test software related to guidance, navigation, and control on a physical system that mimics a planar, frictionless environment. SPOT consists of a $2.4\,\mathrm{m} \times 3.5\,\mathrm{m}$ granite surface with air-bearing spacecraft platforms ($0.3\,\mathrm{m} \times 0.3\,\mathrm{m} \times 0.3\,\mathrm{m}$) that glide across the plane due to negligible friction.

Prior to this thesis work, the laboratory consisted of two such spacecraft platforms, called RED and BLACK, built and extensively documented by Kirk Hovell in his Master's thesis [68]. A contribution of this work, with the assistance of another student, was to construct a third platform called BLUE as documented in Appendix C. A photograph of the laboratory is shown in Fig. 7.1.

The platforms each hold an air tank at 4500 psi that supplies air to two key features of the platform. To float the platform, three air-bearings expel air downwards at 60 psi. To propel the spacecraft, eight miniature air nozzles (two on each side) expel

Figure 7.1: The Spacecraft Proximity Operations Testbed as of June 2022.

air at 80 psi. The platforms are composed of three stacked, modular decks. The plumbing of the platforms sits on the lowest deck, called the bus deck. An image of the top and bottom of the bus deck for BLUE is provided in Appendix C, Figs. C.1 and C.2.

The electrical system of the platforms consists of a rechargeable 24 V battery that is used to power 5 V and 12 V regulators. The 5 V line goes to powering the onboard computers: the main Raspberry Pi 3 (RP3) Model B computer, and the emergency stop system Arduino Pro Mini computer. The 12 V line is used to power solenoid valves that actuate the air-bearings and thrusters. An updated electrical diagram for the platforms, shown in Fig. 7.2, has been made for this thesis that corrects a mistake in the previous version. Furthermore, the updated diagram includes the circuitry of the control panel and the interface of the general purpose input/output (GPIO) pins of the RP3 to the thruster breadboard, modelled in Fig. C.5. The thrusters are numbered according to Fig. C.6. The electronics of the platforms sit on the middle deck, called the avionics deck. An image of the avionics deck for BLUE is provided in Fig. C.3.

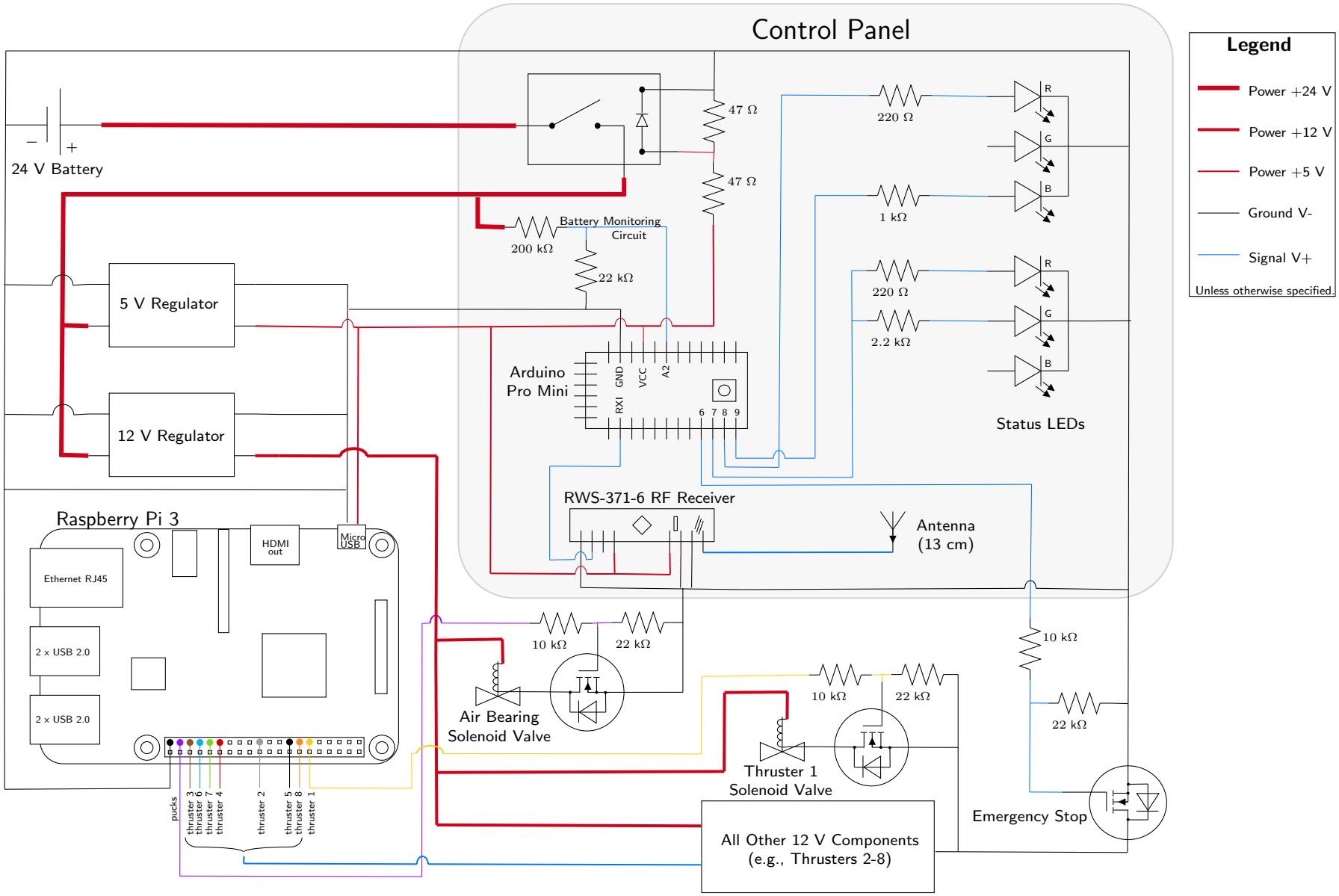


Figure 7.2: Electrical diagram of the platforms. GPIO numbering is specific to BLACK and BLUE.

Visible in Fig. 7.1 are six of the ten total cameras that make up the PhaseSpace motion capture system. This system is used to track the motion of the platforms in real-time on the granite surface using four light-emitting diodes (LEDs) on the top corners of each platform. The state information in the inertial reference frame is sent wirelessly to the RP3 computers on the platforms, which use the information in their real-time controllers.

7.2 Platform Controllers

The target and obstacle (BLACK and BLUE, respectively) use a simple Proportional Derivative (PD) controller to track a pre-computed trajectory. The chaser (RED) implements the guidance algorithm as developed in this thesis. To minimize the effect of signal noise and oscillatory behaviour of the PD controllers on the path-planning of RED, the translational and angular velocity data of BLACK and BLUE is fed through a moving average filter before being input into the model predictive controller. A more sophisticated navigation filter, such as an extended Kalman filter [69], is beyond the scope of this work. From the solution of the optimal control problem, the first set of control forces are passed to a control allocator which then determines the appropriate duty cycles for each of the eight thrusters to produce the desired control effort.

The controllers on the platforms are based upon a Simulink diagram that contains driver blocks to interface with the experimental hardware. The diagram is converted into C++ using MATLAB's Embedded Coder, compiled, and then executed on the RP3 computers. At the end of an experiment, the RP3 computers send their data to a ground station computer.

7.3 Sources of Error

Inconsistency of simulations and experiments using this facility can come from a number of discrepancies between the two environments: friction between the air bearings and the granite slab, table slope, air resistance, signal delays, offsets of the center of mass measurements, thrusters mounted off-axis, and plume impingement.

Chapter 8

Two-Dimensional Simulations

This chapter formulates the optimal control problem for a two-dimensional rendezvous scenario that represents the SPOT laboratory. Both the translational and rotational motion of spacecraft are considered, necessitating the formulation of orientation-specific inequality constraints. Simulation results for three different test cases are presented.

8.1 Problem Formulation

This scenario models the *final approach* and *mating* phases of the rendezvous problem (see §3.1.1), whereby the chaser spacecraft must approach the target spacecraft along an entry corridor and gently make mechanical contact.

8.1.1 Cost Function

Similar to the simulations in Chapter 6, the objective of the chaser spacecraft is to close the distance between itself and its desired position whilst minimizing the associated control effort, given by the equivalent compact cost function described in §4.5,

$$J(\mathbf{z}) = \mathbf{z}^T\mathbf{H}\mathbf{z} - 2\mathbf{z}_t^T\mathbf{H}\mathbf{z}, \tag{8.1}$$

where the desired position of the chaser over the horizon is defined by the vector

$$\mathbf{z}_t \triangleq \begin{bmatrix} 0 & \mathbf{x}_t^{(k+1)} & 0 & \dots & \mathbf{x}_t^{(k+N-1)} & 0 & \mathbf{x}_t^{(k+N)} \end{bmatrix}^T. \tag{8.2}$$

For this RVD scenario, the desired position $\mathbf{x}_t$ of the chaser is directly in front of the docking port of the target spacecraft, as shown in Fig. 8.1, but at a certain holding radius r_{hold}. As the chaser's state meets certain conditions, elaborated further in §8.1.3, the holding radius decreases until the chaser is docked with the target.

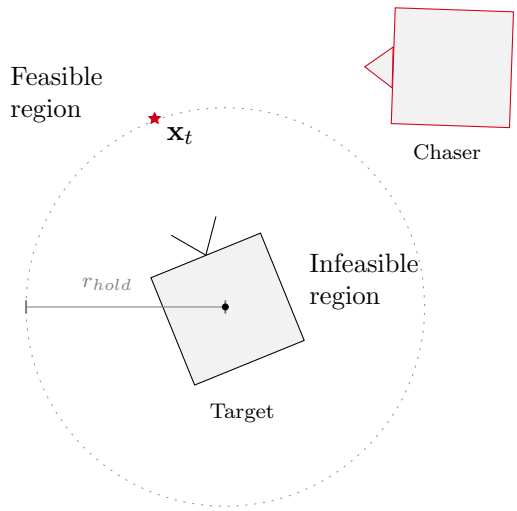


Figure 8.1: Desired position on the dynamic holding radius.

8.1.2 Equality Constraints

The dynamics of this problem consist of two degrees of freedom (2DOF) in translational motion and 1DOF in rotational motion.

From Hill's equations in Eq. (2.15), using the assumption that $n \to 0$, which is the case for large orbital radii and also applies over short rendezvous periods, the double-integrator planar dynamics are obtained:

$$\begin{cases} \ddot{x} &= \frac{u_x}{m} \\ \ddot{y} &= \frac{u_y}{m} \end{cases} \tag{8.3}$$

The expressions in Eq. (8.3) are simply a direct formulation of Newton's second

law. For the problem at hand, an additional DOF is added to consider the double-integrator rotational motion about the z-axis:

$$\begin{cases} \ddot{x} &= \frac{u_x}{m} \\ \ddot{y} &= \frac{u_y}{m} \\ \ddot{\theta} &= \frac{\tau}{I_z} \end{cases} \tag{8.4}$$

Here, m denotes the mass of the chaser platform, I_z is the moment of inertia about the vertical axis, and u_x, u_y, and τ are the control forces and torque, respectively.

As done in §6.2.2, the equations of motion in Eq. (8.4) are discretized using a zero-order hold as given by Eq. (3.3) to be expressed in the form of the discrete-time state-space equation in Eq. (3.2).

8.1.3 Inequality Constraints

This rendezvous scenario considers limitations to the state and control variables in the form of maximum thrust limits, obstacle avoidance constraints, a dynamic holding radius, and an entry cone, as will be mathematically formulated in this section.

Thruster Limitations

The thrusters onboard each platform have a maximum thrust force u_{max} that can be supplied. The torque generated by the thrusters is similarly limited by the maximum thrust force. The maximum torque τ_{max} is given by multiplying the maximum thrust force by the offset distance of the thruster from the center of mass (COM) of the platform.

$$\begin{cases} |u_x^{(k)}| \leq u_{max} \\ |u_y^{(k)}| \leq u_{max} \\ |\tau^{(k)}| \leq \tau_{max} \end{cases} \tag{8.5}$$

The thruster constraints are linear and so can be written in compact form by defining the matrix $\mathbf{U} \in \mathbb{R}^{mN \times n_z}$ from Eq. (6.6) and the vector $\mathbf{k} \in \mathbb{R}^{mN}$ as

$$\mathbf{k} = \begin{bmatrix} u_{max} & u_{max} & \tau_{max} & u_{max} & u_{max} & \tau_{max} & \dots & u_{max} & u_{max} & \tau_{max} \end{bmatrix}^T, \tag{8.6}$$

such that the thrust constraints are given by

$$\pm \mathbf{U}\mathbf{z} \leq \mathbf{k}. \tag{8.7}$$

Previous experiments at the SPOT facility have shown that the maximum thrust provided by the platform nozzles is about 0.22 N [15]. To ensure that this value is not exceeded, a reduced quantity is used in the controller algorithm: $u_{max} = 0.15$ N.

Obstacle Avoidance

The obstacle constraints for this problem are the same as those introduced in §6.2.3, but now for one less translational DOF.

Following the same arguments that produced Eq. (6.11), the nonlinear obstacle avoidance constraint is given by

$$1 \leq (\mathbf{r}^{(k)} - \mathbf{r}_{obj}^{(k)})^T \mathbf{S} (\mathbf{r}^{(k)} - \mathbf{r}_{obj}^{(k)}), \tag{8.8}$$

where $\mathbf{r}_{obj}^{(k)} \in \mathbb{R}^2$ is the obstacle position, $\mathbf{r}^{(k)} \triangleq \begin{bmatrix} x^{(k)} & y^{(k)} \end{bmatrix}^T$ is the spacecraft position at sampling instant k, and the shape matrix of the obstacle is given by

$$\mathbf{S} = \begin{bmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{bmatrix} \tag{8.9}$$

referring to the axes a and b of the ellipsoid in Fig. 6.5.

From Fig. 8.2, it is clear that the obstacle KOZ for a SPOT platform can be modelled by a circle with radius $a = b = r_{KOZ}$. The smallest value for the KOZ is equal to the distance from the COM of the obstacle to the COM of the chaser spacecraft when aligned diagonally, $r_{KOZ} = \sqrt{2}\,(l_c + l_{obs})$, where $l_c = l_{obs}$ is half the length of one side of the platform. An extra 10% margin is added for safety.

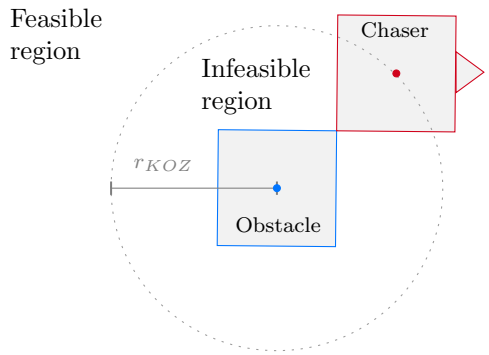


Figure 8.2: The minimum KOZ of the obstacle.

Direct linearization is used to linearize the obstacle constraint, with the expansion points $\mathbf{r}_0^{(k)}$ on the KOZ shown in Fig. 8.3 defined by the reference trajectory from the previous iteration. The linearized obstacle constraint as derived in §6.2.3 is reproduced here as

$$1 \leq 2(\mathbf{r}_0^{(k)} - \mathbf{r}_{obj}^{(k)})^T \mathbf{S}(\mathbf{r}^{(k)} - \mathbf{r}_{obj}^{(k)}) - (\mathbf{r}_0^{(k)} - \mathbf{r}_{obj}^{(k)})^T \mathbf{S}(\mathbf{r}_0^{(k)} - \mathbf{r}_{obj}^{(k)}), \tag{8.10}$$

and can be written in compact notation as per §6.2.3,

$$-2\mathbf{D}\mathbf{z} \leq -\mathbf{1}_N - \mathbf{D}(\bar{\mathbf{r}}_0 + \bar{\mathbf{r}}_{obj}). \tag{8.11}$$

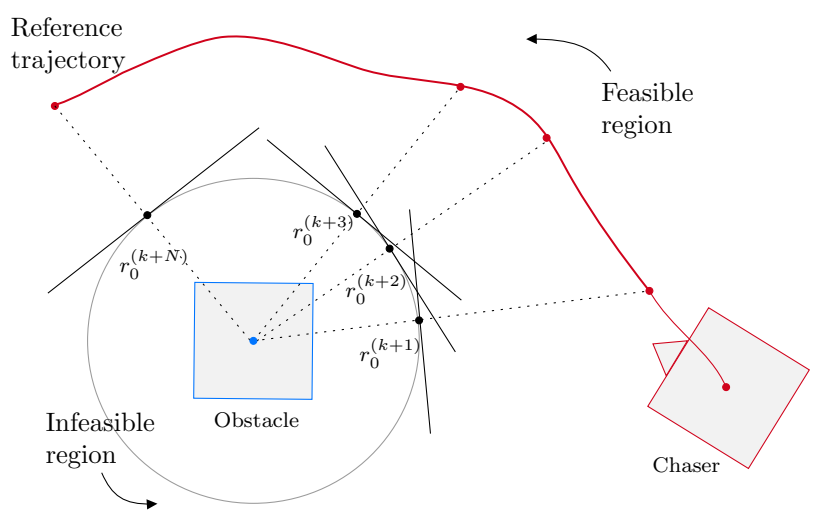


Figure 8.3: A visualization of the resulting hyperplanes from the direct linearization method.

Dynamic Target Collision Constraint

A dynamic holding radius as shown in Fig. 8.1 is used to ensure safe docking [26]. The holding radius is a set value at the beginning of the rendezvous $r_{hold,0}$ and decreases throughout the mission based on two conditions: First, the chaser spacecraft is within a certain distance η from the target holding position $\mathbf{x}_t^{(k)}$; Second, the chaser is within a certain attitude error ζ of the docking port. The condition on the attitude of the chaser ensures that there is precise attitude pointing, and the slow decrease in holding radius ensures that there is soft docking.

If the position and attitude error conditions are met, then the holding radius at the next sampling instant is reduced by a factor of $\gamma \in (0, 1)$,

$$r_{hold}^{(k+1)} = \gamma r_{hold}^{(k)}. \tag{8.12}$$

The constraint itself is written exactly as an obstacle avoidance constraint where the KOZ is of radius r_{hold} centered around the target position $\mathbf{r}_{tar}^{(k)} \triangleq \begin{bmatrix} x_{tar}^{(k)} & y_{tar}^{(k)} \end{bmatrix}$,

$$\left(r_{hold}^{(k)}\right)^2 \leq (\mathbf{r}^{(k)} - \mathbf{r}_{tar}^{(k)})^T (\mathbf{r}^{(k)} - \mathbf{r}_{tar}^{(k)}). \tag{8.13}$$

Similar to the obstacle avoidance constraint, the dynamic holding radius constraint can be convexified using direct linearization.

The holding radius reduces in size until it reaches the lower limit $r_{hold,min}$, which is defined as the distance between the COMs of the target and chaser platforms in the docking position, shown in Fig. 8.4.

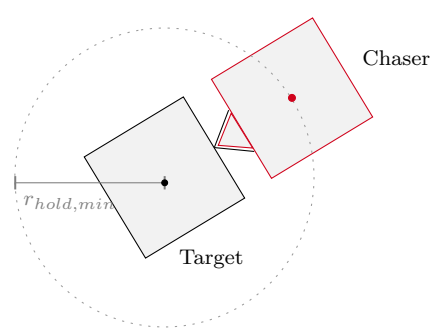


Figure 8.4: Minimum holding radius defined at the docking position.

The algorithm for reducing the holding radius is given in Algorithm 5.

Algorithm 5: Reducing the dynamic holding radius, r_{hold}

Data: min holding radius $r_{hold,min}$; tolerances η, ζ; decrement $\gamma \in (0, 1)$.

if $|(\mathbf{r}^{(k)} - \mathbf{x}^{(k)}_{t,[x,y]})^T(\mathbf{r}^{(k)} - \mathbf{x}^{(k)}_{t,[x,y]})| < \eta^2$ **then**

  `// Condition 1: error from holding position`

  if $|(\theta^{(k)} - \mathbf{x}^{(k)}_{t,[\theta]})| < \zeta$ **then**

    `// Condition 2: error from docking attitude`

    if $r^{(k)}_{hold} > r_{hold,min}$ **then**

      $r^{(k+1)}_{hold} = \gamma r^{(k)}_{hold}$

      if $r^{(k+1)}_{hold} < r_{hold,min}$ **then**

        $r^{(k+1)}_{hold} = r_{hold,min}$ `// set to minimum holding radius`

Entry Cone

Once the chaser spacecraft is within a certain distance r_{cone} from the target, an entry corridor that guides the chaser to the docking port is activated. The entry cone shown in Fig. 8.5 consists of two hyperplanes that intersect at the base of the target's docking port $\mathbf{r}_{dock}$ and create a triangular feasible region with a half angle θ_h.

The constraint for each hyperplane can be defined in terms of the hyperplane

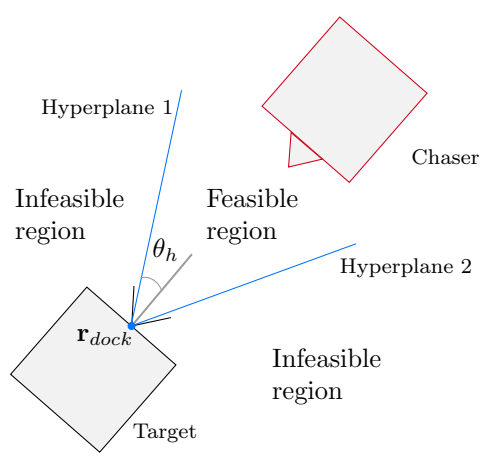


Figure 8.5: Visualization of the feasible and infeasible regions for an entry cone constraint.

normal vector $\hat{\mathbf{n}}^{(k)}$ and the position of the docking port $\mathbf{r}_{dock}^{(k)}$ [23, 26]:

$$\begin{cases} -\hat{\mathbf{n}}_{c1}^{(k)} \cdot \mathbf{r}^{(k)} & \leq -\hat{\mathbf{n}}_{c1}^{(k)} \cdot \mathbf{r}_{dock}^{(k)} \\ \hat{\mathbf{n}}_{c2}^{(k)} \cdot \mathbf{r}^{(k)} & \leq \hat{\mathbf{n}}_{c2}^{(k)} \cdot \mathbf{r}_{dock}^{(k)} \end{cases} \tag{8.14}$$

The normal vector of each hyperplane is defined by its slope $m^{(k)} = dy^{(k)}/dx^{(k)}$,

$$\hat{\mathbf{n}}^{(k)} = \begin{bmatrix} dy^{(k)} & -dx^{(k)} \end{bmatrix}, \tag{8.15}$$

where the slope is given in terms of the entry cone half angle θ_h and the attitude of the target at the current sampling instant $\theta_t^{(k)}$, which is apparent from Fig. 8.6,

$$m^{(k)} = \tan\left(\theta_t^{(k)} \pm \theta_h\right) = \frac{\sin\left(\theta_t^{(k)} \pm \theta_h\right)}{\cos\left(\theta_t^{(k)} \pm \theta_h\right)}. \tag{8.16}$$

Thus, using Eq. (8.15), the normals to the hyperplanes are given by:

$$\hat{\mathbf{n}}_{c1}^{(k)} = \begin{bmatrix} \sin\left(\theta_t^{(k)} + \theta_h\right) & -\cos\left(\theta_t^{(k)} + \theta_h\right) \end{bmatrix} \tag{8.17}$$

$$\hat{\mathbf{n}}_{c2}^{(k)} = \begin{bmatrix} \sin\left(\theta_t^{(k)} - \theta_h\right) & -\cos\left(\theta_t^{(k)} - \theta_h\right) \end{bmatrix} \tag{8.18}$$

The entry cone constraint is linear and can be written in compact notation. Defining $\mathbf{G}_{ci} \in \mathbb{R}^{N \times n_z}$ as a matrix that contains the normals of the hyperplane i over the

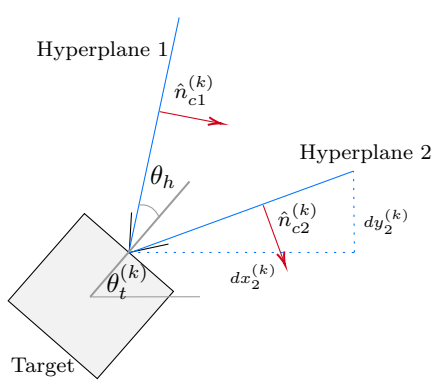


Figure 8.6: The entry cone hyperplane normal vectors and associated parameters.

horizon,

$$\mathbf{G}_{ci} = \begin{bmatrix} 0 & \hat{\mathbf{n}}_{ci}^{(k+1)} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{\mathbf{n}}_{ci}^{(k+2)} & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \hat{\mathbf{n}}_{ci}^{(k+N)} & 0 \end{bmatrix}, \tag{8.19}$$

and the column vector $\overline{\mathbf{r}}_{dock} \in \mathbb{R}^{n_z}$ containing the position of the apex of the entry cone over the horizon,

$$\overline{\mathbf{r}}_{dock} = \begin{bmatrix} 0 & \mathbf{r}_{dock}^{(k+1)} & 0 & \mathbf{r}_{dock}^{(k+2)} & 0 & \dots & 0 & \mathbf{r}_{dock}^{(k+N)} & 0 \end{bmatrix}^T, \tag{8.20}$$

the compact constraint for each hyperplane can be written as:

$$\begin{cases} -\mathbf{G}_{c1}\mathbf{z} & \leq -\mathbf{G}_{c1}\overline{\mathbf{r}}_{dock} \\ \mathbf{G}_{c2}\mathbf{z} & \leq \mathbf{G}_{c2}\overline{\mathbf{r}}_{dock} \end{cases} \tag{8.21}$$

8.1.4 Formulation

Using the cost function in Eq. (8.1), equality constraints, and inequality constraints as defined in the last section, the optimal control problem can be formulated. Similar to Chapter 6, both a nonlinear and linear optimal control problem are written. The nonlinear problem uses the nonlinear obstacle avoidance and dynamic holding radius constraints, Eq. (8.8) and Eq. (8.13), respectively, whereas the linear problem uses the linearized constraint given by Eq. (8.11). Both problems include the linear maximum thrust and entry cone constraints, Eq. (8.7) and Eq. (8.21), respectively.

