Development of Control and Autonomy Algorithms for Docking to Complex Tumbling Satellites

by

Amer Fejzić

Bachelor of Science
University of Washington, 2006

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author . . . .
Department of Aeronautics and Astronautics
August 27, 2008

Certified by . . . .
David W. Miller
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by . . . .
Alvar Saenz-Otero
Research Scientist, Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Prof. David L. Darmofal
Associate Department Head
Chairman, Committee on Graduate Students

Development of Control and Autonomy Algorithms for Docking to Complex Tumbling Satellites

by

Amer Fejzić

Submitted to the Department of Aeronautics and Astronautics
on August 27, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The capability of automated rendezvous and docking is a key enabling technology for many government and commercial space programs. Future space systems will employ a high level of autonomy to acquire, repair, refuel, and reconfigure satellites. Several programs have demonstrated a subset of the necessary autonomous docking technology; however, none has demonstrated online path planning in-space necessary for safe automated docking. Particularly, when a docking mission is sent to service an uncooperative spacecraft that is freely tumbling. In order to safely maneuver about an uncontrolled satellite, an online trajectory planning algorithm with obstacle avoidance employed in a GN&C architecture is necessary.

The main research contributions of this thesis is the development of an efficient sub-optimal path planning algorithm coupled with an optimal feedback control law to successfully execute safe maneuvers for docking to tumbling satellites. First, an autonomous GN&C architecture is presented that divides the docking mission into four phases, each uniquely using the algorithms within to perform their objectives. For reasons of safety and fuel efficiency, a new sub-optimal spline-based trajectory planning algorithm with obstacle avoidance of the uncooperative spacecraft is presented. This algorithm is shown to be computationally efficient and computes desirable trajectories to a complex moving docking port of the tumbling spacecraft.

As a realistic space system includes external disturbances and noises in sensor measurement and control actuation, a closed-loop form of control is necessary to maneuver the spacecraft. Therefore, several optimal feedback control laws are developed to track a trajectory provided by the path planner. Performance requirements for the tracking controllers are defined for the case of two spacecraft docking. With these requirements, the selection of a controller is narrowed down to a phase-plane switching between LQR and servo-LQR control laws.

The autonomous GN&C architecture with the spline-based path planning algorithm and phase-plane controller is validated with simulations and hardware experiments using the Synchronized Position Hold Engage and Reorient Satellites (SPHERES) testbed aboard the International Space Station (ISS). Utilizing the unique space environment provided by the ISS, the experiment is the first in-space demonstration of an online path planning algorithm. Both the flight and simulation tests successfully validated the capabilities of the autonomous control system to dock to a complex tumbling satellite. The contributions in this thesis advance and validate a GN&C architecture that builds on a legacy in autonomous docking of spacecraft.

Thesis Supervisor: David W. Miller
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Alvar Saenz-Otero
Title: Research Scientist, Aeronautics and Astronautics

Acknowledgments

There several people that helped me throughout my graduate life and supported my research endeavors in this thesis.

First of all, I would like to thank Professor David Miller and Dr. Alvar Saenz-Otero for their guidance and support for me to do the research that I love. In addition, I would like to extend my gratitude to my fellow colleagues at the MIT Space Systems Laboratory: Christophe Mandy, Swati Mohan, Jacob Katz, Brent Tweddle, Christine Edwards, and Georges Aoude from the ACL. In particularly I thank the wisest research scientist I have known, Dr. Simon Nolet, for his many advices in research and life that has significantly helped me in achieving great research. Also to mention my two officemates, Andrzej Stewart and Jaime Ramirez for providing the in-depth discussion of the many ideas that came around. I thank the SPHERES team and Aurora Flight Sciences for the extraordinary opportunity to validate my algorithm in space. This was remarkable. Finally, my last special thank you goes to the Assistant Dean for Graduate Student Christopher Jones, for all the assistance he provided when the times where tough and for the MSRP program that helped bring into this wonderful institution. Thank you all.

Contents

1 Introduction **15**

1.1 Motivation . . . 16

1.2 Docking Scenarios . . . 16

1.3 Thesis approach . . . 22

2 Autonomous GN&C Architecture **25**

2.1 Previous GN&C Architecture . . . 25

2.2 GN&C Architecture Modules . . . 26

2.2.1 Algorithms of the GN&C Architecture . . . 27

2.2.2 Capabilities of Previous Algorithms . . . 31

2.2.3 Previous Mission & Vehicle Management Module . . . 31

2.3 Advancements in Module Algorithms . . . 37

2.3.1 Advancements in MVM Module . . . 39

2.3.2 Conclusion of Advancements . . . 42

2.4 Summary . . . 42

3 Trajectory Planning **45**

3.1 Path Planning Problem Formulation . . . 46

3.1.1 Optimal Path Planning Problem Formulation General . . . 50

3.2 Path Planning Problem Formulation for Docking . . . 51

3.2.1 Cost Functional for Docking . . . 52

3.2.2 State Transition Equation for Docking . . . 53

3.2.3 Terminal States for Docking . . . 55

3.2.4 Obstacles for Docking 63
3.2.5 Planning Problem Formulation for Docking Summary 65
3.3 Variational Technique to Optimal Path Planning 66
3.3.1 Euler-Lagrange Equations General 66
3.3.2 Euler-Lagrange Equations for Docking 69
3.4 Spline-Based Trajectory Planning Algorithm 80
3.5 Comparison of Trajectory Planning Algorithms 92
3.5.1 Docking to Fixed Target Facing Forwards 95
3.5.2 Docking to Fixed Rotating Target In-Plane 101
3.5.3 Docking to Fixed Coning Target Facing Backwards 109
3.5.4 Comparison Summary 115
3.6 Summary 115

4 Trajectory Tracking **117**
4.1 PD/PID Controllers 120
4.2 LQR Controller 126
4.3 Servo-LQR Controller 131
4.4 Phase-Plane LQR Controller 135
4.5 Summary 141

5 Simulation and Experimental Autonomous Docking **143**
5.1 Simulation Docking 147
5.1.1 Docking to Rotating Spacecraft Out-of-Plane 148
5.1.2 Docking to Coning Spacecraft backwards 153
5.2 Experimental Docking aboard the ISS 157
5.2.1 Docking to Fixed Non-Tumbling Spacecraft Facing Backwards 158
5.3 Summary 166

6 Conclusions and Recommendations **167**
6.1 Thesis Summary 167
6.2 Issues and Recommendations 168

List of Figures

1-1 Tumbling Dynamics of Target Spacecraft and its Docking Port Axis Motion 18

1-2 Docking Scenarios of a Tumbling Spacecraft 21

2-1 Previous GN&C Architecture for Autonomous Docking [10] 27

2-2 Glideslope Approach Velocity Profile [10] 30

2-3 Docking to a Fixed Target Satellite Facing Forward 32

2-4 Docking to a Rotating Target Satellite Out-of-Plane 35

2-5 Docking to a Coning Target Satellite Facing Backwards. 36

2-6 Docking to a Coning Target Satellite Facing Forward 36

2-7 Hierarchical Depiction of the GN&C Architecture for Autonomous . . 38

2-8 2D Example of Docking to a Rotating Target Satellite In-Plane . . . 40

2-9 Attitude Planning Logic for Autonomous Docking 41

3-1 State Planning with Differential Constraints 47

3-2 Infinite Feasible State Trajectories 48

3-3 Optimal State Trajectory 49

3-4 Optimal State Trajectory Satisfying Control and State Constraints . 51

3-5 Hill's Relative Equations of Motion 54

3-6 Hill's Equations and Double Integrator Bode Plots [12] 56

3-7 Docking Port Vector in Body and Global Coordinates 61

3-8 Example of the Transformation to the Final State of a Rotating. . . . 62

3-9 Modeling Obstacles for Docking of Two Spacecraft 64

3-10 Relative Distance Obstacle Cost Penalization 72

3-11 Relative Velocity Obstacle Cost Penalization 74

3-12 Obstacle Sphere and Way-Points Depiction 82
3-13 Process of the Spline-Based Planning Algorithm 83
3-14 Cubic Spline Interpolation Algorithm 86
3-15 Minimum Distance Along Trajectory to Obstacle 87
3-16 Introducing the First Waypoint 89
3-17 Docking Scenarios for Path Planner Comparison 94
3-18 State Trajectory of Docking to Fixed Target Facing Forwards 98
3-19 Control and Energy Profile for Docking to Fixed Target Facing 99
3-20 3D Trajectory of Docking to Fixed Target Facing Forwards 100
3-21 Invalid 3D Trajectory of Docking to Fixed Rotating Target In-Plane . 102
3-22 State Trajectory of Docking to Fixed Rotating Target In-Plane 106
3-23 Control and Energy Profile for Docking to Fixed Rotating Target. . . 107
3-24 3D Trajectory of Docking to Fixed Rotating Target In-Plane 108
3-25 State Trajectory of Docking to Fixed Coning Target Facing Backwards 112
3-26 Control and Energy Profile for Docking to Fixed Coning Target . . . 113
3-27 3D Trajectory of Docking to Fixed Coning Target Facing Backwards . 114

4-1 Maximum Tracking Error Dependent On The Closest Distance 120
4-2 PD Controller Performance 123
4-3 PID Controller Performance 125
4-4 LQR Controller Performance 130
4-5 Servo-LQR Controller Performance 134
4-6 Phase-Plane LQR Controller 136
4-7 Phase-Plane LQR Controller Performance 139

5-1 Block Diagram of Trajectory Planning, Control, and Estimation . . . 146
5-2 Planned and Actual State Trajectories for Docking to Rotating 150
5-3 State Differences for Docking to Rotating Target Out-of-Plane 151
5-4 3D State Trajectories for Docking to Rotating Target Out-of-Plane . 152
5-5 Planned and Actual State Trajectories for Docking to Coning 154
5-6 State Differences for Docking to Coning Target Facing Backwards . . 155

5-7 3D State Trajectories for Docking to Coning Target Facing Backwards 156
5-8 SPHERES testbed aboard the ISS 157
5-9 State Estimates of Chaser Spacecraft from Experimental Test of . . . 162
5-10 State Estimates of Target Spacecraft from Experimental Test of . . . 163
5-11 3D Plot of Computed and Actual Trajectories of Chaser Spacecraft . 164
5-12 State Differences between Both Spacecraft from Experimental Test . 165

List of Tables

2.1 MVM phases for docking to fixed target. 33
2.2 MVM phases for docking to rotating target. 37
2.3 MVM phases for any docking scenario. 39
2.4 Attitude planning. 41

3.1 Summary of Trajectory Planning Problem Formulation for Spacecraft 65
3.2 Variational Technique to Optimal Trajectory Planning Algorithm . . 79
3.3 Docking to Fixed Target Facing Forwards Scenario Planning Inputs. . 96
3.4 Energy Cost and Computation Time for Fixed Target Facing 97
3.5 Docking to Fixed Rotating Target In-Plane Planning Inputs. 103
3.6 Energy Cost and Computation Time for Fixed Rotating Target . . . 104
3.7 Docking to Fixed Coning Target Facing Backwards Planning Inputs. . 110
3.8 Energy Cost and Computation Time for Fixed Coning Target Facing 111

4.1 PD/PID Controller Gains Selection. 122
4.2 PD/PID Controller Performance Summary. 124
4.3 LQR Controller Performance Summary. 129
4.4 Servo-LQR Controller Performance Summary. 132
4.5 Phase-Plane LQR Controller Performance Summary. 140

5.1 MVM maneuvers for any docking scenario. 145
5.2 Parameters for DP Axis Alignment and Inline Approach phases . . . 147
5.3 Parameters for DP Axis Alignment and Inline Approach phases for . 159

Chapter 1

Introduction

The first successful docking of two spacecraft was performed on March 16, 1966, when the Gemini 8 capsule docked to an Agena Target Vehicle. To this date, most of the spacecraft docking relies on the same methods performed 40 years ago. This includes having the on board astronauts manually control the last executions of a docking maneuver. This thesis contributes to the current endeavor to supersede this method with an autonomous on board solution that requires little or no human-in-the-loop supervision. In addition, the GN&C architecture and algorithms presented focus on docking scenarios to a tumbling satellite. This refers to a spacecraft that lost control authority about at least one of its axes and so may be tumbling in free space. Missions such as servicing damaged satellites fall under this category. The work focuses on the "terminal" phase of a docking mission, this refers to the last 100 meters [10] before maneuvers are executed for physical mating of the docking ports. It continues off from a GN&C architecture developed by Nolet [10] and presents new algorithms that provide the autonomous system the ability to consider obstacles and dock from any initial configuration of the two spacecraft. The new autonomous control system is tested in hardware on the Synchronized Position Hold Engage and Reorient Satellites (SPHERES) [11] testbed aboard the International Space Station (ISS). The experimental test demonstrated the first in-space online path planning algorithm developed in this thesis. The contributions advance and validate a GN&C architecture that builds on a legacy in autonomous docking of spacecraft.

1.1 Motivation

The capability of automated rendezvous and docking is a key enabling technology for many government and commercial space programs [21, 9, 13]. Future space systems will employ a high level of autonomy to acquire, repair, refuel, and reconfigure satellites. Several programs have demonstrated a subset of the necessary autonomous docking technology; however, none has demonstrated online path planning in-space necessary for safe automated docking. Particularly, when a docking mission is sent to service an uncooperative spacecraft that is freely tumbling.

DARPA's Orbital Express Advanced Technology Demonstration [**?**]orbital) is the most autonomous system tested in-space to this date. It demonstrated technologies for autonomous docking to cooperative satellites, such as close proximity maneuvering, sensor technology, and automatic robotic capture. However, the mission did not employ any online path planning with collision avoidance and was used only on a non-tumbling spacecraft. A mission in which a spacecraft would tumble is if a spacecraft was damaged and lost control authority of its attitude stabilization. The external disturbances in space would initiate a tumble on the satellite. If a servicing mission is desired for repair, an autonomous technology that can avoid obstacles is necessary.

1.2 Docking Scenarios

In order to build the appropriate algorithms for a GN&C architecture, an understanding of the potential docking scenarios of a tumbling spacecraft is necessary. A docking scenario is composed of two parts. First is the motion of the docking port (DP), which is dependent on the tumbling dynamics of the spacecraft. The second is the initial configuration of the two spacecraft, their relative position and attitude. Before continuing, a docking terminology is defined. The vehicle that will intentionally execute the maneuvers necessary to perform the docking approach to another satellite is referred to as the chaser spacecraft. The vehicle the *chaser* will approach and dock to is referred to as the target *spacecraft*.

It is assumed that both spacecraft have a docking port attached rigidly to the vehicles. Therefore, the targets' docking port does not move with respect to the local body axes of the satellite. However, it does move with respect to the chaser and depends on the tumbling dynamics of the target spacecraft. There are two dynamics of the target spacecraft considered in this thesis:

Non-Tumbling The target vehicle holds its attitude throughout the complete docking scenario. The vehicles' angular rotation vector is zero.

Rotational Tumble The target spacecraft performs a steady rotation about its angular rate vector. Its inertia is assumed to be symmetric so no nutation occurs in its attitude dynamics.

Next, the motion of the target spacecraft docking port is considered from the described dynamics. While the target spacecraft is not tumbling, its docking port is fixed and will only translate along with the spacecraft. This is the simplest motion of the docking port. For when the target is performing a rotational tumble there are two motions for the docking port and is dependent on the angular rate vector with respect to the dock port axis. If the angular rate vector is perpendicular to the DP axis, then the docking port performs a circular motion that sweeps a plane going through the center of the target spacecraft, see Figure 1-1. Any other direction of the angular rate vector has the DP port sweep a plane that does not pass through the centroid, see Figure 1-1. In this case, the docking port *axis* sweeps a space cone. This will be considered *Coning*.

Until now, only the target spacecraft is considered. Next we include the chaser spacecraft in the picture and define the possible initial configurations that may occur at the start of the terminal phase of a docking mission. Let's assume that the arriving chaser spacecraft comes in with its docking port pointing towards the target. There will of course be an initial relative position between the two spacecraft. Their magnitudes will not be considered as part of the different docking scenarios. The only state considered is the initial attitude of the target viewed with respect to the chaser spacecraft. There are two possibilities considered:

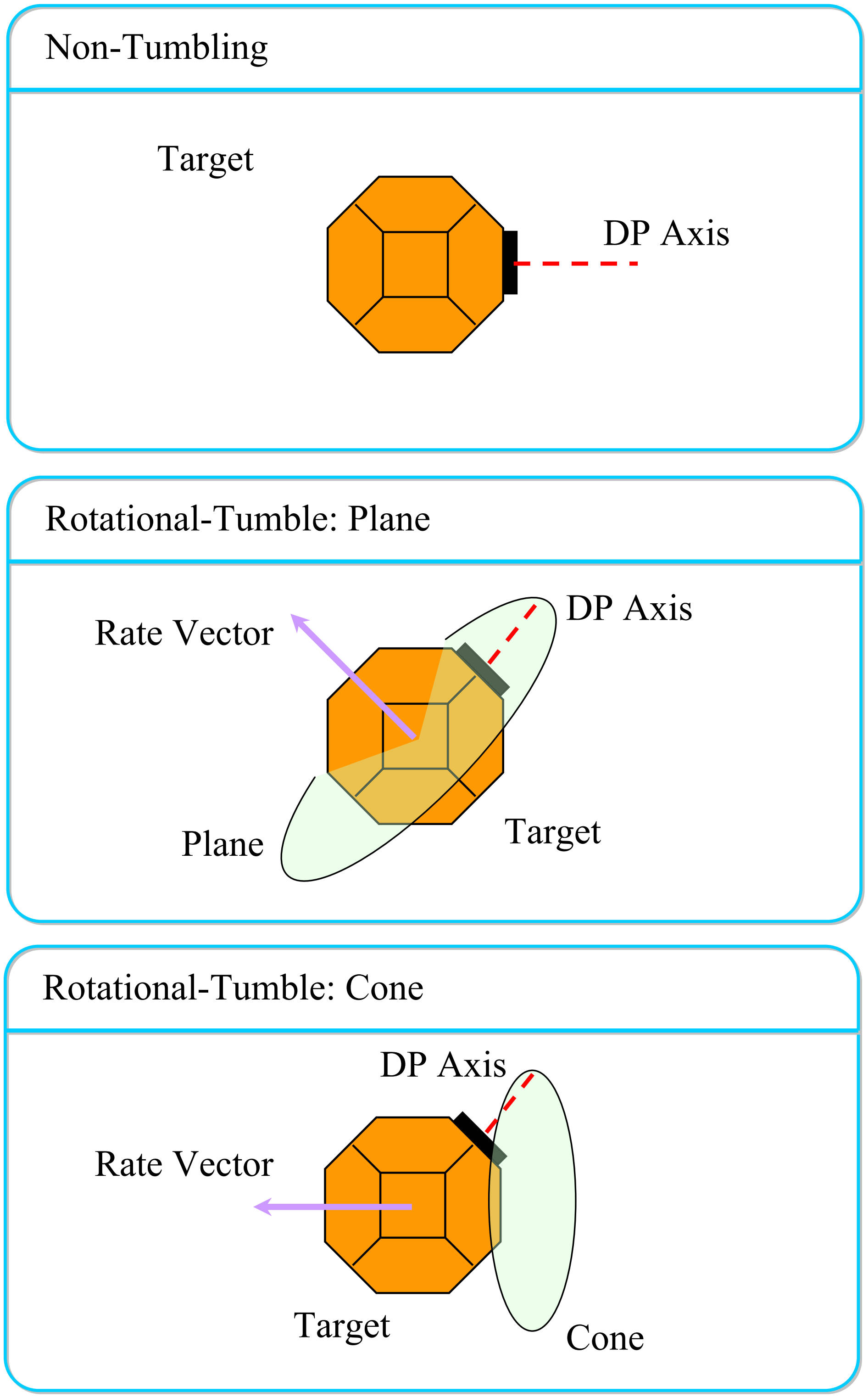


Figure 1-1: Tumbling Dynamics of Target Spacecraft and its Docking Port Axis Motion

Facing Forwards The target spacecraft is facing its docking port towards the chaser

Facing Backwards The target spacecraft DP is flipped 180° and facing away from the chaser.

Combining the two initial configurations and the possible motions of the targets' docking port leads to the docking scenarios of a tumbling spacecraft. It is also added that if the plane the DP axis sweeps has the chaser spacecraft also initially located, then it is referred to as *Rotating In-Plane*. Otherwise, it is known as *Rotating Out-of-Plane*. When the spacecraft is rotating, facing forwards or backwards is not important as that naturally changes with time, but is considered when the target is coning. The following docking scenarios are put together and depicted in Figure 1-2.

Docking to Fixed Non-Tumbling Target Facing Forwards This scenario has both spacecraft face each other for their initial configuration. The target spacecraft has zero angular rotation and is fixed in position and so the initial configuration stays constant throughout the docking scenario. This is the simplest case as the chaser needs to close in the gap linearly between the two vehicles.

Docking to Fixed Non-Tumbling Target Facing Backwards Here the target has its docking port facing away from the chaser. This will require the chaser spacecraft to maneuver around the target spacecraft to get in front of its docking port. Obstacle avoidance is necessary in this case.

Docking to Fixed Rotating Target In-Plane This scenario has the target spacecraft perform a steady rotation with its angular rate vector perpendicular to the DP axis. Therefore, the DP axis sweeps a plane. In addition, the initial location of the chaser spacecraft is constrained to be within this plane. In this scenario, the chaser needs to maneuver only along the plane and thus needs to consider only 2 dimensional motion. Also obstacle avoidance needs to be accounted for as the chaser maneuvers around to get in front of the docking port.

Docking to Fixed Rotating Target Out-of-Plane The target spacecraft rotates with an angular vector perpendicular to its DP axis while it sweeps a plane where the chaser spacecraft is not located. This requires the chaser to maneuver around in three translational degrees-of-freedom. Again, obstacle avoidance is necessary.

Docking to Fixed Coning Target Facing Forwards The targets angular rate vector is not perpendicular to the DP axis and thus the axis sweeps a space cone oriented to face towards the chaser spacecraft. This maneuver may not require obstacle avoidance as the chaser does not need to maneuver around the target.

Docking to Fixed Coning Target Facing Backwards Here, the targets' DP axis is sweeping a cone that is facing away from the chaser. This requires the chaser vehicle to maneuver around the target, obstacle avoidance, and align in front of the docking port matching the coning motion. This is most complex docking scenario considered in this thesis.

The autonomous control system presented in this thesis is developed to work for all the docking scenarios discussed. It is assumed that by proving the architecture to work on the most complicated scenarios, *Docking to Fixed Coning Target Facing Backwards* and *Docking to Fixed Rotating Target Out-of-Plane*, assures it would work on the others. The approach in developing the new autonomous GN&C architecture is discussed in the following section.

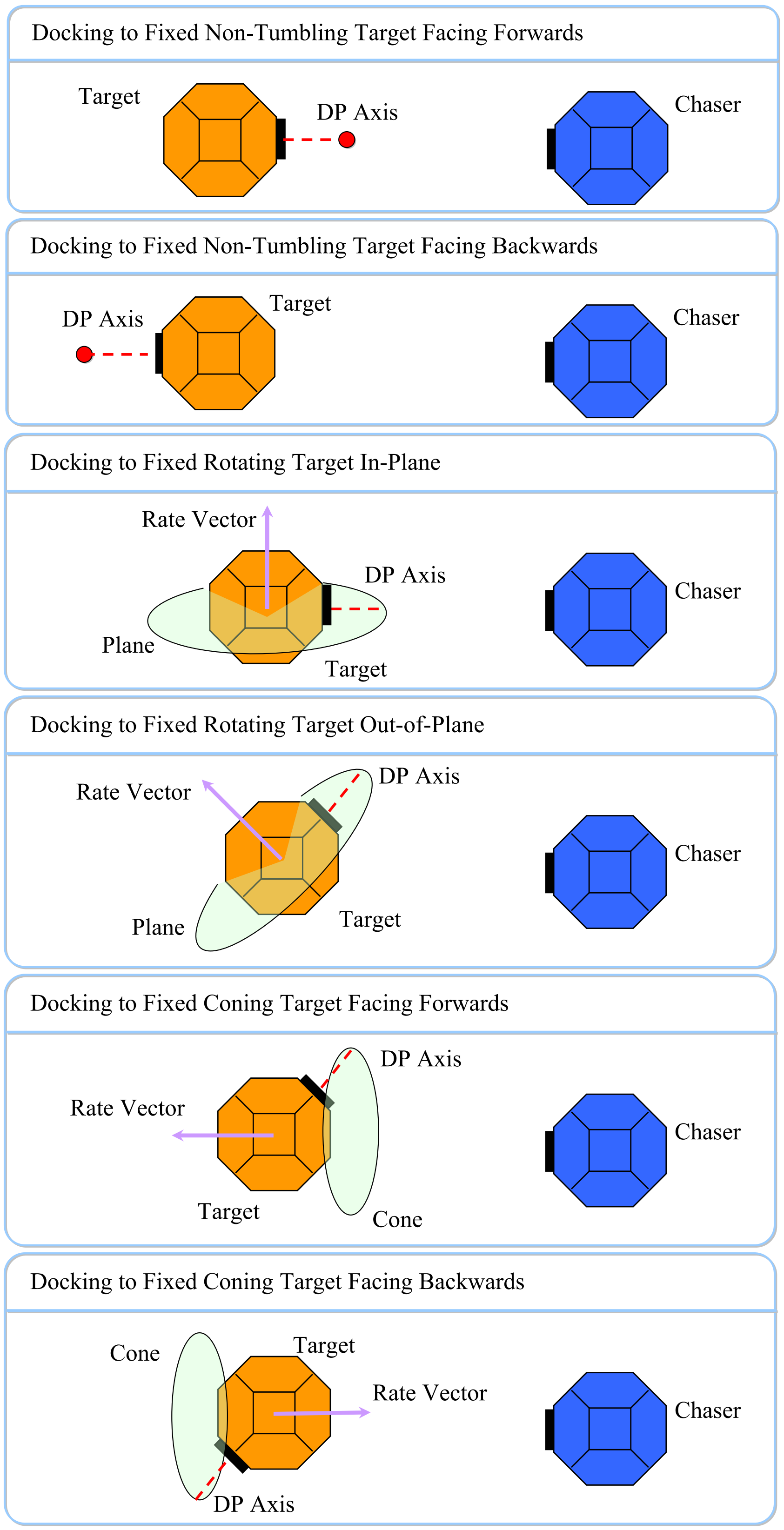

Docking to Fixed Non-Tumbling Target Facing Forwards
Target
DP Axis
Chaser
Docking to Fixed Non-Tumbling Target Facing Backwards
DP Axis
Target
Chaser
Docking to Fixed Rotating Target In-Plane
Rate Vector
DP Axis
Chaser
Plane
Target
Docking to Fixed Rotating Target Out-of-Plane
Rate Vector
DP Axis
Chaser
Plane
Target
Docking to Fixed Coning Target Facing Forwards
DP Axis
Rate Vector
Chaser
Target
Cone
Docking to Fixed Coning Target Facing Backwards
Cone
Target
Rate Vector
Chaser
DP Axis

Figure 1-2: Docking Scenarios of a Tumbling Spacecraft

1.3 Thesis approach

Chapter 2 lays down the higher level organization of the autonomous GN&C architecture. It introduces a previous architecture and the algorithms populating each of its modules. Then the flaws of using these algorithms is exploited for docking to tumbling spacecraft. Several solution methods are proposed that do not require a change in the previous algorithms; however, the most complicated docking scenario is not attainable. The necessary improvements to the solver module by introducing a new algorithm is discussed. Then four high level phases of a docking mission are presented to work for all the docking scenarios. This covers how a trajectory planning algorithm and tracking controllers are used to achieve these scenarios in simulation and experiment.