With proper reordering of the rows of the inequality constraints as per §4.4, the inequality constraints of the linear problem can be written compactly as $\mathbf{Pz} \leq \mathbf{h}$.

Nonlinear Optimal Control Problem

$$
\begin{aligned}
\text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T\mathbf{H}\mathbf{z} - 2\mathbf{z}_t^T\mathbf{H}\mathbf{z} \\
\text{subject to} \quad & \mathbf{C}\mathbf{z} = \mathbf{b} \\
& \pm\mathbf{U}\mathbf{z} \leq \mathbf{k} \\
& 1 \leq (\mathbf{r}^{(k)} - \mathbf{r}_{obj,j}^{(k)})^T\mathbf{S}_j(\mathbf{r}^{(k)} - \mathbf{r}_{obj,j}^{(k)}), \quad k = 2, \ldots, N+1, \quad j = 1, 2, \ldots \\
& \left(r_{hold}^{(k)}\right)^2 \leq (\mathbf{r}^{(k)} - \mathbf{r}_{tar}^{(k)})^T(\mathbf{r}^{(k)} - \mathbf{r}_{tar}^{(k)}) \\
& -\mathbf{G}_{c1}\mathbf{z} \leq -\mathbf{G}_{c1}\overline{\mathbf{r}}_{dock} \\
& \mathbf{G}_{c2}\mathbf{z} \leq \mathbf{G}_{c2}\overline{\mathbf{r}}_{dock}
\end{aligned}
\tag{8.22}
$$

Linear Optimal Control Problem

$$
\begin{aligned}
\text{minimize} \quad & J(\mathbf{z}) = \mathbf{z}^T\mathbf{H}\mathbf{z} - 2\mathbf{z}_t^T\mathbf{H}\mathbf{z} \\
\text{subject to} \quad & \mathbf{C}\mathbf{z} = \mathbf{b} \\
& \mathbf{P}\mathbf{z} \leq \mathbf{h}
\end{aligned}
\tag{8.23}
$$

8.2 Numerical Simulations

Simulations were run using the same hardware as in Chapter 6, and similar to Chapter 6, the six different solvers are compared for scenarios using a chaser, target, and two obstacles. In this series of tests, the target spacecraft is both translating and rotating, the obstacles are translating, and the chaser must compute the optimal path to dock to the target whilst avoiding the obstacles. The scenarios are limited in scope to the size of SPOT's granite table.

Three test cases are presented with the spacecraft initial conditions given in Table 8.1. Note that the chaser spacecraft always begins at rest, while the target and obstacles have nonzero initial velocities. The rotation rates of the target represent slow-tumbling ($< 5°/\text{s}$), yet the chosen rates are faster than similar MPC experiments performed [26]. The constants used in the simulation are given in Table 8.2. Notably, the discretization time used is 2 s, the horizon length is 15, and the dynamic holding

radius decreases by 5% every iteration when the conditions described in §8.1.3 are satisfied.

Table 8.1: Initial states of the chaser spacecraft, target spacecraft, and two obstacles for Test Cases A, B, and C. States are given in the form $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ with translational units of m and m/s and rotational units of ° and °/s.

Parameter		Value		
		Test Case A	Test Case B	Test Case C
Chaser	Initial position	(0.8,2.2,0)	(3.0,1.3,180)	(3.2,2.2,90)
	Initial velocity	(0,0,0)	(0,0,0)	(0,0,0)
Target	Initial position	(1.25,0.5,180)	(1.1,1.3,270)	(0.2,0.2,0)
	Initial velocity	(0.02,0.01,-3.5)	(0.015,-0.008,4.0)	(0.02,0.01,-2.5)
Obstacle 1	Initial position	(0.5,1.5,0)	(1.0,2.0,0)	(1.3,1.0,0)
	Initial velocity	(0.02,0,0)	(0,-0.005,0)	(0.01,-0.01,0)
Obstacle 2	Initial position	(1.5,1.25,0)	(1.0,0.5,0)	(2.1,1.0,0)
	Initial velocity	(0.01,0.01,0)	(0.01,0,0)	(0.012,0.01,0)

Table 8.2: The optimization problem parameters for the simulations.

Parameter		Value
Problem Variables	Horizon N	15
	Discretization time T_d, s	2
	$\mathbf{Q}$	diag(10,10,10,300,300,500)
	$\mathbf{R}$	diag(10,10,10)
	Max iterations	10
	Initial guess	(-0.14,0,0)
Thruster Constraint	Maximum thrust u_{max}, N	0.15
Entry Cone Constraint	Entry cone half angle θ_h, °	20
	Entry cone range r_{cone}, m	0.75
Obstacle Constraint	Obstacle KOZ r_{KOZ}, m	$1.1 \times \sqrt{2} \times 0.3$
Dynamic Holding Radius	Initial holding radius $r_{hold,0}$, m	1.15
	Decrement factor γ	0.95
	Update r_{hold}: η tolerance, m	$\sqrt{0.5}$
	Update r_{hold}: ζ tolerance, °	10
Custom Solver	Fixed κ	0.05
	Backtracking algorithm: α	0.01
	Backtracking algorithm: β	0.95
	Residual tolerance ϵ	10^{-9}

8.2.1 Test Case A

The trajectory of the chaser for Test Case A without obstacles is shown in Fig. 8.7 for each of the solvers at four different snapshots. The chaser spacecraft is represented by the red platform and the target is represented by the black platform. The docking interfaces of the platforms are drawn to represent SPOT's RED and BLACK platforms and are notably off-axis. All solvers calculate a solution that successfully docks with the target. At $k = 18$ it is shown that the chaser spacecraft is within the entry cone activation radius r_{cone}. As such, it remains within the entry cone as the dynamic holding radius decreases until it finally docks at $k = 40$.

The corresponding trajectory for Test Case A with obstacles is shown in Fig. 8.8. Here, the two obstacles are additionally depicted as the blue platforms. Comparing the trajectories between Figs. 8.7 and 8.8, it is clear that the chaser spacecraft was actively maneuvering around the obstacles until it docked at $k = 40$. Furthermore, the presence of the obstacles altered the chaser trajectory such that the entry cone constraint was not activated until after $k = 18$ in Fig. 8.8.

The computation time and control effort per iteration of the solution is plotted for each solver in Fig. 8.9. Similar to the results found in Chapter 6, Figs. 8.9a and 8.9c, show that `IPOPT` is the slowest solver. In this case, the computation time for `IPOPT NMPC` and `IPOPT MPC` are very similar, which was not the case for the three-dimensional simulations that found `IPOPT MPC` to be significantly slower. From Figs. 8.9a and 8.9c it is also clear that, unlike in Chapter 6, the `ISNM` and `Fast MPC` solvers do not exhibit periods of high computation time.

The control effort for each solver, shown in Figs. 8.9b and 8.9d, is quite comparable. Similar to the results found in Chapter 6, the control force required by the `IPOPT NMPC` solution is more oscillatory as compared to the other solvers, especially seen in Fig. 8.9b from $k = 23$ to 40.

The average computation times and total impulse for the solutions calculated by each solver are summarized in Table 8.3. Focusing on the results for Test Case A, `quadprog` has the fastest average computation time ($\sim$0.008 s), followed closely by ISNM ($\sim$0.02 s) and Fast MPC ($\sim$0.05 s). The total impulse of the trajectories found by `quadprog` are the lowest, and the total impulse for the trajectories from `IPOPT`

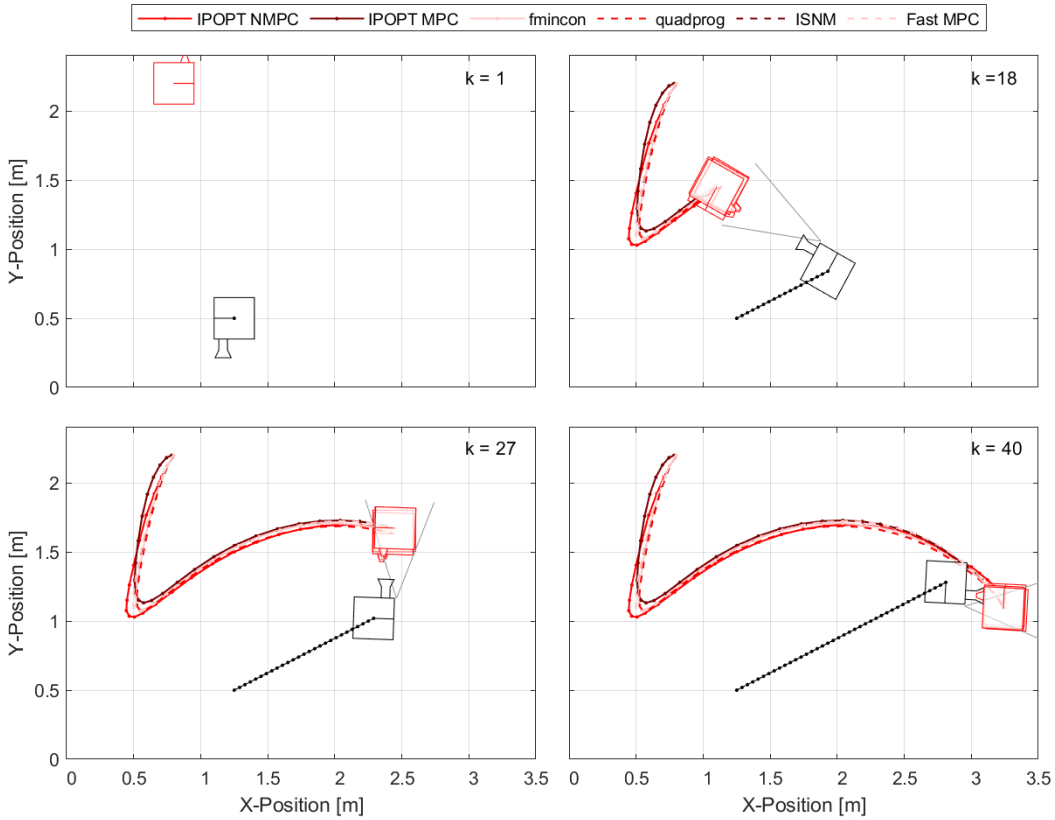


Figure 8.7: Snapshots of the chaser trajectory for Test Case A without obstacles.

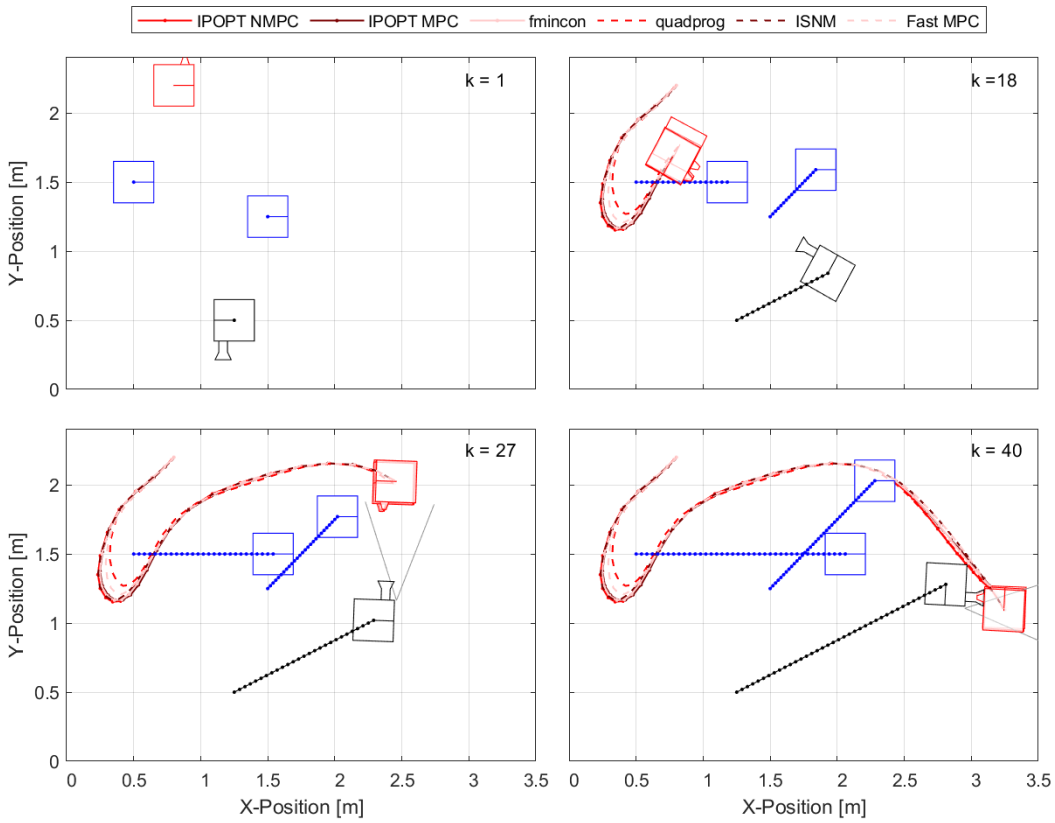
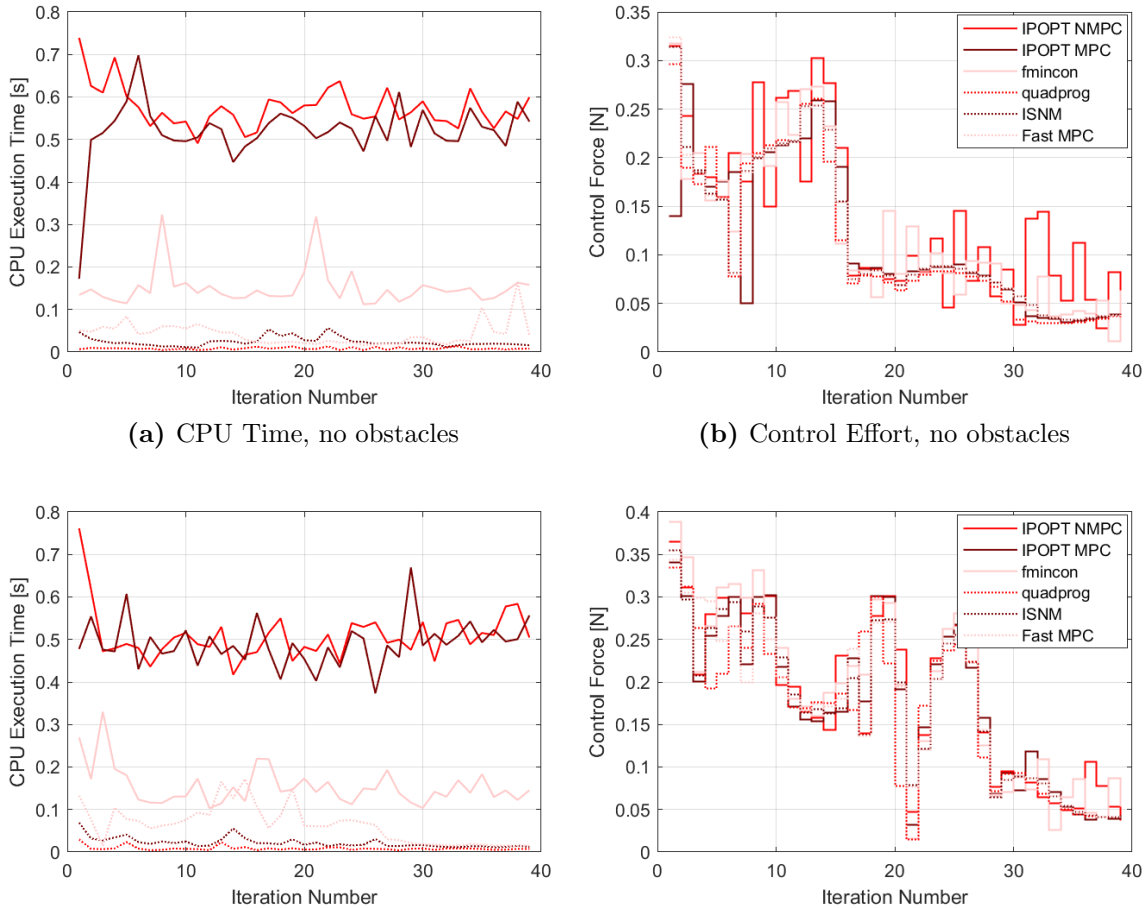


Figure 8.8: Snapshots of the chaser trajectory for Test Case A with obstacles.



(a) CPU Time, no obstacles

(b) Control Effort, no obstacles

(c) CPU Time, with obstacles

(d) Control Effort, with obstacles

Figure 8.9: The CPU time of each solver and control effort of the resulting solution at each iteration for Test Case A with and without obstacles.

Table 8.3: Summary of mean solver computation times and total maneuver impulse for each solver.

Case	Algorithm	Average CPU Time, s		Total Impulse, N s	
		no obstacle	with obstacle	no obstacle	with obstacle
A	`IPOPT NMPC`	0.5746	0.5088	10.70	14.15
	`IPOPT MPC`	0.5200	0.4912	8.90	13.57
	`fmincon`	0.1508	0.1544	9.61	14.36
	`quadprog`	0.0083	0.0086	8.68	12.81
	`ISNM`	0.0242	0.0221	9.07	13.39
	`Fast MPC`	0.0420	0.0638	9.06	13.37
B	`IPOPT NMPC`	0.6174	0.5407	9.51	12.00
	`IPOPT MPC`	0.6008	0.5619	8.34	12.07
	`fmincon`	0.1718	0.1683	9.07	13.29
	`quadprog`	0.0092	0.0082	8.12	11.80
	`ISNM`	0.0269	0.0258	8.43	11.84
	`Fast MPC`	0.0541	0.0546	8.43	11.84
C	`IPOPT NMPC`	0.6251	0.5638	9.16	10.72
	`IPOPT MPC`	0.5632	0.6290	7.98	10.71
	`fmincon`	0.1620	0.1648	8.60	11.62
	`quadprog`	0.0089	0.0092	7.72	9.92
	`ISNM`	0.0262	0.0245	7.61	10.59
	`Fast MPC`	0.0388	0.0498	7.59	10.45

`NMPC` are some of the highest. This is an unexpected development and may be related to the linear obstacle constraint smoothing out the trajectory due to its conservative nature.

8.2.2 Test Case B

Snapshots of Test Case B without obstacles is shown in Fig. 8.10, and the corresponding scenario with obstacles is shown in Fig. 8.11. As seen in Fig. 8.10 at $k = 16$, the trajectories calculated by each solver differ as the chaser accelerates to the desired holding position. The chaser docks to the target at $k = 35$ in the absence of obstacles.

In the case that obstacles are present, shown in Fig. 8.11, the chaser must take a longer path to avoid collisions and so only achieves docking at $k = 43$. Seen at the $k = 29$ snapshot, the chaser enters a challenging situation as it must avoid the obstacle and also stay within the entry cone. Each solver agrees on the course of

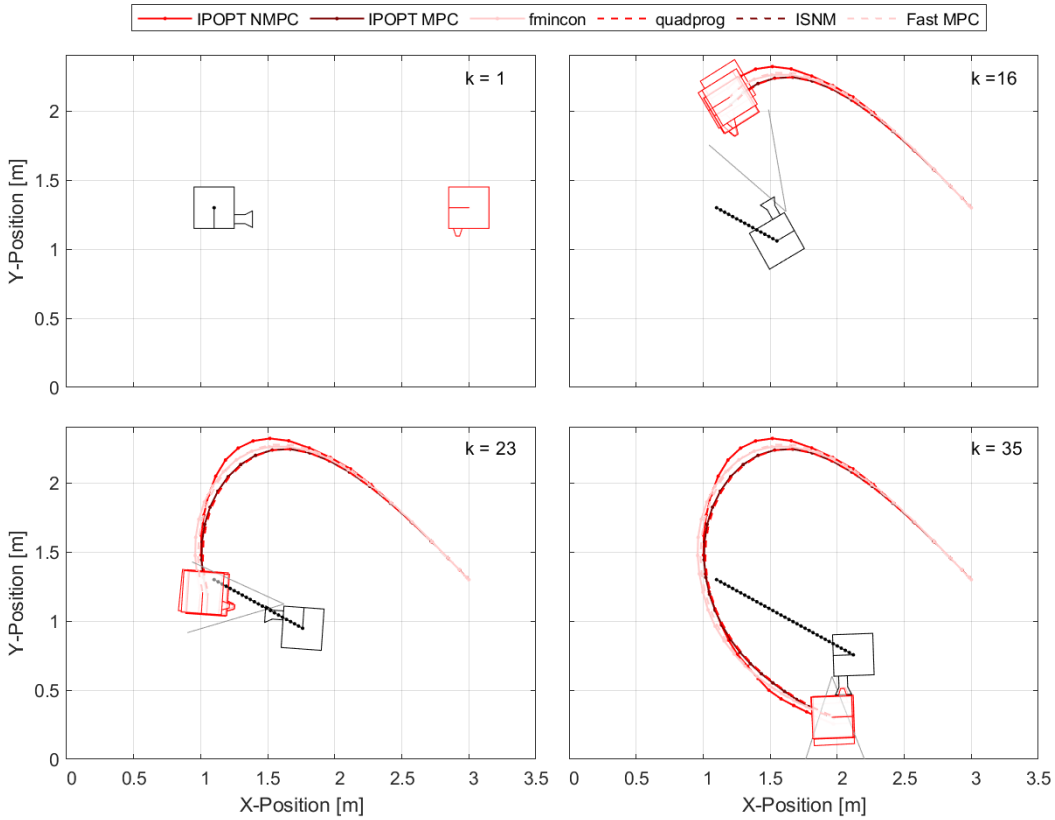


Figure 8.10: Snapshots of the chaser trajectory for Test Case B without obstacles.

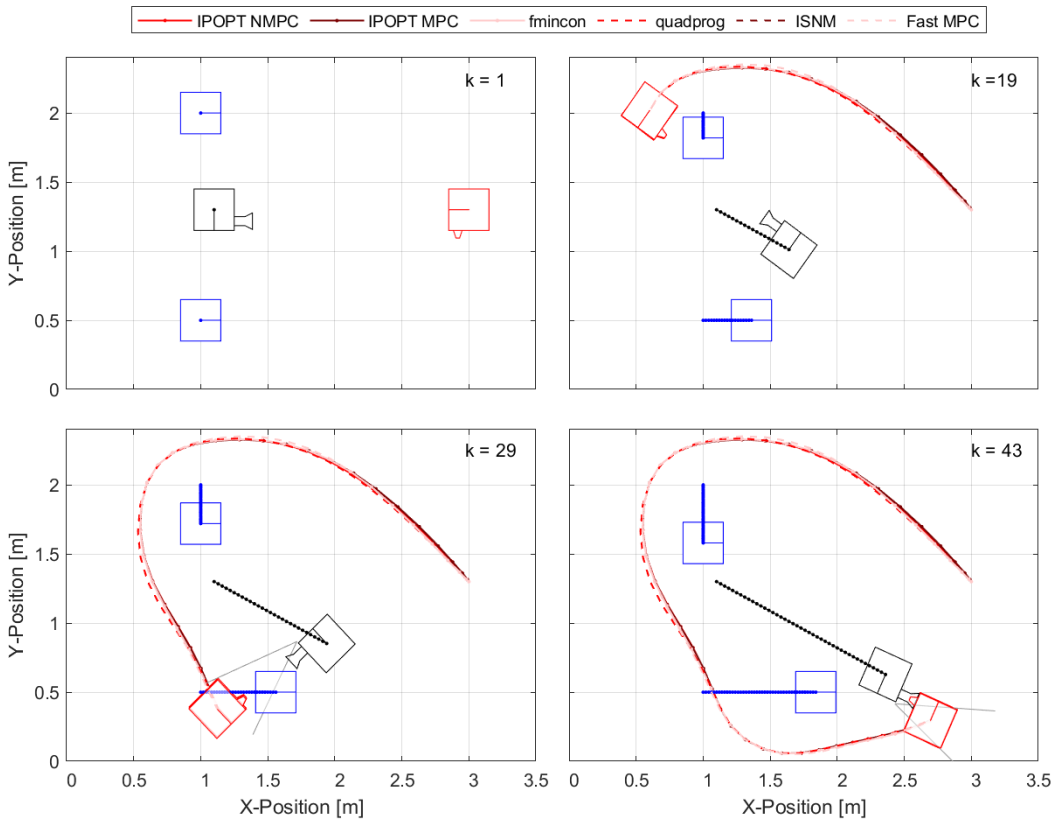
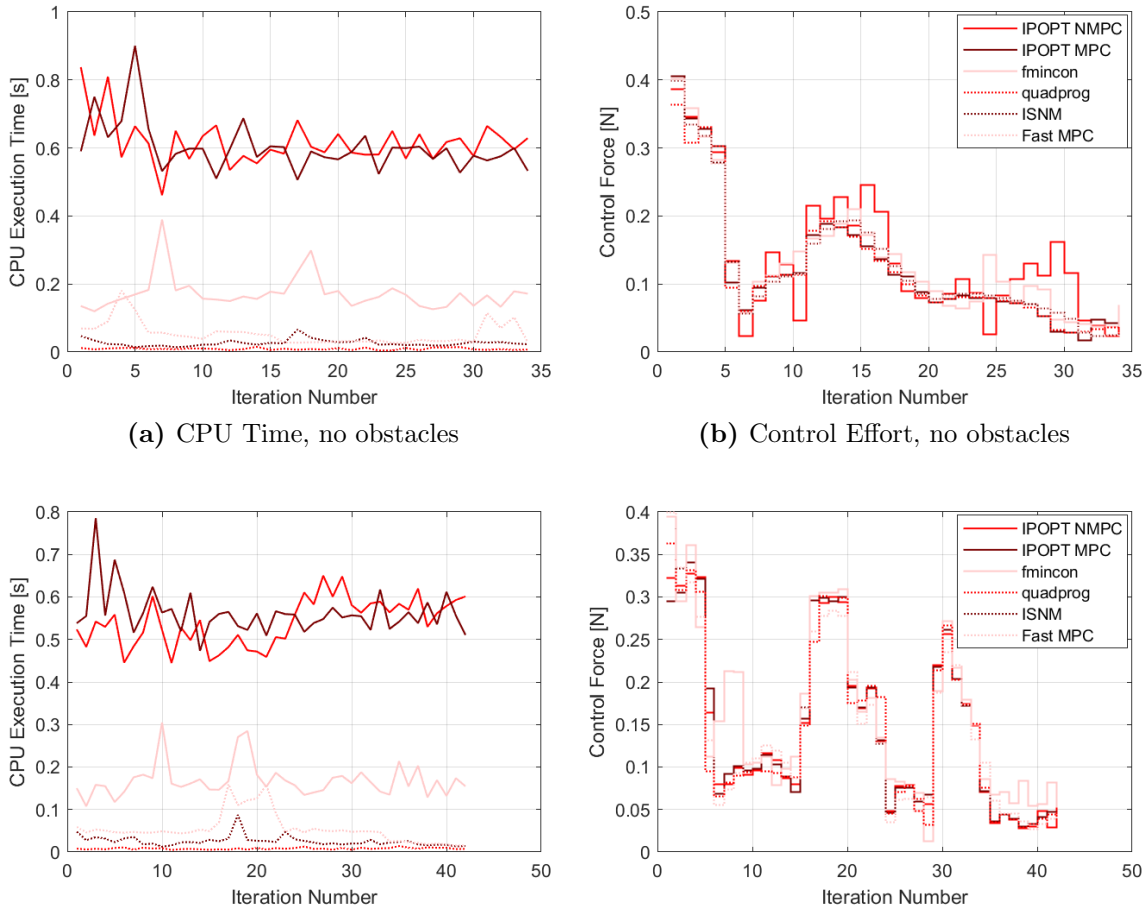


Figure 8.11: Snapshots of the chaser trajectory for Test Case B with obstacles.