Chapter 3 presents the first in-space online trajectory planning algorithm. Two trajectory planning algorithms are developed, where one is used as a benchmark comparison for the new algorithm tested aboard the ISS. First, a general formulation of optimal planning is introduced. Then the specific dynamics and constraints for docking of two spacecraft is developed. An optimal control problem for docking with obstacle clearance is presented. Next, a calculus of variation technique is used to solve this problem by forming the first-order necessary Euler-Lagrange equations for optimality. Solving these equations is computationally expensive and this technique to planning shows undesirable characteristics for implementation. Therefore, a new sub-optimal spline-based trajectory planning algorithm is presented. It shows to be efficient and provides reasonable trajectories for docking. The two planners are compared to test the sub-optimality of the spline-based algorithm.

Chapter 4 investigates the performance of several introduced LQR tracking controllers: LQR, servo-LQR, and phase-plane LQR/servo-LQR. First, the performance requirements of the tracking controllers for docking purposes is defined. Then their performance is studied and compared as they are presented. Each controller has desirable and undesirable characteristics. The phase-plane controller attempts to bring together the positive characteristics of the LQR and servo-LQR controllers. This leads to the best performing tacking controller that is chosen to be coupled with the trajectory planning algorithm.

Chapter 5 combines the spline-based trajectory planning algorithm from Section 3.4 and the phase-plane LQR controller from Section 4.4 into the autonomous GN&C architecture from Chapter 2 for validating the ability to dock to tumbling spacecraft. Two simulations are studied for the two most complex docking scenarios, *Docking to Fixed Coning Target Facing Backwards* and *Docking to Fixed Rotating Target Out-of-Plane*. Then an experimental test using SPHERES aboard the ISS is discussed for a *Docking to Fixed Non-Tumbling Target Facing Backwards* scenario. This experiment tests the ability of the new spline-based planning algorithm, which is the first online path planner test in micro-gravity. The results show the need of a planner that includes obstacle avoidance and emphasizes the importance of an accurate tracking controller.

Chapter 6 summarizes the contributions of this research and presents recommendations for future work.

Chapter 2

Autonomous GN&C Architecture

In order for a spacecraft to determine its location, compute a path for docking, and execute the maneuver completely by itself, an autonomous GN&C architecture is necessary. The architecture defines the organization of how the hardware and software inter-connect and operate to achieve these objectives. It is decomposed into several modules where each have a specific function to accomplish. The most necessary functions are estimation, control, and actuation. The performance of each module is dependent on the algorithms that employ its function. In this chapter, an autonomous GN&C architecture is introduced for docking from previous work and its capabilities are expanded by upgrading the algorithms that populate the low performing modules.

2.1 Previous GN&C Architecture

This section summarizes a previously developed and implemented GN&C architecture for autonomous docking [10]. This architecture already achieved numerous docking scenarios, such as to fixed and tumbling spacecraft. However, it contains certain limitations dependent on the algorithms which populate the modules within. First, the autonomous GN&C architecture is summarized and the algorithms employed are discussed to determine the capabilities of the system for docking scenarios. It is found that with the previous algorithms, the architecture can work only on specialized cases of docking to tumbling satellites. These deficiencies are exploited and some approaches are discussed that can slightly expand its capabilities without changing the algorithms.

2.2 GN&C Architecture Modules

Fehse [1], first introduced a typical docking architecture in his book entitled *Automated Rendezvous and Docking of Spacecraft*. However, this architecture is aimed at traditional docking of spacecraft with dependency on human-in-the-loop supervision. In order to achieve fully autonomous docking, Nolet [10] extended the architecture with the inclusion of an autonomous fault detection, isolation, and recovery (FDIR) and solver module shown in Fig. 2-1. The grayed areas of the architecture in Figure 2-1 are common to Fehse, while the rest are extensions introduced by Nolet. A description of each module and its function is stated:

GN&C mode: estimation module This module receives data from hardware sensors and fuses them together through an estimation algorithm to determine the state of the system. The state refers to a representation of spacecraft position and attitude.

GN&C mode: control module The best estimated state from the estimation module is sent to the control module to be compared with a desired state of the system provided by the Mission & Vehicle Management (MVM) module. Then the module uses a control law algorithm to determine the appropriate actuation necessary to achieve the desired state.

Solver module This module executes the complex algorithms employed to determine a state trajectory with start and end states defined by the MVM module.

FDIR module The FDIR module is active at several levels and linked to multiple modules to autonomously asses any failure such as invalid state estimation from measurements. In case of failure, the FDIR module would execute a collision avoidance maneuver (CAM).

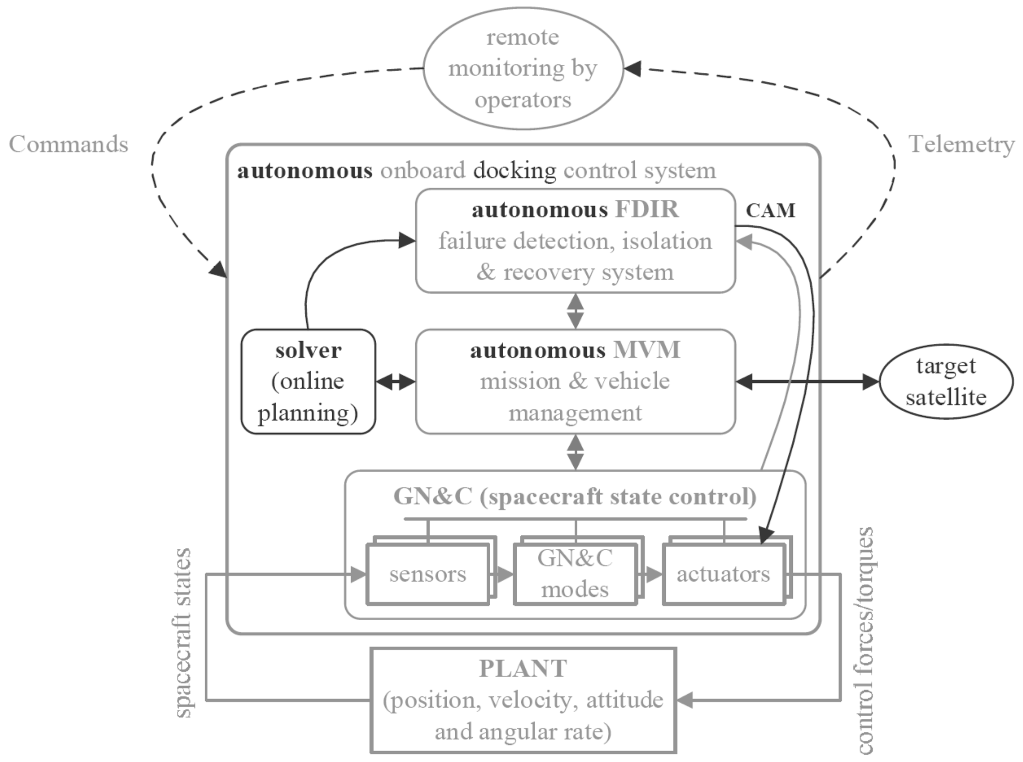

remote monitoring by operators
Commands
Telemetry
autonomous onboard docking control system
autonomous FDIR failure detection, isolation & recovery system
CAM
solver (online planning)
autonomous MVM mission & vehicle management
target satellite
GN&C (spacecraft state control)
sensors
GN&C modes
actuators
spacecraft states
control forces/torques
PLANT (position, velocity, attitude and angular rate)

Figure 2-1: Previous GN&C Architecture for Autonomous Docking [10]

MVM module This is the highest autonomy level module that manages the solver, FDIR, and GN&C modes to accomplish a mission objective such as docking to a spacecraft.

The architecture is well established to work for autonomous docking to complex tumbling satellites; however, the *capabilities are very limited by the algorithms employed in each module.* Next, the algorithms that populate the GN&C modes modules and solver module are reviewed to determine the architectures capabilities for docking to tumbling satellites.

2.2.1 Algorithms of the GN&C Architecture

The algorithms of the GN&C modes, state estimation and control modules, and the solver module is reviewed. The study reveals any insufficient abilities of each module to provide the required function for docking to tumbling spacecraft. The requirements are mentioned as the algorithms are reviewed.

The 6 degree-of-freedom (DOF) state of the spacecraft is described by its position $\mathbf{r}$, velocity $\mathbf{v}$, attitude $\mathbf{q}$, and angular rates $\boldsymbol{\omega}$:

$$\mathbf{x} = [r_x \quad r_y \quad r_z \quad v_x \quad v_y \quad v_z \quad q_1 \quad q_2 \quad q_3 \quad q_4 \quad \omega_x \quad \omega_y \quad \omega_z]^T \tag{2.1}$$

The state in Eq. (2.1) is with reference to an inertial coordinate system. One example is the Earth as a non-moving reference for an orbiting spacecraft. The unit vector quaternion $\mathbf{q}$ is used to describe the nonlinear attitude representation of the spacecraft due to its non-singular properties and ease of numerical maintenance. The quaternion describes a single rotation of amount θ of the global coordinate system about a unit normal eigenaxis $\mathbf{n} = [n_x \quad n_y \quad n_z]^T$. The resulting quaternion formulation is [17]:

$$\mathbf{q} = \left[n_x \sin\left(\frac{\theta}{2}\right) \quad n_y \sin\left(\frac{\theta}{2}\right) \quad n_z \sin\left(\frac{\theta}{2}\right) \quad \cos\left(\frac{\theta}{2}\right) \right]^T \tag{2.2}$$

Thus, Eq. (2.2) provides the attitude representation of the body axis of the spacecraft. This is a requirement for docking purposes as both position and attitude need to be controlled to successfully mate with another spacecraft.

Extended Kalman Filter Estimator

The ability to estimate the required state of the system is necessary for the spacecraft to know where to maneuver in order to dock. This is accomplished by the estimation module through the use of an Extended Kalman Filter (EKF) [10]. The estimator effective to nonlinear systems such as the attitude dynamics of the spacecraft. The approach of the algorithm is to propagate the system dynamics nonlinearly, but linearize at the current time step for the Kalman gain $\mathbf{K}_k$ calculation and state $\hat{\mathbf{x}}_k^{(+)}$ and covariance matrix $\mathbf{P}_k^{(+)}$ update. The Kalman gain weighs the trust in the estimator between the incoming sensor measurements and the model of the dynamics.

The EKF has been used extensively in the aerospace field and has gained confidence in attitude determination when the attitude is changing slowly compared to the rate of the filter. For docking scenarios to tumbling satellites, the EKF is sufficient at determining the full state of the system for docking purposes.

PID-type Controllers

For the control module of the GN&C modes, the standard PID-type controllers are employed. These controllers are widely used and almost a standard in the aerospace community. The two control algorithms that are employed in the control module are the proportional derivative (PD) and proportional integral derivative (PID) controllers. Each use the state error $\tilde{\mathbf{x}}$,

$$\tilde{\mathbf{x}} = \mathbf{x}_d - \mathbf{x} \tag{2.3}$$

the difference between the desired state $\mathbf{x}_d$ and current state $\mathbf{x}$, as an input to calculate the desired forces $\mathbf{f}$ and torques $\boldsymbol{\tau}$ commands that drive the state error to zero:

$$\mathbf{u} = \begin{bmatrix} f_x & f_y & f_z & \tau_x & \tau_y & \tau_z \end{bmatrix}^T \tag{2.4}$$

The controllers are decoupled for position and attitude control. The position control law is also decoupled from each axis and is of the form [10],

$$\mathbf{f} = \begin{bmatrix} K_P\tilde{r}_x + K_I\int\tilde{r}_x dt + K_D\tilde{v}_x \\ K_P\tilde{r}_y + K_I\int\tilde{r}_y dt + K_D\tilde{v}_y \\ K_P\tilde{r}_z + K_I\int\tilde{r}_z dt + K_D\tilde{v}_z \end{bmatrix} \tag{2.5}$$

where K_P, K_I, and K_D are the proportional, integral, and derivative gains. The torque commands $\boldsymbol{\tau}$ for attitude control are determined by a nonlinear-type PID controller of the form [10]:

$$\boldsymbol{\tau} = \begin{bmatrix} 2\cdot K_P\cdot sgn(\tilde{q}_4)\cdot q_1 + 2\cdot K_I\cdot\int(sgn(\tilde{q}_4)\cdot q_1)dt + K_D\cdot\tilde{\omega}_x \\ 2\cdot K_P\cdot sgn(\tilde{q}_4)\cdot q_2 + 2\cdot K_I\cdot\int(sgn(\tilde{q}_4)\cdot q_2)dt + K_D\cdot\tilde{\omega}_y \\ 2\cdot K_P\cdot sgn(\tilde{q}_4)\cdot q_3 + 2\cdot K_I\cdot\int(sgn(\tilde{q}_4)\cdot q_3)dt + K_D\cdot\tilde{\omega}_z \end{bmatrix} \tag{2.6}$$

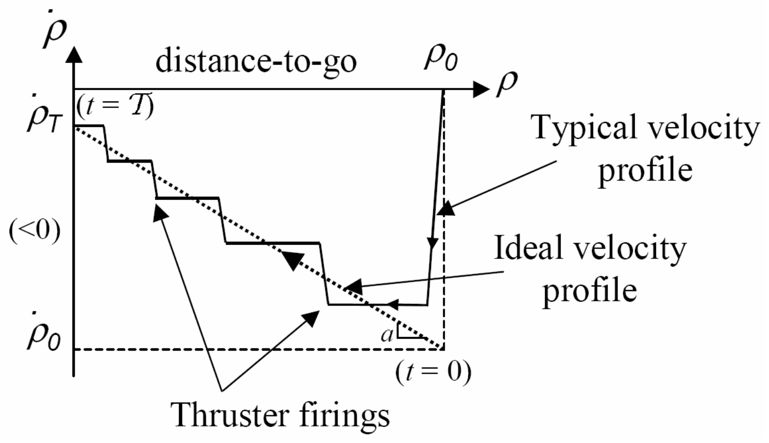


Figure 2-2: Glideslope Approach Velocity Profile [10]

The controller in Eq. (2.6) is extracted from Wie [20] who has shown that the PD version (when integral gain K_I is set to zero) is globally asymptotically stable. With the control laws ability to reach the desired states provided by the MVM module, they show no limiting capability towards the docking scenarios.

Glideslope Algorithm

The previous algorithm for the solver module of a "partial" path planner is done with the glideslope algorithm [10], which is a hybrid between a path planner and velocity controller. Therefore, the algorithm belongs partially to the *solver* and *control module* in the *GN&C modes* from Figure 2-1. The algorithm creates a velocity profile on a linear trajectory in the phase plane to follow by defining a safe *arrival velocity* ($\dot{\rho}_T$), *maneuver period*, and *number of thruster firings*. Figure 2-2 shows a velocity pattern ($\dot{\rho}$) that linearly decreases with distance-to-go (ρ).

The algorithm has been previously used in space operations (Apollo, Shuttle) and works well for a straight line approach along the docking axis. It does not account for any obstacles nor minimize fuel or energy as most other optimal path planners. There is a requirement for obstacle avoidance as stated in Section 1.2. As a result, this module does not fully perform its desired function for docking to tumbling spacecraft.

2.2.2 Capabilities of Previous Algorithms

Depending on the complexity of each algorithm in the modules depicted in Figure 2-1, certain limits arise in the satellite's capabilities to perform a complex tumbling docking scenario. From previous work [10], the lower level algorithms, EKF and PID controllers, allow the spacecraft to successfully estimate its state and maneuver a reference trajectory to within a sufficient accuracy. However, the glideslope algorithm "path planner" contains certain limitations that enable the spacecraft to perform only simplified versions of docking scenarios. As mentioned in Section 2.2.1, the algorithm computes a linear trajectory and does not account for any obstacles, such as the target satellite. Thus, the use of the glideslope algorithm works appropriately when the chaser spacecraft is aligned with the docking port (DP) axis of the target satellite. However, realistic scenarios do not occur with a specific initial configuration of the two spacecraft before docking. Therefore, the *solver module* is further developed in this thesis to extend the autonomous GN&C architecture capabilities for more realistic docking scenarios. From the limiting capabilities of the previous algorithm, the *Mission & Vehicle Management module* (MVM) can utilize the *GN&C modes* and *solver module* to accomplish only simplified docking scenarios.

2.2.3 Previous Mission & Vehicle Management Module

The MVM is the highest level module that manages the *solver* and *GN&C modules* to achieve the objectives of a mission, such as docking to a satellite. In this module, several *phases* of the mission are defined for a docking scenario. Due to the limitations of the glideslope algorithm, there are a different set of *phases* specific to the docking scenario and not a general sequence that works for any case. These phases are discussed in the next section from the previous MVM module, which are applicable to only specialized initial configurations of a docking scenario.

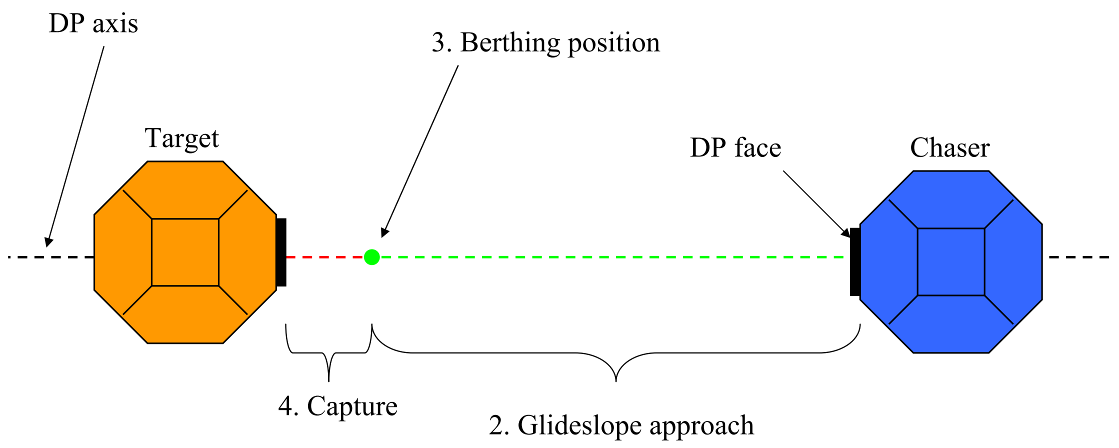


Figure 2-3: Docking to a Fixed Target Satellite Facing Forward

Docking to a Fixed Non-Tumbling Target Spacecraft

The first docking scenario discussed is the simplest one where the target satellite stays in a fixed position and attitude. Even in this simple scenario, the previous algorithms limit the initial configuration of the satellites. The limiting configurations would be any that require the use of a path planner with obstacle avoidance as this is unattainable by the glidslope algorithm. One such initial configuration is if the target spacecraft is facing its back towards the chaser. This requires the chaser spacecraft to maneuver around the target, avoid it as an obstacle, and get in front of the docking port for mechanical mating. The only initial configuration applicable with the glideslope algorithm is when the target spacecraft docking port is facing the chaser, see Figure 2-3. The initial attitude of the chaser satellite is allowed to be arbitrary.

Once the satellites are in the initial configuration shown in Figure 2-3, a set of *phases* are executed in sequence by the MVM module. Each phase has certain *termination conditions* before proceeding to the next. These are summarized in Table 2.1.

For the first phase, the chaser spacecraft maintains its current relative position and adjusts its attitude to point towards the target. Next, the glideslope algorithm

Table 2.1: MVM phases for docking to fixed target.

Phases	Controllers	Termination Conditions
1. Pointing	PD/PID controllers	time limit
2. Glideslope approach	glideslope along DP axis, PD/PID perpendicular	position error < tol and time limit
3. Berthing	PID controllers	state error < tol
4. Capture	Open-loop thrust	time limit

executes the velocity profile along the DP axis while a PD/PID controller is used perpendicularly to stay along the axis. During this phase, the attitude is regulated to orient the chaser's docking port to be within the mechanical alignment for the connection. The approach phase is planned to end at the *berthing position*, a small but safe offset distance from the face of the docking port. In the berthing phase, the chaser spacecraft maintains this state (position and regulated attitude) until the tight constraints are satisfied before a final thrust to *capture*.

The discussed phase sequence works only for initial configurations where the chaser satellite is aligned along the DP axis (as shown in Figure 2-3). This is a limitation brought upon from the glideslope algorithm. One solution without changing the algorithm is to add a pre-phase that moves the chaser to the docking port axis. This pre-phase must maintain a minimum distance from the target satellite for safety. Due to the straight line path planning available from the glideslope algorithm, there is an issue in a configuration when the target is facing backwards. The introduced pre-phase is only applicable to configurations when a linear path from the chaser to the front of the targets' docking port does not go through the target spacecraft.

The specified MVM module has been experimentally tested to work for a fixed non-rotating target spacecraft facing towards the chaser [10]. Therefore, there is good assurance to expand on these docking phases for an improved autonomous docking control system. Next, the changes to the MVM module to account for tumbling dynamics of the target is discussed.

Docking to Tumbling Target Spacecraft

Docking to tumbling satellites with pure rotation has been experimentally demonstrated by Nolet [10] when the chaser starts initially along the DP axis; however, more realistic docking scenarios require expanding the MVM module. As mentioned before, the initial configuration of the spacecraft for a fixed non-tumbling target is limited to a "forward" facing target spacecraft. Likewise when the target spacecraft is performing a rotating tumble, the only working initial configuration is when the chaser is initially aligned with the targets' docking port axis. To free up this constraint to other configurations without changing the algorithms, certain "pre-phases" are introduced. There are two pre-phases required before the *glideslope approach* (Table 2.1), for docking to a rotating target satellite from any initial configuration.

Go To Plane Of Rotation After pointing to the target satellite, the chaser moves to the closest point in the plane of rotation of the target satellite.

Wait For Target Facing The chaser waits at this point as the target satellite continues its rotation until they both point at each other within a certain angle tolerance.

These two "pre-phases" combined with the previous set of phases introduced earlier in Table 2.1 is depicted in Figure 2-4, for a docking scenario of a rotating target where the DP sweeps a plane where the chaser satellite is not initially located. This is a more complicated scenario compared to the chaser satellite already being in the plane of rotation. If this was the case, then the **Go To Plane Of Rotation** phase would be automatically satisfied at the start of the scenario and thus the following phases would proceed. The new expanded phase sequence viable for a rotating tumbling target from any initial configuration is summarized in Table 2.2.

The next step up in the complexity of the target satellite tumbling dynamics is when the docking port is sweeping a cone. This is also a pure rotating tumble; however, the rotation axis is not perpendicular to the docking port axis. In this scenario, an initial configuration where the cone being swept by the DP is behind the

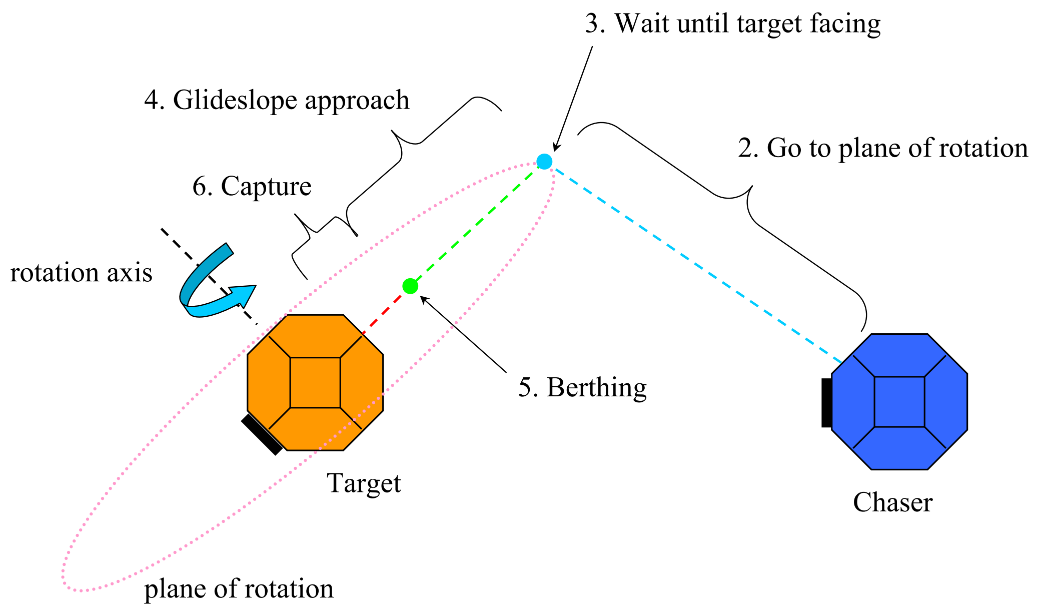


Figure 2-4: Docking to a Rotating Target Satellite Out-of-Plane

target spacecraft relative to the chaser's point-of-view, would be infeasible to by the previous algorithms, see Figure 2-5. This would again require the chaser to plan a path with obstacle avoidance rather than the linear planning provided by the glideslope algorithm. Therefore, the only feasible docking scenario with the glidslope algorithm is when the cone faces towards the chaser. The phase sequence from Table 2.1 with the "pre-phase" to align with the DP axis is applicable in this scenario.

The MVM module's set of phase sequences are specialized to fit varying docking scenarios rather than having a general form that works for all cases. The algorithms used also limit the initial configuration of the spacecraft and thus represent non-fully realistic docking scenarios. The following section introduces the upgraded algorithms of the modules and a new phase sequence in the MVM module that works for all the various docking scenarios with arbitrary initial configurations.