(a) CPU Time, no obstacles

(b) Control Effort, no obstacles

(c) CPU Time, with obstacles

(d) Control Effort, with obstacles

Figure 8.12: The CPU time of each solver and control effort of the resulting solution at each iteration for Test Case B with and without obstacles.

action at this point in the mission, shown by looking at the resulting trajectories in the $k = 43$ snapshot, likely a result of a small feasible region.

The computation time of each solver and control effort per iteration required by their solutions is plotted in Fig. 8.12. Similar arguments for the performance of the solvers in terms of computation time and control effort as was made for Test Case A can be made here: `IPOPT` (solving both the linear and nonlinear problem) has the slowest computational speed followed by `fmincon`. The control forces, shown in Figs. 8.12b and 8.12d, required by the solutions of each solver are quite comparable, especially for the case with obstacles where the trajectories of each solver are nearly

indistinguishable.

The summarized results in Table 8.3 for Test Case B agree with statements made for Test Case A where `quadprog` is the fastest solver that also computes the solution with the lowest total impulse. The computation times of `ISNM` (∼0.03 s) and `Fast MPC` (∼0.05 s) are consistently better than `fmincon` (∼0.17 s) and `IPOPT` (∼0.6 s), but slower than `quadprog` (∼0.009 s). `ISNM` and `Fast MPC` also produce solutions that have a similar total impulse as `quadprog`: comparing `quadprog`'s 8.12 N s to the custom solver's 8.43 N s for the scenario without obstacles, and `quadprog`'s 11.80 N s to the custom solver's 11.84 N s for the scenario with obstacles. The total impulse for the other solvers in the scenario with obstacles is higher by comparison (≥ 12 N s).

8.2.3 Test Case C

Snapshots of Test Case C without obstacles is shown in Fig. 8.13, and the corresponding scenario with obstacles shown in Fig. 8.14. The solutions in Fig. 8.13 are nearly indistinguishable until the chaser enters the final approach of the target within the entry cone and, after $k = 18$, slowly rotates and advances towards the docking port. The trajectories of the solvers for the case with obstacles, from Fig. 8.14, similarly agree. It is clear, however, in the $k = 20$ snapshot, that the solution from `quadprog` 'cut the corner' that was taken by the other solvers.

The computation time of each solver and control effort per iteration required by their solutions is plotted in Fig. 8.15. Similar arguments for the performance of the solvers in terms of computation time and control effort as was made for Test Cases A and B can be made here.

For the case without obstacles, Table 8.3 shows that the `IPOPT NMPC` solver is slowest with an average of ∼0.63 s. The control effort for `IPOPT NMPC`, shown in Fig. 8.15b, is also the most oscillatory of the solutions. From Table 8.3 this coincides with the fact that `IPOPT NMPC` has the largest total impulse at 9.16 N s, followed next by `fmincon` at 8.60 N s. `Fast MPC` offers a solution with the lowest total impulse at 7.59 N s and the solution from `ISNM` close at 7.61 N s.

The time-to-dock when obstacles are present is different for the solutions of each solver: the solutions from `IPOPT` dock at $k = 39$, the solution from `quadprog` docks

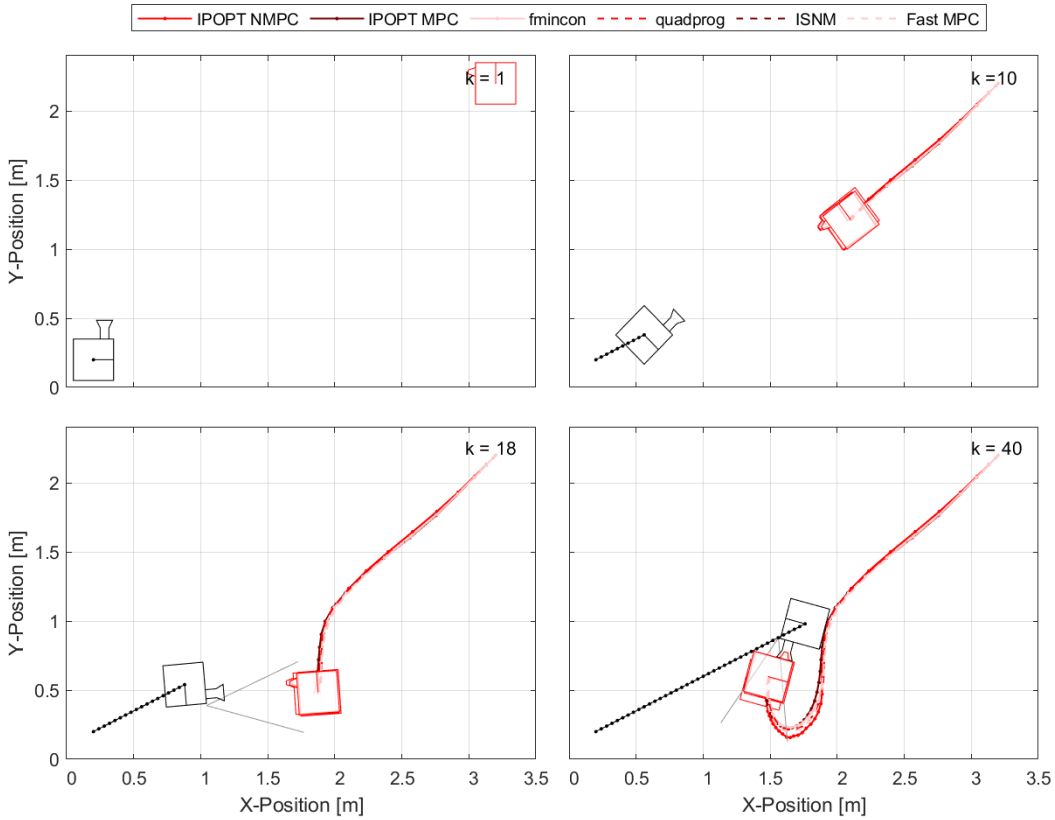


Figure 8.13: Snapshots of the chaser trajectory for Test Case C without obstacles.

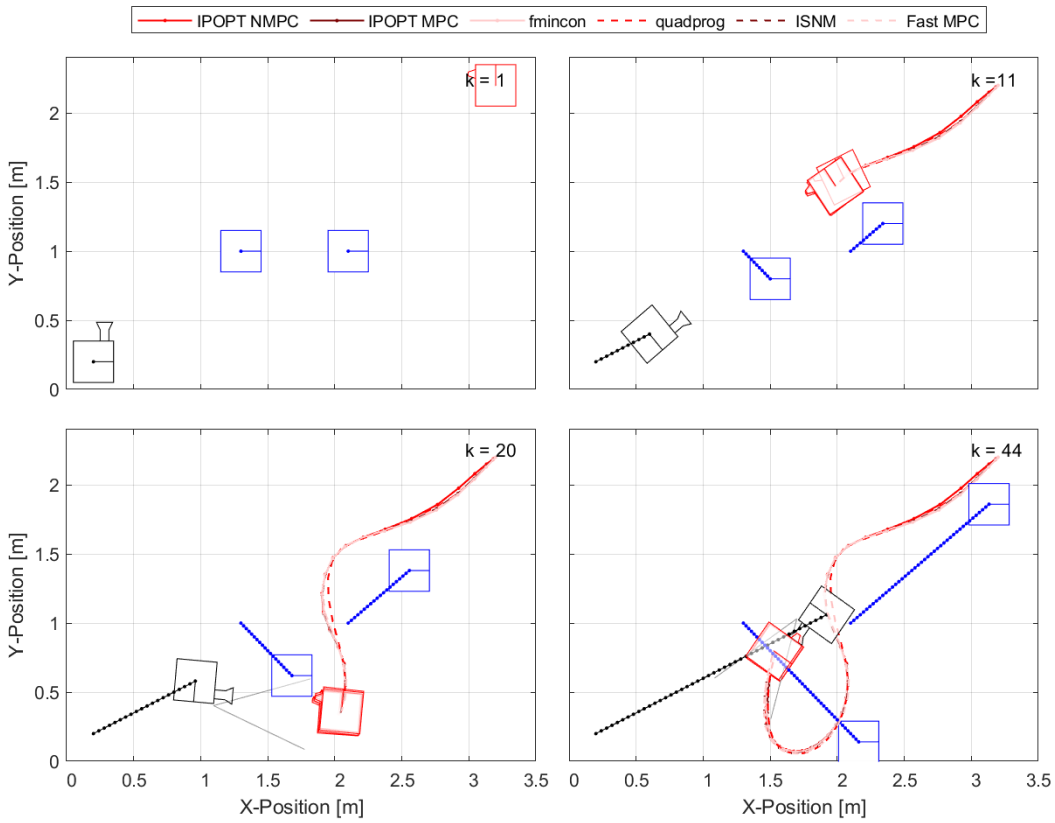
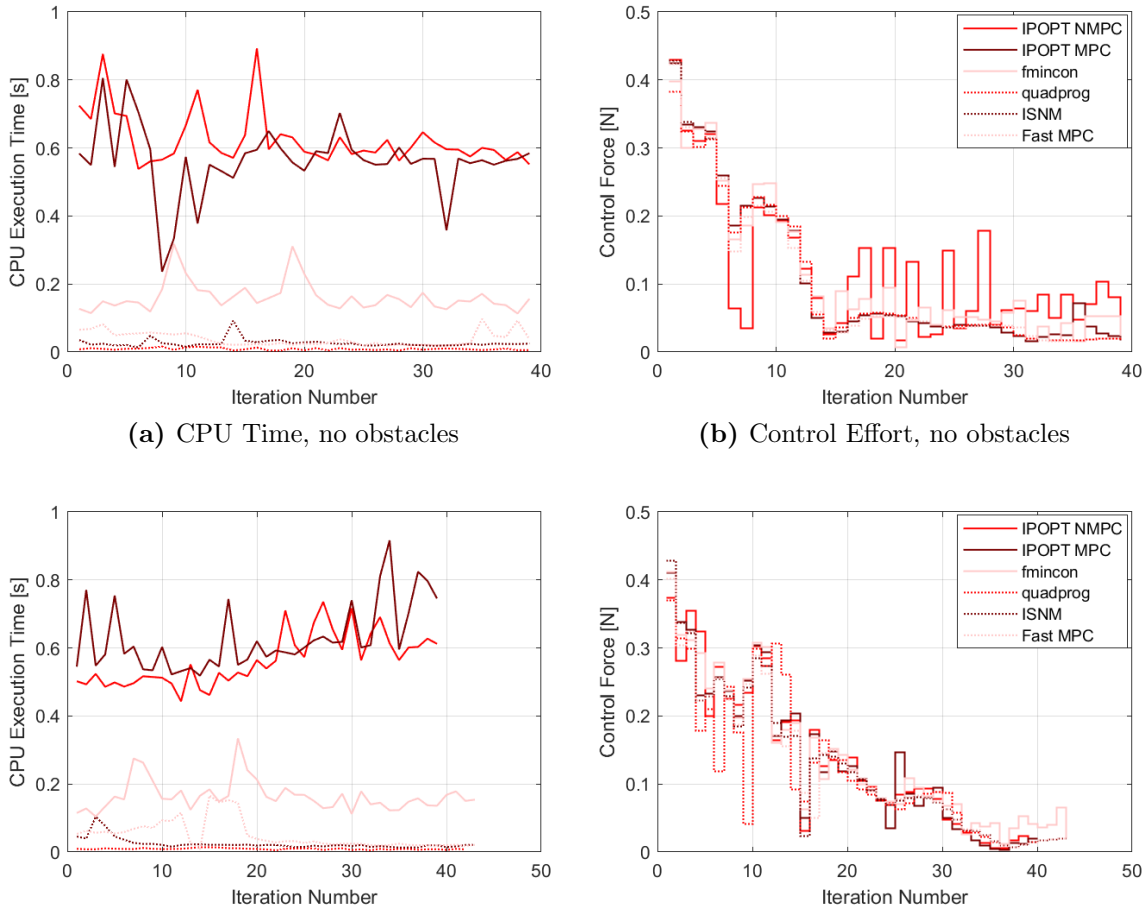


Figure 8.14: Snapshots of the chaser trajectory for Test Case C with obstacles.



(a) CPU Time, no obstacles

(b) Control Effort, no obstacles

(c) CPU Time, with obstacles

(d) Control Effort, with obstacles

Figure 8.15: The CPU time of each solver and control effort of the resulting solution at each iteration for Test Case C with and without obstacles.

at $k = 42$, and the solutions from `fmincon`, `ISNM`, and `Fast MPC` all dock at $k = 43$. The control effort among the solutions is similar as shown in Fig. 8.15d. From Table 8.3, the slowest solver is `IPOPT MPC` ($\sim$0.63 s), followed by `IPOPT NMPC` ($\sim$0.56 s). The fastest solver is `quadprog` ($\sim$0.009 s), which also achieves the solution with the lowest total impulse (9.92 N s).

Chapter 9

Experimental Validation

The penultimate chapter of this thesis validates the performance of the guidance algorithm in real-time on hardware and simultaneously tests the robustness of the algorithm to unmodelled disturbances and sensor delays. The SPOT facility, as introduced in Chapter 7, is used for this purpose.

9.1 Formulation

The optimal control problem as formulated in Chapter 8 is used in this section. The relevant constraints on the guidance problem are: planar dynamics, maximum thrust limits, obstacle avoidance, a dynamic holding radius, and when relevant, an entry cone. As opposed to Chapter 8, the problem in this section is limited to avoiding a single moving obstacle. There are two additional limitations to the experimental validation of this work.

9.1.1 Limitation 1: Embedded Coder

As mentioned in Chapter 7, the MATLAB/Simulink diagram is converted into C++ using MATLAB's Embedded Coder before being compiled and uploaded to the RP3s onboard the platforms. Due to limitations of the Embedded Coder, the `IPOPT` and `fmincon` solvers cannot be implemented on the facility hardware. As a result, the guidance algorithm may only be validated with the `quadprog`, `ISNM`, and `Fast MPC` solvers.

9.1.2 Limitation 2: Data Buffering with PhaseSpace

Preliminary experimental tests using `ISNM` and `Fast MPC` as solvers of the guidance algorithm uncovered problems with the current design of the facility architecture: The state information from PhaseSpace is communicated directly to RED, who then transfers the relevant information to BLACK and BLUE. Running the computationally expensive guidance algorithm on RED leads to buffering of the PhaseSpace data sent to BLACK and BLUE, ultimately causing problems with their PD controllers. As a result, the experimental validation of the guidance algorithm using the custom solvers can currently only be done with a static target and obstacle. Tests with `quadprog` show that this solver runs fast enough that the data buffering is not as significant.

The custom solvers could be implemented with a translating/rotating target and moving obstacle following an upgrade to the facility, as will be discussed further in Chapter 10.

9.1.3 Summary

The guidance algorithm using the `quadprog` solver can be experimentally validated using a translating and rotating target and a moving obstacle. The guidance algorithm using the `ISNM` and `Fast MPC` solvers can be experimentally validated using a static target and obstacle.

9.2 Set-Up

At the beginning of an experiment, the chaser, target, and obstacle platforms sit on the granite table as the PhaseSpace system acquires a strong lock with the LEDs. Then, the platforms float and move to their initial positions. Before the guidance algorithm on the chaser is implemented, the target and obstacle platforms are given a two second head start to accelerate to their target velocities. The chaser's guidance algorithm runs every two seconds and uses the state information of BLACK and BLUE smoothed over one second via a simple moving average filter.

9.3 Experiments using `quadprog`

This section compares the simulation and experimental results for the linear optimal control problem solved using `quadprog`. Three test cases are presented with different chaser, target, and obstacle initial conditions given in Table 9.1. The constants used in the algorithm are given in Table 9.2, and are the same as those used in Chapter 8.

Table 9.1: Initial states of the chaser spacecraft, target spacecraft, and obstacle for Test Cases A, B, and C. States are given in the form $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ with translational units of m and m/s and rotational units of ° and °/s.

Parameter		Value		
		Test Case A	Test Case B	Test Case C
Chaser	Initial position	(3.0,2.0,0)	(3.3,1.8,0)	(1.5,2.0,180)
	Initial velocity	(0,0,0)	(0,0,0)	(0,0,0)
Target	Initial position	(0.5,0.5,315)	(2.0,2.2,270)	(2.2,0.2,270)
	Initial velocity	(0.01,0.01,1)	(-0.01,-0.005,-1.25)	(-0.005,0.005,2.5)
Obstacle	Initial position	(1.5,1.25,0)	(2.5,1.5,0)	(2.5,1.2,0)
	Initial velocity	(0.01,0.01,0)	(-0.01,-0.01,0)	(-0.01,0,0)

Table 9.2: The optimization problem parameters for the simulations and experiments with `quadprog`.

Parameter		Value
Problem Variables	Horizon N	15
	Discretization time T_d, s	2
	$\mathbf{Q}$	diag(10,10,10,300,300,500)
	$\mathbf{R}$	diag(10,10,10)
	Max iterations	10
	Initial guess	(-0.14,0,0)
Thruster Constraint	Maximum thrust u_{max}, N	0.15
Entry Cone Constraint	Entry cone half angle θ_h, °	20
	Entry cone range r_{cone}, m	0.75
Obstacle Constraint	Obstacle KOZ r_{KOZ}, m	$1.1 \times \sqrt{2} \times 0.3$
Dynamic Holding Radius	Initial holding radius $r_{hold,0}$, m	1.15
	Decrement factor γ	0.95
	Update r_{hold}: η tolerance, m	$\sqrt{0.5}$
	Update r_{hold}: ζ tolerance, °	10

9.3.1 Test Case A

The simulated trajectories of the chaser for Test Case A for the scenario with and without the obstacle are shown in Fig. 9.1. The snapshots of the trajectory are shown at different stages within the maneuver: the initial condition, two intermediate stages, and the position when docking was achieved. The trajectory for the scenario without the obstacle present is given by the red dotted line. By comparison, the trajectory for the scenario including the obstacle is given by the solid red line. It is apparent through comparing these solutions that the chaser trajectory is modified by the presence of the obstacle. At 66% completion of the maneuver for both cases with and without the obstacle, the chaser spacecraft is within the entry cone at the holding position. For both cases, the chaser achieves docking at iteration $k = 37$.

The experimental results of Test Case A are shown in Fig. 9.2. In total, five trials

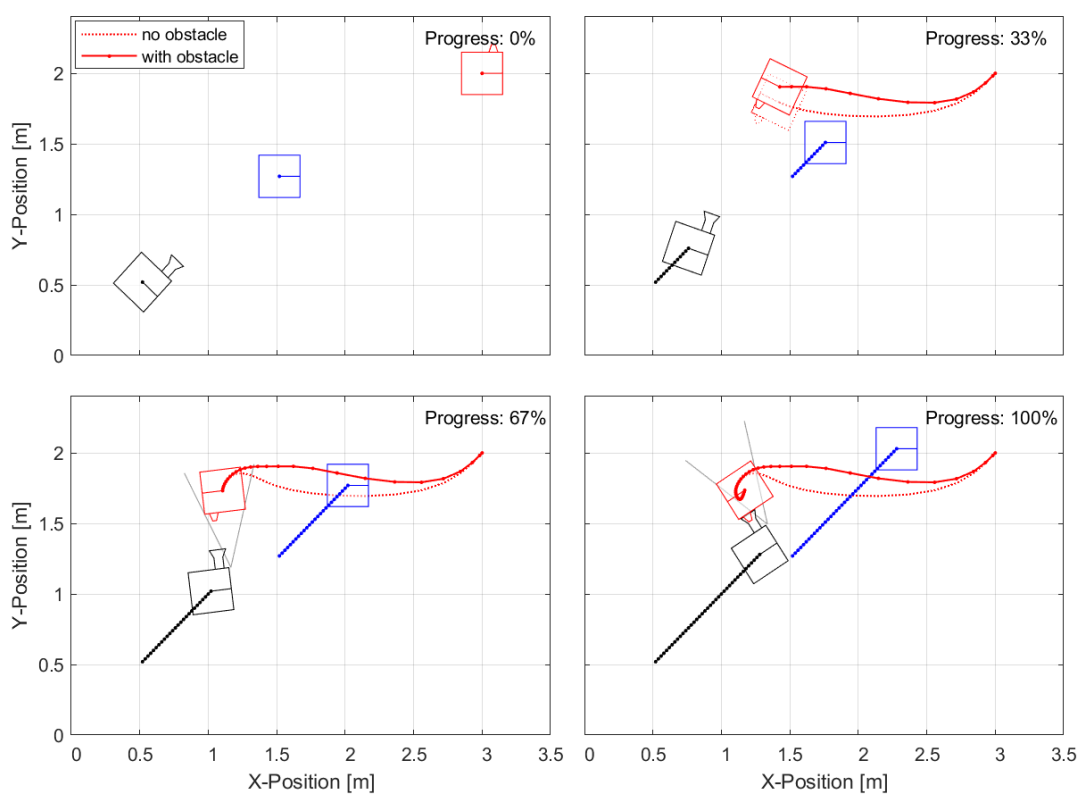


Figure 9.1: Snapshots of the chaser trajectory for Test Case A as simulated using `quadprog` with and without obstacles.

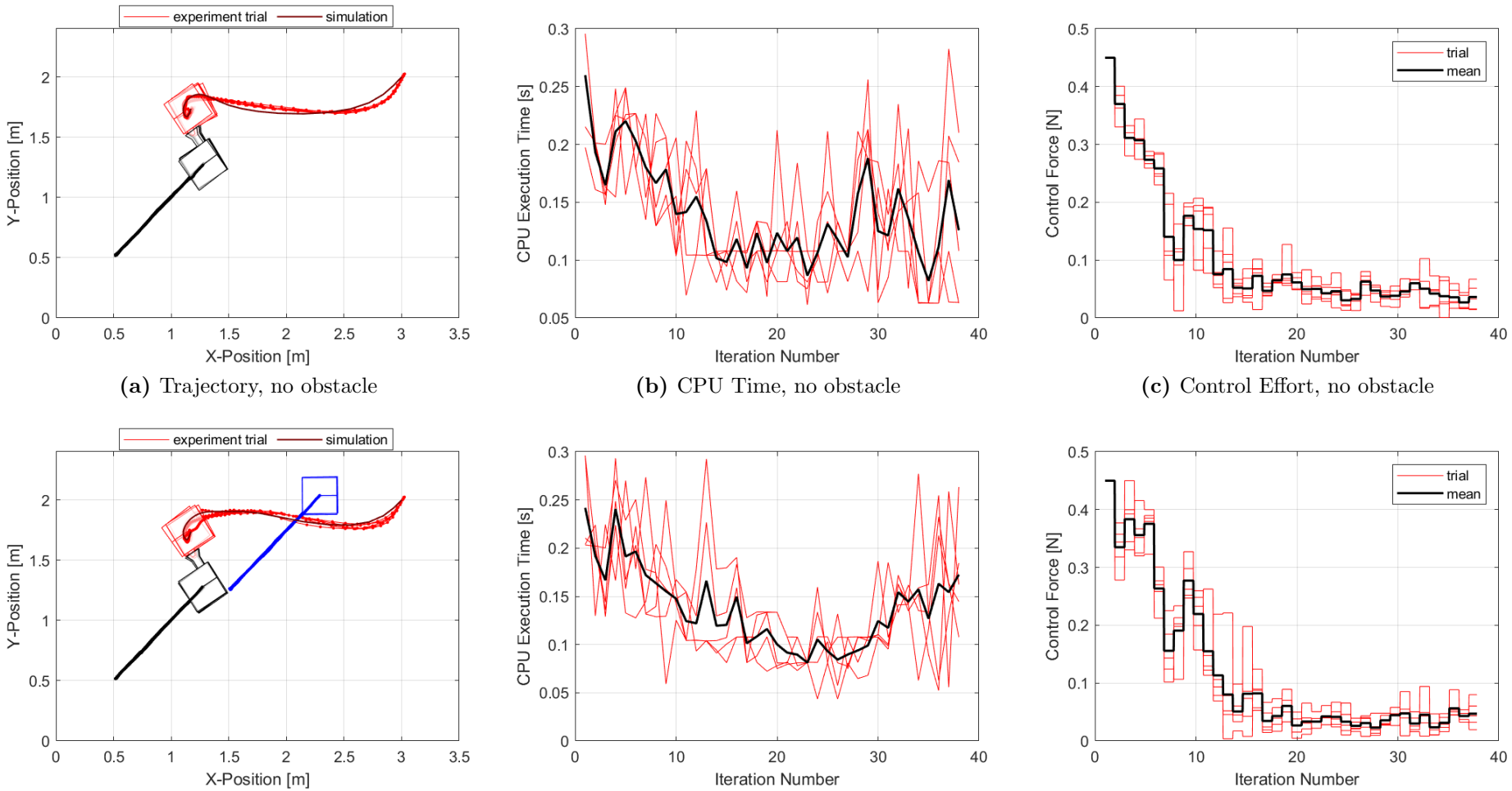
were performed for each scenario to verify reproducibility of the results. The top row of Fig. 9.2 corresponds to the scenario without the obstacle, and the bottom row corresponds to the scenario with the obstacle present. In Figs. 9.2a and 9.2d, the trajectories taken by the chaser in experiment are plotted with the bright red lines. The corresponding trajectory calculated in simulation is overlaid as the dark red line. The trajectories taken by the target and obstacle during the experiment for all five trials are plotted with the black and blue lines, respectively, although the individual trials for these platforms are indistinguishable. Comparisons of the experimental versus simulation trajectories shown in Figs. 9.2a and 9.2d are very similar validating the use of the guidance algorithm in real-time implementation.