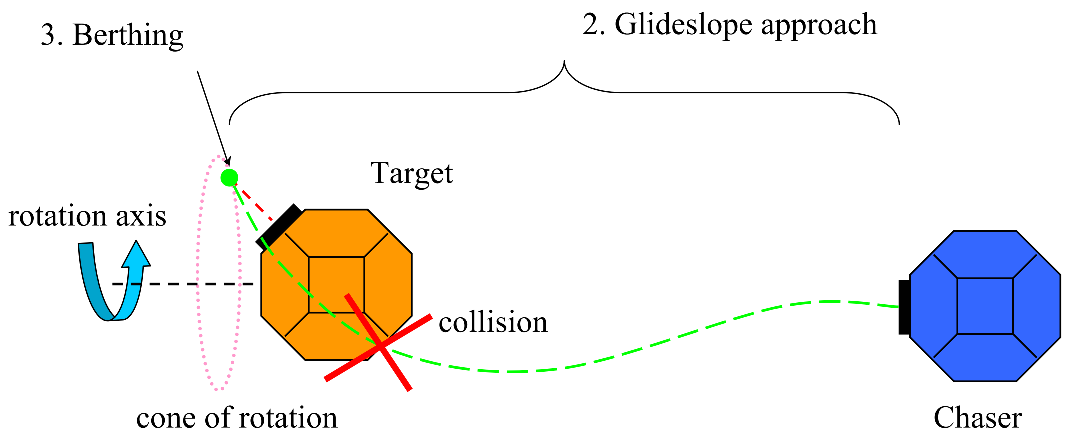


Figure 2-5: Docking to a Coning Target Satellite Facing Backwards using Glideslope Algorithm

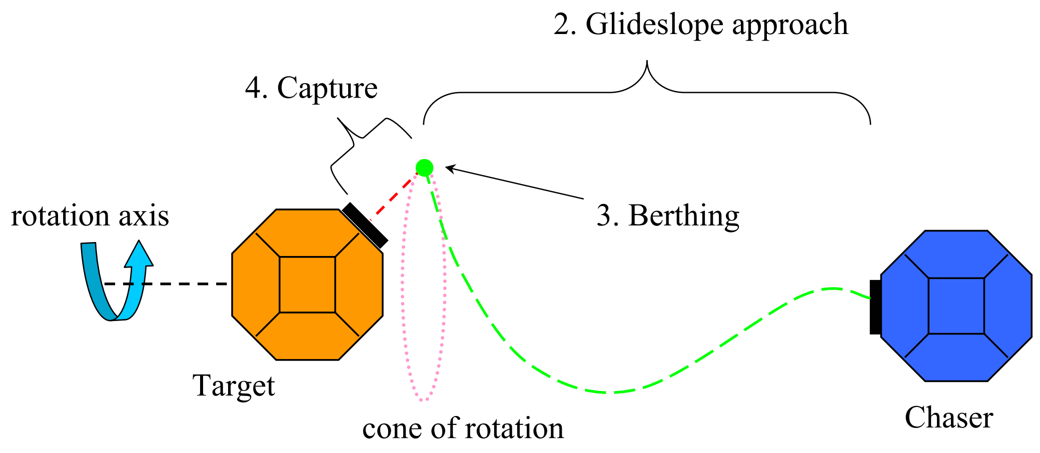


Figure 2-6: Docking to a Coning Target Satellite Facing Forward

Table 2.2: MVM phases for docking to rotating target.

Phase	Controllers	Termination Conditions
1. Pointing	PD/PID controllers	time limit
2. Go to plane of rotation	PD controllers	state error < tol
3. Wait for target facing	PD/PID controllers	state error < tol
4. Glideslope approach	glideslope along DP axis, PD/PID perpendicular	position error < tol and time limit
5. Berthing	PID controllers	state error < tol
6. Capture	Open-Loop Thrust	time limit

2.3 Advancements in Module Algorithms

The module that limits the capabilities of the GN&C architecture the most is the *solver* module. Thus, upgrading the previous glideslope algorithm with an appropriate path planner that handles obstacles would eliminate any constraints on the initial configurations of the docking scenarios. The path planner algorithm allows to plan a path from the chaser's initial position to in front of the target's docking port while considering the target satellite as an obstacle. This provides the chaser the capability to begin from any position and safely move to align with the target spacecraft docking port axis. The specifics of the path planner are discussed in the proceeding Chapter.

In addition to the path planner, improved trajectory tracking controllers are developed for more accurate following of the path. The improved controllers consist of a linear quadratic regulator (LQR), servo-LQR, and a phase-plane switching LQR/servo-LQR tracking controller. Each of these controllers have their own advantages and disadvantages that are discussed in Chapter 4.

The modules composing the previously introduced GN&C architecture from Figure 2-1 exhibit different levels of autonomy. Therefore, a new depiction shown in Figure 2-7 explains the hierarchical levels of autonomy with the MVM module being the highest to the control actuation as the lowest. The autonomous failure detection, isolation, and recovery system (FDIR) module is grayed out because it is not used in the docking scenarios presented in this thesis.

The algorithms of the lower and medium levels of autonomy: control and solver

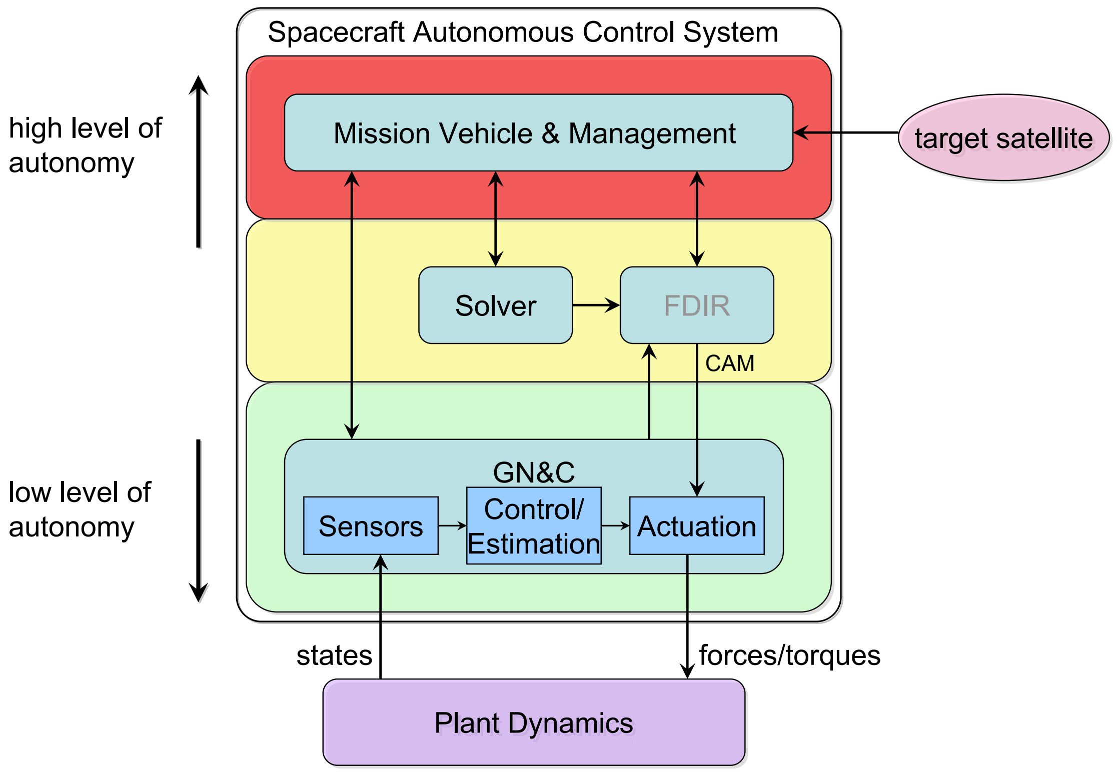


Figure 2-7: Hierarchical Depiction of the GN&C Architecture for Autonomous Docking

module, are upgraded to work for any docking scenario. Next, the MVM module is robustly designed into a single set of phases that work for any docking scenario.

2.3.1 Advancements in MVM Module

The upgraded solver and control modules provide the MVM larger flexibility in creating a more general phase sequence that works for all realistic docking scenarios. The improved MVM module handles the chaser spacecraft position and attitude planning separately. The attitude planning is dependent on the chaser's position relative to the target as explained in later in Section 2.3.1. Even though the attitude planning is coupled with the position in the MVM module, they are decoupled algorithmically in the solver module.

Position Planning

The phase sequence for the position planning is summarized in Table 2.3.

Table 2.3: MVM phases for any docking scenario.

Phase	Controllers	Termination Conditions
1. DP Axis Alignment	Path planner & LQR tracking controllers	time limit
2. Inline Approach	Path planner & LQR tracking controllers	time limit
3. Berthing	LQR controllers	state error $<$ tol
4. Capture	Open-Loop Thrust	time limit

The two new phases introduced, *DP Axis Alignment* and *Close In*, use the advanced solver module which uses a path planner with obstacle avoidance and one of the LQR-type controllers for precise tracking. A visual depiction of the phase sequence is shown in Figure 2-8 and described below.

DP Axis Alignment The chaser satellite uses a path planner and LQR-type controller to follow a safe path avoiding the target satellite as an obstacle to an offset distance, DP alignment position, along the DP axis of the target satellite.

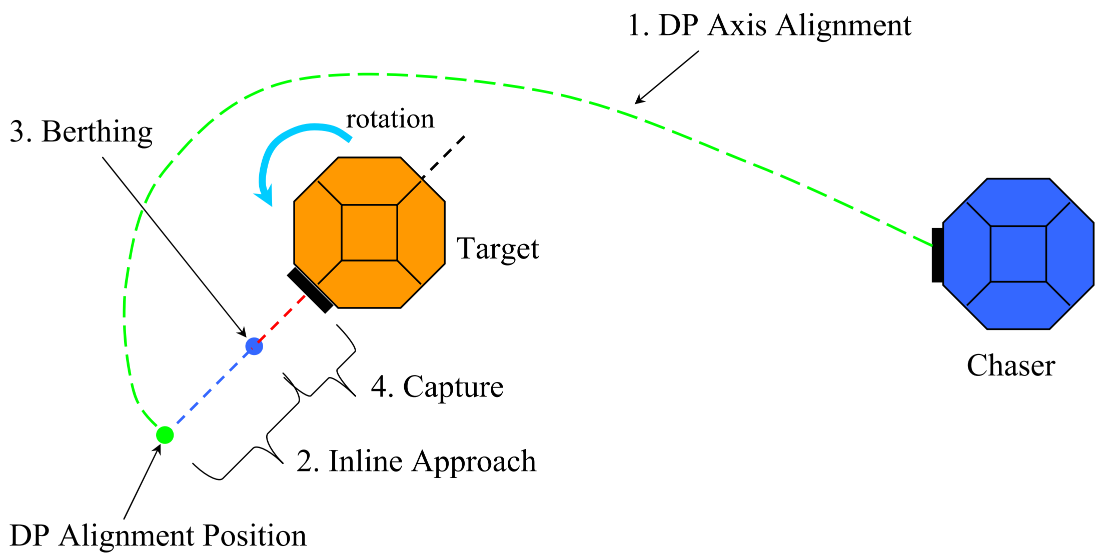


Figure 2-8: 2D Example of Docking to a Rotating Target Satellite In-Plane with the New Phase Sequence

The DP alignment position places the chaser along the DP axis to prepare for an inline approach towards the berthing position.

Close In From the DP alignment position, the chaser plans a second path to follow to the berthing position where it waits until very accurate position and attitude alignment before the capture thrust.

Attitude Planning

As the chaser spacecraft follows the phase sequence in position, the attitude planning switches between two states depending on the position relative to the target satellite.

Point to Target The chaser satellite uses a nonlinear PID controller to continuously point its DP towards the target satellite. The reason to point continuously is drawn from the assumption that sensors are placed on the same side of the DP used for relative estimation of the target satellite. Thus, pointing at the target is required to know its location for safety.

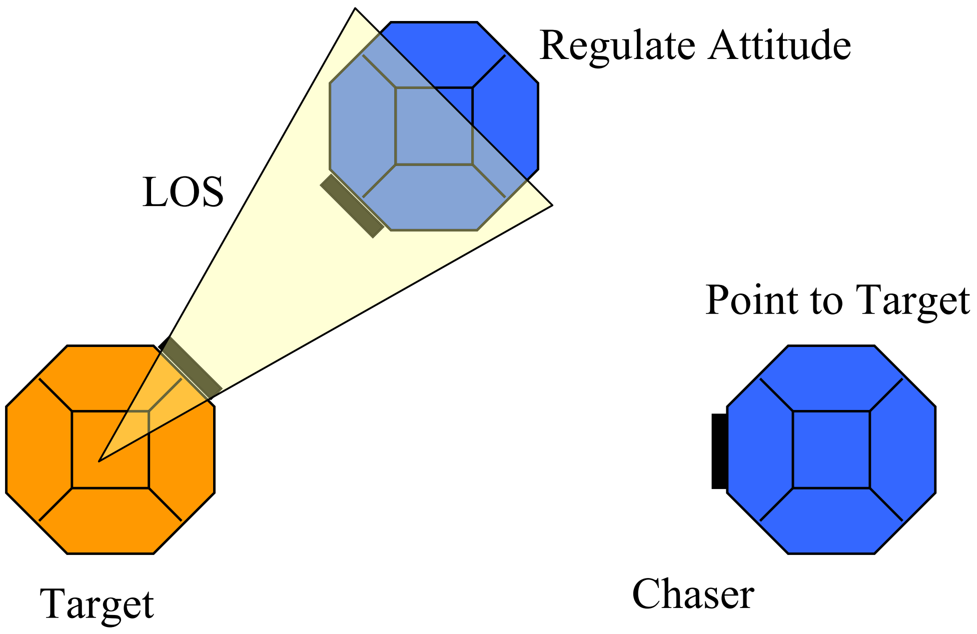


Figure 2-9: Attitude Planning Logic for Autonomous Docking

Regulate Attitude The attitude of the chaser satellite adjusts to have the two docking ports become mechanically aligned for capture.

The decision between to **Point to Target** or **Regulate Attitude** is made by whether the chaser satellite position is within the *line-of-sight* (LOS) of the target spacecraft, see Figure 2-9 and Table 2.4. The LOS is currently described by a space cone extending in front of the targets' docking port. Therefore, if the chaser satellite is within the LOS space cone, then it is close to prepare for a capture and thus decides to regulate the attitude for DP mechanical alignment. Otherwise, being outside the LOS, the chaser's attitude continuously points to the target satellite for relative state estimation.

Table 2.4: Attitude planning.

Relative Position	State
Inside LOS	Regulate Attitude
Outside LOS	Point to Target

2.3.2 Conclusion of Advancements

The online path planning algorithm in the solver module is to be upgraded to a planner that accounts for non-stationary terminal conditions and obstacle avoidance. In addition, the algorithms in the control module are improved with LQR-type controllers. With these two upgrades, an improved sequence of phases for position planning in the MVM module is developed. The new phases are robustly written to accomplish docking to tumbling spacecraft from any initial configuration. This provides testing of realistic conditions when two spacecraft reach each other close enough to execute these phases for docking. The attitude planning has a simple control logic with two step inputs. One is for the chaser to point at the target spacecraft while outside the LOS; otherwise, regulate its attitude to the chaser when inside the LOS to physically mate. The possible required rotation to regulate attitude from pointing might be a maximum of 180 degrees. This is a rather large rotation required for spacecraft and would consume significant amount of fuel. Also, this large rotation may be dangerous at such close proximity with potentially large solar panels interfering. Therefore, there is a recommendation from this thesis for any real application in docking to have the docking port mechanical design built to self-rotate. This allows the DP regulation to be performed mechanically by rotating the relatively small docking port rather than the complete spacecraft.

2.4 Summary

This chapter exploited the deficiencies in the algorithms employed in the previous GN&C architecture and proposed the necessary improvements to successfully accomplish a docking scenario to a tumbling spacecraft. Several solutions are presented that do not require a change to the algorithms, but they would still not be able to perform a docking maneuver to a backwards facing coning target. Therefore, a proposal for a new solver module that consists of a path planner with obstacle avoidance. Also, advancement in tracking controllers is stated. With these upgrades, a new formulation of the phases of a docking mission is presented to be robust for any of the docking scenarios. This chapter builds a final framework that will tie in the solver and control module for simulation and experimental tests of docking to tumbling satellites.

Chapter 3

Trajectory Planning

This chapter covers the upgrade to the previous solver module to a true path planner. A fully functional online planner provides the capability to the GN&C architecture to dock to a tumbling target spacecraft from any initial configuration. First, a general formulation of optimal planning with differential dynamic constraints is defined. Then this formulation is detailed with the application of docking of two spacecraft with collision avoidance. The formulation is composed by defining the system's cost functional to minimize, the chaser spacecraft translational dynamics, the planning terminal conditions dependent on the target spacecraft tumbling dynamics, and the method for modeling obstacles. Afterwards, the development of the solution to the optimal control problem for docking by using the calculus of variation technique is presented. This method develops the necessary conditions for optimality to first-order. These are a set of differential Euler-Lagrange equations that form the Hamiltonian Boundary Value Problem (HBVP). The solution to the HBVP provides a truly optimal result to the path planning problem. However, the solution to the boundary value problem is mathematically complex and too computationally intensive for hardware implementation. Therefore, a highly efficient sub-optimal planning algorithm is developed that is based on cubic splines. The variational technique to optimal planning and the spline-based algorithm are compared to study the level of sub-optimality of the more efficient algorithm. The summary concludes the new spline-based planner is adequate for the problem of docking of two spacecraft and is numerically efficient for hardware implementation.

3.1 Path Planning Problem Formulation

This section first introduces the general formulation of motion planning for dynamical systems that exhibit differential constraints. Afterwards, additional constraints are considered in the state-space for obstacle avoidance. The next section will detail the formulation with specific dynamics and terminal conditions for docking scenarios that is used for investigating the two types of path planning algorithms.

The equations of motion of a dynamical system of order n is described in state-space form by a set of $2n$ first-order differential equations of the form [6],

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3.1}$$

where t is the time variable, $\mathbf{x}(t)$ is an n-dimensional vector with real elements that denotes the state of the system, $\mathbf{u}(t)$ is an m-dimensional vector with real elements that denotes the control input of the system, and $\mathbf{f}$ is a real vector valued function [6].

Equation (3.1) is also referred to as a *state transition equation*. The state $\mathbf{x}(t)$ generally represents the appropriate degrees of freedom n of the dynamical system that lies on a smooth manifold $X \in \Re^n$ called the *state-space*. The control input $\mathbf{u}(t)$ is from a control space U that is a bounded subset of $\Re^m$, where m represents the number of control inputs. The transition equation (3.1) is of general form and may be nonlinear and time-varying. Once a control profile $\mathbf{u}(t)$ is known, the corresponding *state trajectory* $\mathbf{x}(t)$ can be inferred by integrating the transition equation from an initial state $\mathbf{x}(t_0)$ at time t_0 until a final time t_f.

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t_f} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{3.2}$$

A first approach to the objective of a path planning algorithm would be to determine a *state trajectory* $\mathbf{x}(t)$ that drives the system from an initial state $\mathbf{x}(t_0)$ to a

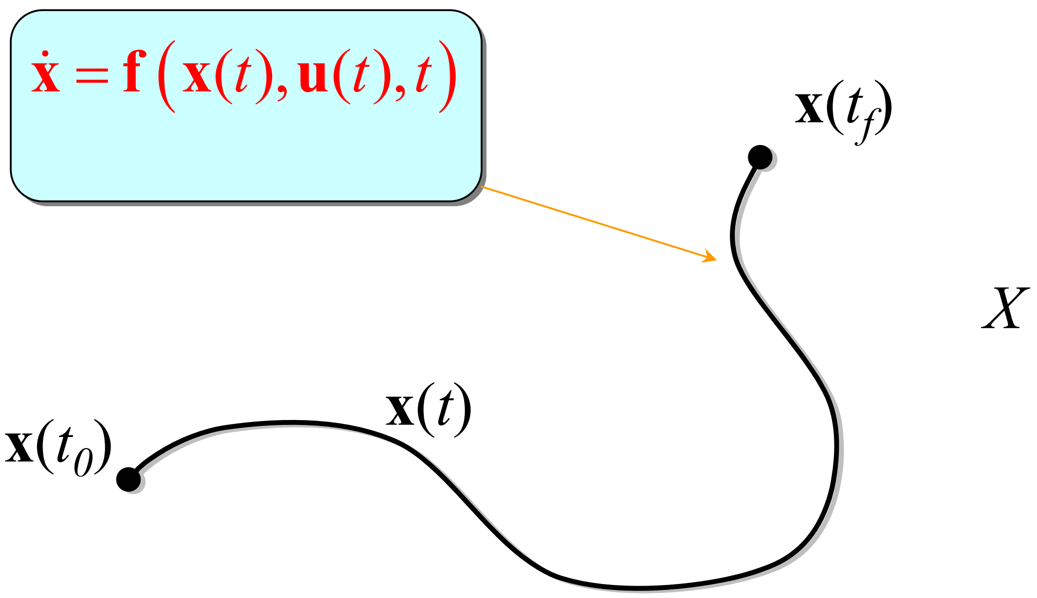

$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right)$
$\mathbf{x}(t_f)$
X
$\mathbf{x}(t)$
$\mathbf{x}(t_0)$

Figure 3-1: State Planning with Differential Constraints

final state $\mathbf{x}(t_f)$ in a fixed final time t_f, while satisfying the differential constraints of the dynamics Eq. (3.1). Figure 3-1 shows such a *trajectory* where the line resembles a multi-dimensional path of the states $\mathbf{x}(t)$ through the state-space X while satisfying the differential constraints $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$ from $\mathbf{x}(t_0)$ until $\mathbf{x}(t_f)$.

The planned state trajectory $\mathbf{x}(t)$ is a solution to a corresponding control input $\mathbf{u}(t)$ from the transition equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$. Thus, an equivalent objective of a path planning algorithm is to find a *control trajectory* $\mathbf{u}(t)$ of the functional form $\mathbf{u}(t) : [0, \infty) \rightarrow U$, which satisfies the differential constraints Eq. (3.1) and drives the state from $\mathbf{x}(t_0)$ to $\mathbf{x}(t_f)$. By definition, a *control trajectory* satisfying all of the constraints is called an admissible solution of a path planning algorithm [6]. However, there may be an infinite number of admissible trajectories for a fully controllable system that drives the states to the goal state, see Figure 3-2. A common approach to a decision strategy to choose between the admissible trajectories is by minimizing a certain cost functional (performance metric) [6],

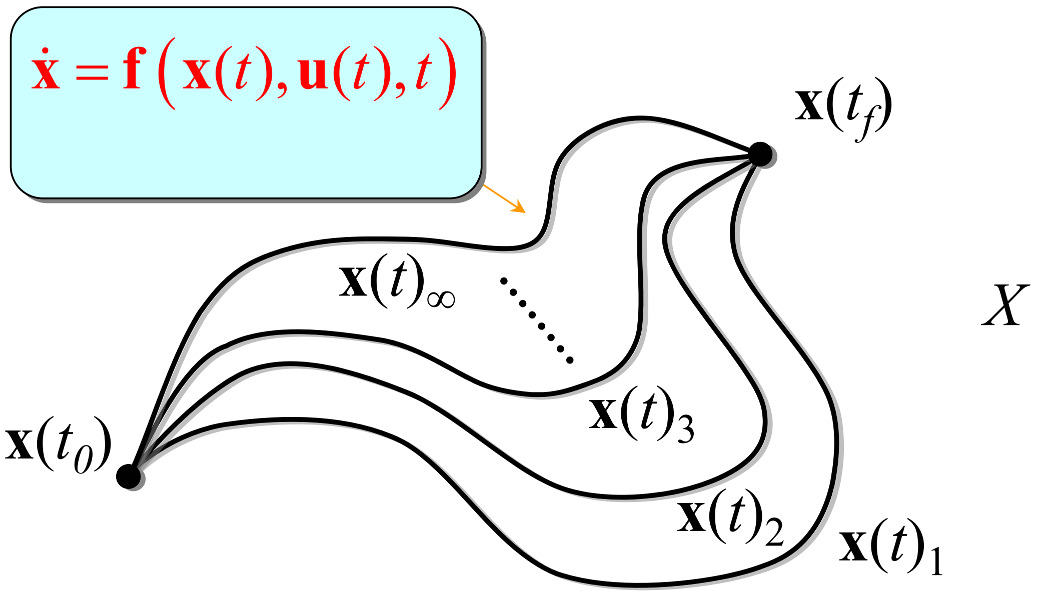


Figure 3-2: Infinite Feasible State Trajectories

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{3.3}$$

where t_0 and t_f are the initial and final time, h and g are real scalar functions where h is specifically considered to be the terminal cost. A control trajectory minimizing Eq. (3.3) falls under the category of optimal path planning, also referred to as an *optimal open-loop control law* from control theory [6]. The cost function is formed such that the trade-off between the states, control, time, and perhaps the final time is optimized. The final time may be handled in two different ways:

t_f **- fixed** The final time is predefined for the path planning.

t_f **- free** The final time is let to vary in the optimization of the cost functional.

When the final time is let to vary, the variable t_f is set to be part of the terminal cost h from Eq. (3.3) with generally a trade-off constant α. This constant implies the importance of having a smaller final time, *large* α, or larger final time, *small* α.

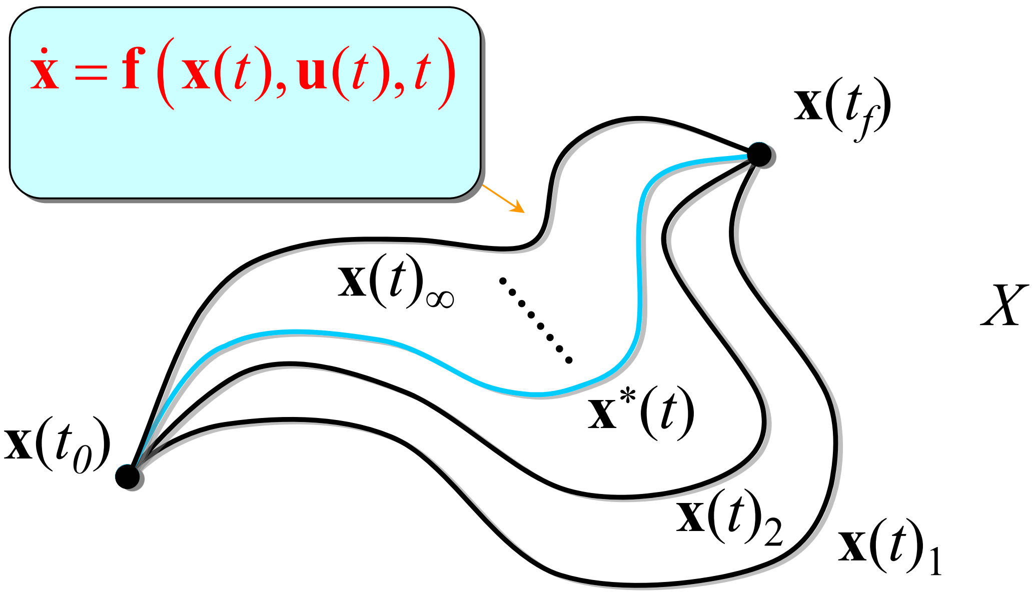


Figure 3-3: Optimal State Trajectory

An example of how the terminal cost function may look like with a free final time is shown in Eq. (3.4).

$$J = \alpha t_f + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{3.4}$$

The *control trajectory* corresponding to the optimal path found by minimizing the cost function Eq. (3.3) and satisfying the differential dynamic constraints Eq. (3.1) is referred to as the *optimal control trajectory* and is denoted with an asterisk $\mathbf{u}^*(t)$. The corresponding *optimal state trajectory* $\mathbf{x}^*(t)$ is again inferred through the state transition equation (3.2), see Figure 3-3.

This concludes the most basic path planning for differential dynamics. Further complexities arise by adding constraints to the control and/or state variables. Typical control constraints consist of lower and upper bounds on the control input:

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{3.5}$$

An example would be the minimum and maximum thrust throttling for the space shuttle main engines during launch. Let's define the control space which satisfies the control constraints to be $U_{feasible}$ and so the optimal control trajectory is constrained to be part of that set [8]:

$$\mathbf{u}^* \in U_{feasible} \tag{3.6}$$

Furthermore, constraints on the state-space can be employed where a certain subset of the original set X is feasible. Therefore, the optimal state trajectory is also constrained to the feasible set [8]:

$$\mathbf{x}^* \in X_{feasible} \tag{3.7}$$

An obvious example for $X_{feasible}$ would be a non-moving obstacle which occupies the state-space set $X_{obstacle}$. Since the full state-space X is considered to be an implicit "universal" set, the feasible state-space is the compliment of X relative to $X_{obstacle}$ [8],

$$X_{feasible} = X \backslash X_{obstacle} \tag{3.8}$$

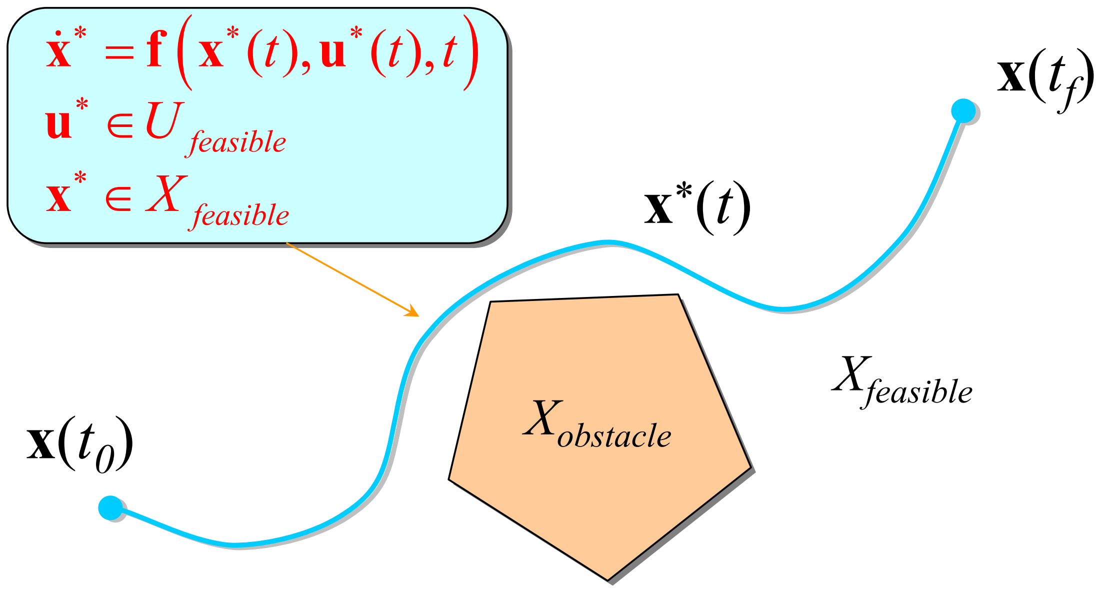
which allows only a feasible region of the state-space to be optimized across. Figure 3-4 shows this example with an optimal state trajectory satisfying all constraints. Finally, a proper problem formulation for motion planning under differential, control, state constraints can be defined.

3.1.1 Optimal Path Planning Problem Formulation General

The objective of the planning algorithm is to find the *optimal control trajectory* $\mathbf{u}^*(t)$ that minimizes the performance metric Eq. (3.3),

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt$$

and satisfies the differential, control, and state constraints:



$\dot{\mathbf{x}}^* = \mathbf{f}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), t\right)$
$\mathbf{u}^* \in U_{feasible}$
$\mathbf{x}^* \in X_{feasible}$
$\mathbf{x}(t_f)$
$\mathbf{x}^*(t)$
$X_{feasible}$
$X_{obstacle}$
$\mathbf{x}(t_0)$

Figure 3-4: Optimal State Trajectory Satisfying Control and State Constraints

$$
\begin{aligned}
&\dot{\mathbf{x}}^*(t) = \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t), t) \\
&\mathbf{u}^* \in U_{feasible} \\
&\mathbf{x}^* \in X_{feasible}
\end{aligned}
\tag{3.9}
$$

The problem formulation is established for any general system, nonlinear or linear, time-variant or time-invariant systems. In the next section, this formulation will be modified and completed for the docking scenarios to a tumbling target satellite.

3.2 Path Planning Problem Formulation for Docking

The cost functional, state transition equation, terminal states, and obstacle constraints are defined for docking scenarios to complex tumbling target satellites for a planning algorithm. The control and autonomy architecture for docking is established in Chapter 2 with Table 2.3 on page 39 detailing the position planning phases. The attitude planning is decoupled from the path planner and thus the planning algorithm needs to only account for translational motion of the chaser spacecraft. From Table 2.3, phases **DP Axis Alignment** and **Inline Approach** require the use of a path planner. Such a docking scenario is depicted in Figure 2-8 on page 40. The translational equations of motion of a spacecraft are developed as the state transition equation (3.1) for the planning algorithm. The final time for the path planning formulation is set to be a fixed value t_f. As the final positions of the two phases that use the path planner are along the DP axis of the spacecraft, the terminal states for the planner depend on the tumbling dynamics of the target spacecraft. These dynamics are modeled and the propagated final state of the target spacecraft is used to determine the final state for the chaser spacecraft. Lastly, the obstacle constraint is modeled as a collision sphere in the state-space. The final problem formulation for docking is summarized to be used to develop the two path planning methods in following sections.

3.2.1 Cost Functional for Docking

As mentioned previously, there may be multiple admissible trajectories that connect the initial $\mathbf{x}(t_0)$ and final states $\mathbf{x}(t_f)$ together. A very common decision logic at filtering out the admissible trajectories for a unique one done is by minimizing some sort of performance metric of the system [6]. For the case of spacecraft docking, an obvious performance metric is one that chooses from the admissible trajectories one that consumes the least fuel. Given that the control effort $\mathbf{u}(t)$ represents fuel consumption, a possible cost functional to be minimized is,

$$J = \int_{t_0}^{t_f} |\mathbf{u}(t)| \, dt \tag{3.10}$$

where $|\cdot|$ is the absolute value. This cost functional has been used repeatedly in spacecraft trajectory planning [14] and is ideal for discrete path planning algorithms. The optimal control law for minimum fuel paths is a Bang-Off-Bang discontinues controller [6]. Since the two planning algorithms in this thesis are of continues form, it is easier to work with a cost functional that provides a continues control law. The cost functional used for the planning algorithms is to minimize the energy of the control input:

$$\boxed{J = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^T(t)\mathbf{u}(t)dt} \tag{3.11}$$

The energy of the system directly relates to the fuel consumed by the spacecraft, so minimizing energy is an acceptable performance metric. The two planning algorithms will take the cost functional Eq. (3.11) into account in their own unique manner. Next, the state transition equation is developed for docking of spacecraft and shows how the control input influences the dynamics.

3.2.2 State Transition Equation for Docking

Docking missions would typically occur in an orbit around Earth and would be governed by orbital equations of motion. Other possible missions discussed in Chapter 1 may be outside the Earth's gravitational sphere of influence and somewhere in deep space, thus the spacecraft would obey a different set of equations of motion. For an orbiting spacecraft, the translational dynamics are governed by the nonlinear two-body equation of relative motion in cartesian coordinates [17],

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\mathbf{r} = \mathbf{f} \tag{3.12}$$

where $\mathbf{r} \in \Re^3$ is the relative position vector from the earth to the spacecraft, $r = \|\mathbf{r}\|$ is the magnitude of the relative position, $\mu = G(M_\oplus + m)$ is the $GM_\oplus$ product, G is the universal gravitational constant, $M_\oplus$ is the mass of the Earth, and m is the mass of the spacecraft.

For rendezvous and docking applications, only the relative position dynamics between the two spacecraft are important [1]. When the docking mission of the two spacecraft is assumed to be in a highly circular orbit, the nonlinear dynamics Eq. (3.12) may be linearized into the well known *Euler-Hill* equations of relative

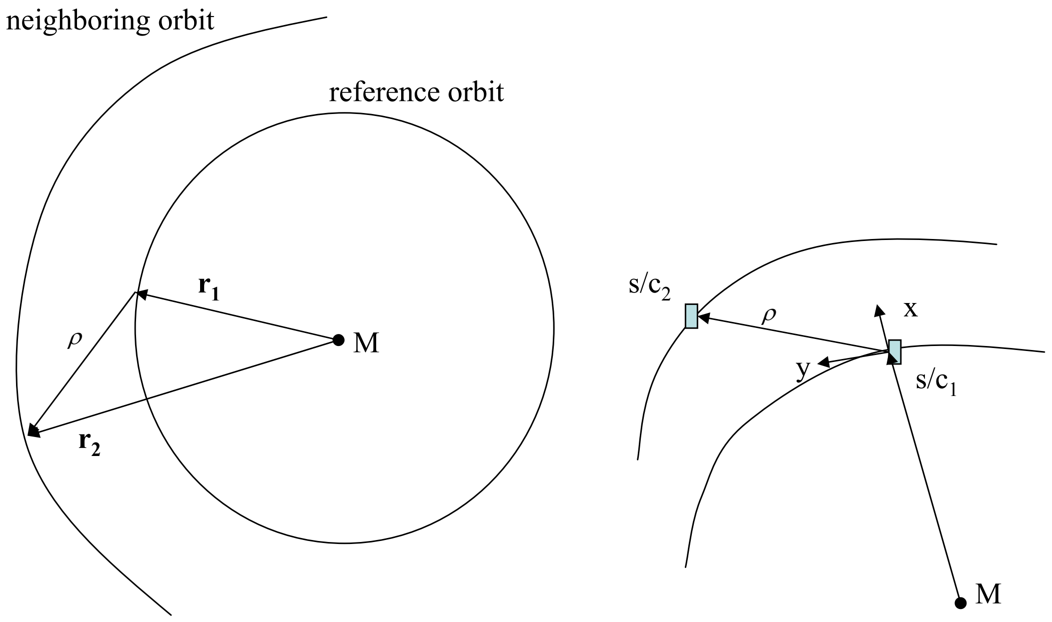


Figure 3-5: Hill's Relative Equations of Motion

motion [17],

$$\begin{aligned} \ddot{x} - 2n\dot{y} - 3n^2x &= f_x \\ \ddot{y} + 2n\dot{x} &= f_y \\ \ddot{z} + n^2z &= f_z \end{aligned} \tag{3.13}$$

where the x, y, and z coordinates are relative to a moving coordinate system being centered at one of the spacecraft center of mass, shown in Figure 3-5. Here n is the angular velocity of the orbit that is assumed constant when the orbit is highly circular.

The *Hill's* equation (3.13) can be further simplified to three decoupled double integrators equations of motion when the spacecraft are maneuvering faster than the angular velocity of the orbit, n. This can be seen by observing the bode plot of the double integrator superimposed on *Hill's* equations and noting that they are identical in frequencies higher than the rate of the orbit n [12]. Figure 3-6 shows an example of the bode plots for an orbital rate of $n = 1 = 10^0$ rad/sec on the x-axis. The development of control and autonomy algorithms in this thesis for docking missions are focused on the terminal phase of a docking mission, referring up to one hundred meter separation between spacecraft [10]. Therefore, any nonlinear effects of orbital dynamics are small and may be considered as disturbances to be compensated by tracking controllers. As a result, the spacecraft is modeled as a constant point-mass with internally generated forces provided by thrusters for maneuvering. The state transition equation for the planning algorithm is,

$$\boxed{\ddot{\mathbf{r}} = \mathbf{a}} \tag{3.14}$$

where $\ddot{\mathbf{r}}$ is the acceleration of the position $\mathbf{r} = [r_x \quad r_y \quad r_z]^T$, and $\mathbf{u} = \mathbf{a} = [a_x \quad a_y \quad a_z]^T$ is the acceleration control input. The spacecraft is set to unit mass for simplification while the relation $F = ma$ is used to determine the force F necessary to provide the appropriate acceleration a, where m is the mass of the spacecraft. The transition equation Eq. (3.14) is with respect to a non-moving global coordinate frame of reference. The state transition equation defines the differential dynamics constraint for propagating an initial state to another final state. The specific formulation of these state is developed in the next section.

3.2.3 Terminal States for Docking

The terminal states $\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$ with respect to a global coordinate frame are defined for docking scenarios to complex tumbling target satellites. For either of the two phases that use the path planner, **DP Axis Alignment** and **Inline Approach**, the initial state $\mathbf{x}(t_0)$ is defined as the initial state of the chaser spacecraft when the MVM module decides to execute the planning algorithm.

The final state is much more complicated as it is a time-varying state dependent

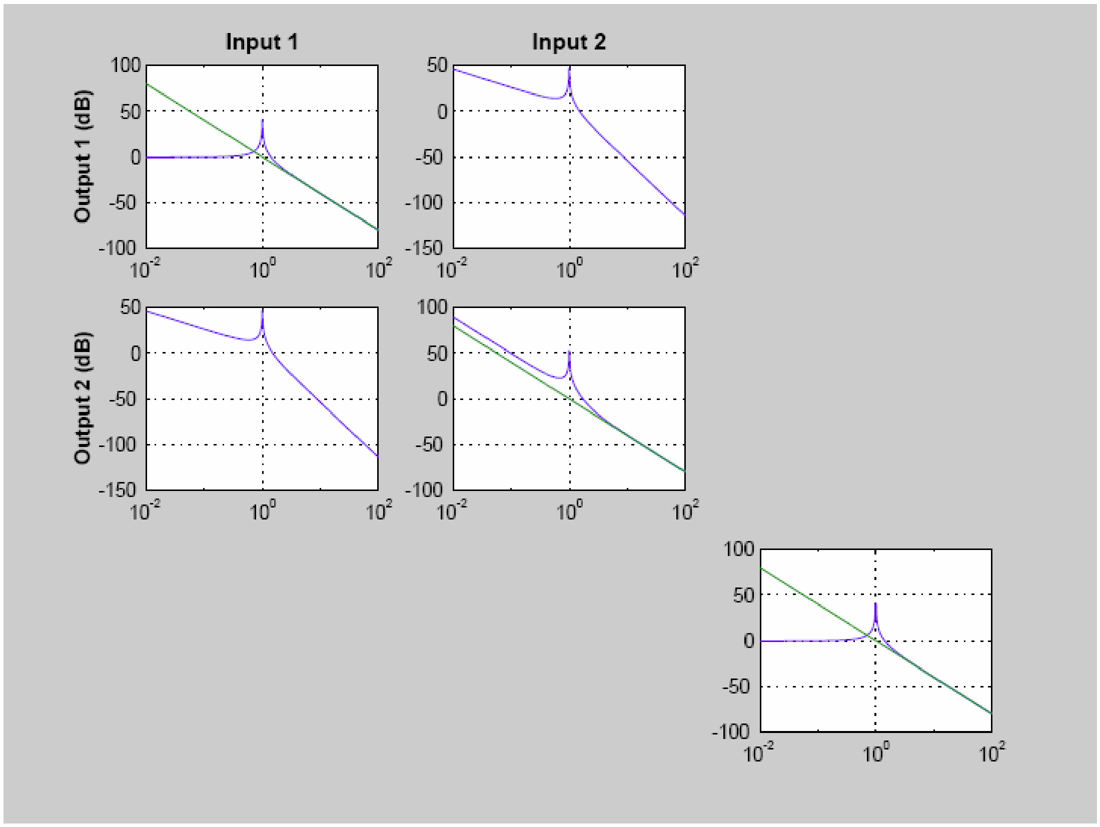


Figure 3-6: Hill's Equations and Double Integrator Bode Plots [12]

on the tumbling dynamics of the target spacecraft and its position with respect to the chaser. For the **DP Axis Alignment** phase, the final state is set to be the *DP Axis Alignment Position*, which is an offset distance along the target satellite docking port axis at the final time t_f. Also for the **Inline Approach** phase, the final state is the *Berthing Position*, which is another smaller offset distance along the DP Axis, see Figure 2-8 on page 40. The final states for both cases depend on the target spacecraft time-varying docking port axis at time t_f. As the DP axis depends on the orientation of the target spacecraft, the derivation of the final state depends on the attitude and position dynamics.

Spacecraft Position and Tumbling Attitude Dynamics

The full position and attitude state dynamics of the target spacecraft are developed for use in forming the final state for the chaser spacecraft. The state vector that describes both the position and attitude of the spacecraft was introduced in Section 2.2.1 on page 27 as Eq. (2.1):

$$\mathbf{x} = \begin{bmatrix} r_x & r_y & r_z & v_x & v_y & v_z & q_1 & q_2 & q_3 & q_4 & \omega_x & \omega_y & \omega_z \end{bmatrix}^T$$

The translational equations of motion of the spacecraft are represented by Eq. (3.14) that describe the propagation of position $\mathbf{r}$ and velocity $\mathbf{v}$ when the spacecraft is considered to be a unit mass. This assumption is used for the path planner while accurate propagation of the position state is represented by,

$$\ddot{\mathbf{r}} = \frac{1}{m}\mathbf{f} \tag{3.15}$$

where $\mathbf{f} \in \Re^3$ is the total force applied to the system.

The elements in the state vector representing the attitude is the quaternion vector $\mathbf{q} \in \Re^4$ and the angular rate vector $\boldsymbol{\omega} \in \Re^3$ with respect to its local body coordinate frame. The equations of motion that represent the propagation of the attitude state is described by the attitude kinematics and dynamics equations. The kinematics define how the quaternion attitude representation propagates with time while the dynamics define the angular velocities propagation. The nonlinear kinematics is a function of the angular rates of the state vector and represented as [17],

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}(\omega)\mathbf{q} \tag{3.16}$$

where $\mathbf{\Omega}(\omega)$ is defined as:

$$\mathbf{\Omega}(\omega) \equiv \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{3.17}$$

The spacecraft rigid-body attitude dynamics describes the time derivative of the angular momentum vector $\mathbf{h} \in \Re^3$,

$$\dot{\mathbf{h}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{h} \tag{3.18}$$

where $\boldsymbol{\tau} = \begin{bmatrix}\tau_x & \tau_y & \tau_z\end{bmatrix}^T$ is the total moment/torque applied to the system. The angular momentum vector is defined as,

$$\mathbf{h} = \mathbf{I}\boldsymbol{\omega} \tag{3.19}$$

with $\mathbf{I}$ being the spacecraft moment of inertia tensor:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.20}$$

When the spacecraft is assumed to be a rigid-body, the off-diagonal terms become identical $I_{xy} = I_{yz}$, $I_{xz} = I_{zx}$, and $I_{yz} = I_{zy}$; thus, the inertia tensor is symmetric. By using the product rule for differentiation of $\mathbf{h}$, the time derivative of the angular momentum is:

$$\dot{\mathbf{h}} = \dot{\mathbf{I}}\boldsymbol{\omega} + \mathbf{I}\dot{\boldsymbol{\omega}} \tag{3.21}$$

A standard assumption is made that the spacecraft inertia does not vary with time, then combining equations (3.18) and (3.21) with $\dot{\mathbf{I}} = 0$ defines the time rate change of the angular velocities:

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left[-\boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) + \boldsymbol{\tau}\right] \tag{3.22}$$

As a result, the complete spacecraft attitude equations of motion are represented by equations (3.16), (3.17) and (3.22) and so the full equations of motion describing the propagation of the state $\mathbf{x}$ is composed:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \frac{1}{m}\mathbf{f} \\ \frac{1}{2}\boldsymbol{\Omega}(\omega)\mathbf{q} \\ \mathbf{I}^{-1}\left[-\boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) + \boldsymbol{\tau}\right] \end{bmatrix} \tag{3.23}$$

When the target satellite is purely tumbling, no control inputs are executed $\mathbf{f} = \mathbf{0}$ and $\boldsymbol{\tau} = \mathbf{0}$. Thus, the transient solution of equation (3.23) describes the tumbling dynamics of the target satellite.

The dynamics in equation (3.23) describe the equation of motion of a general spacecraft. Let's define the state of the *target* spacecraft as $\mathbf{x}_{obs}$ since it will be considered as an obstacle in the path planner. Then the initial state of the target $\mathbf{x}_{obs}(t_0)$ is numerically integrated with dynamics Eqs. (3.23) using $\mathbf{f} = \mathbf{0}$ and $\boldsymbol{\tau} = \mathbf{0}$ until t_f to find the final state of the target spacecraft $\mathbf{x}_{obs}(t_f)$:

$$\mathbf{x}_{obs}(t) = \mathbf{x}_{obs}(t_0) + \int_{t_0}^{t_f} \left\{ \begin{bmatrix} \mathbf{v}_{obs} \\ \mathbf{0} \\ \frac{1}{2}\boldsymbol{\Omega}(\omega)\mathbf{q}_{obs} \\ \mathbf{I}_{obs}^{-1}\left[-\boldsymbol{\omega}_{obs} \times (\mathbf{I}_{obs}\boldsymbol{\omega}_{obs})\right] \end{bmatrix} \right\} dt \tag{3.24}$$

As mentioned, the final state of the chaser spacecraft depends on the final state of the target (obstacle) spacecraft. The transformation to the chaser spacecraft final state $\mathbf{x}(t_f)$ with respect to a global reference frame is developed in the next section.

Transformation to Terminal States

When the MVM module decides to execute the path planner, the initial state of the target satellite is numerically integrated with Eqs. (3.24) until the final time. The final state of the target satellite from the transient solution is used to find the final state of the chaser spacecraft. This is performed by transforming the desired final state described with respect to the target spacecraft coordinate system to the global frame of reference that is fed to the path planner.

Typically, the docking port is rigidly attached to a spacecraft and thus the docking port axis is fixed relative to the local body coordinate frame. Let's define the docking port axis unit vector as $\hat{\mathbf{d}}_{body} \in \Re^3$ relative to the body frame of the target, see Figure 3-7. Now, the motion of the DP axis in the global frame is a function of a rotational transformation from the quaternion state representation,

$$\hat{\mathbf{d}} = \mathbf{R}(\mathbf{q}_{obs})\hat{\mathbf{d}}_{body} \tag{3.25}$$

where the general $\mathbf{R}(\mathbf{q})$ (not necessarily $\mathbf{R}(\mathbf{q}_{obs})$) is the direction cosine matrix [17]:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2\left(q_1q_2 - q_3q_4\right) & 2\left(q_1q_3 + q_2q_4\right) \\ 2\left(q_1q_2 + q_3q_4\right) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2\left(q_2q_3 - q_1q_4\right) \\ 2\left(q_1q_3 - q_2q_4\right) & 2\left(q_2q_3 + q_1q_4\right) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.26}$$

The offset distances of *DP Axis Alignment Position* and *Berthing Position* shown in Figure 2-8 on page 40 are along the DP axis with a magnitude of z and b, respectively. Therefore the alignment and berthing position vectors in the body frame are:

$$\begin{aligned} \mathbf{z}_{body} &= z\hat{\mathbf{d}}_{body} \\ \mathbf{b}_{body} &= b\hat{\mathbf{d}}_{body} \end{aligned} \tag{3.27}$$

The vectors are fixed for any docking scenario unless the docking port is repositioned.

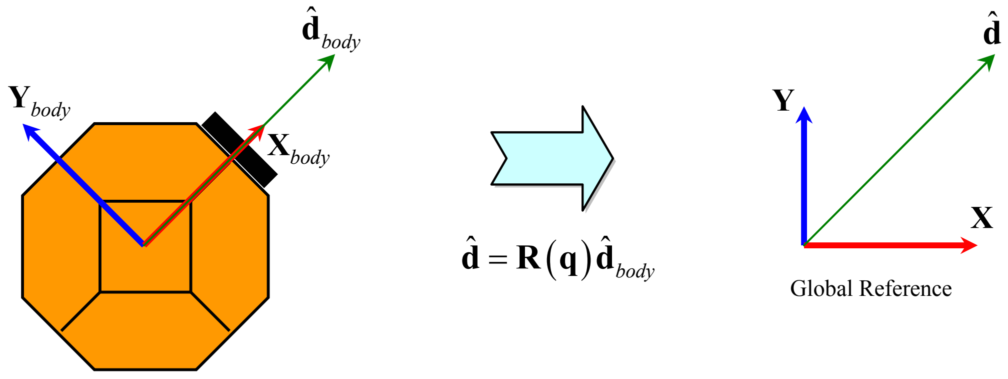

$\hat{\mathbf{d}}_{body}$
$\mathbf{Y}_{body}$
$\mathbf{X}_{body}$
$\hat{\mathbf{d}} = \mathbf{R}(\mathbf{q})\hat{\mathbf{d}}_{body}$
$\hat{\mathbf{d}}$
Y
X
Global Reference

Figure 3-7: Docking Port Vector in Body and Global Coordinates

They are transformed to the global frame by the rotation matrix:

$$\begin{aligned} \mathbf{z} &= \mathbf{R}(\mathbf{q}_{obs})\mathbf{z}_{body} \\ \mathbf{b} &= \mathbf{R}(\mathbf{q}_{obs})\mathbf{b}_{body} \end{aligned} \tag{3.28}$$

These position vectors are fixed in the body frame but vary in the global frame due to the time-varying rotation matrix from the continuously varying attitude.