The computation time per iteration of the `quadprog` solver for each experimental trial is plotted in red in Figs. 9.2b and 9.2e for the scenario without and with the obstacle, respectively. The mean of the trials is plotted as the black line. A similar convention is used to plot the trials and mean control effort of the solutions in Figs. 9.2c and 9.2f. There is considerable structure in the plots of computation time and control effort for the experiments, indicating the validity of the experimental data.

The mean computation time for the scenarios with and without the obstacle can be compared, as is shown in Fig. 9.3c. The data from the scenario with the obstacle is plotted in red and the data from the scenario without the obstacle is plotted in black. A similar comparison of the mean control effort between the scenarios can be compared, as is shown in Fig. 9.3d. These experimental results are compared against the trends seen in simulation, shown in Figs. 9.3a and 9.3b.

Figures 9.3a and 9.3c show that there is no significant increase in computation time when an obstacle is present, likely due to the trajectory not being significantly altered when the obstacle is present. Figure 9.3c shows that there is a higher computational load both at the beginning and end of the mission.

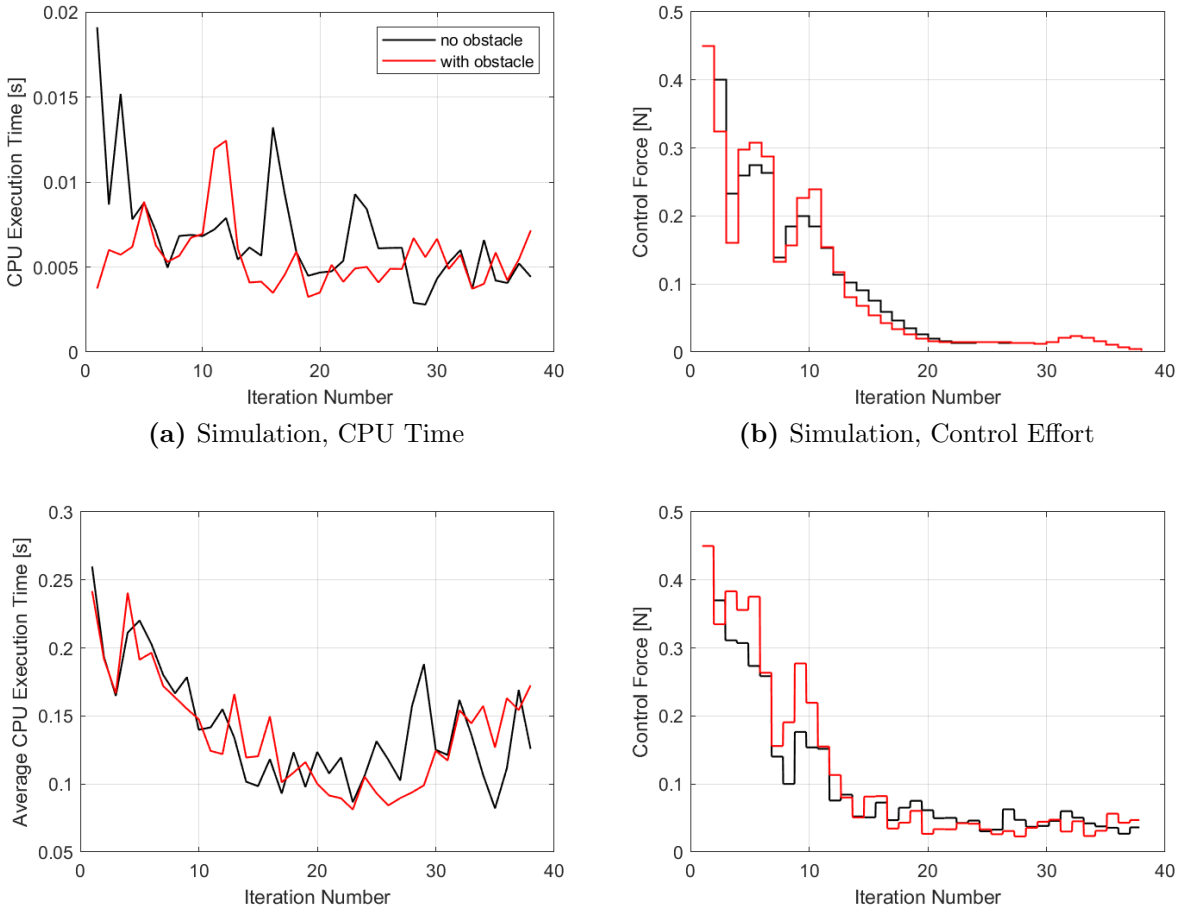
Comparisons of the control effort required by the simulation and experiment solutions, Figs. 9.3b and 9.3d, show that the experimental solution follows the expected trends, if a bit noisier. In particular, experimental data for the scenario with the obstacle shows an increase in the control effort around $k = 4$ and another significant increase at $k = 9$, which is the same trend seen in simulations.



(a) Trajectory, no obstacle **(b)** CPU Time, no obstacle **(c)** Control Effort, no obstacle

(d) Trajectory, with obstacle **(e)** CPU Time, with obstacle **(f)** Control Effort, with obstacle

Figure 9.2: The experimental results of five trials for Test Case A using `quadprog`. The trajectories, computation times, and control efforts are shown for each trial for tests with and without an obstacle.



(a) Simulation, CPU Time

(b) Simulation, Control Effort

(c) Experiment, Mean CPU Time

(d) Experiment, Mean Control Effort

Figure 9.3: Comparison of the mean computation time and mean control effort between simulations and experiments for Test Case A with and without obstacles using `quadprog`.

Table 9.3: Summary of mean solver computation times and total maneuver impulse for simulations and experiments using `quadprog`.

Case	Type	Average CPU Time, s		Total Impulse, N s	
		no obstacle	with obstacle	no obstacle	with obstacle
A	Simulation	0.0068	0.0056	7.11	6.89
	Experiment	0.1411	0.1378	8.10	8.81
B	Simulation	0.0056	0.0074	3.20	6.76
	Experiment	0.1156	0.1756	5.25	9.61
C	Simulation	0.0055	0.0058	7.61	9.60
	Experiment	0.1487	0.1444	9.79	12.08

The average computation time of the solver and total impulse of the associated solution for both simulations and experiments are summarized in Table 9.3 for the scenarios with and without the obstacle. The data shows that in simulation the `quadprog` solver is about twenty times faster than its implementation run on the RP3 during experiments. This discrepancy is mainly attributed to the different processors used, noting that the RP3[1] uses a Quad Core 1.2 GHz Broadcom BCM2837 64bit CPU with 1 GB RAM.

It is also clear from Table 9.3 that the total impulse of the maneuver in simulation is less than the total impulse during experiment, a phenomenon similarly found in [23]. The scenario without the obstacle shows that the experiment total impulse is 1 N s (14%) larger than the simulation total impulse. Similarly, for the scenario with the obstacle the experiment total impulse is about 1.9 N s (28%) larger.

9.3.2 Test Case B

The simulated trajectories of the chaser for Test Case B for the scenario with and without the obstacle are shown in Fig. 9.4. The dotted lines indicate the scenario without the obstacle, and the solid lines indicate the scenario with the obstacle. The difference in trajectory of the chaser spacecraft between the scenarios is clear.

The experimental results for five trials of Test Case B are shown in Fig. 9.5. From the trajectory plots in Figs. 9.5a and 9.5d the experimental trials plotted in bright red are compared to the simulation trial in dark red and show remarkable agreement.

Similar comparisons of the mean experiment control force to the simulation control force can be done from Figs. 9.6b and 9.6d to show a strong correlation. For the scenario with obstacles, the experiment solution experiences a wide, short peak control effort centered around $k = 23$ as opposed to the sharp, tall peak control effort experienced in simulation around $k = 26$.

From Fig. 9.6c, it is clear that during experiments the scenario with the obstacle required a higher computational time as compared to the scenario without the obstacle. This trend is not seen in the simulations, shown in Fig. 9.6a.

[1] Raspberry Pi, "Raspberry Pi 3 Model B," n.d. (accessed July 18, 2022) https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

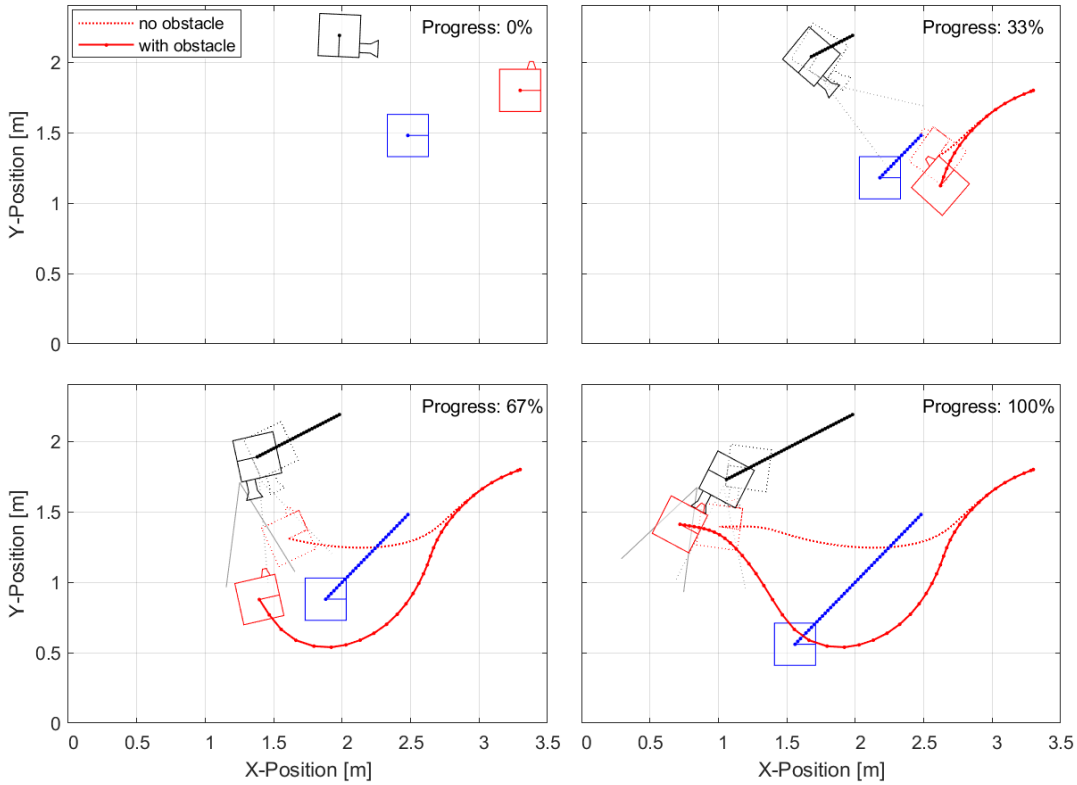
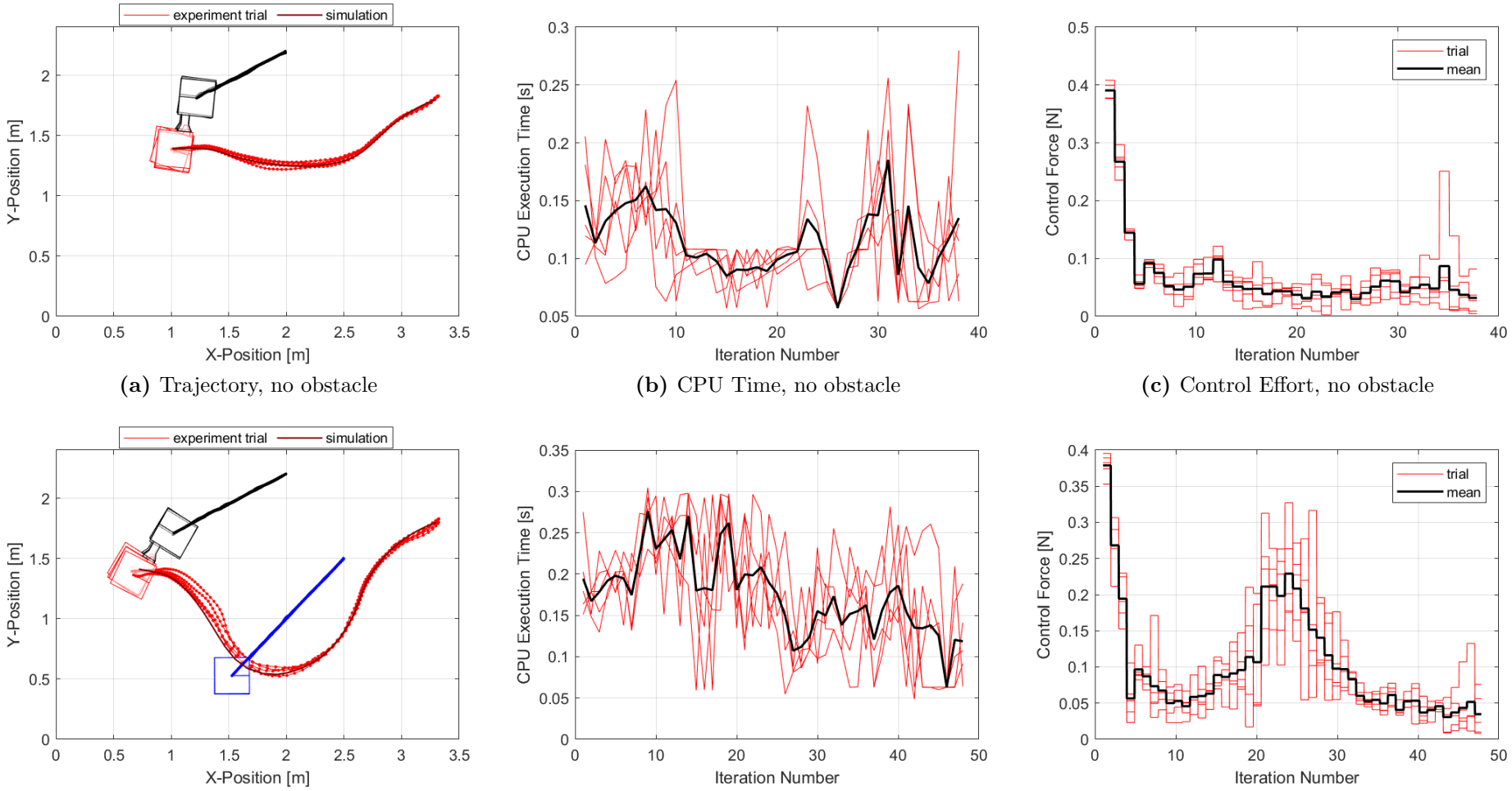


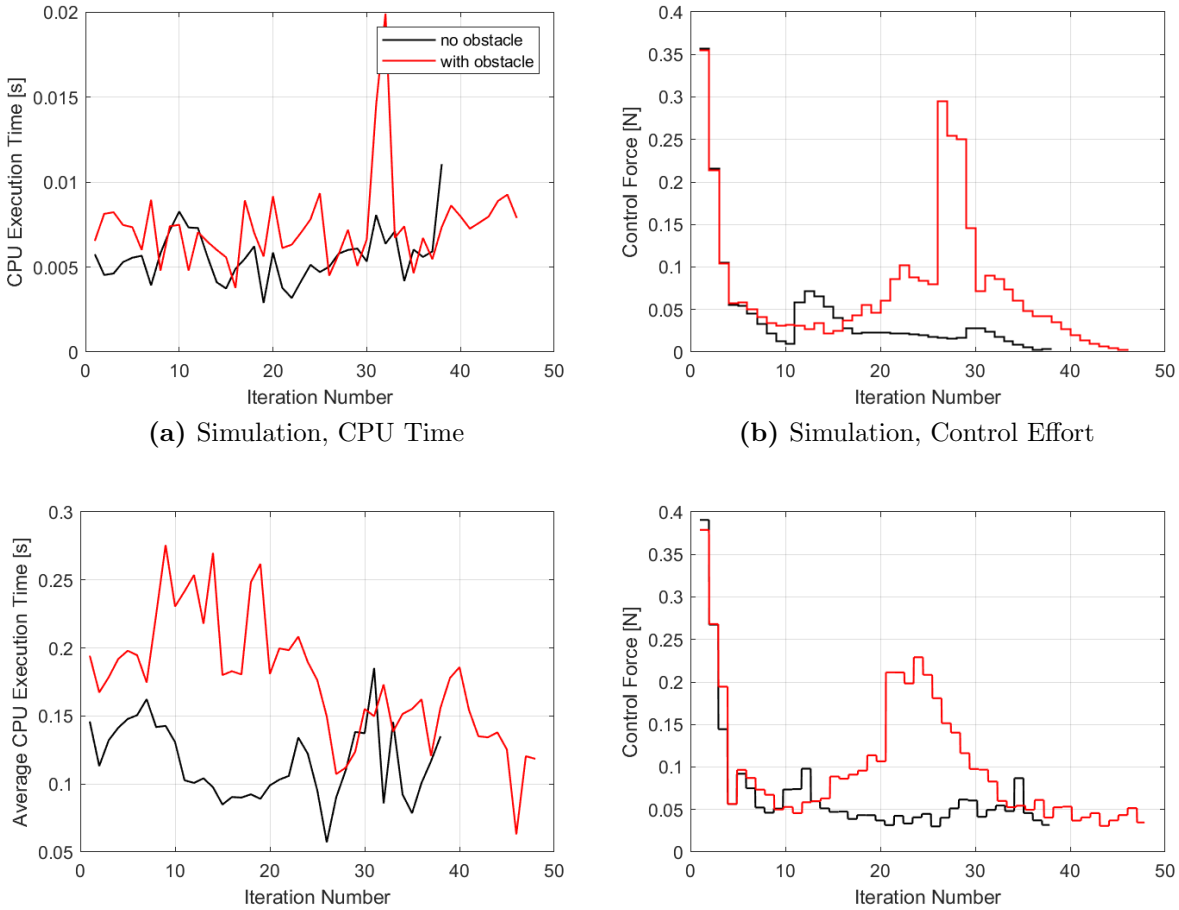
Figure 9.4: Snapshots of the chaser trajectory for Test Case B as simulated using `quadprog` with and without obstacles.



(a) Trajectory, no obstacle **(b)** CPU Time, no obstacle **(c)** Control Effort, no obstacle

(d) Trajectory, with obstacle **(e)** CPU Time, with obstacle **(f)** Control Effort, with obstacle

Figure 9.5: The experimental results of five trials for Test Case B using `quadprog`. The trajectories, computation times, and control efforts are shown for each trial for tests with and without an obstacle.



(a) Simulation, CPU Time

(b) Simulation, Control Effort

(c) Experiment, Mean CPU Time

(d) Experiment, Mean Control Effort

Figure 9.6: Comparison of the mean computation time and mean control effort between simulations and experiments for Test Case B with and without obstacles using `quadprog`.

From the average computation times summarized in Table 9.3 for Test Case B, it is again noted that the `quadprog` solver in simulation runs about twenty times faster than its implementation on the RP3. From Table 9.3 it is also found that the total impulse of experiment solutions is 64% and 42% larger than the simulation solutions for the scenario without and with the obstacle, respectively. Discrepancies of similar magnitude were also found in [23] and may be attributed to oscillations and sensor noise in the target and obstacle states, signal delays, and unmodelled disturbances including air resistance, table friction, plumbing leaks, and plume impingement. Significant sources of error are elaborated upon in §9.5.

9.3.3 Test Case C

Test Case C is the final test scenario to validate the guidance algorithm using `quadprog` as the solver in real-time. The simulated trajectories of the chaser for Test Case C for the scenario with and without the obstacle are shown in Fig. 9.7. The experimental results for five trials of Test Case C are shown in Fig. 9.8.

The trajectories of the experimental trials shown in Figs. 9.8a and 9.8d show some dissimilarities from one another, but all trials successfully dock to the target. The impact of plume impingement is seen on the trajectory of the obstacle in Fig. 9.8d, though similar plume impingement on the chaser spacecraft is unlikely due to the direction of motion of the obstacle.

The computation time of the solver for simulations and experiments, plotted in Figs. 9.9a and 9.9c, respectively, show that there is no significant difference in solving the scenario with the obstacle and without. Similar to Test Case A, this is likely due to the lack of significant change of the chaser's trajectory with the presence of the obstacle. The control effort as required by the solution in the simulation and experiment show similar trends, seen in Figs. 9.9b and 9.9d.

The average computation times summarized in Table 9.3 for Test Case C indicate that the simulation solver runs about twenty-five times faster than in experiment on the RP3. The total impulse of the solution found in experiment is about 29% and 26% higher than the solution from simulations for the scenario without and with the obstacle, respectively.

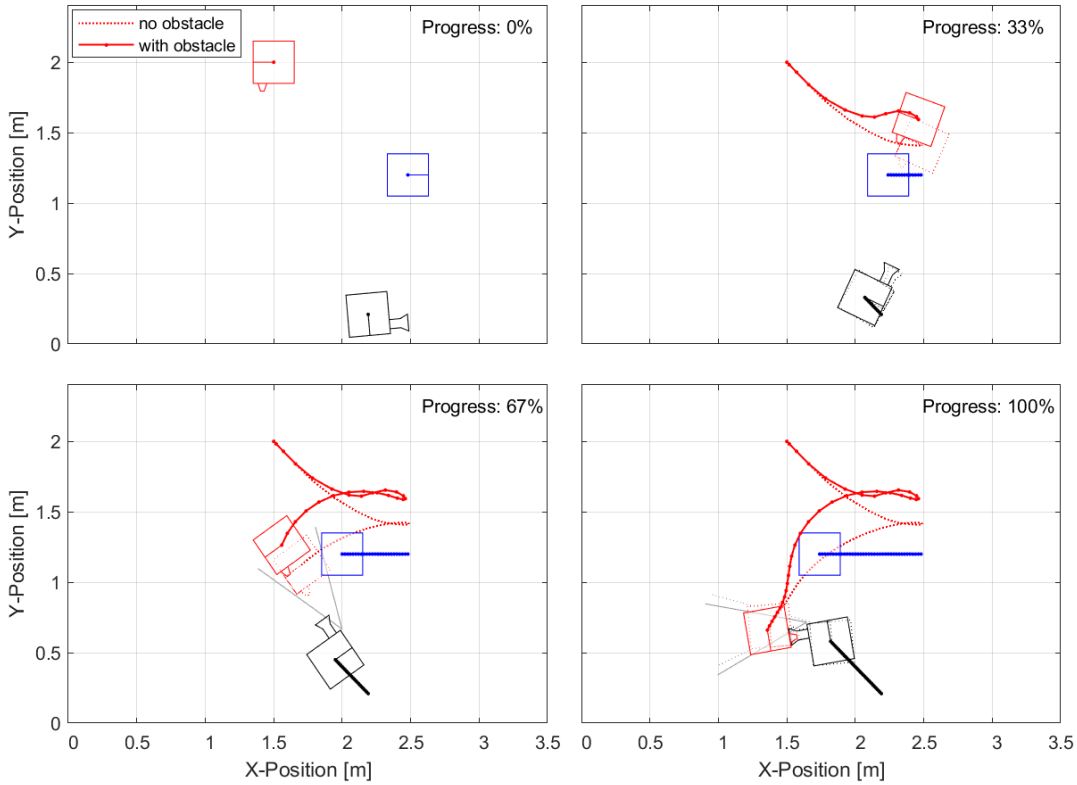
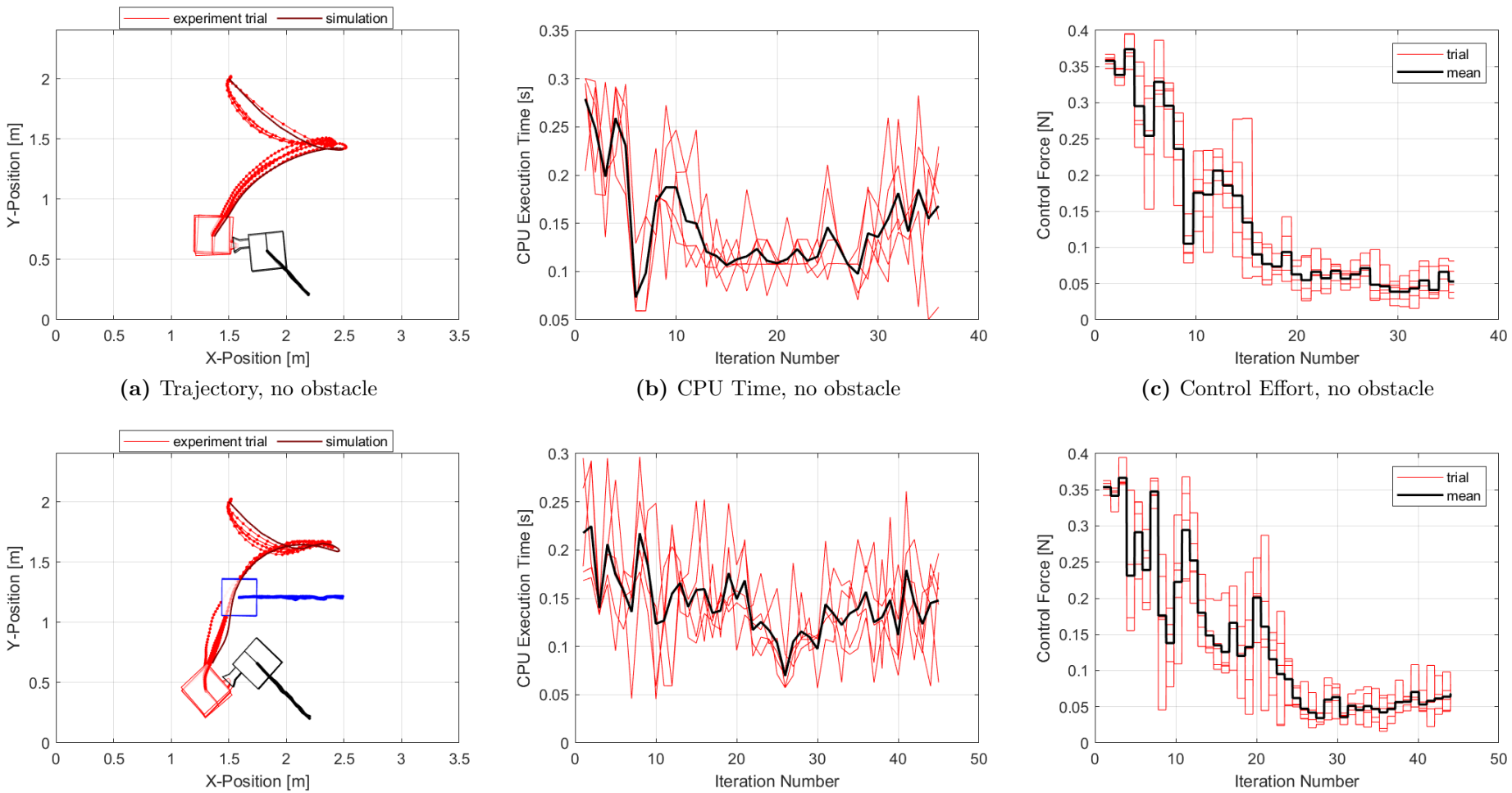


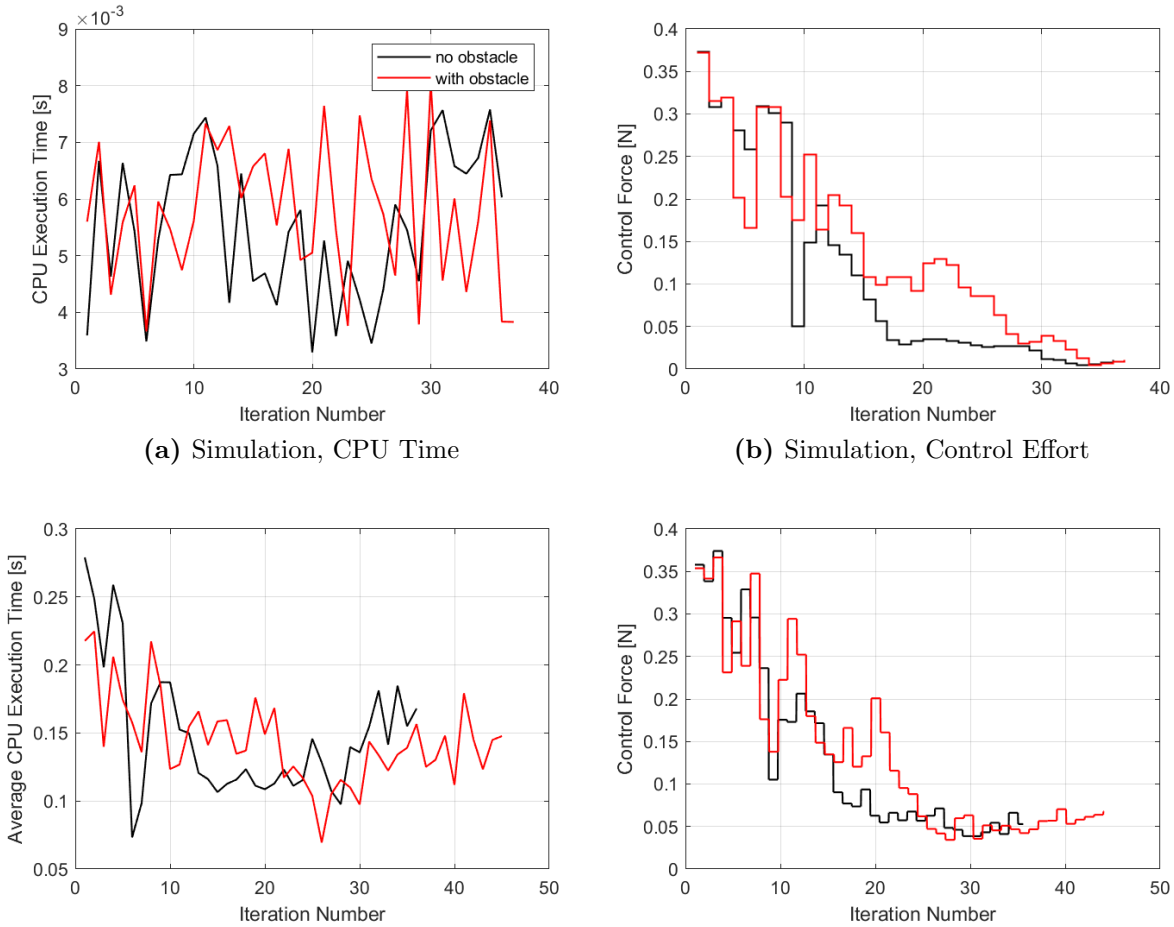
Figure 9.7: Snapshots of the chaser trajectory for Test Case C as simulated using `quadprog` with and without obstacles.



(a) Trajectory, no obstacle **(b)** CPU Time, no obstacle **(c)** Control Effort, no obstacle

(d) Trajectory, with obstacle **(e)** CPU Time, with obstacle **(f)** Control Effort, with obstacle

Figure 9.8: The experimental results of five trials for Test Case C using `quadprog`. The trajectories, computation times, and control efforts are shown for each trial for tests with and without an obstacle.



(a) Simulation, CPU Time

(b) Simulation, Control Effort

(c) Experiment, Mean CPU Time

(d) Experiment, Mean Control Effort

Figure 9.9: Comparison of the mean computation time and mean control effort between simulations and experiments for Test Case C with and without obstacles using `quadprog`.

9.4 Experiments using the Custom Solvers

This section compares the simulation and experimental results for the linear optimal control problem solved using `ISNM` and `Fast MPC`. A single test case is presented with the chaser, target, and obstacle initial conditions given in Table 9.4. Note that all platforms start at rest and the target and obstacle remain fixed for the duration of the experiment.

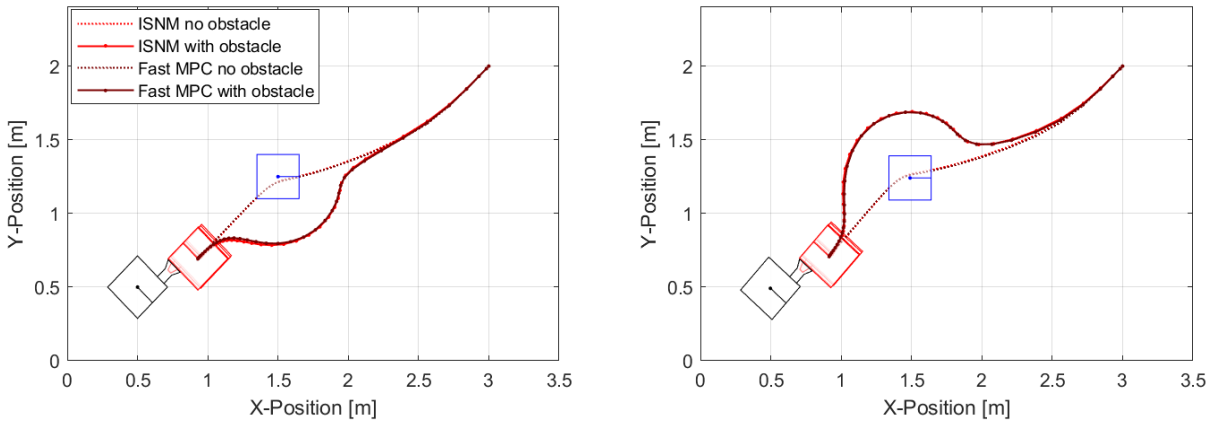
Table 9.4 gives the "nominal" positions of the platforms for the test case, and similarly gives a "perturbed" position corresponding to the actual positions of the obstacle and target used in the experimental trials. The perturbed position of the obstacle and target is offset from the nominal position by $< 2\,\text{cm}$, but as shown in the simulation results of Fig. 9.10, the trajectory computed by the guidance algorithm is significantly different between the nominal and perturbed test case when the obstacle is present.

The cause of this difference is due to the linear obstacle constraint, specifically how the reference trajectory from previous iterations is used to form the hyperplanes that guide the chaser around the obstacle. As seen by comparing Figs. 9.10a and 9.10b, the trajectory of the chaser when the obstacle is absent crosses on either side of the COM of the obstacle's set position. The trajectory taken by the chaser spacecraft with the obstacle present remains on that same side of the obstacle's COM.

The test case used for the simulations in the remainder of the section, therefore, is the perturbed test case. The constants used in the algorithm to solve this test case are given in Table 9.5. For the trials of this test case, the horizon length has been decreased to 10 to promote faster computations and the entry cone half angle has been increased to 45° to ensure that a feasible solution is found by the algorithm. It

Table 9.4: Initial states of the chaser spacecraft, target spacecraft, and obstacle. States are given in the form (x, y, θ) in units of m and °.

Parameter	Value	
	Test Case A (Nominal)	Test Case A (Perturbed)
Chaser Initial State	(3.0,2.0,0)	(3.0,2.0,0)
Target Initial State	(0.5,0.5,315)	(0.4973,0.4919,318.16)
Obstacle Initial State	(1.5,1.25,0)	(1.4897,1.2410,0)



(a) Nominal position **(b)** Perturbed position

Figure 9.10: Simulations of the trajectories taken by the chaser spacecraft, solved using `ISNM` and `Fast MPC`, for the (a) nominal and (b) perturbed test case.

Table 9.5: The optimization problem parameters for the simulations and experiments with `ISNM` and `Fast MPC`.

Parameter		Value
Problem Variables	Horizon N	10
	Discretization time T_d, s	2
	$\mathbf{Q}$	diag(10,10,10,300,300,500)
	$\mathbf{R}$	diag(10,10,10)
	Max iterations	10
	Initial guess	(-0.14,0,0)
Thruster Constraint	Maximum thrust u_{max}, N	0.15
Entry Cone Constraint	Entry cone half angle θ_h, °	45
	Entry cone range r_{cone}, m	0.75
Obstacle Constraint	Obstacle KOZ r_{KOZ}, m	$1.1 \times \sqrt{2} \times 0.3$
Dynamic Holding Radius	Initial holding radius $r_{hold,0}$, m	1.15
	Decrement factor γ	0.95
	Update r_{hold}: η tolerance, m	$\sqrt{0.5}$
	Update r_{hold}: ζ tolerance, °	10
Custom Solver	Fixed κ, ISNM	0.05
	Fixed κ, Fast MPC	0.1
	Backtracking algorithm: α	0.01
	Backtracking algorithm: β	0.95
	Residual tolerance ϵ	10^{-9}

also should be mentioned that the fixed κ value used in the custom solver is different for `ISNM` and `Fast MPC`. Simulations found that the solution for `Fast MPC` would not converge with a lower κ weight and so is twice as large as the κ value for `ISNM`. Simulations show that there is no significant difference in the solutions of the solvers despite this.

The simulated trajectories of the chaser for the scenario with and without the obstacle are shown in Fig. 9.11. The solutions as found by `ISNM` and `Fast MPC` are indistinguishable. The experimental results for five trials using the guidance algorithm solved with `ISNM` are shown in Fig. 9.12, and similarly the experimental trials solved with `Fast MPC` are shown in Fig. 9.13.

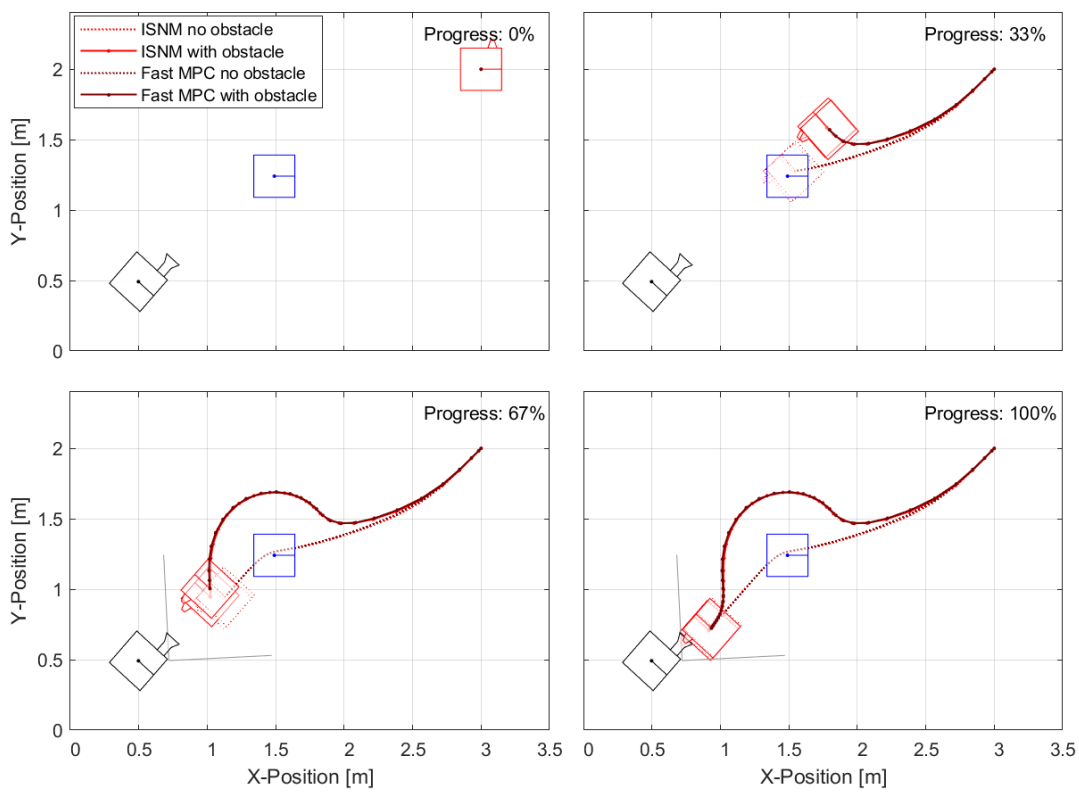
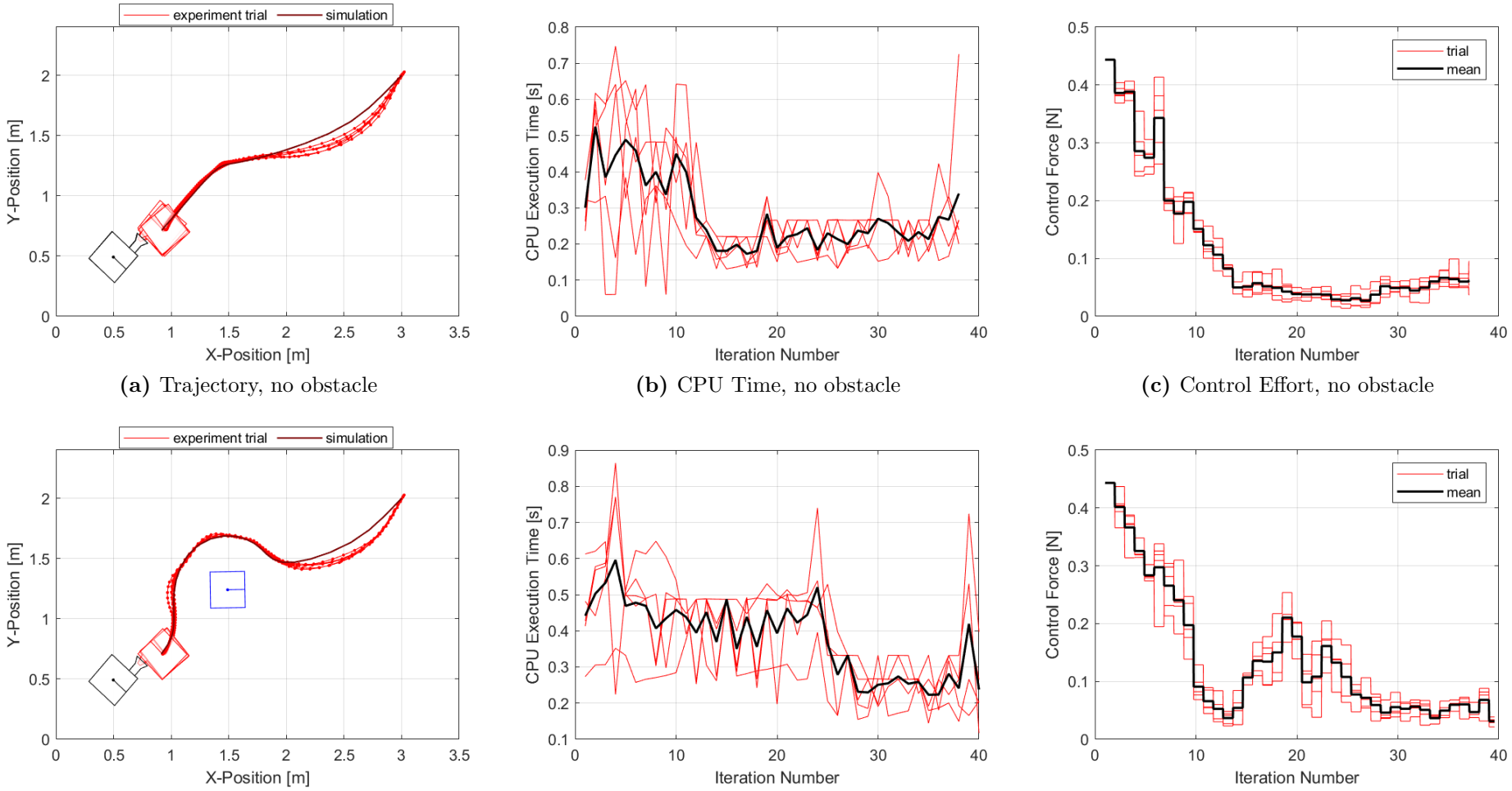


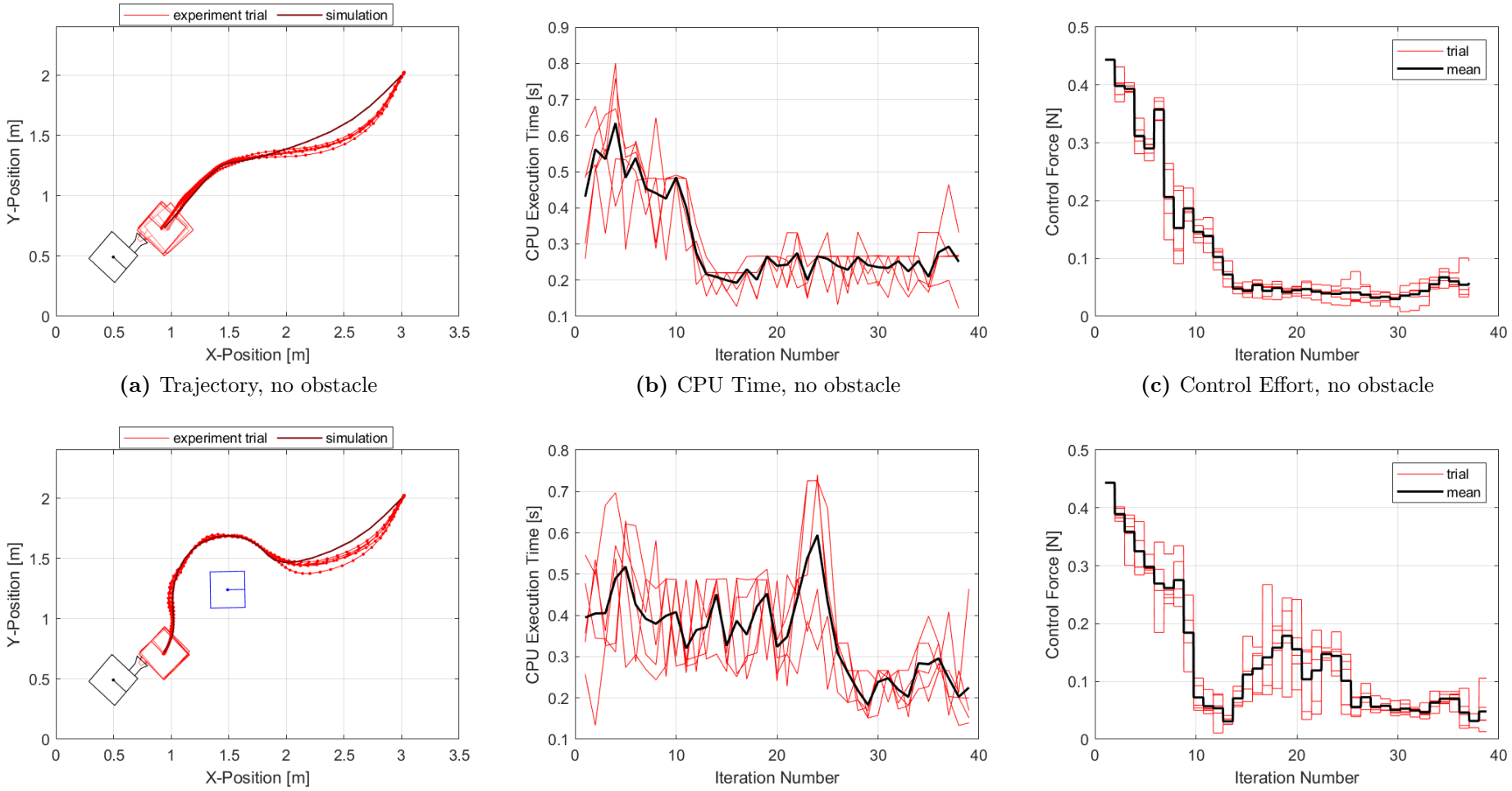
Figure 9.11: Snapshots of the chaser trajectory as simulated using `ISNM` and `Fast MPC` with and without obstacles.



(a) Trajectory, no obstacle **(b)** CPU Time, no obstacle **(c)** Control Effort, no obstacle

(d) Trajectory, with obstacle **(e)** CPU Time, with obstacle **(f)** Control Effort, with obstacle

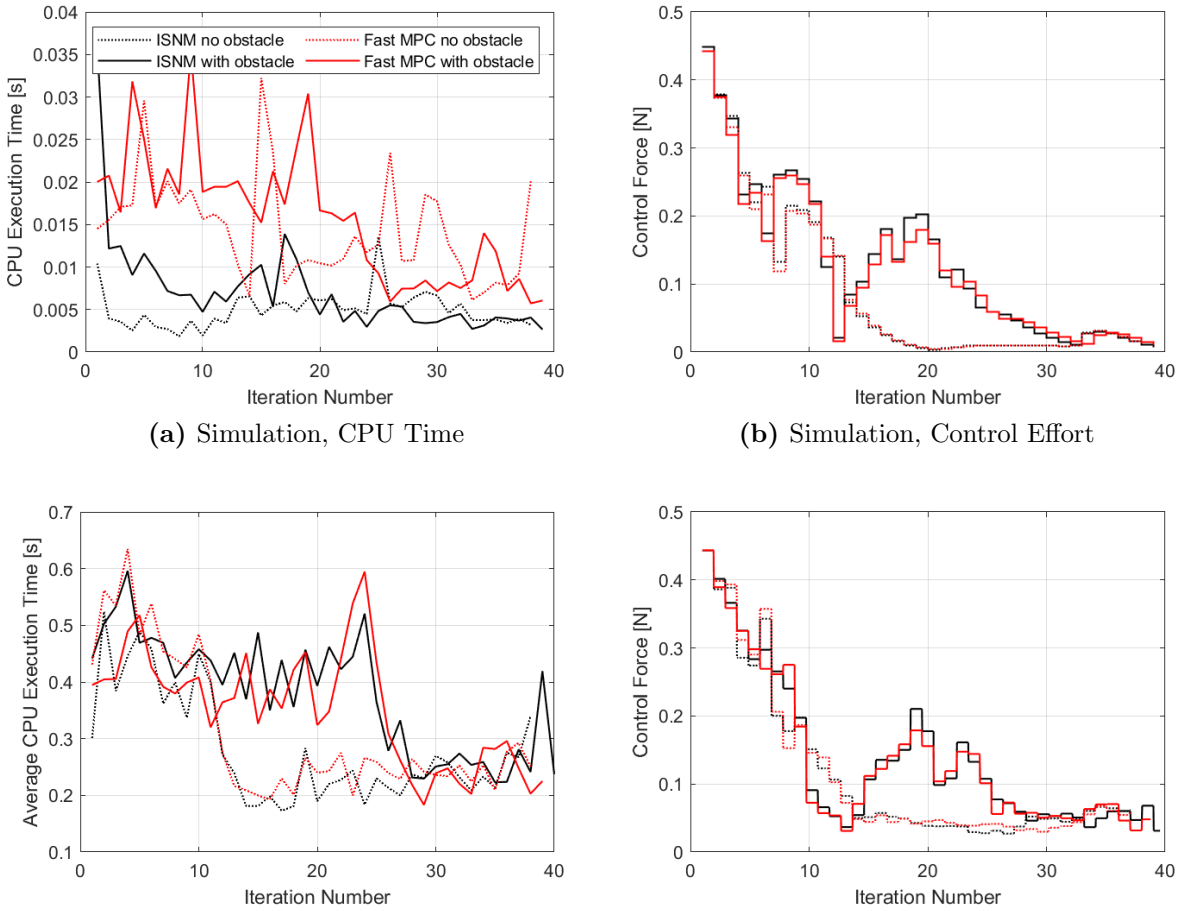
Figure 9.12: The experimental results of five trials using `ISNM`. The trajectories, computation times, and control efforts are shown for each trial for tests with and without an obstacle.