The desired final state in the global coordinates are the vectors in Eqs. (3.28) transformed by the final position and attitude state $\mathbf{x}_{obs}(t_f)$ of the target spacecraft. Therefore, final position of the chaser is formed by extracting the quaternion attitude from the obstacle final state $\mathbf{q}_{obs}(t_f) \in \mathbf{x}_{obs}(t_f)$ to form the rotation matrix to apply to $\mathbf{z}_{body}$ and translate by $\mathbf{r}_{obs}(t_f)$:

$$\mathbf{z}(t_f) = \mathbf{R}(\mathbf{q}_{obs}(t_f))\mathbf{z}_{body} + \mathbf{r}_{obs}(t_f) \tag{3.29}$$

This defines the final position $\mathbf{r}_f = \mathbf{z}(t_f)$ for the terminal state $\mathbf{x}(t_f)$ of the **DP Axis Alignment** phase of the chaser spacecraft, and likewise it is found for the **Inline Approach** phase. The final velocity $\mathbf{v}_f$ is the time derivative of equation (3.29), which is the cross product of the angular rate vector $\boldsymbol{\omega}_{obs}$ with $\mathbf{z}_{body}$ transformed to

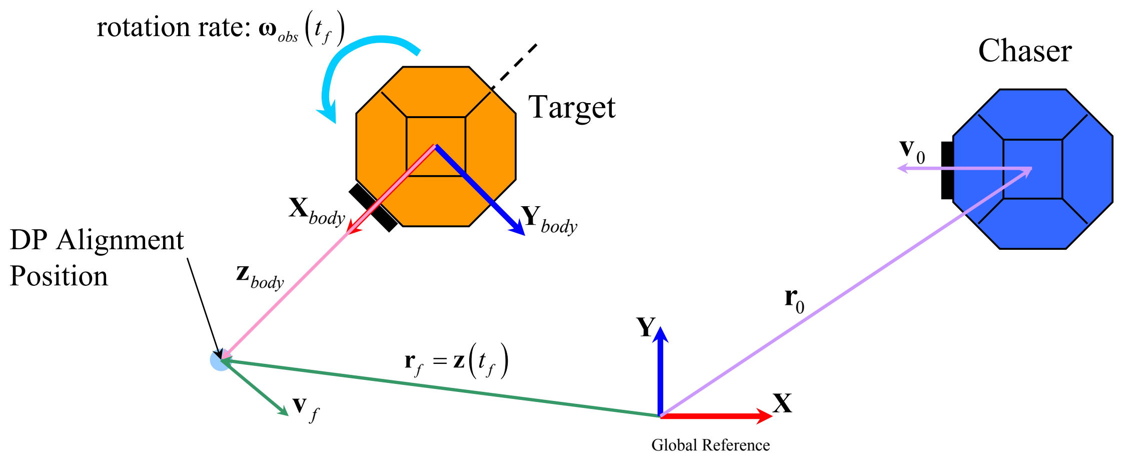


Figure 3-8: Example of the Transformation to the Final State of a Rotating Target Satellite Scenario

the global coordinate system:

$$\mathbf{v}(t_f) = \mathbf{R}(\mathbf{q}_{obs}(t_f))\,[\boldsymbol{\omega}_{obs}(t_f) \times \mathbf{z}_{body}] + \mathbf{v}_{obs}(t_f) \tag{3.30}$$

Therefore the final terminal state for the **DP Axis Alignment** phase is composed:

$$\boxed{\mathbf{x}(t_f) = \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{q}_{obs}(t_f))\mathbf{z}_{body} + \mathbf{r}_{obs}(t_f) \\ \mathbf{R}(\mathbf{q}_{obs}(t_f))\,[\boldsymbol{\omega}_{obs}(t_f) \times \mathbf{z}_{body}] + \mathbf{v}_{obs}(t_f) \end{bmatrix}} \tag{3.31}$$

The same equations apply for the **Inline Approach** phase by replacing $\mathbf{z}_{body}$ with the berthing position vector $\mathbf{b}_{body}$. Finally, the complete terminal states are defined for the chaser spacecraft. The initial state $\mathbf{x}(t_0) = [\mathbf{r}_0 \quad \mathbf{v}_0]^T$ is set to be current state of the chaser spacecraft at time t_0 when the path planner is requested to be executed. The final state is a transformation with the target satellite final state defined by Eq. (3.31). An example of these transformations for the docking scenario of a purely rotational tumbling scenario is depicted in Figure 3-8.

The development of the terminal conditions for path planning is complete. The initial state $\mathbf{x}(t_0)$ is provided by the estimator as the current state of the chaser spacecraft when the path planning is requested by the MVM module. However, the final state $\mathbf{x}(t_f)$ is more complicated and has to be computed. At first, the initial state of the target spacecraft $\mathbf{x}_{obs}(t_0)$ is provided by the estimator and is propagated until t_f, Eq. (3.24). Then equation (3.31) is used to compute the chaser spacecraft final state in the correct global coordinate frame. In the next section, the modeling of obstacles is defined for the planning algorithms.

3.2.4 Obstacles for Docking

The area of the state-space that defines a forbidden region as an obstacle $X_{obstacle}$ can be modeled in various manners. They depend on the shape of the expected obstacles, the accuracy one desires to maneuver around the obstacle, and the path planning algorithm approach. The most useful method is by bounding the obstacle with a combination of several convex polyhedrons. This approach offers the user to adjust the tightness of the bound by increasing or decreasing the dimensions of the polyhedrons. A simpler method is to fit a spherical obstacle around an object. This method does not provide tight bounds about the obstacle but does provide very efficient modeling as this is desired for the new trajectory planning algorithm

For the purposes of docking two spacecraft together, the only obstacles to account for are simply the two spacecraft. Therefore, two spherical obstacles may be used enclose the two spacecraft. The size of the spherical obstacle is defined by its radius. Since a planning algorithm generally determines the trajectory of the centroid to travel, the traveling spacecraft spherical obstacle radius is added to the target spacecraft obstacle radius. Thus the only obstacle to account for in the planning algorithm is the target spacecraft as it also accounts for the size of the chaser. The spherical obstacles are depicted in Figure 3-9.

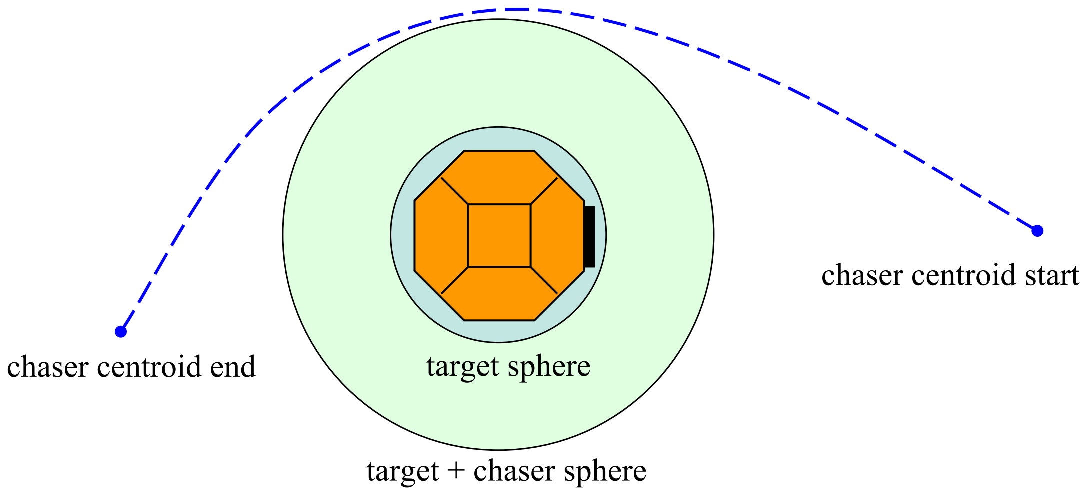


Figure 3-9: Modeling Obstacles for Docking of Two Spacecraft

3.2.5 Planning Problem Formulation for Docking Summary

The cost functional, state transition equation, final time, terminal states, and obstacle constraints are defined for docking scenarios to a tumbling target satellite. This provides enough information to build a planning algorithm that determines a unique trajectory while avoiding any obstacles. There are no constraints on the control input as this adds further complexity to the planning algorithm. For an attempt to assure a non-saturating control profile, a large enough final time is selected. The larger the final time, the lower the maximum control input is along a state trajectory. A summary of the trajectory planning problem formulation for docking is summarized in Table 3.1

Table 3.1: Summary of Trajectory Planning Problem Formulation for Spacecraft Docking.

Description	Formulas	
1. final time	fixed t_f	
2. cost functional	$J = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^T(t)\mathbf{u}(t)dt$	Eq. (3.11)
3. state transition	$\ddot{\mathbf{r}} = \mathbf{a}$	Eq. (3.14)
4. initial state	$\mathbf{x}(t_0) = \mathbf{x}_0 \in \Re^6$	
5. final state	form obstacle initial state: $\mathbf{x}_{obs}(t_0) = \mathbf{x}_{obs,0} \in \Re^{13}$	
	Propagate $\mathbf{x}_{obs}(t_0)$ to t_f: $\mathbf{x}_{obs}(t) = \mathbf{x}_{obs}(t_0) + \int_{t_0}^{t_f} \left\{ \begin{bmatrix} \mathbf{v}_{obs} \\ \mathbf{0} \\ \frac{1}{2}\mathbf{\Omega}(\omega)\mathbf{q}_{obs} \\ \mathbf{I}_{obs}^{-1}\left[-\boldsymbol{\omega}_{obs} \times (\mathbf{I}_{obs}\boldsymbol{\omega}_{obs})\right] \end{bmatrix} \right\} dt$	Eq. (3.24)
	form final state $\mathbf{x}(t_f) \in \Re^6$ using $\mathbf{x}_{obs}(t_f) \in \Re^{13}$: $\mathbf{x}(t_f) = \begin{bmatrix} \mathbf{R}(\mathbf{q}_{obs}(t_f))\mathbf{z}_{body} + \mathbf{r}_{obs}(t_f) \\ \mathbf{R}(\mathbf{q}_{obs}(t_f))\left[\boldsymbol{\omega}_{obs}(t_f) \times \mathbf{z}_{body}\right] + \mathbf{v}_{obs}(t_f) \end{bmatrix}$	Eq. (3.31)
6. obstacle	$X_{obstacle}$ = sphere with radius r_{obs}	

3.3 Variational Technique to Optimal Path Planning

The calculus of variations is a very powerful technique applicable to solve continues optimal control problems. The variational method was initially developed to find optimal curves satisfying certain constraints. A simple example would be to find the shortest curve connecting two points. The obvious answer is a straight line. However, when another constraint is added that the curve must pass along another surface, then the problem is not as straightforward and the calculus of variation method can be used to find the optimal curve.

In regards to the trajectory planning problem, the calculus of variations technique is applied to determine the optimal control and state trajectories while satisfying differential dynamic constraints and minimizing a cost functional. This method provides the necessary set of differential equations for optimality to first-order. In this section, these set of equations are defined for a general system, referred to as the Euler-Lagrange equations. Next, the specific set of necessary equations are found for the docking problem and methods for numerical solutions are mentioned.

3.3.1 Euler-Lagrange Equations General

In Section 3.1.1, a general formulation of an optimal control problem was stated. The general form considers possibly free constrained final time t_f and state $\mathbf{x}(t_f)$. Since only fixed final constraints are used for the planning methods in this thesis, the optimal control problem setup is simplified. Thus, the terminal cost $h(\mathbf{x}(t_f), t_f)$ is excluded from the cost functional. The optimal control problem setup for the variational technique approach for path planning is defined.

Problem 1 (**Optimal Control Problem**). *Determine the optimal control* $\mathbf{u}^*(t)$ *and state* $\mathbf{x}^*(t)$ *trajectory that minimizes the cost functional,*

$$J(\mathbf{x}(t), \mathbf{u}(t), t) = \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt$$

and satisfies the differential equations of motion constraint,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

where:

- t_0 *and* t_f *fixed*
- $\mathbf{x}(t_0)$ *and* $\mathbf{x}(t_f)$ *fixed*

The calculus of variations approach for optimization is similar to the method used for determining the minimum of a curve. Finding an extremum of a curve by standard calculus methods is done by setting the first derivative of the curve function to zero and having the second derivative define whether it is a maxima or minima. In this case, the curve function is dependent of only an independent variable. The calculus of variations approach is similar, but its goal is to minimize a function dependent of another function such as the performance metric to be minimized $J(\mathbf{x}(t), \mathbf{u}(t), t)$. This is referred to as a functional. Therefore, the calculus of variations method requires the variation (first derivative) of the functional to equal zero. The necessary conditions for the variation being zero is developed [6].

First, the differential dynamic constraints are handled by augmenting them to the cost functional with time-varying Lagrange multipliers $\mathbf{p}(t) = [p_1(t), \ldots, p_n(t)]$, referred to as the *costate* variables:

$$J_a = \int_{t_0}^{t_f} \left\{ g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t) \left[\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}(t) \right] \right\} dt \tag{3.32}$$

The number of costate variables is equivalent to the number of state variables. The variation of J_a is found by introducing the variations $\delta\mathbf{x}$, $\delta\dot{\mathbf{x}}$, $\delta\mathbf{u}$, and $\delta\mathbf{p}$ [6]:

$$\begin{aligned} \delta J_a = \int_{t_0}^{t_f} \Bigg\{ \left[\frac{\partial g}{\partial \mathbf{x}} + \mathbf{p}^T(t) \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right] \delta\mathbf{x}(t) + \left[\frac{\partial g}{\partial \mathbf{u}} + \mathbf{p}^T(t) \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right] \delta\mathbf{u}(t) \\ + \left[\mathbf{f} - \dot{\mathbf{x}} \right]^T \delta\mathbf{p}(t) - \mathbf{p}^T(t) \delta\dot{\mathbf{x}} \Bigg\} dt \end{aligned} \tag{3.33}$$

Eq. (3.33) can be cleaned up by introducing the **Hamiltonian**, a real valued scalar function:

$$H(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3.34}$$

Then the variation becomes:

$$\delta J_a = \int_{t_0}^{t_f} \left\{ \frac{\partial H}{\partial \mathbf{x}} \delta\mathbf{x}(t) + \frac{\partial H}{\partial \mathbf{u}} \delta\mathbf{u}(t) + [\mathbf{f} - \dot{\mathbf{x}}]^T \delta\mathbf{p}(t) - \mathbf{p}^T(t)\delta\dot{\mathbf{x}} \right\} dt \tag{3.35}$$

The term with $\delta\dot{\mathbf{x}}$ in equation (3.35) can be simplified after integrating the term by parts:

$$-\int_{t_0}^{t_f} \mathbf{p}^T(t)\delta\dot{\mathbf{x}} dt = -\mathbf{p}^T(t_f)\left(\delta\mathbf{x}(t_f) - \dot{\mathbf{x}}(t_f)\delta t_f\right) + \int_{t_0}^{t_f} \dot{\mathbf{p}}^T(t)\delta\mathbf{x} dt \tag{3.36}$$

Since the terminal conditions of t_f and $\mathbf{x}(t_f)$ are fixed, the terms outside the integrand are not considered in equation (3.36). After combining Eq. (3.36) and Eq. (3.35), the variation is rewritten to be:

$$\delta J_a = \int_{t_0}^{t_f} \left\{ \left[\frac{\partial H}{\partial \mathbf{x}} + \dot{\mathbf{p}}^T(t) \right] \delta\mathbf{x}(t) + \frac{\partial H}{\partial \mathbf{u}} \delta\mathbf{u}(t) + [\mathbf{f} - \dot{\mathbf{x}}]^T \delta\mathbf{p}(t) \right\} dt \tag{3.37}$$

The integral must vanish for $\delta J_a = 0$, which defines an extremal. Thus, the terms multiplying the variations $\delta\mathbf{x}(t)$, $\delta\mathbf{u}(t)$, and $\delta\mathbf{p}(t)$ must be set to zero. This defines the necessary conditions that minimizes J_a subject to any boundary constraints [6],

$$\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*, t) \tag{3.38a}$$

$$\dot{\mathbf{p}}^* = -\frac{\partial H}{\partial \mathbf{x}^*} \tag{3.38b}$$

$$\frac{\partial H}{\partial \mathbf{u}^*} = 0 \tag{3.38c}$$

while the boundary conditions are fixed for the docking problem:

$$
\begin{aligned}
&t_0 \quad and \quad t_f \quad fixed \\
&\mathbf{x}(t_0) = \mathbf{x}_0 \\
&\mathbf{x}(t_f) = \mathbf{x}_f
\end{aligned}
$$

The first two set of coupled differential equations from $\dot{\mathbf{x}}^*$ and $\dot{\mathbf{p}}^*$, equations (3.38a) and (3.38b), are referred to as the Euler-Lagrange equations. The optimal control trajectory $\mathbf{u}^*$ solution from $\frac{\partial H}{\partial \mathbf{u}^*} = 0$ is substituted into the system dynamics $\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*, t)$. Then the coupled *state* $\dot{\mathbf{x}}^*$ and *costate* $\dot{\mathbf{p}}^*$ differential equations form the Hamiltonian Boundary Value Problem (HBVP). These differential equations may be linear when the system dynamics are also linear and the cost functional contains only quadratic terms and so may be solved analytically. Otherwise, they are nonlinear differential equation in which numerical methods are mainly used to find a solution. The dimension of the states, n, and costates is equivalent. Therefore, the numerical method must find a solution to $2n$ coupled nonlinear differential equations with n initial conditions from $\mathbf{x}_0$ and n final conditions from $\mathbf{x}_f$. There are various numerical methods for solving such a problem, but the technique used here is based on the collocation method [16]. The solution to the HBVP solves the general optimal control problem and the specific setup of the Euler-Lagrange equations for docking scenarios is developed in the next section.

3.3.2 Euler-Lagrange Equations for Docking

Section 3.2.5 summarizes the specific formulation of terminal conditions, system dynamics, and obstacle modeling for docking purposes. The information from Table 3.1 is used to formulate the optimal control problem for docking in addition to certain new methods that will be introduced. First, the state transition equation (3.14) defines the differential equations of motion constraint. The three decoupled double integrator differential dynamics are rewritten in state-space form as,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3.39}$$

with $\mathbf{A}$ and $\mathbf{B}$ being,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3x3} & \mathcal{I}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{3x3} \\ \mathcal{I}_{3x3} \end{bmatrix} \tag{3.40}$$

where $\mathcal{I}$ is the identity matrix, $\mathbf{x} \in \Re^6$ is the x, y, z position and velocity states, and $\mathbf{u} = \mathbf{a} = [a_x \quad a_y \quad a_z]^T$ is the acceleration control input. The goal of the trajectory planning is to determine a control profile that reduces the total fuel/energy consumption. If an unconstrained trajectory initially passes through an obstacle, then surely a trade-off is required to balance between minimizing fuel and finding a path that avoids the obstacle. Therefore, the cost functional is formed in order to take into account fuel minimization and obstacle clearance.

$$J = \int_{t_0}^{t_f} \left[g_{control}(\mathbf{u}(t)) + g_{obstacle}(\mathbf{x}(t)) \right] dt \tag{3.41}$$

The fuel consideration is taken care of by minimizing the total energy from Eq. (3.11) in the $g_{control}$ cost functional,

$$g_{control}(\mathbf{x}(t), \mathbf{u}(t), t) = \frac{1}{2}\rho \mathbf{u}^T(t)\mathbf{u}(t) \tag{3.42}$$

where $\rho \geq 0$ is a weight factor evenly scaled to all three axis of control. The choice for minimizing energy is that it represents a quadratic cost functional and results in a continues optimal control law derived later. The obstacle clearance is accounted for by penalizing the relative distance d_r between the spacecraft and the obstacle. Thus, there is a search for a distance function that has large values near the obstacle and zero far away. The proposed distance metric is a piecewise cubic [3],

$$g_r(d_r) = \begin{cases} k\left(a_1 d_r^3 + a_2 d_r^2 + a_3 d_r + a_4\right) & \text{if } d_r < s \\ 0 & \text{else} \end{cases} \tag{3.43}$$

where s is a buffer distance of where the cost value is nonzero, k is the maximum cost at $d_r = 0$, and the coefficients $a_1, \ldots, a_4$ are found to satisfy the Lipschitz smoothness conditions for the Euler-Lagrange differential equations. This condition is satisfied if the distance metric $g_r(d_r)$ is C^2. The cubic polynomial already satisfies such a condition, so the piece-wise connection between the cubic polynomial and the value zero at $g_r(s)$ needs to be constrained to C^2. This is accomplished by solving for the four coefficients satisfying the following conditions:

$$\begin{aligned} g_r(0) &= k \\ g_r(s) &= 0 \\ \frac{\partial g_r(d_r)}{\partial d_r}(s) &= 0 \\ \frac{\partial^2 g_r(d_r)}{\partial d_r^2}(s) &= 0 \end{aligned} \tag{3.44}$$

This gives four linear algebraic equations to be solved for four unknowns. Therefore, Eq. (3.44) can be rewritten in matrix form as:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ s^3 & s^2 & s & 1 \\ 3s^2 & 2s & 1 & 0 \\ 6s & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} k \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.45}$$

A plot of the distance metric cost functional is shown in Figure 3-10. The relative distance function is defined as the euclidean two-norm of the objects face-to-face relative position vector,

$$d_r(t) = \|\mathbf{r}(t) - \mathbf{r}_{obs}(t)\|_2 - R - R_{obs} \tag{3.46}$$

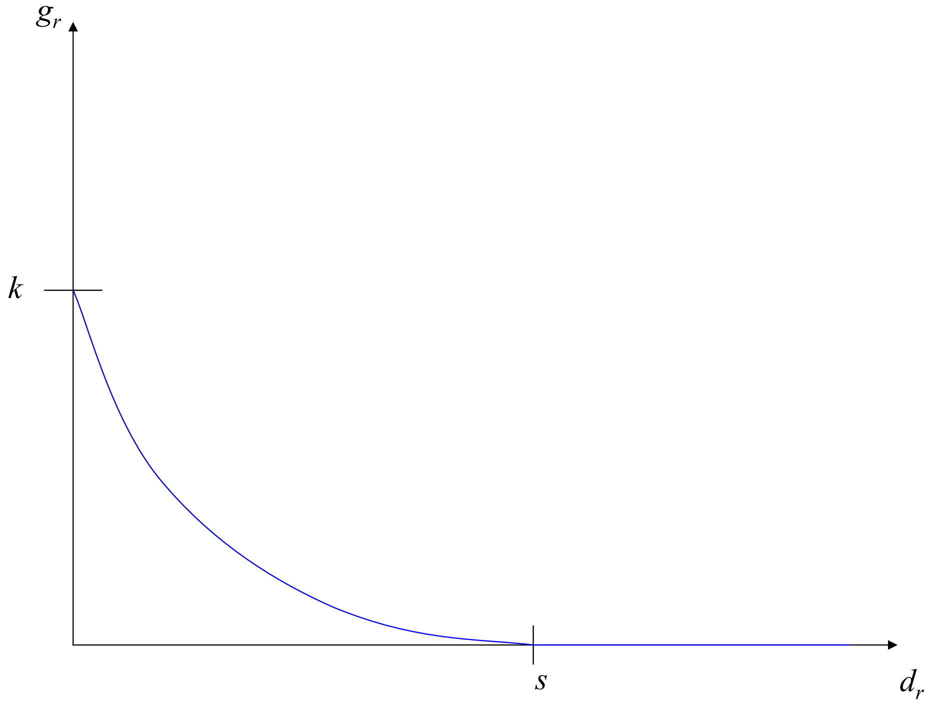


Figure 3-10: Relative Distance Obstacle Cost Penalization

where $\mathbf{r}(t) \in \mathbf{x}(t)$ is the position vector of the chaser spacecraft that is a subset of the state $\mathbf{x}(t)$, while $\mathbf{r}_{obs}(t)$ is the position vector to the centroid of the obstacle. The scalar values R is the radius of the obstacle sphere to cover the chaser spacecraft and R_{obs} is the radius of the obstacle sphere covering the target spacecraft. As shown in Figure 3-9, both obstacle spheres are added together and applied to the target spacecraft, referred to as the obstacle in this planning algorithm. Therefore, the radii subtract off from relative centroid-to-centroid distance $\|\mathbf{r} - \mathbf{r}_{obs}\|_2$. Since the obstacle position vector is a function of time in Eq. (3.46), this method accounts for time-varying obstacles.

Let's consider the possible ways to minimize Eq. (3.43) over the time t_0 to t_f. One strategy for the spacecraft is to distance the trajectory further away from the obstacle while another applicable approach is to increase velocity and spend as little time close to the obstacle [3]. As the second strategy is undesirable, a cost function $g_v(v_r)$ dependent on the relative velocity v_r is introduced and multiplied with the relative distance cost function $g_r(d_r)$. The proposed relative velocity cost function is [3],

$$g_v(v_r) = \begin{cases} v_r & \text{if } v_r > v_c \\ b_1 v_r^3 + b_2 v_r^2 + b_3 v_r + b_4 & \text{else} \end{cases} \tag{3.47}$$

where the coefficients $b_1, \ldots, b_4$ are chosen to satisfy the C^2 continuity at the connecting velocity v_c. The relative velocity cost function $g_v(v_r)$ increases with larger values of v_r linearly after v_c. The issue with keeping the cost function just the magnitude of the relative velocity is the non-differentiable property at $v_r = 0$. Thus, the cubic polynomial is introduced at v_c, a very small relative velocity, and to have a zero cost at $v_r = 0$. A plot of the cost function $g_v(v_r)$ is shown in Figure 3-11. The cubic polynomial coefficients of $g_v(v_r)$ are chosen by satisfying:

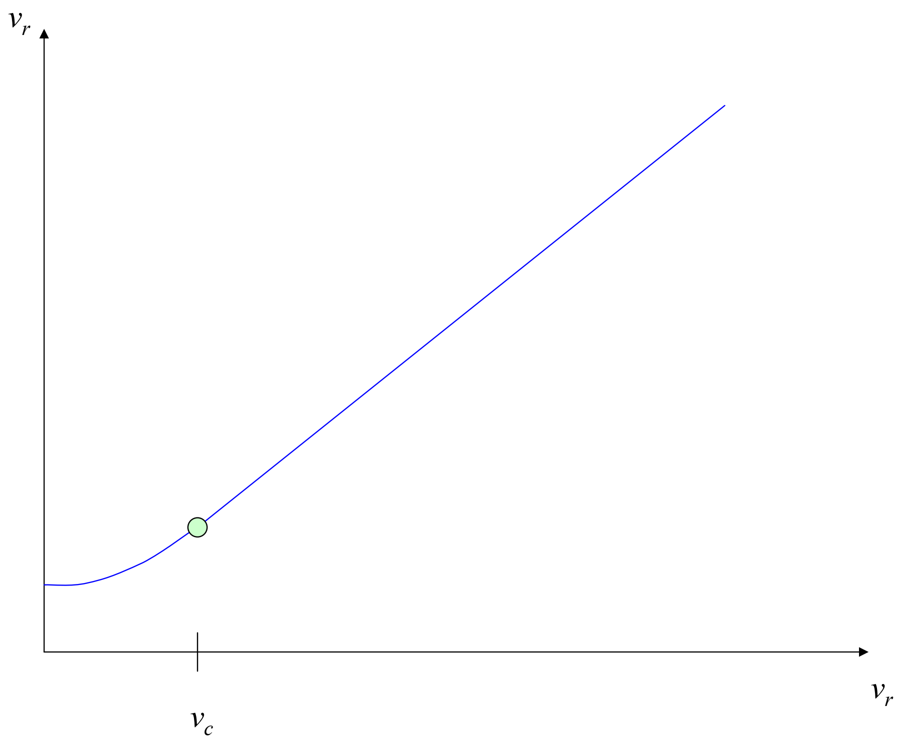


Figure 3-11: Relative Velocity Obstacle Cost Penalization

$$\begin{aligned} g_v(v_c) &= v_c \\ \frac{\partial g_v(v_r)}{\partial v_r}(v_c) &= 1 \\ \frac{\partial g_v(v_r)}{\partial v_r}(0) &= 0 \\ \frac{\partial^2 g_v(v_r)}{\partial v_r^2}(v_c) &= 0 \end{aligned} \tag{3.48}$$

Then rewritten in matrix form as:

$$\begin{bmatrix} v_c^3 & v_c^2 & v_c & 1 \\ 3v_c^2 & 2v_c & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 6v_c & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} v_c \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.49}$$

The relative velocity v_r is also a two-norm of the relative velocity vectors between the chaser spacecraft and obstacle from $\mathbf{x} = [\mathbf{r} \quad \mathbf{v}]^T$:

$$v_r(t) = \|\mathbf{v}(t) - \mathbf{v}_{obs}(t)\|_2 \tag{3.50}$$

Therefore, the total obstacle cost functional is composed of equations (3.43) and (3.47) as,

$$g_{obstacle}(\mathbf{x}(t)) = \alpha g_r(d_r) g_v(v_r) \tag{3.51}$$

where $\alpha \geq 0$ is total weighting factor on the obstacle penalization [3]. As a result, the primary weights ρ and α may be used to trade-off optimizing energy and obstacle clearance. Multiple obstacles may be considered by summing up the cost function to each obstacle as,

$$g_{obstacle}(\mathbf{x}(t)) = \sum_i \alpha_i g_{r,i}(d_{r,i}) g_{v,i}(v_{r,i}) \tag{3.52}$$

for the i^{th} obstacle [3]. However, only a single obstacle, the target spacecraft, is considered for the docking scenarios. This concludes the setup of the cost functional Eq. (3.41). Thus, the system dynamics constraints and cost functional are completely formulated for the development of the Euler-Lagrange equations.