(a) Trajectory, no obstacle **(b)** CPU Time, no obstacle **(c)** Control Effort, no obstacle

(d) Trajectory, with obstacle **(e)** CPU Time, with obstacle **(f)** Control Effort, with obstacle

Figure 9.13: The experimental results of five trials using `Fast MPC`. The trajectories, computation times, and control efforts are shown for each trial for tests with and without an obstacle.



(a) Simulation, CPU Time

(b) Simulation, Control Effort

(c) Experiment, Mean CPU Time

(d) Experiment, Mean Control Effort

Figure 9.14: Comparison of the mean computation time and mean control effort between simulations and experiments with and without obstacles using `ISNM` and `Fast MPC`.

As was seen in the experimental results using `quadprog`, the experimental trials for both `ISNM` in Fig. 9.12 and `Fast MPC` in Fig. 9.13 show remarkable consistency. Comparing the trajectories taken in experiment versus the simulated trajectories in Figs. 9.12a, 9.12d, 9.13a, and 9.13d show good agreement.

The mean computation time and control effort for the scenarios with and without the obstacle can be compared for both solvers, as is shown in Fig. 9.14, for the simulations and experiments. From Fig. 9.14a, it is clear that `ISNM` is faster than `Fast MPC` in simulation. Experimental results shown in Fig. 9.14c show that the solvers have similar performance and that there is considerable structure in the trend

of the computation time associated with the presence of the obstacle, which is shown by both solvers.

Figures 9.14b and 9.14d plot the control effort required by the simulation and experiment solutions, respectively. These plots show both the remarkable agreement of the solvers to each other and the experimental solution to the simulation solution.

The average computation time of the solver over each iteration and the total impulse of the maneuver for simulations and experiments of both solvers are summarized in Table 9.6 for the scenarios with and without the obstacle. Interestingly, `ISNM` performs about fifty-five times faster in simulation than experiment, whereas `Fast MPC` performs about twenty times faster. Notably, the solutions calculated using `Fast MPC` have a lower total impulse than those of `ISNM`. Similar to the results found using `quadprog`, the total impulse for the solutions in experiment are higher than those in simulation by ∼27% for the scenario without the obstacle and ∼10% for the scenario with the obstacle for both solvers.

Table 9.6: Summary of mean solver computation times and total maneuver impulse for simulations and experiments using `ISNM` and `Fast MPC`.

Algorithm	Type	Average CPU Time, s		Total Impulse, N s	
		no obstacle	with obstacle	no obstacle	with obstacle
`ISNM`	Simulation	0.0050	0.0071	6.85	9.97
	Experiment	0.2824	0.3782	8.65	10.93
`Fast MPC`	Simulation	0.0144	0.0156	6.70	9.66
	Experiment	0.3123	0.3529	8.65	10.69

9.5 Sources of Error

During this testing campaign, there was a significant leak in the plumbing of the chaser platform caused by a broken fitting that has since been repaired. The leak was causing a persistent disturbance that may account for some of the discrepancies between the experiment and simulated results. Further sources of error come from plume impingement pushing the chaser, which was observed on several occasions when the chaser was passing very close to the obstacle, and a variable COM across trials caused by the air tank being at different levels of capacity for each test.

Chapter 10

Conclusion

The final chapter of this thesis reviews the motivation, methodology, and results of the research that was conducted. A list of topics that could be explored further are indicated.

10.1 Thesis Summary

Autonomous rendezvous and docking is a well-established technology used for on-orbit servicing missions. A threat to the safety of rendezvous missions is collisions with other space objects, particularly defunct satellites, used rocket launchers, and their fragments, that make up the category of objects called space debris. The guidance algorithm onboard the chaser spacecraft, which is responsible for planning a path to a target spacecraft, must actively avoid this space debris for the success of the mission.

Next generation guidance algorithms are considered a high priority technology. Key features of these algorithms include optimality and real-time implementability. To accomplish these requirements of a guidance algorithm, this thesis used Model Predictive Control (MPC) to formulate an optimal guidance algorithm in real-time for autonomous path-planning with moving obstacles. Commercial black box solvers `IPOPT`, `fmincon`, and `quadprog`, as well as custom solvers specifically designed for convex programming problems, referred to as `ISNM` and `Fast MPC`, were used to solve the formulated optimal control problem.

Numerical results in two- and three-dimensions proved the functionality of the

algorithm. Scenarios were modelled using Hill's equations and planar dynamics with two moving obstacles. A variety of constraint equations were used: nonlinear obstacle constraints, linearized obstacle constraints, maximum thrust limits, a dynamic holding radius, and an entry cone, where applicable. It was found that the guidance algorithm using `quadprog` as a solver had the fastest computation time and generally produced solutions with the lowest total impulse.

To validate the real-time collision avoidance capabilities of the guidance algorithm using the test facility at Carleton University, a third platform was constructed.

The guidance algorithm was validated through experiments using the upgraded laboratory facility: tests with a translating and rotating target and a moving obstacle were solved using `quadprog`, and tests with a static target and obstacle were solved using `ISNM` and `Fast MPC`. Results show excellent correlation to simulation trajectories, with the caveat that the computational time required to solve the problem in experiment is about twenty times longer than in simulations for `quadprog` and `Fast MPC` and about fifty times longer for `ISNM`. Furthermore, the total impulse required by the experimental solutions is 10 to 64% larger than required by simulations.

The successful experiments were performed in the presence of unmodelled disturbances, including plume impingement and a significant plumbing leak, demonstrating the inherent robustness of the real-time guidance algorithm. To the best of the author's knowledge, these experiments are the first to demonstrate an MPC-based guidance algorithm with moving obstacle avoidance capabilities and this work is also the first implementation of Fast MPC in a spacecraft RVD guidance algorithm.

10.2 A Note on Practicality

For the successful execution of the MPC-based guidance algorithm developed in this work, complete and accurate knowledge of the environment is required. In other words, since the optimal trajectory is determined by predicting the motion of the chaser, target, and obstacles from their current states, these states must be accurately known at every sampling instant. In practical applications, there will be sensor noise

and measurement uncertainty associated with the state data, which impacts the performance of the guidance algorithm. Furthermore, deviations of the linearized model from the true nonlinear system also introduce errors in the predicted motion, especially when the linear model is applied to scenarios where its underlying assumptions no longer hold true, e.g., slightly elliptical target orbits and larger relative spacecraft separations. Although the MPC-based guidance algorithm has demonstrated its inherent robustness to measurement uncertainties and minor model errors, further robust techniques may need to be integrated into the algorithm for scenarios with less accurate knowledge of the platform states or larger model errors to ensure that the predicted path can be executed.

The docking strategy considered in this work assumes that the chaser spacecraft has accurate knowledge of the docking port position on the target spacecraft and that low control forces can be employed. The degree to which the chaser pursues the docking port of the target depends on the relative weights of the state error and control forces in the cost function as well as the rate at which the dynamic holding radius decreases. By tuning these parameters, the user has the ability to control the speed that the chaser closes in on the target. If not tuned appropriately, the chaser may approach too quickly and collide with the target or bounce out of the docking port, or it may approach too slowly and have difficulties reaching the docking position.

10.3 Recommendations for Future Work

Ideas for future work that expand upon this thesis are provided in this section. The work falls under three categories of extending the functionality and reliability of the algorithm, improving the speed of the algorithm, and follow-up investigations using SPOT.

10.3.1 Improving the Performance of the Guidance Algorithm

Tuning Parameters

The trajectories output by the guidance algorithm have a large amount of variability depending on the problem parameters. For example, increasing the horizon length

will cause the algorithm to optimize a trajectory over a larger period and so provide a more optimal solution, but at the cost of increased computation time. The behaviour of the solution can also be modified through the weights of the optimization variables in the cost function. By increasing the weight of the state errors, the chaser spacecraft will use more fuel to quickly approach the target spacecraft. The opposite effect is seen if the weights on the control forces are increased. Specific to the custom solver, the parameters used in the backtracking line search can be modified to find a finer or coarser step size. A finer step provides a better solution at the cost of increased computation time.

Additional work on this topic should include an analysis on these tuning parameters with the goal of determining which are most critical to influencing the performance of the guidance algorithm. The study should compare trends seen in the average computation times and total impulses from which statements on the real-time implementability and optimality of the algorithm can be made, respectively.

Robustness

Although experimental validation of the algorithm demonstrated its inherent robustness to disturbances, a formal technique should be implemented that tackles robustness of the algorithm to handle uncertainties in the states. One common technique used in the literature is constraint tightening [28, 35].

Developing a Dedicated Feasibility Search

The custom algorithm developed in this thesis is quite bare compared to commercial black box solvers in terms of its error handling. An important property of the infeasible start Newton method is that the inequality constraints are satisfied by the initial guess of the solution. With the time-varying constraints used in this thesis, i.e., obstacle avoidance, entry cone, and dynamic holding radius, the initial guess is unlikely to always satisfy the updated constraint equations. This is especially seen when the entry cone constraint is initially activated since the previous optimal control problem was not limited by this constraint. Hence, a dedicated feasibility search must be developed to ensure that the inequality constraints are satisfied.

Implementing Soft Constraints

To form the equality constrained optimal control problem, the inequality constraints are implicitly written into the cost function using a logarithmic barrier. The logarithmic barrier is considered a *hard constraint* since violating the constraint creates an undefined value for the logarithm, and hence the source of the problem of the last topic. A *soft constraint* can instead be implemented which allows for constraint violation with a less harsh penalty [49, 70]. This is not necessarily an alternative to implementing a feasibility search, but would likely reduce the need for a feasibility search.

10.3.2 Improving the Speed of the Guidance Algorithm

Introducing a KOZ Activation Distance

The speed of the algorithm depends on the number of optimization variables, which depends on the horizon length, and the number of constraint equations. Similar to the activation of the entry cone constraint, a similar activation of the obstacle avoidance constraints can be implemented so that these extra constraints are not considered when the obstacle is out of the way of the chaser. This strategy is employed by [24].

Fast Implementation in C

Instead of writing the Fast MPC algorithm in MATLAB and then using MATLAB's Embedded Coder to convert this code to C++, it would be beneficial to write the solver directly in C and call the solver from MATLAB. This would hopefully reduce the inefficiencies introduced into the code through the code generation process and give significantly faster results.

10.3.3 Further Testing after Facility Upgrades

The `ISNM` and `Fast MPC` algorithms were not experimentally validated with a translating/rotating target and moving obstacle due to the current architecture of the facility software. The facility is currently being upgraded to remove these problems.

First, the method of communication with the PhaseSpace system is being updated. A ground station Raspberry Pi 3 computer will communicate to the PhaseSpace system and relay the real-time state information to the platforms. This will remove the chaser platform from the communication loop to the target and obstacle with the hope that this method will reduce the buffering of the state data such that the controllers on the target and obstacle will work properly.

Second, the facility is looking into upgrading the Raspberry Pi 3 computers to a faster alternative, the Jetson Xavier NX. The hope is that the computation time of the algorithm will be greatly improved with the better processor on this unit.

Through these improvements to the facility, it should be possible to experimentally validate the guidance algorithm with the custom solver with a translating/rotating target and moving obstacle.

References

[1] Biesbroek, R., Aziz, S., Wolahan, A., Cipolla, S., Richard-Noca, M., and Piguet, L., "The ClearSpace-1 Mission: ESA and ClearSpace Team Up to Remove Debris," *Proceedings of the 8th European Conference on Space Debris (virtual)*, ESA Space Debris Office, Darmstadt, Germany, April 2021.

[2] Coll, G. T., Webster, G., Pankiewicz, O., Schlee, K., Aranyos, T., Nufer, B., Fothergill, J., Tamasy, G., Kandula, M., Felt, A., and Hicks, N., "Satellite Servicing Projects Division Restore-L Propellant Transfer Subsystem Progress 2020," *AIAA Propulsion and Energy 2020 Forum*, Virtual, August 2020, p. 3795.

[3] Woffinden, D. C. and Geller, D. K., "Navigating the Road to Autonomous Orbital Rendezvous," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, 2007, pp. 898–909.

[4] Saponara, M., Barrena, V., Bemporad, A., Hartley, E. N., Maciejowski, J., Richards, A., Tramutola, A., and Trodden, P., "Model Predictive Control Application to Spacecraft Rendezvous in Mars Sample Return Scenario," *EUCASS Proceedings Series - Advances in AeroSpace Sciences*, edited by Array, Vol. 6, Munich, Germany, 2013, pp. 137–158.

[5] Nolet, S., Kong, E., and Miller, D. W., "Design of an Algorithm for Autonomous Docking With a Freely Tumbling Target," *Modeling, Simulation, and Verification of Space-based Systems II*, Vol. 5799, SPIE, Orlando, FL, May 2005, pp. 123–134.

[6] Cowardin, H., "Orbital Debris: Quarterly News," *Orbital Debris Quarterly News*, Vol. 26, No. 2, 2022.

[7] Cowardin, H., "Orbital Debris: Quarterly News," *Orbital Debris Quarterly News*, Vol. 26, No. 1, 2022.

[8] Liou, J.-C., "Orbital Debris: Quarterly News," *Orbital Debris Quarterly News*, Vol. 11, No. 2, 2007.

[9] Kessler, D. J. and Cour-Palais, B. G., "Collision Frequency of Artificial Satellites: The Creation of a Debris Belt," *Journal of Geophysical Research: Space Physics*, Vol. 83, No. A6, 1978, pp. 2637–2646.

[10] Starek, J. A., Açıkmeşe, B., Nesnas, I. A., and Pavone, M., "Spacecraft Autonomy Challenges for Next Generation Space Missions," *Advances in Control System Technology for Aerospace Applications*, Springer, 2016, pp. 1–48.

[11] Chai, R., Tsourdos, A., Savvaris, A., Chai, S., Xia, Y., and Chen, C. P., "Review of Advanced Guidance and Control Algorithms for Space/Aerospace Vehicles," *Progress in Aerospace Sciences*, Vol. 122, 2021, pp. 100696.

[12] Pearson, D. J., "The Glideslope Approach," *Orbital Mechanics and Mission Design*, 1989, pp. 109–123.

[13] Nolet, S., *Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[14] Hovell, K. and Ulrich, S., "Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 254–264.

[15] Hovell, K. C., *Deep Reinforcement Learning as Guidance for Aerospace Robotics*, Ph.D. thesis, Carleton University, 2022.

[16] Richards, A., Schouwenaars, T., How, J. P., and Feron, E., "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 755–764.

[17] Richards, A. and How, J., "Performance Evaluation of Rendezvous Using Model Predictive Control," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, TX, August 2003, p. 5507.

[18] Richards, A. and How, J. P., "Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility," *Proceedings of the 2003 American Control Conference*, Vol. 5, IEEE, Denver, CO, June 2003, pp. 4034–4040.

[19] Park, H., Di Cairano, S., and Kolmanovsky, I., "Linear Quadratic Model Predictive Control Approach to Spacecraft Rendezvous and Docking," *Proceedings of 21st AAS/AIAA Space Flight Mechanics Meeting, Spaceflight Mechanics, Part III of Advances in the Astronautical Sciences*, Vol. 140, New Orleans, LA, February 2011.

[20] Di Cairano, S., Park, H., and Kolmanovsky, I., "Model Predictive Control Approach for Guidance of Spacecraft Rendezvous and Proximity Maneuvering," *International Journal of Robust and Nonlinear Control*, Vol. 22, No. 12, 2012, pp. 1398–1427.

[21] Petersen, C., Jaunzemis, A., Baldwin, M., Holzinger, M., and Kolmanovsky, I., "Model Predictive Control and Extended Command Governor for Improving Robustness of Relative Motion Guidance and Control," *24th AAS/AIAA Spaceflight Mechanics Meeting*, Santa Fe, NM, January 2014, pp. 701–718.

[22] Weiss, A., Baldwin, M., Erwin, R. S., and Kolmanovsky, I., "Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies," *IEEE Transactions on Control Systems Technology*, Vol. 23, No. 4, 2015, pp. 1638–1647.

[23] Park, H., Zagaris, C., Virgili Llop, J., Zappulla, R., Kolmanovsky, I., and Romano, M., "Analysis and Experimentation of Model Predictive Control for Spacecraft Rendezvous and Proximity Operations with Multiple Obstacle Avoidance," *AIAA/AAS Astrodynamics Specialist Conference*, Long Beach, CA, September 2016, p. 5273.

[24] Zagaris, C., Park, H., Virgili-Llop, J., Zappulla, R., Romano, M., and Kolmanovsky, I., "Model Predictive Control of Spacecraft Relative Motion with Convexified Keep-Out-Zone Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 2054–2062.

[25] Jewison, C., Erwin, R. S., and Saenz-Otero, A., "Model Predictive Control with Ellipsoid Obstacle Constraints for Spacecraft Rendezvous," *IFAC-PapersOnLine*, Vol. 48, No. 9, 2015, pp. 257–262.

[26] Park, H., Zappulla, R., Zagaris, C., Virgili-Llop, J., and Romano, M., "Nonlinear Model Predictive Control for Spacecraft Rendezvous and Docking with a Rotating Target," *27th AAS/AIAA Spaceflight Mechanics Meeting*, Vol. 2, San Antonio, TX, February 2017.

[27] Vromen, S., *Design of Convex Guidance for the Final Phase of Satellite Rendezvous: Tested using Hardware-in-the-Loop Simulations*, Master's thesis, Delft University of Technology, 2015.

[28] Richards, A. and How, J. P., "Robust Variable Horizon Model Predictive Control for Vehicle Maneuvering," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, Vol. 16, No. 7, 2006, pp. 333–351.

[29] Gavilán, F., Vázquez, R., and Camacho, E. F., "Robust Model Predictive Control for Spacecraft Rendezvous with Online Prediction of Disturbance Bounds," *Aerospace Guidance, Navigation and Flight Control Systems IFAC Workshop (AGNFCS)*, 2009.

[30] Gavilán, F., Vázquez, R., and Camacho, E. F., "Chance-Constrained Model Predictive Control for Spacecraft Rendezvous with Disturbance Estimation," *Control Engineering Practice*, Vol. 20, No. 2, 2012, pp. 111–122.

[31] Buckner, C. and Lampariello, R., "Tube-based Model Predictive Control for the Approach Maneuver of a Spacecraft to a free-tumbling Target Satellite," *2018 Annual American Control Conference (ACC)*, IEEE, Milwaukee, WI, June 2018, pp. 5690–5697.

[32] Lampariello, R., *Optimal Motion Planning for Object Interception and Capture*, Ph.D. thesis, Technical University of Darmstadt, 2021.

[33] Albee, K., Oestreich, C., Specht, C., Terán Espinoza, A., Todd, J., Hokaj, I., Lampariello, R., and Linares, R., "A Robust Observation, Planning, and Control Pipeline for Autonomous Rendezvous with Tumbling Targets," *Frontiers in Robotics and AI*, 2021, pp. 234.

[34] Albee, K., Specht, C., Mishra, H., Oestreich, C., Brunner, B., Lampariello, R., and Linares, R., "Autonomous Rendezvous with an Uncertain, Uncooperative Tumbling Target: the Tumbledock Flight Experiments," *16th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, Noordwijk, the Netherlands, June 2022.

[35] Mammarella, M., Capello, E., Park, H., Guglieri, G., and Romano, M., "Tube-based Robust Model Predictive Control for Spacecraft Proximity Operations in the Presence of Persistent Disturbance," *Aerospace Science and Technology*, Vol. 77, 2018, pp. 585–594.

[36] Lopez, I. and Mclnnes, C. R., "Autonomous Rendezvous Using Artificial Potential Function Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 2, 1995, pp. 237–241.

[37] Dong, H., Hu, Q., and Akella, M. R., "Dual-Quaternion-Based Spacecraft Autonomous Rendezvous and Docking Under Six-Degree-of-Freedom Motion Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 5, 2018, pp. 1150–1162.

[38] Hough, J. and Ulrich, S., "Lyapunov Vector Fields for Thrust-Limited Spacecraft Docking with an Elliptically-Orbiting Uncooperative Tumbling Target," *AIAA Scitech 2020 Forum*, Orlando, FL, January 2020, p. 2078.

[39] McCamish, S. B., Romano, M., Nolet, S., Edwards, C. M., and Miller, D. W., "Flight Testing of Multiple-Spacecraft Control on SPHERES During Close-Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 46, No. 6, 2009, pp. 1202–1213.

[40] Bevilacqua, R., Lehmann, T., and Romano, M., "Development and Experimentation of LQR/APF Guidance and Control for Autonomous Proximity Maneuvers of Multiple Spacecraft," *Acta Astronautica*, Vol. 68, No. 7-8, 2011, pp. 1260–1275.

[41] Muñoz, J. D., *Rapid Path-Planning Algorithms for Autonomous Proximity Operations of Satellites*, Ph.D. thesis, University of Florida, 2011.

[42] Zappulla, R. I., Park, H., Virgili-Llop, J., and Romano, M., "Experiments on Autonomous Spacecraft Rendezvous and Docking using an Adaptive Artificial Potential Field Approach," *Proceedings of the Space Flight Mechanics Meeting*, AAS Paper AAS 16-459, Univelt Inc., San Diego, CA, February 2016, pp. 4477–4494.

[43] Zappulla, R. I., *Experimental Evaluation Methodology for Spacecraft Proximity Maneuvers in a Dynamic Environment*, Ph.D. thesis, Naval Postgraduate School, 2017.

[44] Zappulla, R., Park, H., Virgili-Llop, J., and Romano, M., "Real-Time Autonomous Spacecraft Proximity Maneuvers and Docking Using an Adaptive Artificial Potential Field Approach," *IEEE Transactions on Control Systems Technology*, Vol. 27, No. 6, 2018, pp. 2598–2605.

[45] Boyarko, G. A., *Spacecraft Guidance Strategies for Proximity Maneuvering and Close Approach with a Tumbling Object*, Ph.D. thesis, Naval Postgraduate School, 2010.

[46] Ciarcià, M., Grompone, A., and Romano, M., "A Near-Optimal Guidance for Cooperative Docking Maneuvers," *Acta Astronautica*, Vol. 102, 2014, pp. 367–377.

[47] Ventura, J., Ciarcià, M., Romano, M., and Walter, U., "An Inverse Dynamics-Based Trajectory Planner for Autonomous Docking to a Tumbling Target," *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, January 2016, p. 0876.

[48] Virgili-Llop, J., Zagaris, C., Park, H., Zappulla, R., and Romano, M., "Experimental Evaluation of Model Predictive Control and Inverse Dynamics Control for Spacecraft Proximity and Docking Maneuvers," *CEAS Space Journal*, Vol. 10, No. 1, 2018, pp. 37–49.

[49] Fear, A. and Lightsey, E. G., "Implementation of Small Satellite Autonomous Rendezvous Using Model Predictive Control," *AIAA SCITECH 2022 Forum*, San Diego, CA & Virtual, January 2022, p. 0838.

[50] Wang, Y. and Boyd, S., "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2009, pp. 267–278.

[51] Hartley, E. N. and Maciejowski, J. M., "Field Programmable Gate Array Based Predictive Control System for Spacecraft Rendezvous in Elliptical Orbits," *Optimal Control Applications and Methods*, Vol. 36, No. 5, 2015, pp. 585–607.

[52] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Vol. 16, Cambridge University Press, 2003.

[53] ISO Central Secretary, "Space systems — Rendezvous and Proximity Operations (RPO) and On Orbit Servicing (OOS) — Programmatic principles and practices," Standard ISO 24330:2022, International Organization for Standardization, Geneva, CH, 2022.

[54] Oberg, J., "Rendezvous and Proximity Operations Handbook," Tech. rep., NASA, Johnson Space Center, Houston, TX, 1988.

[55] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, No. 6, 2000, pp. 789–814.

[56] Eren, U., Prach, A., Koçer, B. B., Raković, S. V., Kayacan, E., and Açıkmeşe, B., "Model Predictive Control in Aerospace Systems: Current State and Opportunities," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1541–1566.

[57] Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004.

[58] Di Cairano, S. and Kolmanovsky, I. V., "Real-time Optimization and Model Predictive Control for Aerospace and Automotive Applications," *2018 Annual American Control Conference (ACC)*, IEEE, Milwaukee, WI, June 2018, pp. 2392–2409.

[59] Rao, A. V., "A Survey of Numerical Methods for Optimal Control," *Advances in the Astronautical Sciences*, Vol. 135, No. 1, 2009, pp. 497–528.

[60] Wächter, A. and Biegler, L. T., "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57.

[61] Wächter, A., *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*, Ph.D. thesis, Carnegie Mellon University, 2002.

[62] Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer, 1999.

[63] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 10, No. 3, 1984, pp. 282–298.

[64] Henshaw, C. G., Glassner, S., Naasz, B., and Roberts, B., "Grappling Spacecraft," *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 5, 2022, pp. 137–159.

[65] Shoemaker, M. A., Vavrina, M., Gaylor, D. E., Mcintosh, R., Volle, M., and Jacobsohn, J., "OSAM-1 Decommissioning Orbit Design," *AAS/AIAA Astrodynamics Specialist Conference*, South Lake Tahoe, CA, August 2020.

[66] Park, H., Di Cairano, S., and Kolmanovsky, I., "Model Predictive Control for Spacecraft Rendezvous and Docking with a Rotating/Tumbling Platform and for Debris Avoidance," *Proceedings of the 2011 American Control Conference*, IEEE, San Francisco, CA, June 2011, pp. 1922–1927.

[67] Bloise, N., Capello, E., Dentis, M., and Punta, E., "Obstacle Avoidance with Potential Field Applied to a Rendezvous Maneuver," *Applied Sciences*, Vol. 7, No. 10, 2017, pp. 1042.

[68] Hovell, K., *Detumbling Space Debris Using Tethers*, Master's thesis, Carleton University, 2017.

[69] Fraser, C. T., *Adaptive Extended Kalman Filtering Strategies for Autonomous Relative Navigation of Formation Flying Spacecraft*, Master's thesis, Carleton University, 2019.

[70] Richards, A., "Fast Model Predictive Control with Soft Constraints," *European Journal of Control*, Vol. 25, 2015, pp. 51–59.

Appendix A

Relative Motion of Spacecraft

This section provides the mathematical derivation of Hill's equations. First, the mathematical notation used in the derivation is described. The non-linear equations of relative motion are constructed, and using two key assumptions, the linearized equations of motion are obtained.

A.1 Mathematical Notation

To mathematically describe the relative positions of spacecraft in a reference frame, a vector is required.

Consider a reference frame $\mathcal{F}_a$ with a basis formed by the unit vectors $\{\hat{x}_a, \hat{y}_a, \hat{z}_a\}$. A generic vector $\vec{r}$ can be written in terms of its components in this given reference frame:

$$\vec{r} = r_x\hat{x}_a + r_y\hat{y}_a + r_z\hat{z}_a \qquad \text{or} \qquad \vec{r} = \begin{bmatrix} \hat{x}_a & \hat{y}_a & \hat{z}_a \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{A.1}$$

The *scalar components* of $\vec{r}$ in $\mathcal{F}_a$ are the scalar coefficients r_x, r_y, and r_z. The scalar components of a vector in a particular reference frame is denoted by the boldface

vector symbol with a subscript indicating the reference frame,

$$\mathbf{r}_a = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}. \tag{A.2}$$

The unconventional matrix containing the basis vectors of the reference frame is called a *vectrix* and is denoted by the related reference frame symbol with an overhead arrow,

$$\vec{\mathcal{F}}_a = \begin{bmatrix} \hat{x}_a \\ \hat{y}_a \\ \hat{z}_a \end{bmatrix}. \tag{A.3}$$

Thus, the vector $\vec{r}$ can be written in terms of its components in the $\mathcal{F}_a$ reference frame compactly as

$$\vec{r} = \vec{\mathcal{F}}_a^T \mathbf{r}_a. \tag{A.4}$$

To compute the time derivative of a vector $\vec{r}$ in a general reference frame $\mathcal{F}_a$, one simply uses the product rule:

$$\vec{r} = \vec{\mathcal{F}}_a^T \mathbf{r}_a \tag{A.5}$$

$$\dot{\vec{r}} = \dot{\vec{\mathcal{F}}}_a^T \mathbf{r}_a + \vec{\mathcal{F}}_a^T \dot{\mathbf{r}}_a \tag{A.6}$$

$$\ddot{\vec{r}} = \ddot{\vec{\mathcal{F}}}_a^T \mathbf{r}_a + 2\dot{\vec{\mathcal{F}}}_a^T \dot{\mathbf{r}}_a + \vec{\mathcal{F}}_a^T \ddot{\mathbf{r}}_a \tag{A.7}$$

If the reference frame $\mathcal{F}_a$ is non-inertial, its time-derivative will not be zero. In relation to an inertial reference frame $\mathcal{F}_b$, let $\vec{\omega}_{ab}$ denote the angular velocity that $\mathcal{F}_a$ is rotating with respect to $\mathcal{F}_b$. The time derivatives of the unit vectors of $\mathcal{F}_a$ can be written as

$$\dot{\vec{\mathcal{F}}}_a^T = \vec{\omega}_{ab} \times \vec{\mathcal{F}}_a^T \tag{A.8}$$

$$\dot{\vec{\mathcal{F}}}_a^T = \vec{\mathcal{F}}_a^T \boldsymbol{\omega}_{ab}^{\times} \tag{A.9}$$

and

$$\ddot{\vec{\mathcal{F}}}_a^T = \dot{\vec{\omega}}_{ab} \times \vec{\mathcal{F}}_a^T + \vec{\omega}_{ab} \times \dot{\vec{\mathcal{F}}}_a^T \tag{A.10}$$

$$\ddot{\vec{\mathcal{F}}}_a^T = \vec{\mathcal{F}}_a^T \left(\dot{\boldsymbol{\omega}}_{ab}^{\times} + \boldsymbol{\omega}_{ab}^{\times} \boldsymbol{\omega}_{ab}^{\times} \right), \tag{A.11}$$

where the superscript $\times$ indicates the *skew-symmetic cross product operator matrix*, which is defined for some general component matrix $\mathbf{u} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ as

$$\mathbf{u}^{\times} \triangleq \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{A.12}$$

A.2 Nonlinear Relative Dynamics

The two-body problem as discussed in §2.2 applies to both the target and chaser spacecraft that orbit the planetary body such that their equations of motion are, respectively:

$$\ddot{\vec{r}}_t = -\frac{\mu}{|\vec{r}_t|^3}\vec{r}_t \qquad\qquad \ddot{\vec{r}}_c = -\frac{\mu}{|\vec{r}_c|^3}\vec{r}_c \tag{A.13}$$

The position of the chaser spacecraft relative to the target spacecraft is given by the vector $\vec{\rho}$, as shown in Fig. A.1. In the LVLH frame, the components of this vector are designated as $\boldsymbol{\rho}_L \triangleq \begin{bmatrix} x & y & z \end{bmatrix}^T$, such that that the vector $\vec{\rho}$ can be written as

$$\vec{\rho} \triangleq \vec{\mathcal{F}}_L^T \boldsymbol{\rho}_L = \vec{\mathcal{F}}_L^T \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{A.14}$$

From Fig. A.1, it is apparent that $\vec{\rho}$ is related to $\vec{r}_c$ and $\vec{r}_t$ via

$$\vec{\rho} = \vec{r}_c - \vec{r}_t. \tag{A.15}$$

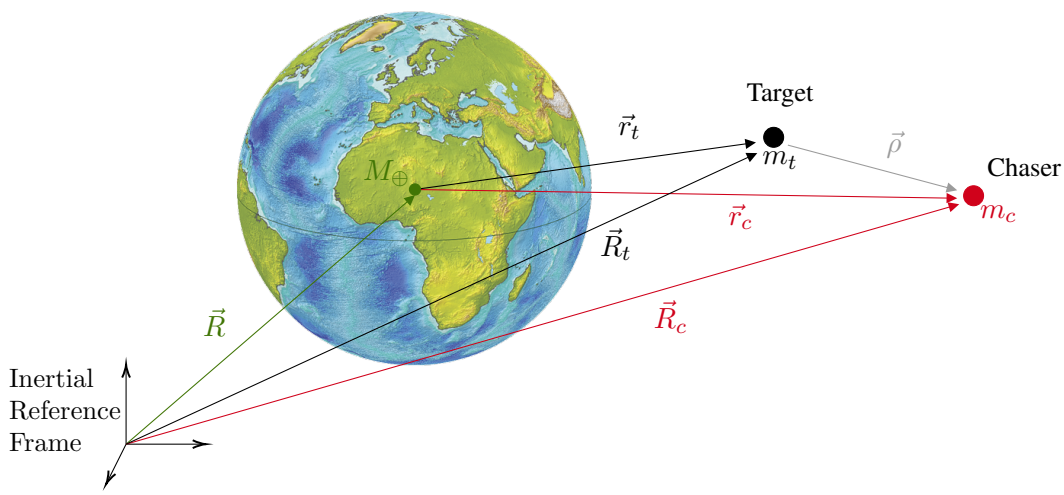


Figure A.1: The positions of a planetary body, chaser spacecraft, and target spacecraft as seen in an inertial reference frame.

Of interest are the equations of motion of the chaser with respect to the target spacecraft. These equations are found by the double-time derivative of the relative position,

$$\ddot{\vec{\rho}} = \ddot{\vec{r}}_c - \ddot{\vec{r}}_t. \tag{A.16}$$

Starting with Eq. (A.16), the two-body equations of motion from Eq. (A.13) for the spacecraft are substituted in, and the result is equated to the definition of the second time derivative of a vector in a non-inertial reference frame, Eq. (A.7), particularly for the LVLH frame. Note that the notation for the magnitude of the vectors has been simplified, $|\vec{r}_t| = r_t$ and $|\vec{r}_c| = r_c$. The relations in Eq. (A.9) and Eq. (A.11) are

used.

$$
\begin{aligned}
\ddot{\vec{\rho}} &= \ddot{\vec{\rho}} \\
\ddot{\vec{r}}_c - \ddot{\vec{r}}_t &= \ddot{\vec{r}}_c - \ddot{\vec{r}}_t \\
-\frac{\mu}{r_c^3}\vec{r}_c + \frac{\mu}{r_t^3}\vec{r}_t &= \ddot{\vec{r}}_c - \ddot{\vec{r}}_t \\
-\frac{\mu}{r_c^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{cL} + \frac{\mu}{r_t^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{tL} &= \left(\ddot{\vec{\mathcal{F}}}_L^T\mathbf{r}_{cL} + 2\dot{\vec{\mathcal{F}}}_L^T\dot{\mathbf{r}}_{cL} + \vec{\mathcal{F}}_L^T\ddot{\mathbf{r}}_{cL}\right) - \left(\ddot{\vec{\mathcal{F}}}_L^T\mathbf{r}_{tL} + 2\dot{\vec{\mathcal{F}}}_L^T\dot{\mathbf{r}}_{tL} + \vec{\mathcal{F}}_L^T\ddot{\mathbf{r}}_{tL}\right) \\
-\frac{\mu}{r_c^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{cL} + \frac{\mu}{r_t^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{tL} &= \ddot{\vec{\mathcal{F}}}_L^T\left(\mathbf{r}_{cL} - \mathbf{r}_{tL}\right) + 2\dot{\vec{\mathcal{F}}}_L^T\left(\dot{\mathbf{r}}_{cL} - \dot{\mathbf{r}}_{tL}\right) + \vec{\mathcal{F}}_L^T\left(\ddot{\mathbf{r}}_{cL} - \ddot{\mathbf{r}}_{tL}\right) \\
-\frac{\mu}{r_c^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{cL} + \frac{\mu}{r_t^3}\vec{\mathcal{F}}_L^T\mathbf{r}_{tL} &= \vec{\mathcal{F}}_L^T\left(\dot{\boldsymbol{\omega}}_{LI}^{\times} + \boldsymbol{\omega}_{LI}^{\times}\boldsymbol{\omega}_{LI}^{\times}\right)\left(\mathbf{r}_{cL} - \mathbf{r}_{tL}\right) + 2\vec{\mathcal{F}}_L^T\boldsymbol{\omega}_{LI}^{\times}\left(\dot{\mathbf{r}}_{cL} - \dot{\mathbf{r}}_{tL}\right) + \vec{\mathcal{F}}_L^T\left(\ddot{\mathbf{r}}_{cL} - \ddot{\mathbf{r}}_{tL}\right) \\
-\frac{\mu}{r_c^3}\mathbf{r}_{cL} + \frac{\mu}{r_t^3}\mathbf{r}_{tL} &= \left(\dot{\boldsymbol{\omega}}_{LI}^{\times} + \boldsymbol{\omega}_{LI}^{\times}\boldsymbol{\omega}_{LI}^{\times}\right)\left(\mathbf{r}_{cL} - \mathbf{r}_{tL}\right) + 2\boldsymbol{\omega}_{LI}^{\times}\left(\dot{\mathbf{r}}_{cL} - \dot{\mathbf{r}}_{tL}\right) + \left(\ddot{\mathbf{r}}_{cL} - \ddot{\mathbf{r}}_{tL}\right)
\end{aligned}
\tag{A.17}
$$

Now the components of the vectors in the LVLH frame must be defined. Recall that the LVLH frame is defined such that the radial direction points along the vector from the ECI origin to the target spacecraft, thus the components of the position vector of the target spacecraft (and its derivatives) in the LVLH frame are simply:

$$
\mathbf{r}_{tL} = \begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} \qquad \dot{\mathbf{r}}_{tL} = \begin{bmatrix} \dot{r}_t \\ 0 \\ 0 \end{bmatrix} \qquad \ddot{\mathbf{r}}_{tL} = \begin{bmatrix} \ddot{r}_t \\ 0 \\ 0 \end{bmatrix} \tag{A.18}
$$

Similarly, the position of the chaser spacecraft written in the LVLH frame can be written simply from the relation $\vec{r}_c = \vec{r}_t + \vec{\rho} = \vec{\mathcal{F}}_L^T\left(\mathbf{r}_{tL} + \boldsymbol{\rho}_L\right)$. Thus:

$$
\mathbf{r}_{cL} = \begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} \qquad \dot{\mathbf{r}}_{cL} = \begin{bmatrix} \dot{r}_t + \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \qquad \ddot{\mathbf{r}}_{cL} = \begin{bmatrix} \ddot{r}_t + \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{A.19}
$$

Lastly, the rotation rate of the LVLH frame with respect to the ECI frame is known to be perpendicular to the orbital plane:

$$\boldsymbol{\omega}_{LI} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \qquad \dot{\boldsymbol{\omega}}_{LI} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix} \tag{A.20}$$

Substituting the component matrices into Eq. (A.17) gives

$$\begin{aligned}
-\frac{\mu}{r_c^3}\begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} + \frac{\mu}{r_t^3}\begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} &= \left(\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}^{\times} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}^{\times} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}^{\times} \right) \left(\begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} - \begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} \right) + 2 \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}^{\times} \left(\begin{bmatrix} \dot{r}_t + \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \begin{bmatrix} \dot{r}_t \\ 0 \\ 0 \end{bmatrix} \right) + \left(\begin{bmatrix} \ddot{r}_t + \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} - \begin{bmatrix} \ddot{r}_t \\ 0 \\ 0 \end{bmatrix} \right) \\
-\frac{\mu}{r_c^3}\begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} + \frac{\mu}{r_t^3}\begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} &= \begin{bmatrix} -\ddot{\theta} y \\ \ddot{\theta} x \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\theta}^2 x \\ -\dot{\theta}^2 y \\ 0 \end{bmatrix} + 2 \begin{bmatrix} -\dot{\theta}\dot{y} \\ \dot{\theta}\dot{x} \\ 0 \end{bmatrix} + \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix},
\end{aligned} \tag{A.21}$$

which can be written as:

$$\ddot{x} - \ddot{\theta} y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 x + \frac{\mu}{r_c^3}(r_t + x) - \frac{\mu}{r_t^2} = 0 \tag{A.22}$$

$$\ddot{y} + \ddot{\theta} x + 2\dot{\theta}\dot{x} - \dot{\theta}^2 y + \frac{\mu}{r_c^3} y = 0 \tag{A.23}$$

$$\ddot{z} + \frac{\mu}{r_c^3} z = 0 \tag{A.24}$$

Neglecting perturbations, the angular momentum for the target spacecraft $h = r_t^2\dot{\theta}$ should be conserved such that its time derivative is given by

$$\dot{h} = 2r_t\dot{r}_t\dot{\theta} + r_t^2\ddot{\theta} = 0, \tag{A.25}$$

which implies for the LVLH frame centered at the target spacecraft that

$$\ddot{\theta} = -2\frac{\dot{r}_t}{r_t}\dot{\theta}. \tag{A.26}$$

Substituting Eq. (A.26) into Eqs. (A.22)-(A.24) reduces these expressions to the exact, nonlinear equations of motion of the chaser spacecraft relative to the target spacecraft in the LVLH frame:

$$\ddot{x} + 2\frac{\dot{r}_t}{r_t}\dot{\theta}y - 2\dot{\theta}\dot{y} - \dot{\theta}^2x + \frac{\mu}{r_c^3}(r_t + x) - \frac{\mu}{r_t^2} = 0 \tag{A.27}$$

$$\ddot{y} - 2\frac{\dot{r}_t}{r_t}\dot{\theta}x + 2\dot{\theta}\dot{x} - \dot{\theta}^2y + \frac{\mu}{r_c^3}y = 0 \tag{A.28}$$

$$\ddot{z} + \frac{\mu}{r_c^3}z = 0 \tag{A.29}$$

A.3 Linearized Relative Dynamics

The non-linear equations of motion can be linearized using two assumptions.

Assumption 1

The distance from the target to the chaser spacecraft is much smaller than the target orbit radius, i.e., $r_t \gg |\vec{\rho}| = ||\boldsymbol{\rho}_L|| = \sqrt{x^2 + y^2 + z^2}$.

If Assumption 1 holds, the orbit radius of the chaser spacecraft r_c can be simplified:

$$
\begin{aligned}
r_c &= \sqrt{(r_t + x)^2 + y^2 + z^2} \\
&= \sqrt{r_t^2 + 2r_t x + x^2 + y^2 + z^2} \\
&= |r_t| \sqrt{1 + \frac{2x}{r_t} + \cancelto{0}{\frac{x^2 + y^2 + z^2}{r_t^2}}} \\
&\approx |r_t| \sqrt{1 + \frac{2x}{r_t}}
\end{aligned}
\tag{A.30}
$$

The term of interest is r_c^{-3},

$$
r_c^{-3} \approx r_t^{-3} \left(1 + \frac{2x}{r_t}\right)^{-3/2}, \tag{A.31}
$$

which can be further simplified through use of a binomial expansion, $(1+x)^n \approx 1+nx$:

$$
\begin{aligned}
r_c^{-3} &\approx r_t^{-3} \left[1 - \frac{3}{2}\left(\frac{2x}{r_t}\right)\right] \\
&= r_t^{-3} \left(1 - \frac{3x}{r_t}\right)
\end{aligned}
\tag{A.32}
$$

Then, the term for the two-body equation of motion of the chaser spacecraft, Eq. (A.13), can be simplified by neglecting higher order terms:

$$
\begin{aligned}
-\frac{\mu}{r_c^3}\begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} &\approx -\frac{\mu}{r_t^3}\left(1 - \frac{3x}{r_t}\right)\begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} \\
&\approx -\frac{\mu}{r_t^3}\begin{bmatrix} (r_t + x)\left(1 - \frac{3x}{r_t}\right) \\ y\left(1 - \frac{3x}{r_t}\right) \\ z\left(1 - \frac{3x}{r_t}\right) \end{bmatrix} \\
&\approx -\frac{\mu}{r_t^3}\begin{bmatrix} r_t + x - 3x - \cancelto{0}{\frac{3x^2}{r_t}} \\ y - \cancelto{0}{\frac{3xy}{r_t}} \\ z - \cancelto{0}{\frac{3xz}{r_t}} \end{bmatrix} \\
&\approx -\frac{\mu}{r_t^3}\begin{bmatrix} r_t - 2x \\ y \\ z \end{bmatrix}
\end{aligned}
\tag{A.33}
$$

As a result, the exact nonlinear equations of motion in Eqs. (A.27)-(A.29) can be simplified to the following form:

$$\ddot{x} - \ddot{\theta}y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 x - 2\frac{\mu}{r_t^3}x = 0 \tag{A.34}$$

$$\ddot{y} + \ddot{\theta}x + 2\dot{\theta}\dot{x} - \dot{\theta}^2 y + \frac{\mu}{r_t^3}y = 0 \tag{A.35}$$

$$\ddot{z} + \frac{\mu}{r_t^3}z = 0 \tag{A.36}$$

Assumption 2

The orbit of the target spacecraft is circular.

If the target orbit is circular, then the rotation rate of the reference frame is constant, $\ddot{\theta} = 0$, and equal to the mean orbital motion $\dot{\theta} = n = \sqrt{\frac{\mu}{r_t^3}}$. With this assumption,

Eqs. (A.34)-(A.36) are simplified to:

$$\ddot{x} - 3n^2x - 2n\dot{y} = 0 \tag{A.37}$$

$$\ddot{y} + 2n\dot{x} = 0 \tag{A.38}$$

$$\ddot{z} + n^2z = 0 \tag{A.39}$$

These are the linearized equations of relative motion, also referred to as Hill's equations.

Appendix B

Algorithms

This section provides the mathematical algorithms used in Fast MPC, namely, forward and backward substitution and Cholesky factorization.

B.1 Forward Substitution Algorithm

Forward substitution is used to solve the linear system $\mathbf{Lx} = \mathbf{b}$ where $\mathbf{L}$ is a lower triangular matrix. This equation can be written as:

$$\begin{bmatrix} \ell_{11} & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \dots & \ell_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{B.1}$$

Through expanding this equation the following relationships are determined:

$$x_1 = \frac{b_1}{\ell_{11}} \tag{B.2}$$

$$x_2 = \frac{b_2 - \ell_{21} x_1}{\ell_{22}} \tag{B.3}$$

$$\vdots$$

$$x_n = \frac{b_n - \sum_{j=1}^{n-1} \ell_{nj} x_j}{\ell_{nn}} \tag{B.4}$$

For $\mathbf{x}$ and $\mathbf{b}$ with more than one column, the formulation is very similar:

$$\begin{bmatrix} \ell_{11} & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \dots & \ell_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} \\ x_{21} & x_{22} & \dots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1j} \\ b_{21} & b_{22} & \dots & b_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nj} \end{bmatrix} \tag{B.5}$$

The general relationships are:

$$x_{1j} = \frac{b_{1j}}{\ell_{11}} \tag{B.6}$$

$$x_{2j} = \frac{b_{2j} - \ell_{21} x_{1j}}{\ell_{22}} \tag{B.7}$$

$$\vdots$$

$$x_{nj} = \frac{b_{nj} - \sum_{i=1}^{n-1} \ell_{ni} x_{ij}}{\ell_{nn}} \tag{B.8}$$

B.2 Backward Substitution Algorithm

Backward substitution is used to solve the linear system $\mathbf{Ux} = \mathbf{b}$ where $\mathbf{U}$ is an upper triangular matrix. This equation can be written as:

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{B.9}$$

Through expanding this equation the following relationships are determined:

$$x_n = \frac{b_n}{u_{nn}} \tag{B.10}$$

$$x_{n-1} = \frac{b_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}} \tag{B.11}$$

$$\vdots$$

$$x_1 = \frac{b_1 - \sum_{j=2}^{n} u_{1j}x_j}{u_{11}} \tag{B.12}$$

Similarly, for larger dimension $\mathbf{x}$ and $\mathbf{b}$ the formulation is written as:

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} \\ x_{21} & x_{22} & \dots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1j} \\ b_{21} & b_{22} & \dots & b_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nj} \end{bmatrix} \tag{B.13}$$

The general relationships are:

$$x_{nj} = \frac{b_{nj}}{u_{nn}} \tag{B.14}$$

$$x_{n-1,j} = \frac{b_{n-1,j} - u_{n-1,n}x_{nj}}{u_{n-1,n-1}} \tag{B.15}$$

$$\vdots$$

$$x_{1j} = \frac{b_{1j} - \sum_{i=2}^{n} u_{1i}x_{ij}}{u_{11}} \tag{B.16}$$

B.3 Cholesky Factorization

The idea behind Cholesky factorization is to find a lower triangular matrix $\mathbf{L}$ such that $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. Now, writing this out in matrices, the lower triangular matrix is defined as

$$\mathbf{L} \triangleq \begin{bmatrix} \lambda_{11} & 0 \\ \ell_{21} & \mathbf{L}_{22} \end{bmatrix}, \tag{B.17}$$

where λ_{11} is a scalar, ℓ_{21} is a column vector, and $\mathbf{L}_{22}$ is a matrix of one lesser dimension in each axis of the original matrix. The matrix $\mathbf{A}$ is defined similarly,

$$\mathbf{A} \triangleq \begin{bmatrix} \alpha_{11} & 0 \\ a_{21} & \mathbf{A}_{22} \end{bmatrix}. \tag{B.18}$$

Now, the relation can be expanded such that:

$$\begin{aligned} \mathbf{A} &= \mathbf{L}\mathbf{L}^T \\ \begin{bmatrix} \alpha_{11} & 0 \\ a_{21} & \mathbf{A}_{22} \end{bmatrix} &= \begin{bmatrix} \lambda_{11} & 0 \\ \ell_{21} & \mathbf{L}_{22} \end{bmatrix} \begin{bmatrix} \lambda_{11} & \ell_{21}^T \\ 0 & \mathbf{L}_{22}^T \end{bmatrix} \\ \begin{bmatrix} \alpha_{11} & 0 \\ a_{21} & \mathbf{A}_{22} \end{bmatrix} &= \begin{bmatrix} \lambda_{11}^2 & \lambda_{11}\ell_{21}^T \\ \ell_{21}\lambda_{11} & \ell_{21}\ell_{21}^T + \mathbf{L}_{22}\mathbf{L}_{22}^T \end{bmatrix} \end{aligned} \tag{B.19}$$

Looking at the components, the entries of the lower triangular matrix can be calculated as

$$\lambda_{11} = \sqrt{\alpha_{11}}, \tag{B.20}$$

$$\ell_{21} = \frac{a_{21}}{\lambda_{11}}, \tag{B.21}$$

$$\mathbf{L}_{22}\mathbf{L}_{22}^T = \mathbf{A}_{22} - \ell_{21}\ell_{21}^T, \tag{B.22}$$

from which another Cholesky factorization is performed on the submatrix $\mathbf{L}_{22}$ and so on until the size of the submatrix reduces to a scalar, i.e., the last entry of the lower triangular matrix.