First, the *Hamiltonian* is formed as:

$$H = g_{control}(\mathbf{u}(t)) + g_{obstacle}(\mathbf{x}(t)) + \mathbf{p}^T(t)\left[\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\right] \tag{3.53}$$

and then the optimal control law is found by using equation (3.38c), $\frac{\partial H}{\partial \mathbf{u}^*} = 0$. Since equation (3.38c) takes a partial derivative with respect to the control input $\mathbf{u}(t)$, only the terms with $\mathbf{u}(t)$ are extracted from the Hamiltonian,

$$H(\mathbf{u}(t)) = \frac{1}{2}\rho\mathbf{u}^T(t)\mathbf{u}(t) + \mathbf{p}^T(t)\mathbf{B}\mathbf{u}(t) \tag{3.54}$$

where the costates are $\mathbf{p} = [p_1, \ldots, p_6]$. Evaluating Eq. (3.38c) onto Eq. (3.54),

$$\frac{\partial H}{\partial \mathbf{u}^*} = \rho\mathbf{u}^*(t) + \mathbf{p}^*(t)^T\mathbf{B} = 0 \tag{3.55}$$

provides the general optimal control law for linear system dynamics:

$$\mathbf{u}^*(t) = -\frac{1}{\rho}\mathbf{p}^*(t)^T\mathbf{B} \tag{3.56}$$

The optimal control law Eq. (3.56) is a function of the time-varying costates $\mathbf{p}^*(t)$. Since for the double integrator dynamics the first $3x3$ elements of the $\mathbf{B}$ matrix are all zeros $\mathbf{0}_{3x3}$ from Eq. (3.40) and the bottom $3x3$ elements is the identity matrix $\mathcal{I}_{3x3}$, the control law is dependent of only the last three costates $p_4, \ldots, p_6$. The control law can then be simplified as:

$$\mathbf{u}^*(t) = -\frac{1}{\rho}\begin{bmatrix} p_4^*(t) \\ p_5^*(t) \\ p_6^*(t) \end{bmatrix} \tag{3.57}$$

Next, the general optimal control law from Eq. (3.56) is plugged into the differential dynamics constraint Eq. (3.38a):

$$\dot{\mathbf{x}}^*(t) = \mathbf{A}\mathbf{x}^*(t) - \frac{1}{\rho}\mathbf{B}\mathbf{p}^*(t)^T\mathbf{B} \tag{3.58}$$

For the double integrator dynamics, this equation is simplified by using the control law from Eq. (3.57) instead:

$$\dot{\mathbf{x}}^*(t) = \mathbf{A}\mathbf{x}^*(t) - \frac{1}{\rho}\mathbf{B}\begin{bmatrix} p_4^*(t) \\ p_5^*(t) \\ p_6^*(t) \end{bmatrix} \tag{3.59}$$

This forms the six state differential equations of the Euler-Lagrange equations. Next, the differential costate equations are found from Eq. (3.38b), which requires only the terms with the state variable $\mathbf{x}(t)$ from the Hamiltonian Eq. (3.53):

$$H(\mathbf{x}(t)) = \alpha g_r(d_r)g_v(v_r) + \mathbf{p}^T(t)\mathbf{A}\mathbf{x}(t) \tag{3.60}$$

Evaluating Eq. (3.38b) onto Eq. (3.60):

$$\dot{\mathbf{p}}^* = -\alpha\frac{\partial g_r(d_r)g_v(v_r)}{\partial \mathbf{x}^*} - \mathbf{p}^{*T}(t)\mathbf{A} \tag{3.61}$$

Observing $\mathbf{p}^{*T}(t)\mathbf{A}$ for the double integrator dynamics signifies that only the first three costates $p_1, \ldots, p_3$ couple into these six differential equations. However, the last three costates $p_4, \ldots, p_6$ are part of the state differential equations. Eq. (3.61) can be simplified to:

$$\dot{\mathbf{p}}^* = -\alpha\frac{\partial g_r(d_r)g_v(v_r)}{\partial \mathbf{x}^*} - \begin{bmatrix} \mathbf{0}_{3x1} \\ p_1^*(t) \\ p_2^*(t) \\ p_3^*(t) \end{bmatrix} \tag{3.62}$$

The twelve coupled nonlinear differential Euler-Lagrange equations are formed with equations (3.59) and (3.62). These equations are solved using a collocation method based on [16]. The numerical approach uses piecewise cubic polynomials to approximate the solution at predefined mesh points. Table 3.2 on page 79 describes all the necessary steps in detail for the variational approach to trajectory planning for docking scenarios. Several details are extracted from the summary of the problem formulation for docking in Table 3.1. Once the solution to the Euler-Lagrange equations is found, then the path is defined by the optimal state trajectory $\mathbf{x}^*(t)$ and the optimal control trajectory $\mathbf{u}^*(t)$ can be evaluated with Eq. (3.57) using the optimal costates $\mathbf{p}^*(t)$. The numerical computation of the solution to the Hamiltonian Boundary Value Problem is significantly high. This is due to the complexity of solving a boundary value problem. Also, a solution is not guaranteed as the numerical methods require a good initial guess of the solution. There are different approaches that attempt to improve the computation of solving the optimal control problem [5]. The variational technique algorithm also requires large enough memory for all the necessary computation to code. The next section develops a sub-optimal path planner that is highly efficient in computation time and requires little memory for storage.

Table 3.2: Variational Technique to Optimal Trajectory Planning Algorithm for Docking.

Action	Formulas	Equations
1. Define fixed final time	t_f	
2. Define radii	R - chaser, R_{obs} - obstacle	
3. Define weights	$\rho \geq 0$ - weight on control input $\alpha \geq 0$ - weight on obstacle clearance	
4. Define obstacle terms	s - buffer distance (where cost is non-zero) k - maximum cost at zero relative distance v_c - connecting velocity near $v_r = 0$	
5. Solve for coefficients	$\begin{bmatrix} 0 & 0 & 0 & 1 \\ s^3 & s^2 & s & 1 \\ 3s^2 & 2s & 1 & 0 \\ 6s & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} k \\ 0 \\ 0 \\ 0 \end{bmatrix}$	Eq. (3.45)
	$\begin{bmatrix} v_c^3 & v_c^2 & v_c & 1 \\ 3v_c^2 & 2v_c & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 6v_c & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} v_c \\ 1 \\ 0 \\ 0 \end{bmatrix}$	Eq. (3.49)
6. Form initial state	$\mathbf{x}(t_0) = \mathbf{x}_0 \in \Re^6$ initial state of chaser spacecraft at t_0	
7. Form obstacle init state	$\mathbf{x}_{obs}(t_0) = \mathbf{x}_{obs,0} \in \Re^{13}$ initial state in $\Re^{13}$ of target spacecraft	
8. Propagate $\mathbf{x}_{obs}(t_0)$ to t_f	$\mathbf{x}_{obs}(t) = \mathbf{x}_{obs}(t_0) + \int_{t_0}^{t_f} \left\{ \begin{bmatrix} \mathbf{v}_{obs} \\ \mathbf{0} \\ \frac{1}{2}\boldsymbol{\Omega}(\omega)\mathbf{q}_{obs} \\ \mathbf{I}_{obs}^{-1}\left[-\boldsymbol{\omega}_{obs} \times (\mathbf{I}_{obs}\boldsymbol{\omega}_{obs})\right] \end{bmatrix} \right\} dt$ integrate numerically for $\mathbf{x}_{obs}(t_f) \in \Re^{13}$	Eq. (3.24)
9. Form final state	$\mathbf{x}(t_f) = \begin{bmatrix} \mathbf{R}(\mathbf{q}_{obs}(t_f))\mathbf{z}_{body} + \mathbf{r}_{obs}(t_f) \\ \mathbf{R}(\mathbf{q}_{obs}(t_f))\left[\boldsymbol{\omega}_{obs}(t_f) \times \mathbf{z}_{body}\right] + \mathbf{v}_{obs}(t_f) \end{bmatrix}$ use $\mathbf{x}_{obs}(t_f) \in \Re^{13}$ to form $\mathbf{x}(t_f) \in \Re^6$	Eq. (3.31)
10. Euler-Lagrange Eqns.	$\dot{\mathbf{x}}^*(t) = \mathbf{A}\mathbf{x}^*(t) - \frac{1}{\rho}\mathbf{B}\mathbf{p}^*(t)^T\mathbf{B}$ $\dot{\mathbf{p}}^* = -\alpha\frac{\partial g_r(d_r)g_v(v_r)}{\partial \mathbf{x}^*} - \mathbf{p}^{*T}(t)\mathbf{A}$ optimal state trajectory is given by $\dot{\mathbf{x}}^*(t)$	Eq. (3.58) Eq. (3.61)

3.4 Spline-Based Trajectory Planning Algorithm

A trajectory planning algorithm that requires low memory and computation power is an ideal algorithm for spacecraft hardware implementation. The main reason being is the limited power a satellite in space can provide to its online computer. An efficient algorithm can also be faster implemented and could undergo sooner in-space testing for advancing its TRL level [10]. There already exist numerous efficient sub-optimal planning algorithms such as Rapid Exploring Random Search Trees (RRTs) [8]. However, it may be required to call the planning algorithm several times for re-planning, so it is desired to have an algorithm that will always output a unique trajectory for the same conditions. Therefore, the RRTs are unsatisfactory as they provide a different trajectory after another execution of the same conditions due to its stochastic nature of planning. The following path planning algorithm finds a unique energy sub-optimal trajectory for the spacecraft while avoiding spherical obstacles. The main approach for avoiding obstacle constraints is done by introducing intermediate way-point states that deviate an initial unconstrained trajectory into the feasible state-space. If the initial unconstrained trajectory does not pass through any obstacles, then the solution is a fully energy optimal trajectory. The introduction of way-points by the planner diminish it to be sub-optimal. The development of the planner is similar to work done by [18].

The problem formulation of path planning for docking scenarios, summarized in Table 3.1 on page 65, is used in developing the algorithm. First, the double integrator dynamics are still used as the spacecraft equations of motion Eq. (3.14):

$$\ddot{\mathbf{r}} = \mathbf{a}$$

An obstacle is modeled as a moving collision sphere defined from the center of the spacecraft,

$$\|\mathbf{r}(t) - \mathbf{r}_{obs}(t)\|_2 \geq (R + R_{obs} + R_{buffer}) \tag{3.63}$$

where $\|\cdot\|_2$ specifies the euclidean 2-norm, $\mathbf{r}_{obs}(t)$ is the center position of the time-varying obstacle, R is a positive scalar radius length of the spacecraft, R_{obs} is the radius length of the obstacle, and R_{buffer} is an added safety zone that accounts for measurement and process uncertainty. The last term is similar to the buffer distance s in the variational technique to path planning in Section 3.3.2. The initial and final conditions for the planning algorithm are,

$$\mathbf{x}(t_0) = \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{v}_0 \end{bmatrix}, \quad \mathbf{x}(t_f) = \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \tag{3.64}$$

where $\mathbf{x}(t_0)$ represents the initial state of position $\mathbf{r}_0$ and velocity $\mathbf{v}_0$ at time t_0, and $\mathbf{x}(t_f)$ is the final state at time t_f. Computing the terminal conditions for spacecraft docking is described in Section 3.2.3. For the rest of the algorithm development, the initial time is set to zero:

$$t_0 = 0 \tag{3.65}$$

The final time t_f is fixed and predefined by the user. The user is considered to be MVM module from the GN&C architecture in Chapter 2. When obstacle constraints are active, several position *way-points* may be introduced by the planning algorithm for the spacecraft to pass through,

$$\mathbf{r}_w^i = \begin{bmatrix} \mathbf{r}_w^1 & \mathbf{r}_w^2 & \cdots & \mathbf{r}_w^N \end{bmatrix} \tag{3.66a}$$

$$t_w^i = \begin{bmatrix} t_w^1 & t_w^2 & \cdots & t_w^N \end{bmatrix} \tag{3.66b}$$

where $\mathbf{r}_w^i$ is the i^{th} way-point position at time t_w^i. A depiction describing the terminal states, way-points, and a collision sphere is shown in Figure 3-12. The objective of the planning algorithm is to find an *energy sub-optimal control trajectory* $\mathbf{a}^*(t)$, $t \in [t_0, t_f]$ that minimizes the performance metric Eq. (3.11),

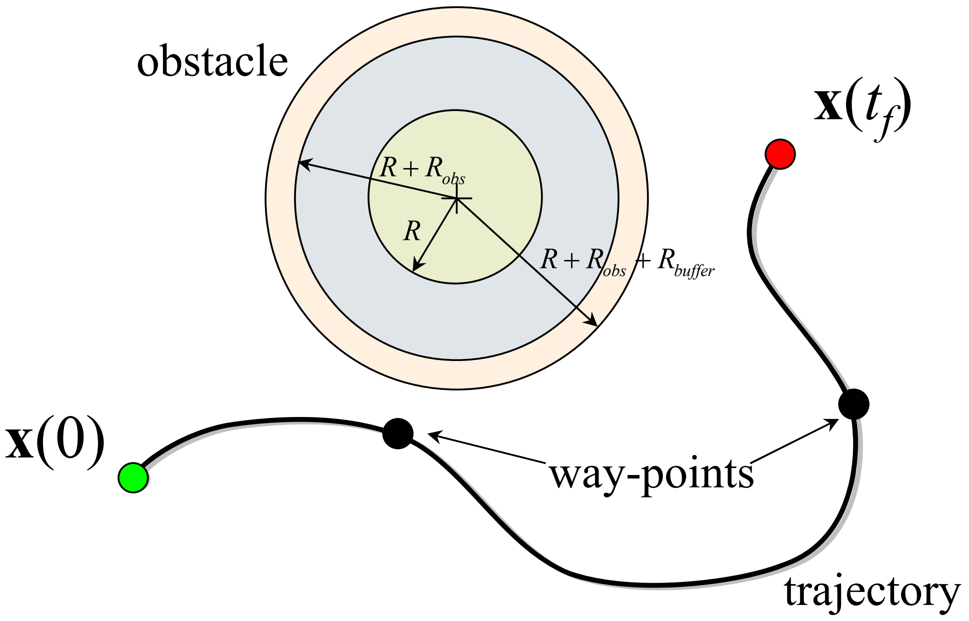


Figure 3-12: Obstacle Sphere and Way-Points Depiction

$$J = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{a}^T(t)\mathbf{a}(t)dt$$

while satisfying differential constraints of Eq. (3.14) and passes through way-points $\mathbf{r}_w^i$. The total energy is directly related to the fuel consumed by the spacecraft, thus optimizing for energy also minimizes fuel consumption. The approach of the planning algorithm will be to first calculate the optimal trajectory without obstacles and then introduce way-points that move the trajectory outside any obstacles. A depiction of the approach is shown Figure 3-13 while the details follow next.

Sultan [18] provided an important result that the energy optimal trajectory of $\ddot{\mathbf{r}} = \mathbf{a}$ dynamics is a piecewise cubic polynomial of class C^1 in time. The result is restated and extended to specify that the cubic polynomials are equivalent to cubic splines. The Lemma 1 considers the optimum trajectory $r \in \Re^1$ in one dimension along a single axis from $\mathbf{r} \in \Re^3$.

Lemma 1 (**Energy Optimal Trajectory**). *Given the differential constraints of $\ddot{r} = a$, a minimum energy trajectory with terminal states $\mathbf{x}(t_0) \in \Re^2$ and $\mathbf{x}(t_f) \in \Re^2$,*

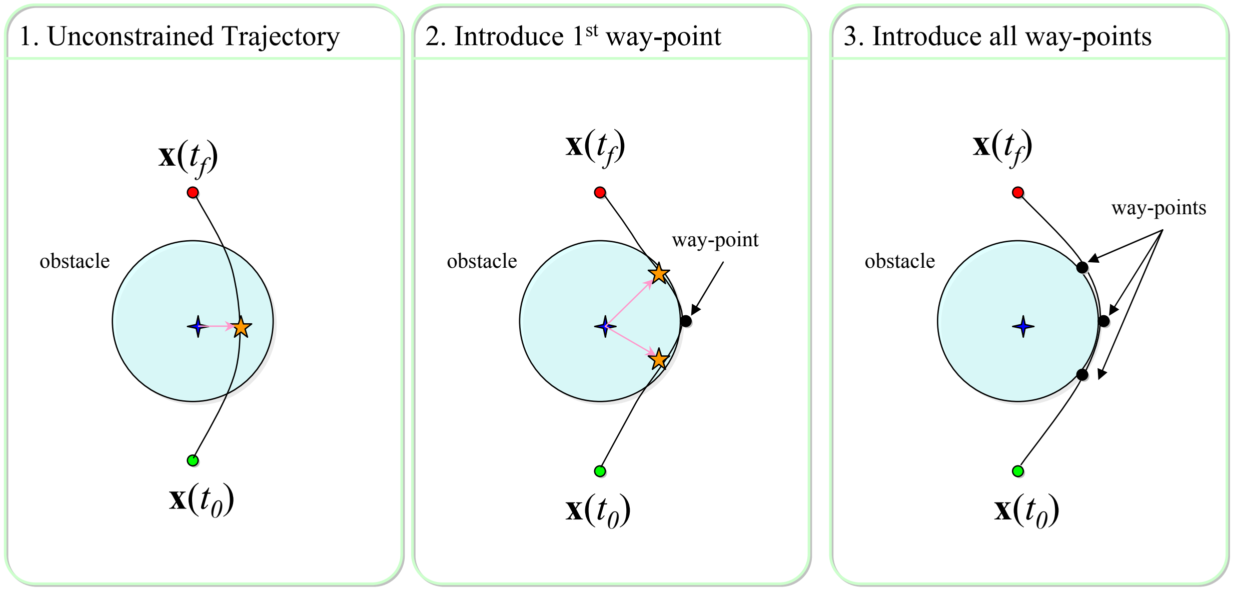


Figure 3-13: Process of the Spline-Based Planning Algorithm

and passing through a sequence of way-points $[r_w^i, i = 1, \ldots, N]$ *is a piecewise cubic spline,*

$$r(t) = \mathcal{A}r_w^i + \mathcal{B}r_w^{i+1} + \mathcal{C}a_i + \mathcal{D}a_{i+1} \tag{3.67a}$$

$$v(t) = \frac{r_w^{i+1} - r_w^i}{t_w^{i+1} - t_w^i} - \frac{3\mathcal{A}^2 - 1}{6}\left(t_w^{i+1} - t_w^i\right)a_i + \frac{3\mathcal{B}^2 - 1}{6}\left(t_w^{i+1} - t_w^i\right)a_{i+1} \tag{3.67b}$$

where [19],

$$\begin{aligned}
\mathcal{A} &= \frac{t_w^{i+1} - t}{t_w^{i+1} - t_w^i} \\
\mathcal{B} &= \frac{t - t_w^i}{t_w^{i+1} - t_w^i} \\
\mathcal{C} &= \frac{1}{6}\left(\mathcal{A}^3 - \mathcal{A}\right)\left(t_w^{i+1} - t_w^i\right)^2 \\
\mathcal{D} &= \frac{1}{6}\left(\mathcal{B}^3 - \mathcal{B}\right)\left(t_w^{i+1} - t_w^i\right)^2
\end{aligned} \tag{3.68}$$

and a_i *is the* 2^{nd} *derivative (acceleration) at way-point* r_w^i *to be determined by satis-*

fying the 1th derivative continuity between all the intervals [19]. The 2nd derivative optimal control trajectory is a continues piecewise set of linear functions:

$$a(t) = \mathcal{A}a_i + \mathcal{B}a_{i+1} \tag{3.69}$$

Proof. Given a single interval $[t_w^i, \quad t_w^{i+1}]$ in one dimension, let's find the optimal control input $a(t)$ that minimizes the energy cost functional Eq. (3.11),

$$J = \frac{1}{2}\int_{t_w^i}^{t_w^{i+1}} a^2(t)dt$$

constrained to the double integrator dynamics with state $\mathbf{x} = [r \quad v]^T$,

$$\dot{\mathbf{x}} = \begin{bmatrix} v \\ a \end{bmatrix} \tag{3.70}$$

and boundary conditions:

$$\mathbf{x}(t_w^i) = \begin{bmatrix} r_w^i \\ v(t_w^i) \end{bmatrix}, \quad \mathbf{x}(t_w^{i+1}) = \begin{bmatrix} r_w^{i+1} \\ v(t_w^{i+1}) \end{bmatrix} \tag{3.71}$$

The calculus of variations method provides the necessary conditions for first-order optimality with equations (3.38a), (3.38b) and (3.38c). The Hamiltonian is formed as,

$$H = \frac{1}{2}a^2 + p_1 v + p_2 a \tag{3.72}$$

where p is the costate variable. Then using Eq. (3.38c),

$$\frac{\partial H}{\partial u} = \frac{\partial H}{\partial a} = a + p_2 = 0 \tag{3.73}$$

and then determining the costates with Eq. (3.38b):

$$
\begin{aligned}
\dot{p_1} &= -\frac{\partial H}{\partial r} = 0 \\
\dot{p_2} &= -\frac{\partial H}{\partial v} = -p_1
\end{aligned}
\tag{3.74}
$$

Combing equations (3.73) and (3.74) yields the optimal control input $a(t)$,

$$
a(t) = b_i t + c_i \tag{3.75}
$$

where b_i and c_i are constants. Inputing the optimal control profile Eq. (3.75) into the system dynamics Eq. (3.70) and applying the boundary conditions from Eq. (3.71) provides the state trajectory $[r(t) \quad v(t)]^T$ and control input $a(t)$ given by Lemma 1.

$\square$

As a result of Lemma 1, a cubic spline interpolation algorithm is used to calculate the optimal trajectory [19]. The execution of this algorithm may be called several times as additional way-points may be introduced for obstacle avoidance. This process of the algorithm is formalized as a sub-algorithm in Figure 3-14 with respect to the overall planning algorithm. The inputs to the spline interpolation algorithm are: terminal states $\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$, position way-points $\mathbf{r}_w^i$, and time sequence t_{seq}, while the outputs are: state $\mathbf{x}(t)$ and control $\mathbf{u}(t)$ trajectories. In cubic spline terminology, the way-points are considered as *knots*, while the terminal states exhibit a *not-a-knot* condition. The *not-a-knot* condition let's the 1st derivative be predefined at the boundaries of the piecewise cubic spline, which corresponds to initial and final velocity conditions. Therefore, the full initial and final terminal states are satisfied in addition to the way-points (knots). The algorithm is quite efficient, as it requires to solve a system of linear equations $Ax = b$ in which the A matrix has a tridiagonal form. The computational complexity of the spline interpolation is on $O(N + 2)$ for each axis by using the tridiagonal algorithm [19], where N is the number of way-points in addition to the two terminal conditions.

The first step of the planning algorithm is to determine the unconstrained tra-

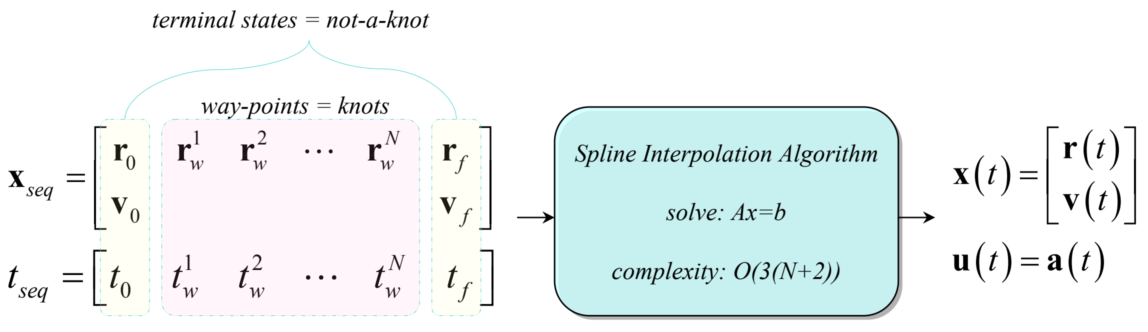

terminal states = not-a-knot
way-points = knots
$\mathbf{x}_{seq} = \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_w^1 & \mathbf{r}_w^2 & \cdots & \mathbf{r}_w^N & \mathbf{r}_f \\ \mathbf{v}_0 & & & & & \mathbf{v}_f \end{bmatrix}$
$t_{seq} = \begin{bmatrix} t_0 & t_w^1 & t_w^2 & \cdots & t_w^N & t_f \end{bmatrix}$
Spline Interpolation Algorithm
solve: Ax=b
complexity: O(3(N+2))
$\mathbf{x}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \mathbf{v}(t) \end{bmatrix}$
$\mathbf{u}(t) = \mathbf{a}(t)$

Figure 3-14: Cubic Spline Interpolation Algorithm

jectory with spline interpolation. The term *unconstrained* in this planning algorithm refers to a piece-wise cubic spline without any way-points that are *introduced* by the algorithm. Therefore, the sequence of states and their corresponding times is composed by only the terminal states,

$$\begin{aligned} \mathbf{x}_{seq} &= \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_f \\ \mathbf{v}_0 & \mathbf{v}_f \end{bmatrix} \\ t_{seq} &= \begin{bmatrix} 0 & t_f \end{bmatrix} \end{aligned} \tag{3.76}$$

where $t_0 = 0$ as the standard initial time. However, the user may initially provide several way-points that are desired for the spacecraft to pass through. This would also be considered an unconstrained trajectory as the algorithm did not introduce the way-points for obstacle avoidance. The user must make sure that the desired way-points are not inside any obstacles. Then the initial state sequence input in Eq. (3.76) is expanded to include the user predefined way-points:

$$\begin{aligned} \mathbf{x}_{seq} &= \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}^1_{w,user} & \cdots & \mathbf{r}_f \\ \mathbf{v}_0 & & \cdots & \mathbf{v}_f \end{bmatrix} \\ t_{seq} &= \begin{bmatrix} 0 & t^1_{w,user} & \cdots & t_f \end{bmatrix} \end{aligned} \tag{3.77}$$

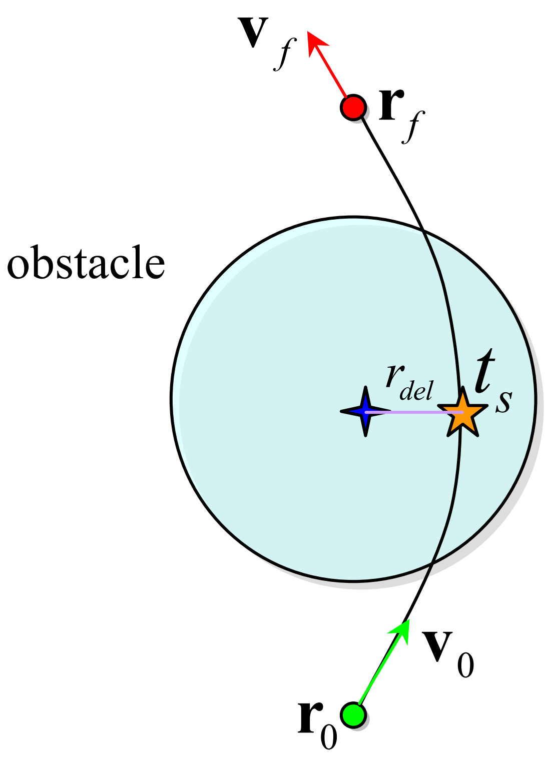


Figure 3-15: Minimum Distance Along Trajectory to Obstacle

where $\mathbf{r}^i_{w,user}$ and $\mathbf{v}^i_{w,user}$ are the user supplied way-points at times $t^i_{w,user}$. These way-points are not considered for the further development of the algorithm. Then the spline interpolation algorithm is executed with inputs Eq. (3.76) to determine the initial unconstrained trajectory.