B.4 Cholesky Factorization of Block Diagonal

The Cholesky factorization of a block tridiagonal matrix, as is the form of the Schur complement in Fast MPC, can be computed using the following formulation. Let

$\mathbf{Y} \triangleq \mathbf{L}\mathbf{L}^T$ where

$$\mathbf{Y} \triangleq \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} & 0 & \dots & 0 & 0 \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} & \mathbf{Y}_{23} & \dots & 0 & 0 \\ 0 & \mathbf{Y}_{32} & \mathbf{Y}_{33} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0\dots & \mathbf{Y}_{N-1,N-1} & \mathbf{Y}_{N-1,N} & \\ 0 & 0 & 0\dots & \mathbf{Y}_{N,N-1} & \mathbf{Y}_{NN} & \end{bmatrix} \tag{B.23}$$

and

$$\mathbf{L} \triangleq \begin{bmatrix} \mathbf{L}_{11} & 0 & 0 & \dots & 0 & 0 \\ \mathbf{L}_{21} & \mathbf{L}_{22} & 0 & \dots & 0 & 0 \\ 0 & \mathbf{L}_{32} & \mathbf{L}_{33} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0\dots & \mathbf{L}_{N-1,N-1} & 0 & \\ 0 & 0 & 0\dots & \mathbf{L}_{N,N-1} & \mathbf{L}_{NN} & \end{bmatrix} \tag{B.24}$$

such that the relation can be written as

$$
\begin{bmatrix}
\mathbf{Y}_{11} & \mathbf{Y}_{12} & 0 & \dots & 0 & 0 \\
\mathbf{Y}_{21} & \mathbf{Y}_{22} & \mathbf{Y}_{23} & \dots & 0 & 0 \\
0 & \mathbf{Y}_{32} & \mathbf{Y}_{33} & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 \dots & \mathbf{Y}_{N-1,N-1} & \mathbf{Y}_{N-1,N} & \\
0 & 0 & 0 \dots & \mathbf{Y}_{N,N-1} & \mathbf{Y}_{NN} &
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{L}_{11} & 0 & 0 & \dots & 0 & 0 \\
\mathbf{L}_{21} & \mathbf{L}_{22} & 0 & \dots & 0 & 0 \\
0 & \mathbf{L}_{32} & \mathbf{L}_{33} & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \dots & \mathbf{L}_{N-1,N-1} & 0 \\
0 & 0 & 0 & \dots & \mathbf{L}_{N,N-1} & \mathbf{L}_{NN}
\end{bmatrix}
\begin{bmatrix}
\mathbf{L}_{11}^T & \mathbf{L}_{21}^T & 0 & \dots & 0 & 0 \\
0 & \mathbf{L}_{22}^T & \mathbf{L}_{32}^T & \dots & 0 & 0 \\
0 & 0 & \mathbf{L}_{33}^T & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \dots & \mathbf{L}_{N-1,N-1}^T & \mathbf{L}_{N,N-1}^T \\
0 & 0 & 0 & \dots & 0 & \mathbf{L}_{NN}^T
\end{bmatrix}. \tag{B.25}
$$

From this expression the following relationships are determined

$$\mathbf{L}_{11}\mathbf{L}_{11}^T = \mathbf{Y}_{11}, \tag{B.26}$$

$$\mathbf{L}_{ii}\mathbf{L}_{i+1,i}^T = \mathbf{Y}_{i,i+1}, \tag{B.27}$$

$$\mathbf{L}_{ii}\mathbf{L}_{ii}^T = \mathbf{Y}_{ii} - \mathbf{L}_{i,i-1}\mathbf{L}_{i,i-1}^T, \tag{B.28}$$

which simply requires Cholesky factorization of the submatrices within $\mathbf{Y}$.

Appendix C

Build of BLUE

This section presents images and models related to the construction of the third platform, BLUE, for the Spacecraft Proximity Operations Testbed at Carleton University's Spacecraft Robotics and Control Laboratory. These images complement those shown in Kirk Hovell's Master's thesis [68].

The platforms are built from three modular decks stacked vertically on four metal rods. The bus deck (bottom) holds the plumbing, the avionics deck (middle) holds the electronics and control panel interface, and the sensor deck (top) provides available space for future components to be integrated with the system. Labelled images of the top and bottom view of the bus deck are shown in Figs. C.1 and C.2. A labelled image of the avionics deck is given in Fig. C.3. A front view of the control panel is given in Fig. C.4, which also shows how power is distributed to each deck of the platform.

A model of the breadboard circuit that controls the actuation of the air-bearings and thrusters is shown in Fig. C.5. The coloured data lines at the bottom of the figure originate from the GPIO pins of the Raspberry Pi 3 computer. The numbering of these pins for BLACK and BLUE is indicated on the electrical diagram in Fig. 7.2. The numbering of the thrusters is shown in Fig. C.6.

Detailed descriptions of the design and inner workings of each system of the platform is given in [68] as well as the laboratory's GitHub Wiki page[1].

[1] https://github.com/Carleton-SRCL/SPOT/wiki

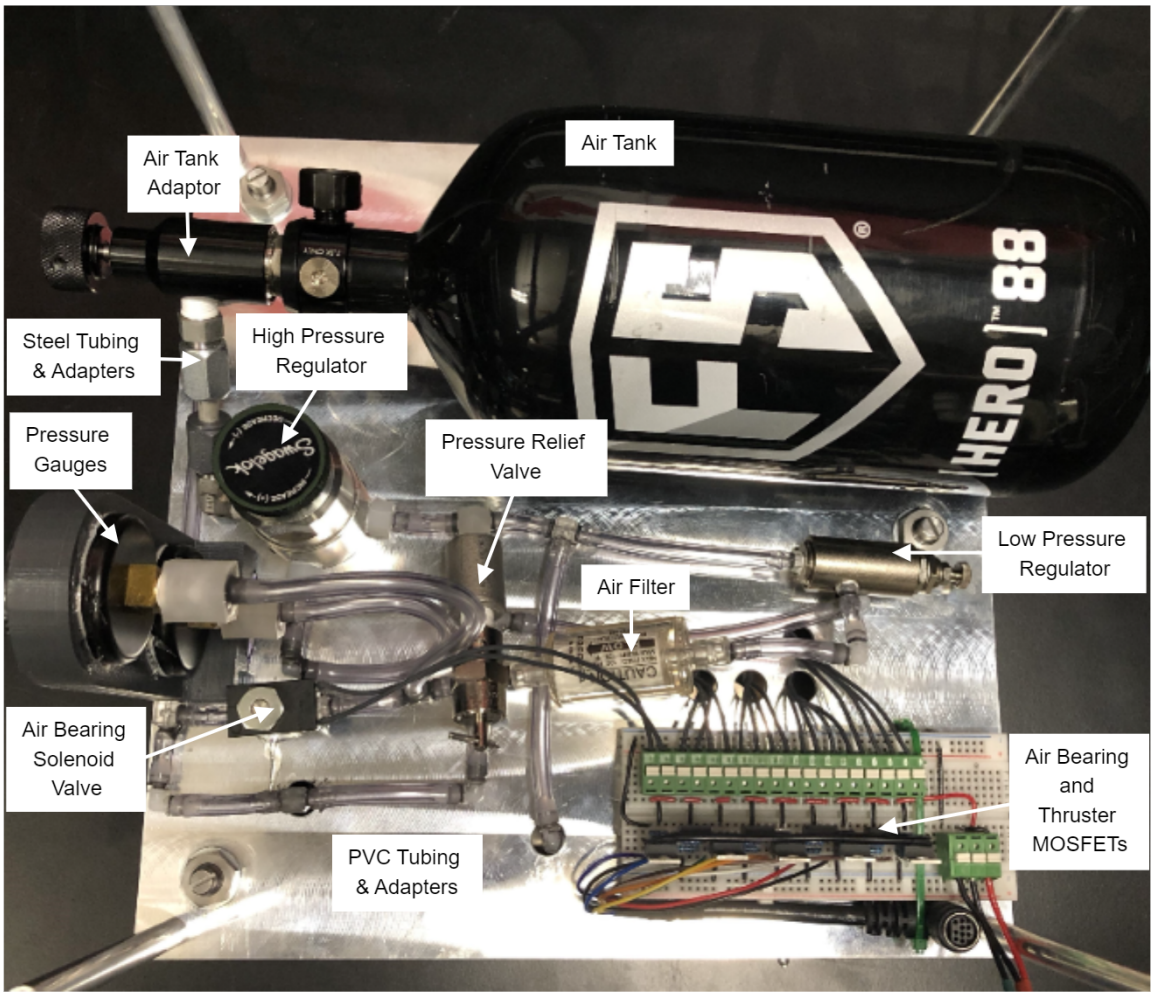


Figure C.1: The top of BLUE's bus deck, which holds the main pieces of plumbing: the air tank adapter, pressure regulators, pressure relief valve, air filter, and pressure gauges. It also holds the MOSFETs used to switch on the air bearings and thrusters.

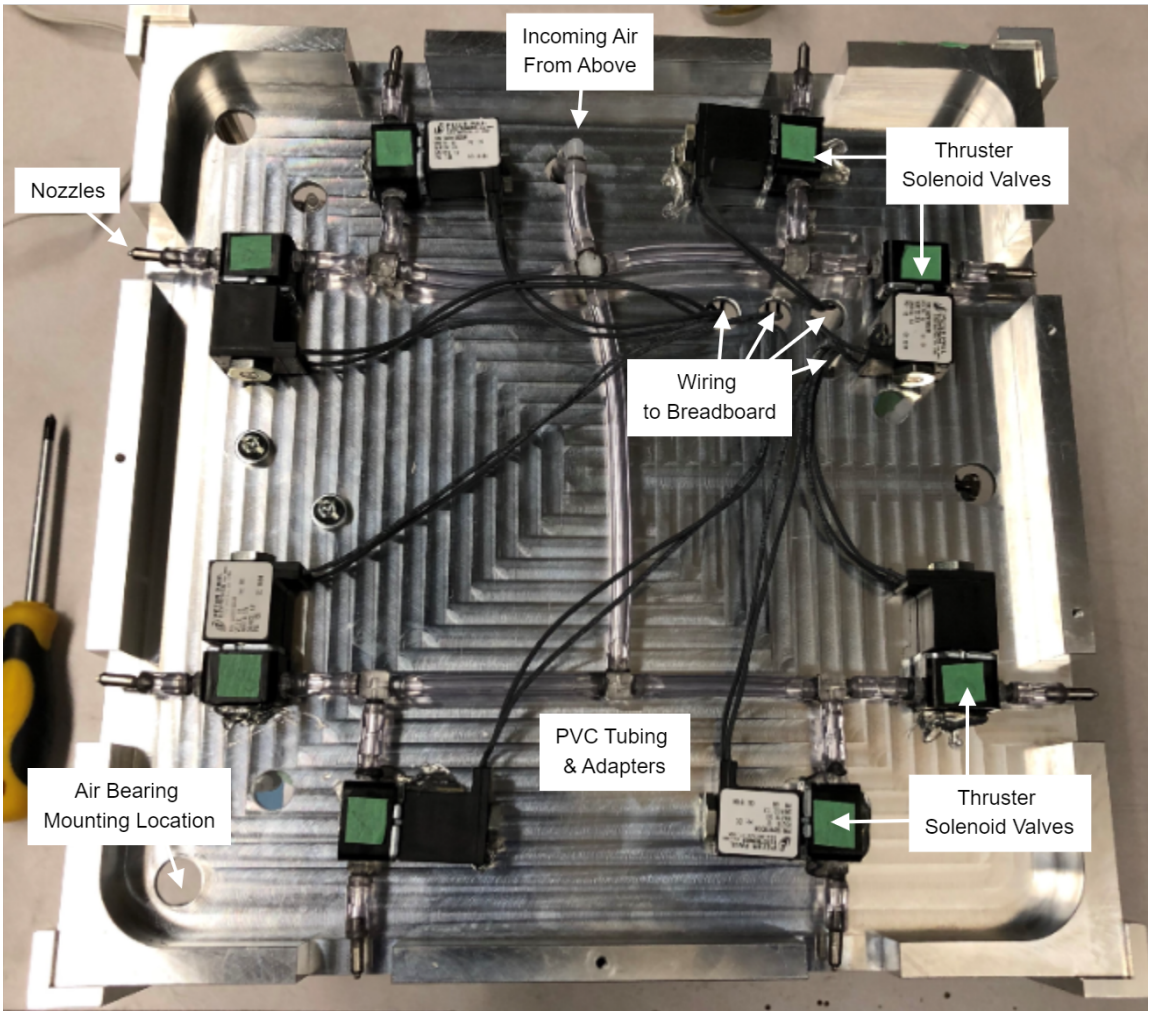


Figure C.2: The bottom of BLUE's bus deck, which holds the thruster solenoid valves and attached nozzles. All wiring feeds to the MOSFETs on top of the bus deck.

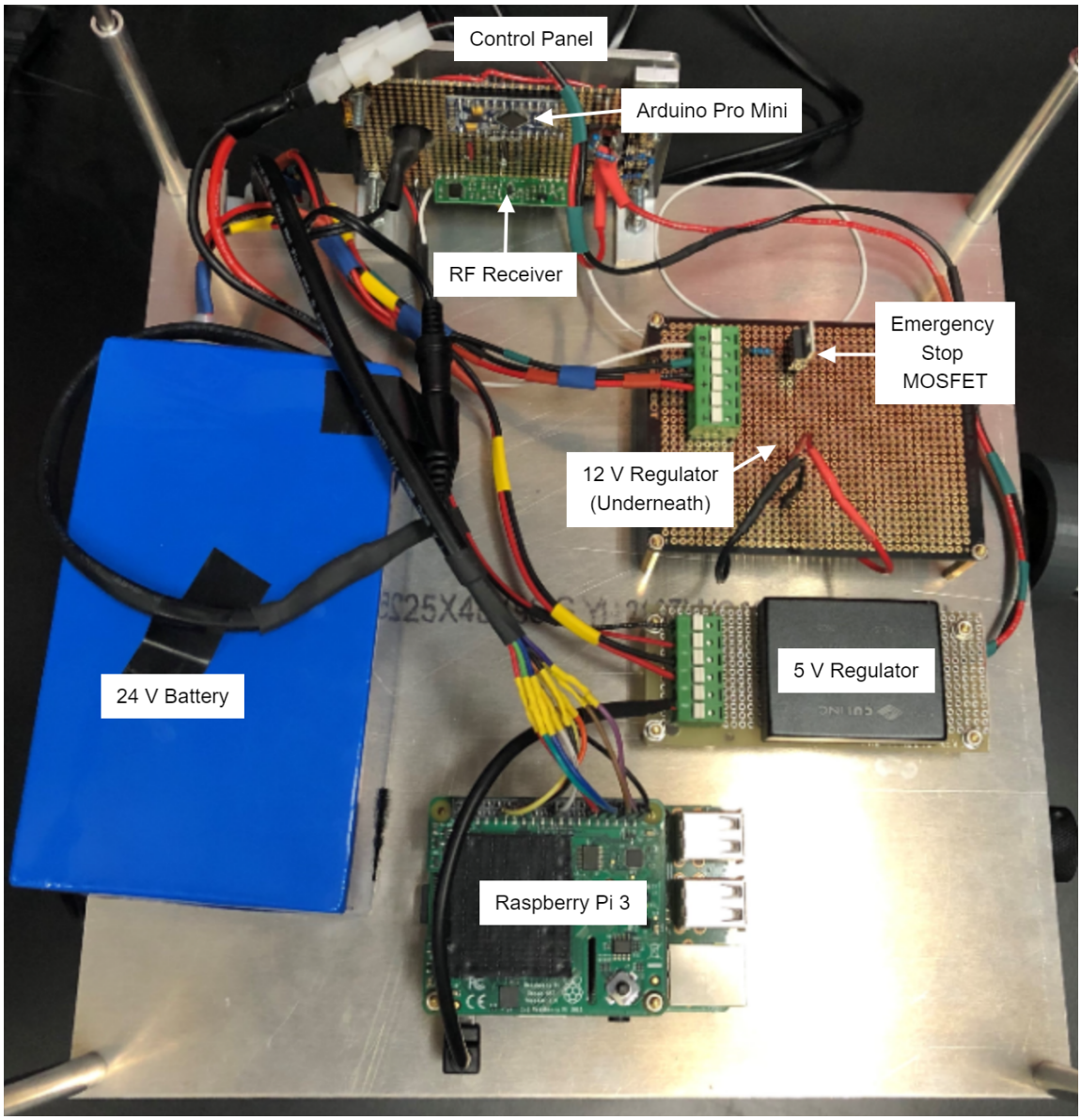


Figure C.3: BLUE's avionics deck, which holds the main electrical components: the control panel, 24 V battery, the 5 V and 12 V regulators, the main computer (Raspberry Pi 3 Model B), the emergency system computer (Arduino Pro Mini), and the circuitry for the emergency stop system.

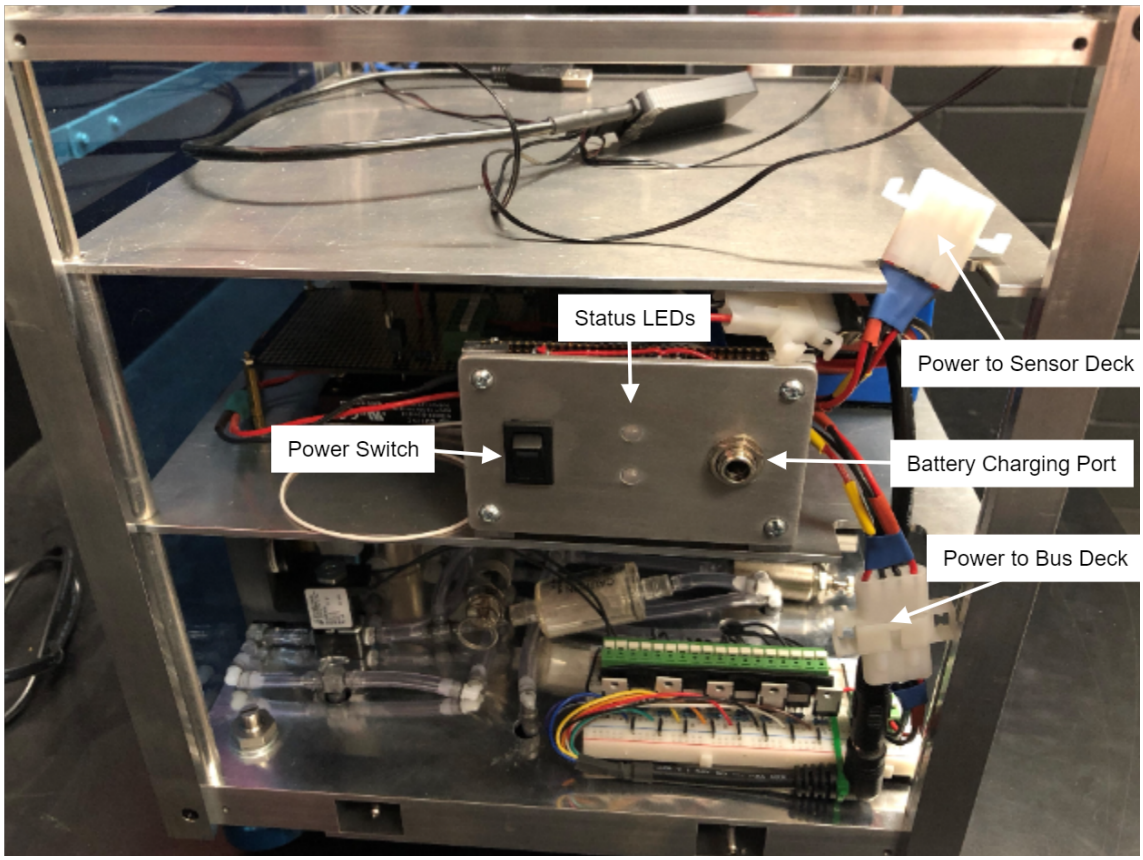


Figure C.4: BLUE's control panel, which allows the platform to be powered on/off, shows the status of the battery (top LED) and emergency stop system (bottom LED), as well as a charging port for the 24 V battery. The power lines to the bus deck and sensor deck are seen on the right side.

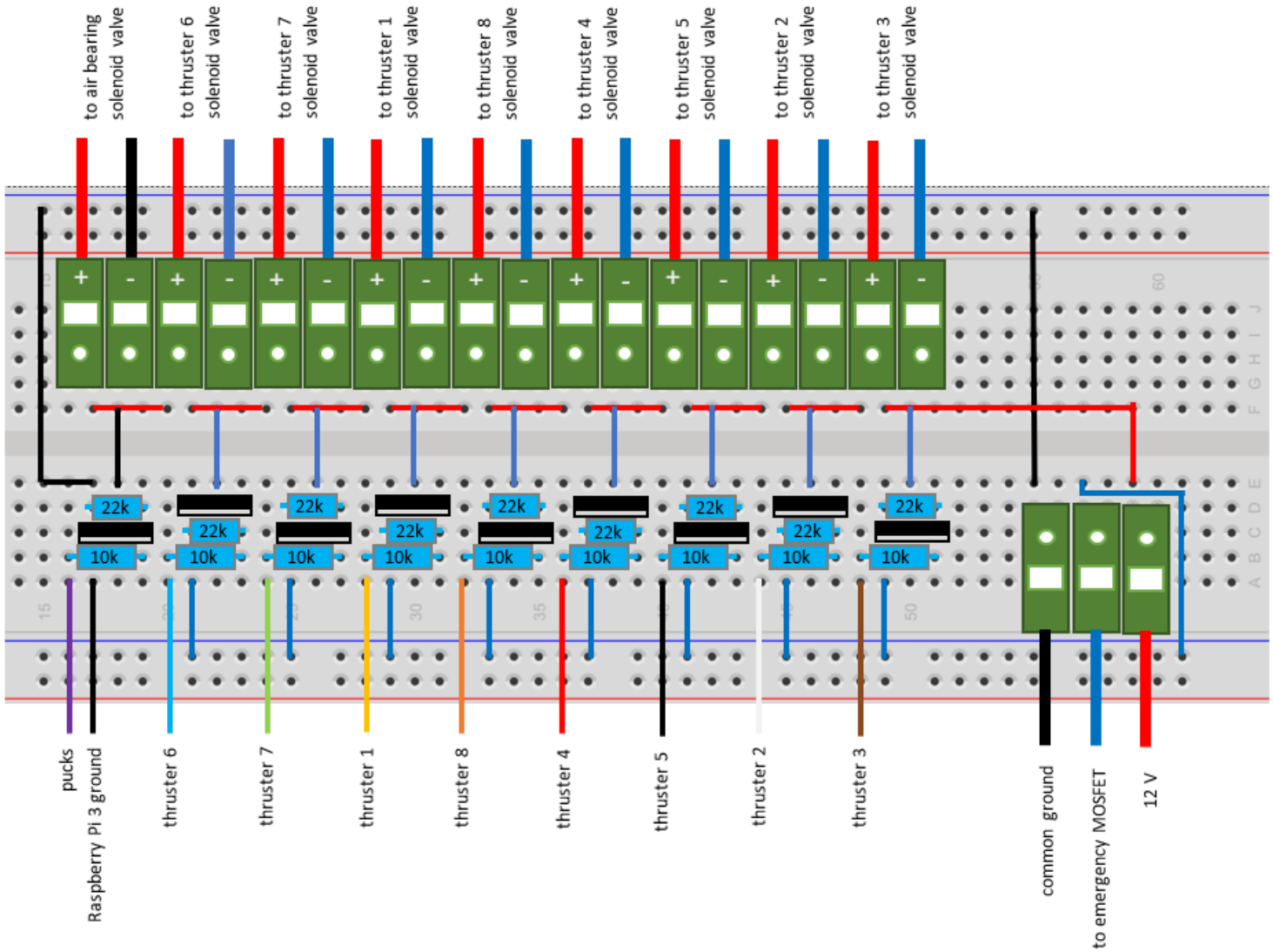


Figure C.5: A model of the breadboard circuit sitting on the bus deck that actuates the thruster and air-bearing solenoid valves using MOSFETs.

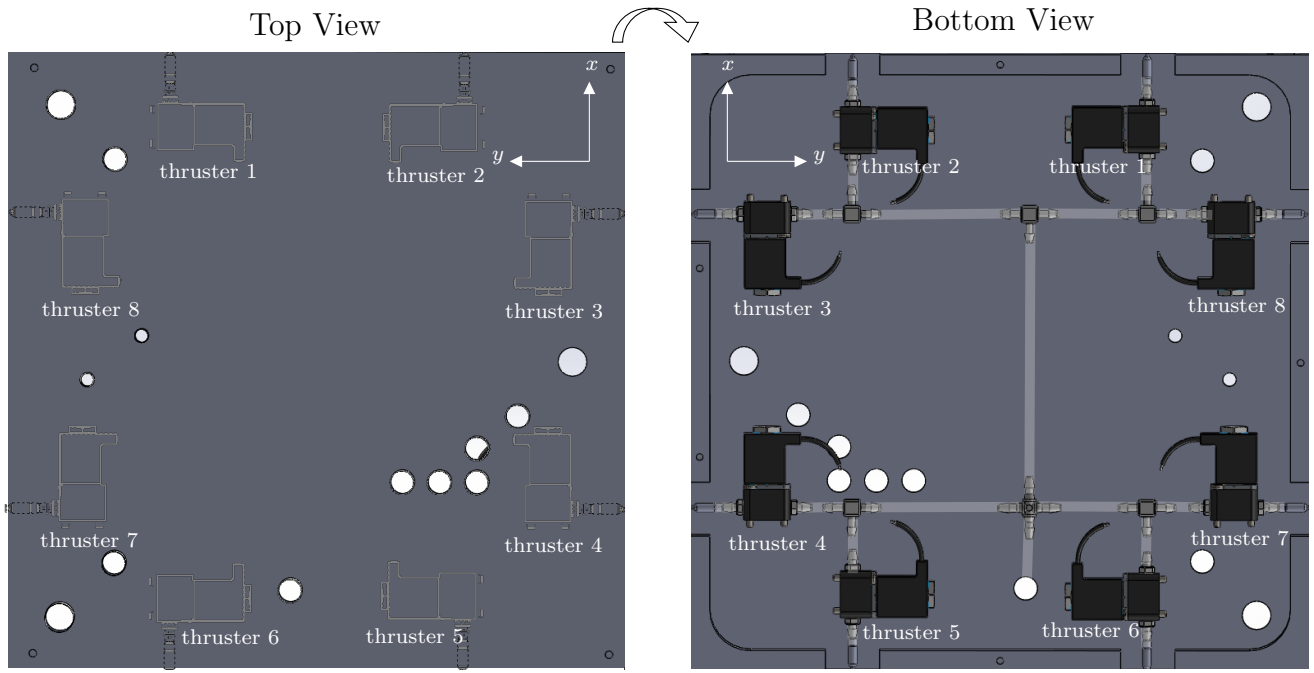


Figure C.6: A model of the bus deck showing the numbering of the thrusters relative to the xy-body axis of the platform.

COURTNEY BASHNICK received her B.Sc (Honours) degree with a double major in Astrophysics and Physics at the University of Calgary in 2020 where she received the Venkatesan Silver Medallion for achieving the highest academic standing in the Department of Physics and Astronomy. She has built solid and hybrid student sounding rockets and has additionally worked on the firmware development, laboratory testing, payload integration, and data analysis of a plasma scientific instrument flown on NASA sounding rocket missions. She has received funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) at the Undergraduate and Master levels in addition to several other scholarships. She will be attending Carleton University in the Fall of 2022 to pursue a Ph.D. degree in Aerospace Engineering.