Afterwards, a feasibility check is performed on the trajectory by making sure no trajectory position in time $\mathbf{r}(t)$ passes through any obstacles, satisfying Eq. (3.63). This is accomplished by first finding the minimum distance along the trajectory to the center of the obstacle, see Figure 3-15. Then the problem formulation is a one-dimensional nonlinear minimization,

$$t_s, r_{del} \longleftarrow \min_t \|\mathbf{r}(t) - \mathbf{r}_{obs}(t)\|_2 \tag{3.78}$$

where the outputs r_{del} is the scalar minimum distance between the trajectory and obstacle at time t_s. The algorithm used for performing the one-dimensional optimization is *Brent's method* of parabolic interpolation [19]. The initial trajectory is feasible if;

$$r_{del} \geq (R + R_{obs} + R_{buffer}) \tag{3.79}$$

otherwise, a new way-point is introduced at time t_s to move the trajectory position $\mathbf{r}(t_s)$ into the feasible state-space. This is accomplished by translating the current point $\mathbf{r}(t_s)$ linearly outward the obstacle. The direction of translating the infeasible position is a unit direction vector from the center of the obstacle to $\mathbf{r}(t_s)$:

$$\hat{\mathbf{r}}_{out} = \frac{\mathbf{r}(t_s) - \mathbf{r}_{obs}(t_s)}{\|\mathbf{r}(t_s) - \mathbf{r}_{obs}(t_s)\|} \tag{3.80}$$

So a new way-point is introduced for the piece-wise cubic spline trajectory at t_s by linearly translating position $\mathbf{r}(t_s)$ in the direction $\hat{\mathbf{r}}_{out}$ by the right amount to be outside be the obstacle,

$$\begin{aligned} \mathbf{r}_w^i &= \mathbf{r}(t_s) + (R_{out} - r_{del})\,\hat{\mathbf{r}}_{out} \\ t_w^i &= t_s \end{aligned} \tag{3.81}$$

where $R_{out} = R + R_{obs} + R_{buffer}$ is the total radius of the obstacle sphere, and $i = 1$ for the first introduction of a way-point. A situation when this method does not work well is when the position $\mathbf{r}(t_s)$ lies on or very close to the center of the obstacle $r_{obs}(t_s)$. If the closest position along the trajectory $\mathbf{r}(t_s)$ to the obstacle is less than some tolerance $r_{del,tol}$,

$$r_{del} < r_{del,tol} \tag{3.82}$$

then a random direction is chosen for $\hat{\mathbf{r}}_{out}$:

$$\hat{\mathbf{r}}_{out} = \frac{RAND_{3x1}}{\|RAND_{3x1}\|} \tag{3.83}$$

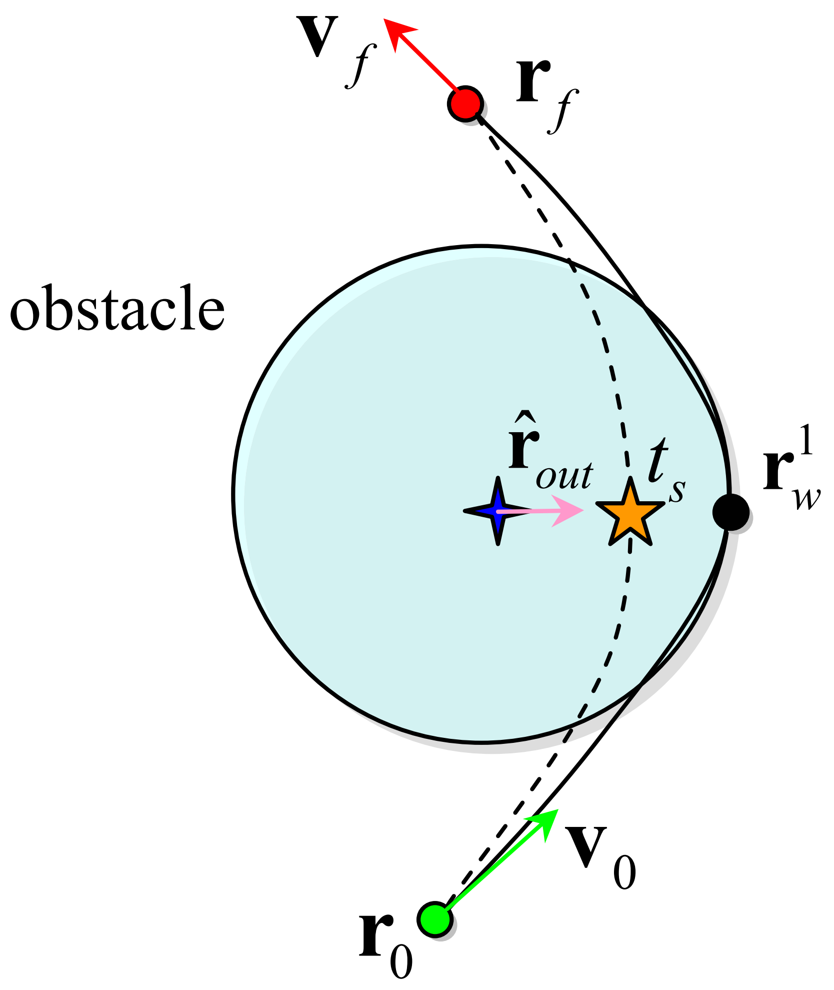


Figure 3-16: Introducing the First Waypoint

After the first new way-point $\mathbf{r}_w^1$ is determined, the spline interpolation algorithm is executed again with inputs,

$$\begin{aligned} \mathbf{x}_{seq} &= \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_w^1 & \mathbf{r}_f \\ \mathbf{v}_0 & & \mathbf{v}_f \end{bmatrix} \\ t_{seq} &= \begin{bmatrix} 0 & t_w^1 & t_f \end{bmatrix} \end{aligned} \tag{3.84}$$

to attain a new state trajectory $\mathbf{r}(t)$ and $\mathbf{v}(t)$ that replaces the previous unconstrained trajectory, see Figure 3-16. Afterwards, the new trajectory is verified again for feasibility by evaluating Eq. (3.78) and checking the obstacle constraint Eq. (3.79). If the new piece-wise cubic spline trajectory is feasible, it is returned as $\mathbf{x}^*(t)$ and the control trajectory $\mathbf{a}^*(t)$. The asterisk denotes the energy sub-optimal trajectory that satisfies collision avoidance constraints. If the new trajectory is not feasible by having $r_{del} < R_{out}$, then another way-point is introduced with equations (3.80) and (3.81). This cycle is repeated until the state trajectory satisfies obstacle constraints. A depiction of the process of the algorithm is shown in Figure 3-13. Also, a detailed description is shown in Algorithm 3.1. The spline-based methods that's implemented and tested as in Algorithm 3.1 does not handle moving obstacles and $\mathbf{r}_{obs}$ is not a function of time in this case. The algorithm can be extended to account time-varying obstacles by replacing any $\mathbf{r}_{obs}$ within the algorithm with $\mathbf{r}_{obs}(t)$. The initial $\mathbf{x}(t_0)$ and final $\mathbf{x}(t_0)$ states supplied to the spline-based algorithm is computed from Table 3.1 on page 65.

An energy sub-optimal trajectory planning algorithm is developed that is based on piece-wise cubic spline interpolation with way-point introduction. The way-points are introduced if the a trajectory passes through an obstacle sphere in a way that deviates the path to be feasible. This is accomplished by moving the closest position along the trajectory to the obstacle outwards the sphere linearly into the feasible state-space. However, if that trajectory passes through the center of the obstacle sphere, then a random direction is chosen for introducing the new way-point. The algorithm repeats this process until the complete trajectory does not avoid the obstacle constraint. If the initial trajectory does not pass through any obstacles and so no way-points are introduced by the algorithm, then the given trajectory is completely energy optimal. A comparison of the computational and energy cost performance between the variational technique to path planning from Section 3.3.2 and the spline-based algorithm is discussed in the following section.

Algorithm 3.1: Spline-Based Trajectory Planning Algorithm

```
input  : $t_f$, $\mathbf{x}(t_0)$, $\mathbf{x}(t_f)$, $\mathbf{r}_{obs}$, $R$, $R_{obs}$, $R_{buffer}$, $r_{del,tol}$
output: $\mathbf{r}^*(t)$, $\mathbf{v}^*(t)$, $\mathbf{a}^*(t)$

$R_{out} = R + R_{obs} + R_{buffer}$
$\mathbf{x}_{seq} = \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_f \\ \mathbf{v}_0 & \mathbf{v}_f \end{bmatrix}, \quad t_{seq} = \begin{bmatrix} 0 & t_f \end{bmatrix}$
$\mathbf{r}(t)$, $\mathbf{v}(t)$, $\mathbf{a}(t) \longleftarrow spline(t_{seq}, \quad \mathbf{x}_{seq})$
$t_s, r_{del} \longleftarrow \min_t \|\mathbf{r}(t) - \mathbf{r}_{obs}\|_2$
$i = 1$
while $r_{del} < R_{out}$ do
    if $r_{del} \geq r_{del,tol}$ then
        $\hat{\mathbf{r}}_{out} = \frac{\mathbf{r}(t_s) - \mathbf{r}_{obs}}{\|\mathbf{r}(t_s) - \mathbf{r}_{obs}\|}$
    else
        $\hat{\mathbf{r}}_{out} = \frac{RAND_{3x1}}{\|RAND_{3x1}\|}$
    end
    $\mathbf{r}_w^i = \mathbf{r}(t_s) + (R_{out} - r_{del})\,\hat{\mathbf{r}}_{out}$
    $t_w^i = t_s$
    $\mathbf{x}_{seq} = \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_w^i & \mathbf{r}_f \\ \mathbf{v}_0 & & \mathbf{v}_f \end{bmatrix}, \quad t_{seq} = \begin{bmatrix} 0 & t_w^i & t_f \end{bmatrix}$
    $\mathbf{r}(t)$, $\mathbf{v}(t)$, $\mathbf{a}(t) \longleftarrow spline(t_{seq}, \quad \mathbf{x}_{seq})$
    $t_s, r_{del} \longleftarrow \min_t \|\mathbf{r}(t) - \mathbf{r}_{obs}\|_2$
    $i = i + 1$
end
return $\mathbf{r}^*(t)$, $\mathbf{v}^*(t)$, $\mathbf{a}^*(t)$
```

3.5 Comparison of Trajectory Planning Algorithms

Two trajectory planning methods are developed. One based on the calculus of variation method for optimal control, which requires to solve a difficult Hamiltonian boundary value problem. The second uses cubic spline interpolation and clever way-point introduction to form an energy sub-optimal trajectory. Since the variational method solves the necessary conditions of optimality for the problem provided, it is used as a reference planner for the spline-based algorithm. The goal of the spline-based planner is to achieve similar trajectories with minimal cost difference with respect to the variational technique to planning. Therefore, a comparison of the two planner is studied at determining how efficient the new spline-based algorithm performs. In addition, the computational performance is compared for the SPHERES application. A discussion on each planners' ability for practical use and implementation concludes the comparison.

The planners will be compared in three different docking scenarios from the simplest one that does not require obstacle avoidance to the most complex tumbling dynamics of the target spacecraft considered in this thesis. In all scenarios, the target spacecraft is fixed in position, so no moving obstacles are considered. Since there are two phases of a docking mission that use the path planner as described in Chapter 2, the more elaborate phase **DP Axis Alignment** is chosen. These scenarios are described below while a depiction of the initial configuration is shown in Figure 3-17.

Docking to Fixed Target Facing Forwards This scenario has both spacecraft face each other for their initial configuration. Since the target spacecraft is fixed, meaning not rotating nor translating, the initial configuration stays constant throughout the docking scenario. The scenario does not require the planners to consider obstacle avoidance, but will be used show reasonable planning for the simplest setup.

Docking to Fixed Rotating Target In-Plane The target spacecraft performs a steady rotational tumble where the docking port axis sweeps the same plane as where the chaser spacecraft is initially located. The targets' position stays fixed through the scenario as the chaser needs to plan a path around the target spacecraft avoiding the obstacle.

Docking to Fixed Coning Target Facing Backwards In this scenario, the target spacecraft turns 180° to face its back to the chaser and performs a steady rotation where its rotation vector is not perpendicular to the docking port axis. This setup causes the docking port axis to sweep a cone. The chaser spacecraft needs again maneuver about the target obstacle and reach its final state.

Before executing either of the two planners, their required inputs such as terminal states, weight parameters, and coefficients are pre-computed and are not considered when analyzing each algorithms computation time. The algorithms are implemented in MATLAB®and executed on a PC with 2.16GHz Intel Core Duo®processor and 2GB of RAM. The trajectory planning is applied using the SPHERES parameters. The specific scenarios are discussed in the next section.

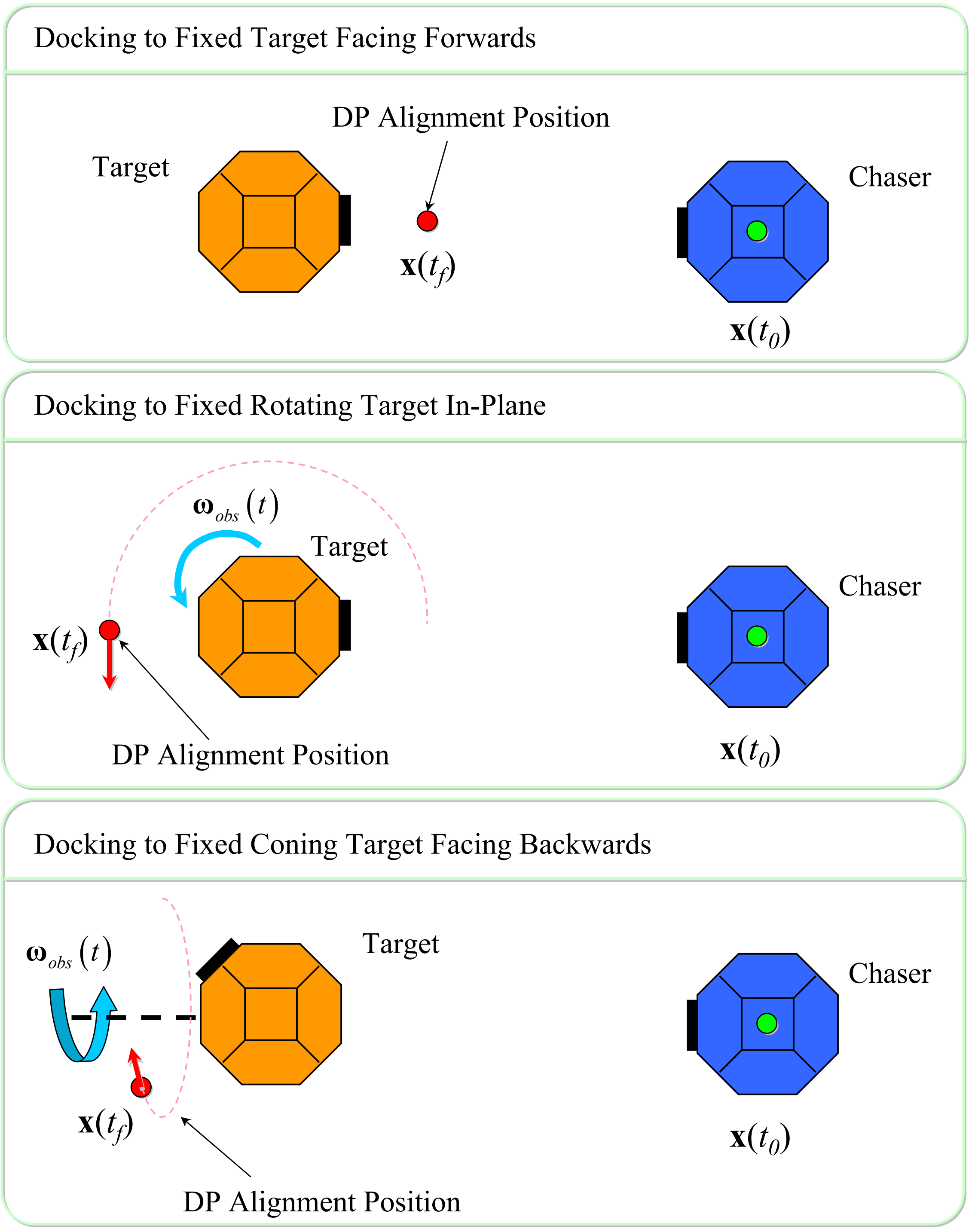


Figure 3-17: Docking Scenarios for Path Planner Comparison

3.5.1 Docking to Fixed Target Facing Forwards

This scenario tests the performance of the planning algorithms on the simplest case scenario. It should show reasonable trajectories from both planners in hope that they are identical. As mentioned that the spline-based planner provides a complete energy optimal trajectory when no obstacles are considered, then it should provide an identical solution to the reference planner. The position and attitude fixed target satellite provides a stationary final state for the chaser spacecraft, no final velocities. Therefore, the planners need to determine the energy optimal trajectory between two stationary points. The known result is a straight line that ends and stops at the terminal positions with zero velocities.

The variational technique path planning algorithm is considered to be solving the Euler-Lagrange equations equations (3.58) and (3.61), see Table 3.2 on page 79. All the required computation for the terms beforehand from Table 3.2 are not considered in the computation time for the reference planner. Some of these pre-computed terms are also necassary for the spline-based planner, such as the terminal states $\mathbf{x}(0)$ and $\mathbf{x}(t_f)$. The computation time of the spline-based planner is composed of everything performed in Algorithm 3.1 on page 91. The inputs required for both planners is shown in Table 3.3 specific for the SPHERES application.

Figure 3-18 shows the computed position and velocity profiles from both planners. The spline-based algorithm computed and equivalent state trajectory as the variational technique to planning. The position path in Figure 3-18(a) shows that the chaser spacecraft translates along the x-axis from its initial position $0m$ to the final position $-0.45m$. Figure 3-18(b) shows that the velocity began initial from zero and ended at zero as well. This is expected and the resulting trajectory is a straight line between the stationary terminal positions. The solutions are reasonable and so there is a gain in confidence about the path planners.

The optimality of the trajectories is compared in Figure 3-19 with the corresponding control and energy profile. As the trajectories are equivalent, so are the control and energy profiles. Notice that a minimum energy control input is a linear func-

Table 3.3: Docking to Fixed Target Facing Forwards Scenario Planning Inputs.

Description	Value
mass of spacecraft	$m = 4.3kg$
final time	$t_f = 100$ seconds
radius of chaser	$R = 0.105m$
radius of target	$R_{obs} = 0.105m$
obstacle position	$\mathbf{r}_{obs} = [-0.7\ 0\ 0]^T$
initial state	$\mathbf{x}(0) = \mathbf{0}$
final state	$\mathbf{x}(t_f) = [-0.45\ 0\ 0\ 0\ 0\ 0]^T$ from Eq. (3.31)
Variational Technique to Planning Terms:	
weight on control input	$\rho = 10000$
weight on obstacle clearance	$\alpha = 100$
buffer distance	$s = 0.03m$
maximum cost at zero relative distance	$k = 0.08$
connecting velocity	$v_c = 0.005m/s$
polynomial coefficients	using equations (3.45) and (3.49)
Spline-Based Algorithm Terms:	
buffer distance	$R_{buffer} = 0.03m$
tolerance on r_{del}	$r_{del,tol} = 0.005m$

tion between $t_0 = 0$ and $t_f = 100$ that satisfies the terminal conditions in position and velocity. The total energy cost of the computed trajectory from both planning algorithm is the same, $2.246 \times 10^{-5} J$.

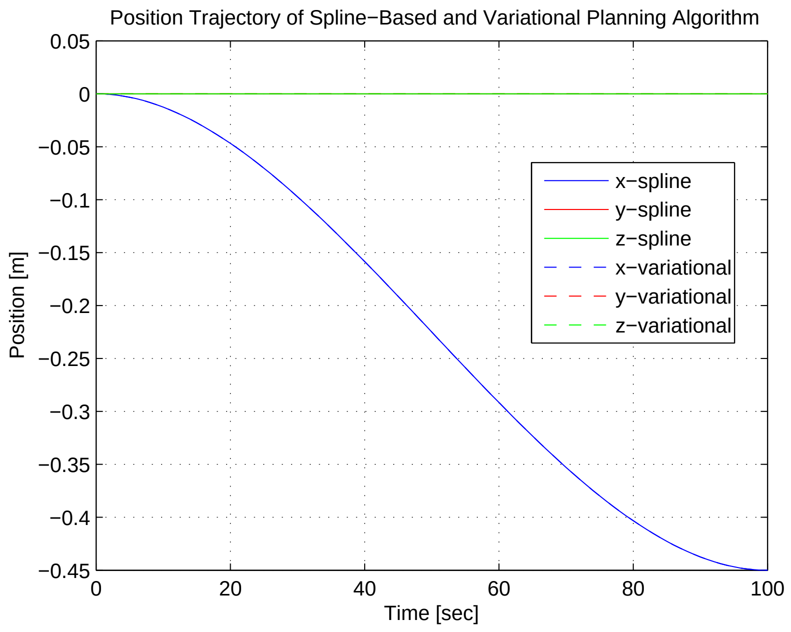
A 3D depiction of the energy optimal paths from both planners is shown in Figure 3-20. The Figure also shows the obstacle spheres where the smallest is that of the target spacecraft alone R_{obs}, the second largest is the combined target and chaser obstacle spheres $R + R_{obs}$, and the largest sphere is the addition of the buffer distance R_{buffer} or s. The green filled circle illustrates the initial position while the red shows the final. As expected, the trajectory is a straight line between the two terminal states.

Since this scenario does not include any obstacle avoidance, so the solution to the Euler-Lagrange equations does not take a lot of computation time as the cost functional is highly simplified. The spline-based algorithm is also fast as it did not introduce any way-points due to no obstacle constraint violation. The total energy cost of the trajectories and computation time for each algorithm is summarized in Table 3.4. The difference in the total energy is 0% for the spline-based algorithm, so it provides the most energy optimal trajectory. As expected, the spline-based algorithm is faster by approximately 2x than the variational technique to planning.

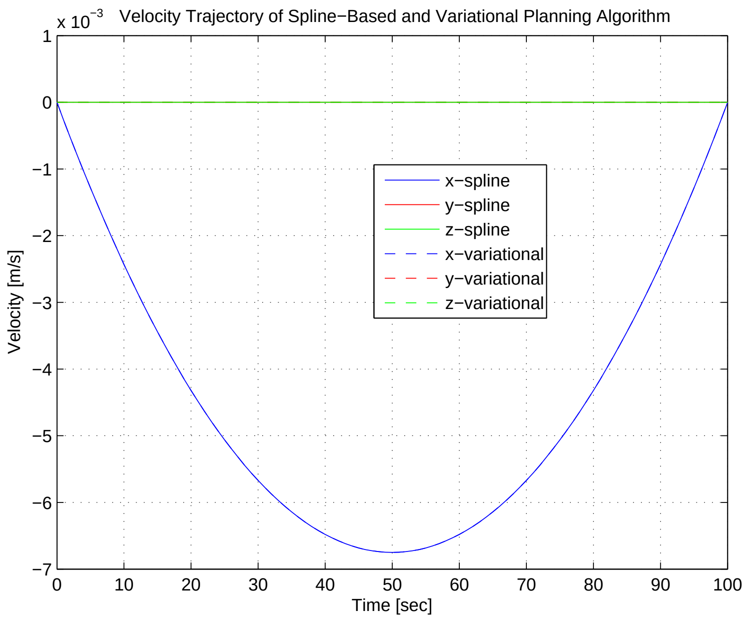
Table 3.4: Energy Cost and Computation Time for Fixed Target Facing Forwards Scenario.

	Variational Technique Planning	Spline-Based Algorithm
Total Energy Cost:	$2.246 \times 10^{-5}\ J$	$2.246 \times 10^{-5}\ J$
Energy Difference:		0%
Computation Time:	0.6756 seconds	0.3260 seconds
Times Faster:		2x

The two planners provided two identical solutions that are reasonable. This validates both algorithms at finding optimal trajectories without obstacle constraints. The next section considers a docking scenario when the obstacle has to be accounting in the planning.

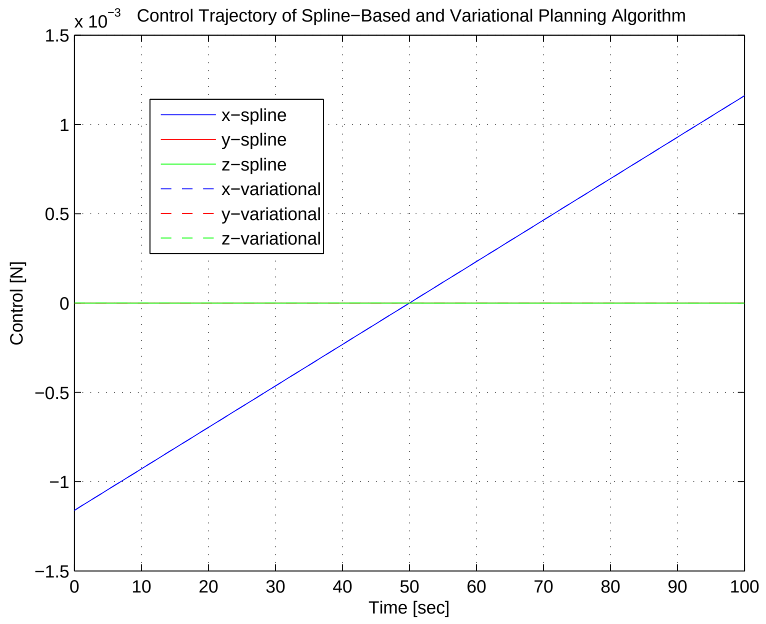


(a) Position Trajectory of Docking to Fixed Target Facing Forwards

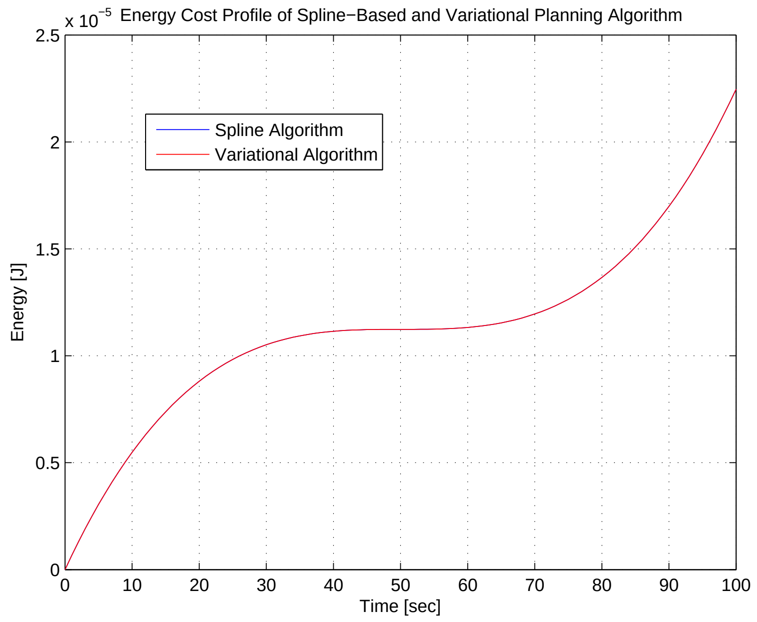


(b) Velocity Trajectory of Docking to Fixed Target Facing Forwards

Figure 3-18: State Trajectory of Docking to Fixed Target Facing Forwards



(a) Control Input of Docking to Fixed Target Facing Forwards



(b) Energy Profile of Docking to Fixed Target Facing Forwards

Figure 3-19: Control and Energy Profile for Docking to Fixed Target Facing Forwards

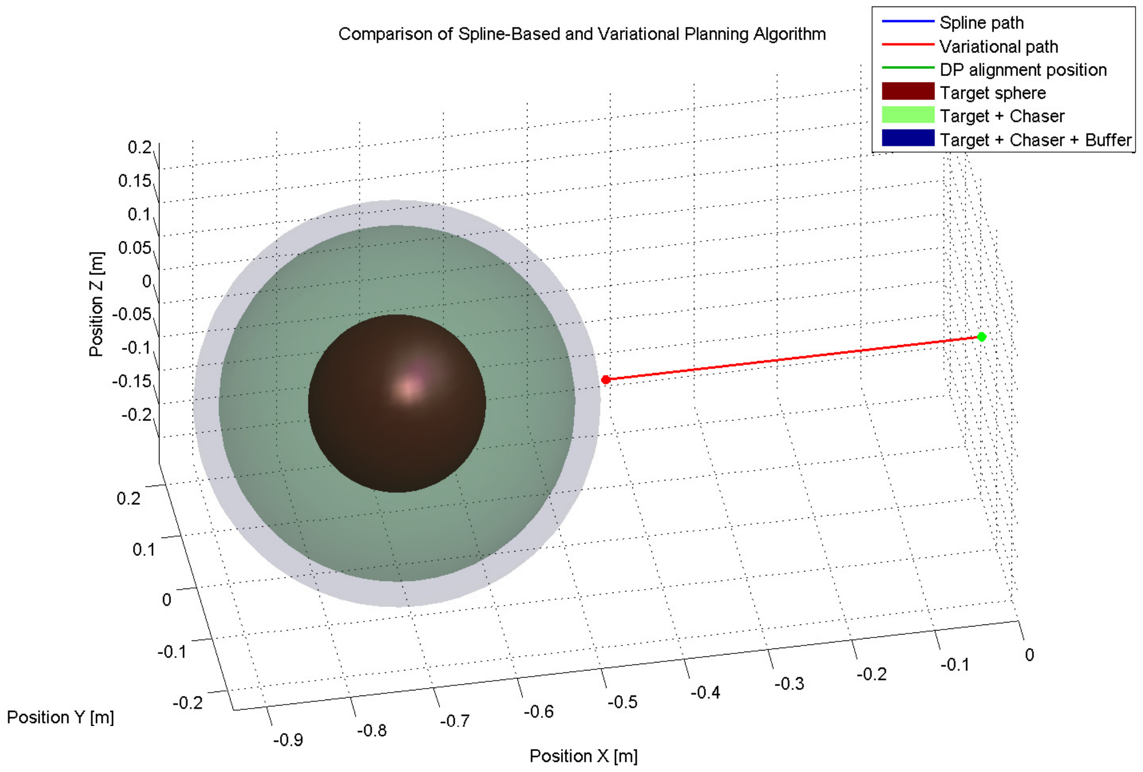


Figure 3-20: 3D Trajectory of Docking to Fixed Target Facing Forwards

3.5.2 Docking to Fixed Rotating Target In-Plane

This scenarios expands to the complexity of the docking scenario and the solution both planning algorithms need to compute. The planning is performed for a tumbling target spacecraft that is rotating at a rate of 2 deg/sec about the z-axis. The docking port axis of the target sweeps a plane in which the chaser spacecraft is initially located. This means that the chaser spacecraft needs to maneuver about the x and y axis to avoid the target obstacle and reach its final state. The initial configuration of both spacecraft has them facing each other. As the target performs a steady rotation at the rate of 2 deg/sec, its state at the final time $t_f = 100$ seconds will be a rotation by 200 degrees. This leads to a final state where the target is facing its back to the chaser and has a fixed angular rate. Therefore, the final state for the chaser is to be at the required distance infront of the docking port with a velocity vector in the direction of moving docking port axis. Computing the terminal states is described in Section 3.2.3.

The inputs to the two planning algorithms for this scenario is similar to that of the previous with the main change being the new terminal conditions. The final time $t_f = 100$ seconds, size of the obstacle, and obstacle position stays the same. However, the cost terms for the variational technique to planning need to be adjusted to achieve a reasonable solution. If all the cost weights stayed the same as in Table 3.3, with the only change to the new terminal conditions, then an invalid trajectory arises that goes through the obstacle, see Figure 3-21. Here, it emphasized on the importance of tuning cost weights for the variational method to path planning.

The red curve shows the trajectory from the variational method to planning as it goes through the target obstacle. The green curve shows the path of the docking port. The reason the planner computed such a trajectory is because there is not enough weight on the obstacle cost functional. Therefore, the value of α on the obstacle clearance is increased to $\alpha = 500$. The new inputs for docking to a fixed rotating target spacecraft scenario is summarized in Table 3.5. Notice in Table 3.5 that there are no tuning variables for the spline-based algorithm, just physical constraints on

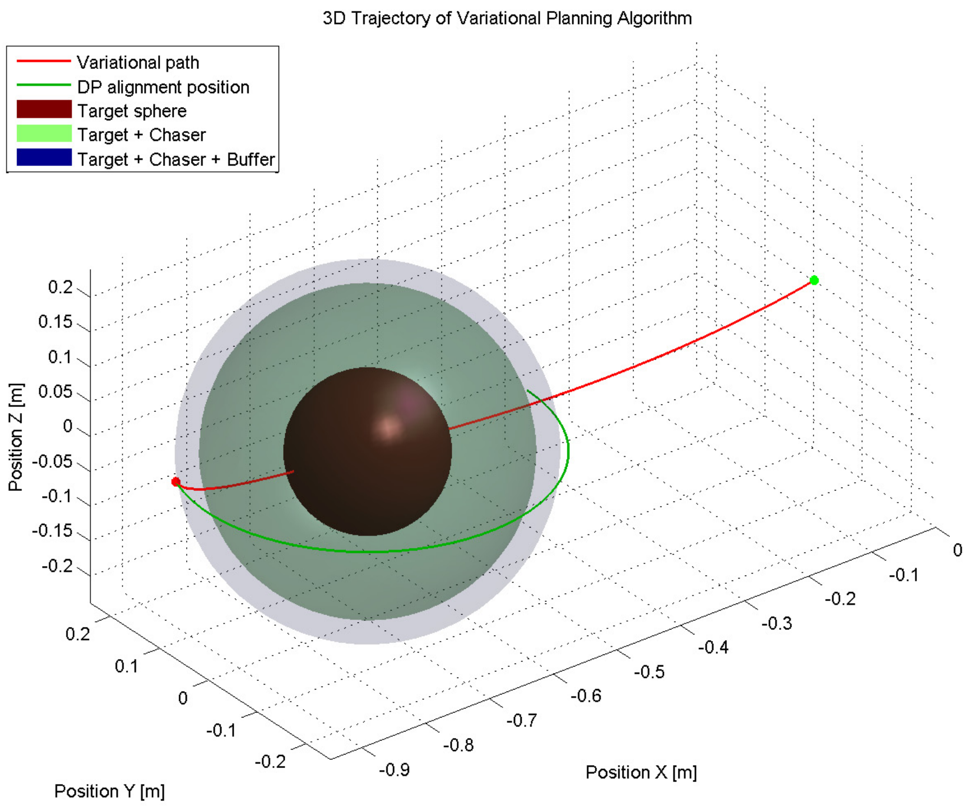


Figure 3-21: Invalid 3D Trajectory of Docking to Fixed Rotating Target In-Plane

the obstacle that obviously do not depend on the tumbling dynamics.

Table 3.5: Docking to Fixed Rotating Target In-Plane Planning Inputs.

Description	Value
mass of spacecraft	$m = 4.3kg$
final time	$t_f = 100$ seconds
radius of chaser	$R = 0.105m$
radius of target	$R_{obs} = 0.105m$
obstacle position	$\mathbf{r}_{obs} = [-0.7\ 0\ 0]^T$
initial state	$\mathbf{x}(0) = \mathbf{0}$
final state	$\mathbf{x}(t_f) = [-0.935\ 0.085\ 0\ 0.003\ 0.008\ 0]^T$ from Eq. (3.31)
Variational Technique to Planning Terms:	
weight on control input	$\rho = 10000$
weight on obstacle clearance	$\alpha = 500$
buffer distance	$s = 0.03m$
maximum cost at zero relative distance	$k = 0.08$
connecting velocity	$v_c = 0.005m/s$
polynomial coefficients	using equations (3.45) and (3.49)
Spline-Based Algorithm Terms:	
buffer distance	$R_{buffer} = 0.03m$
tolerance on r_{del}	$r_{del,tol} = 0.005m$

The state trajectories from both planners is shown in Figure 3-22. The spline-based algorithm trajectory follows a similar path as that from the reference planner with a maximum of about $10cm$ deviation. Even though the trajectories are not identical, both of them do not collide with the obstacle and meet at the same boundary conditions. Both paths do not show any sporadic behavior and behave smoothly.

The spline-based algorithm trajectory is slightly different from that of the reference, but the control and energy profiles in Figure 3-23 show that it is more energy optimal. The reason the variational method provides a less purely energy optimal trajectory is because it is developed to find the optimum solution to the cost functional provided in Eq. (3.41). This cost functional is composed of minimizing the energy and obstacle clearance. Thus, the method finds the optimum solution that trades off the cost between both factors, energy and obstacle clearance, and not purely the energy of the system. Therefore, if a larger weight is added to obstacle clearance, then a less energy optimum solution is computed. Tweaking these weights to find the most minimum energy trajectory while not reducing the cost on obstacle clearance enough for the trajectory to go through as in Figure 3-21 takes too many iteration. Therefore, the benchmark comparison is looked at observing that the spline-based algorithm is very close to the reference and does not need to be identical. The control profile in Figure 3-23(a) for the spline-based algorithm shows it is a piece-wise set of three linear functions as two way-points have been introduced at 64 and 78 seconds. The total energy for the spline-based algorithm is $2.393 \times 10^{-4} J$ and $2.469 \times 10^{-4} J$ for the variational method to planning.

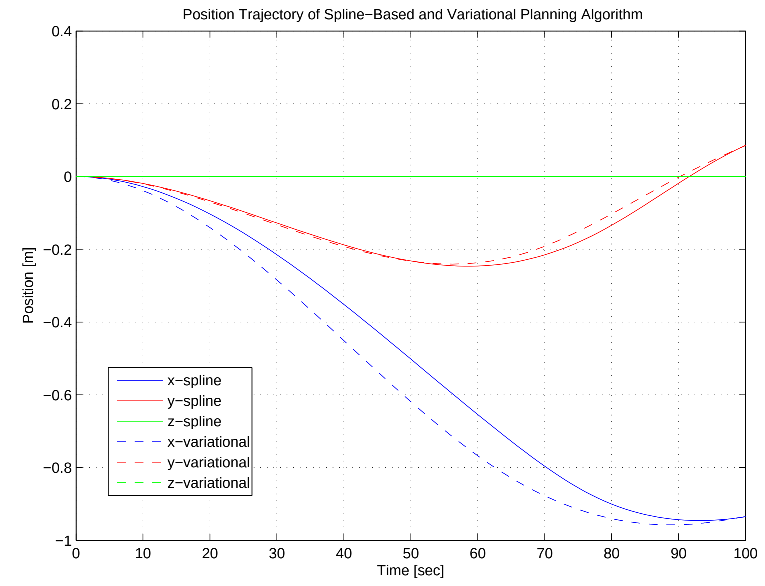
The 3D trajectories in Figure 3-24 show clearly that both paths do not interfere with the obstacle and are admissible. The reason the trajectories curve out to the negative y direction is due from the terminal velocity constraint that has the chaser spacecraft travel along the DP alignment position at t_f. As a result, the trajectories match the velocity of the time-varying DP alignment position at the final time.

The performance comparison of the two algorithms is summarized in Table 3.6. It shows that the spline-based algorithm total energy cost is a small 3% different than that of the reference planner. However, the computation time between the two planners is very significant. The spline-based algorithm took 12x less time to compute by introducing only 2 way-points for avoiding the obstacle.

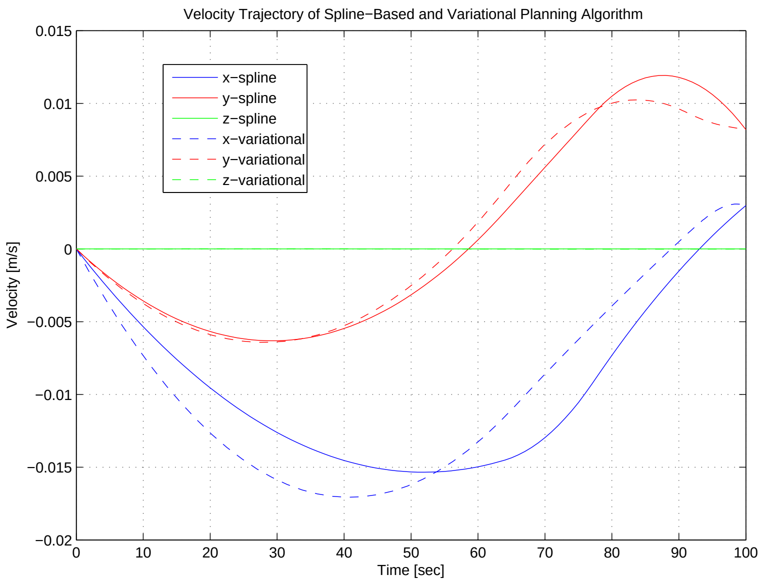
Table 3.6: Energy Cost and Computation Time for Fixed Rotating Target In-Plane Scenario.

	Variational Technique Planning	Spline-Based Algorithm
Total Energy Cost:	$2.469 \times 10^{-4}\ J$	$2.393 \times 10^{-4}\ J$
Energy Difference:		3.04%
Computation Time:	5.036 seconds	0.4112 seconds
Times Faster:		12x

This scenario showed the ability of both planners to plan a feasible path while satisfying obstacle constraints. Both planners minimized the energy as much as they are formulated to perform. Figure 3-21 emphasizes the importance of carefully choosing cost weights for the variational method to planning so that the trajectory will not pass through an obstacle. Therefore, this planner is not ideal for implementation for an autonomous docking system as the MVM module would have to perform the weights tuning. On the other hand, the spline-based algorithm does not require any parameter tuning. It also outperformed the variational method to planning in optimum energy and minimum computation time by being 12x faster. The next scenario is different in the tumbling dynamics of the target spacecraft but only adds slightly more complexity for the planners as the initial position of the chaser spacecraft is moving.

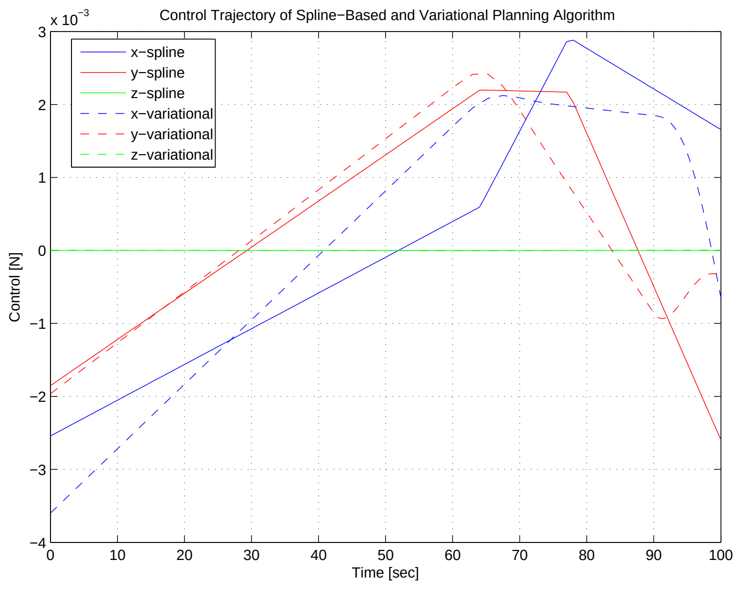


(a) Position Trajectory of Docking to Fixed Rotating Target In-Plane

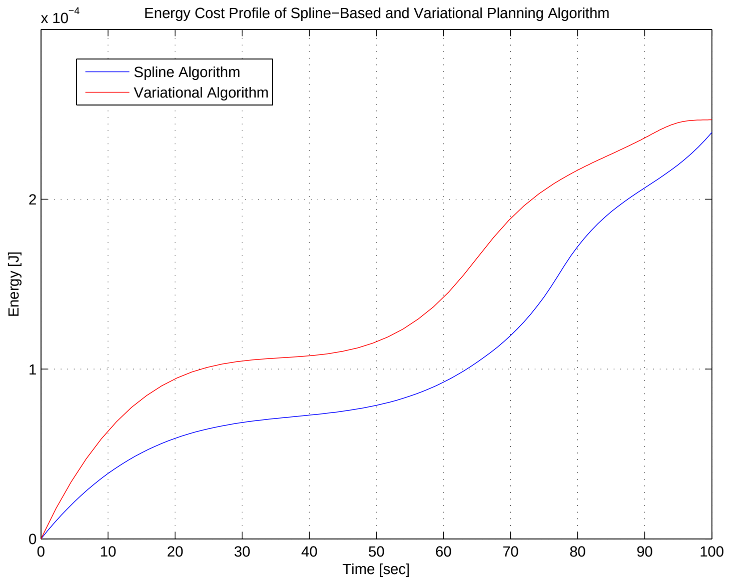


(b) Velocity Trajectory of Docking to Fixed Rotating Target In-Plane

Figure 3-22: State Trajectory of Docking to Fixed Rotating Target In-Plane



(a) Control Input of Docking to Fixed Rotating Target In-Plane



(b) Energy Profile of Docking to Fixed Rotating Target In-Plane

Figure 3-23: Control and Energy Profile for Docking to Fixed Rotating Target In-Plane

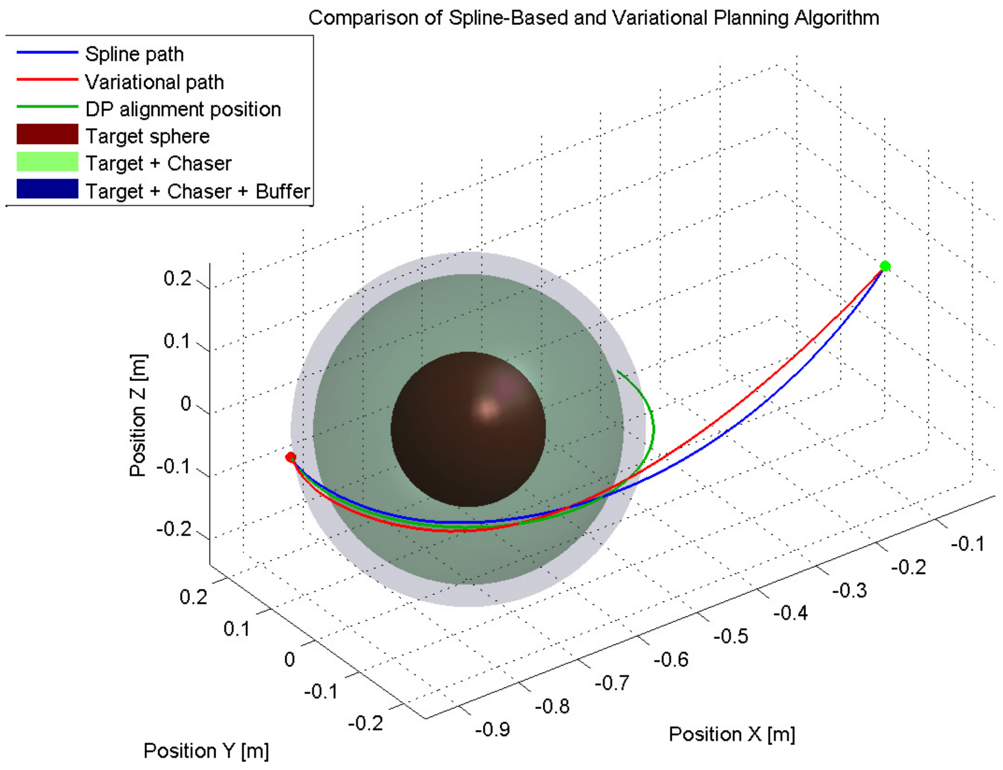


Figure 3-24: 3D Trajectory of Docking to Fixed Rotating Target In-Plane

3.5.3 Docking to Fixed Coning Target Facing Backwards

The tumbling dynamics of the target spacecraft in this scenario is still a pure rotation, but the rotation rate vector of 2 deg/sec is not perpendicular to the docking port axis as in the previous scenario. This causes the docking port axis to sweep a cone. In addition, the target spacecraft faces its back towards the chaser during the complete coning tumbling dynamics. Therefore, it requires the chaser to maneuver in all three x,y, and z axis to avoid the obstacle and reach the terminal position at t_f. Another complexity is added by introducing non-zero initial velocity for the chaser spacecraft. The target will still have zero translational velocity as it is fixed and moving in attitude.

The initial configuration of the two spacecraft and the rotation rate vector is chosen such that the target docking port sweeps a cone of 45 degrees between the generatix and axis, an aperture of 90 degrees. The setup of the docking scenario is shown in Figure 3-17. The position of the target is still the same as before and is fixed. The initial state of the chaser is adjusted to have a slight velocity towards and to the side of the target spacecraft $\mathbf{x}(0) = [0\ 0\ 0\ \ -0.005\ \ -0.008\ 0]^T$. Another adjustment is made to the parameter α penalizing obstacle clearance to $\alpha = 80$. The complete setup for the docking scenario to a coning target spacecraft facing backwards in summarized in Table 3.7.

The Figures 3-25(a) and 3-25(b) show a nearly identical state trajectories from the two planning algorithms. The spline-based algorithm trajectory deviates less than a $1cm$ from the variational method to planning path. It is seen that the initial velocities of the chaser vehicle is non-zero. This non-zero initial velocity exhibits the *not-a-knot* condition for the spline interpolation in the spline-based Algorithm 3.1 on page 91.

The control profiles in Figure 3-26(a) having a similar response. Both are fairly linear and then take a turn at about 70 seconds and continue straight afterwards. This is also when the only way-point is introduced to the spline-based planning algorithm. The energy profile in Figure 3-26(b) show a nicely behaved energy build-up similar to that in Figure 3-19(b) of the fixed non-rotating docking scenario. The total energy of

Table 3.7: Docking to Fixed Coning Target Facing Backwards Planning Inputs.

Description	Value
mass of spacecraft	$m = 4.3kg$
final time	$t_f = 100$ seconds
radius of chaser	$R = 0.105m$
radius of target	$R_{obs} = 0.105m$
obstacle position	$\mathbf{r}_{obs} = [-0.7\ 0\ 0]^T$
initial state	$\mathbf{x}(0) = [0\ 0\ 0$ $-0.005\ \ -0.008\ 0]^T$
final state	$\mathbf{x}(t_f) = [-0.877\ 0.0651\ \ -0.166$ $0\ 0.006\ 0.002]^T$ from Eq. (3.31)
Variational Technique to Planning Terms:	
weight on control input	$\rho = 10000$
weight on obstacle clearance	$\alpha = 80$
buffer distance	$s = 0.03m$
maximum cost at zero relative distance	$k = 0.08$
connecting velocity	$v_c = 0.005m/s$
polynomial coefficients	using equations (3.45) and (3.49)
Spline-Based Algorithm Terms:	
buffer distance	$R_{buffer} = 0.03m$
tolerance on r_{del}	$r_{del,tol} = 0.005m$

the variational technique to planning algorithm is $8.513 \times 10^{-5}J$ and $8.385 \times 10^{-5}J$ for the spline-based algorithm with a 1.5% difference. Both the planners found a very optimal trajectory, but the spline-based algorithm was 13.5x times faster. A performance summary is shown in Table 3.8.

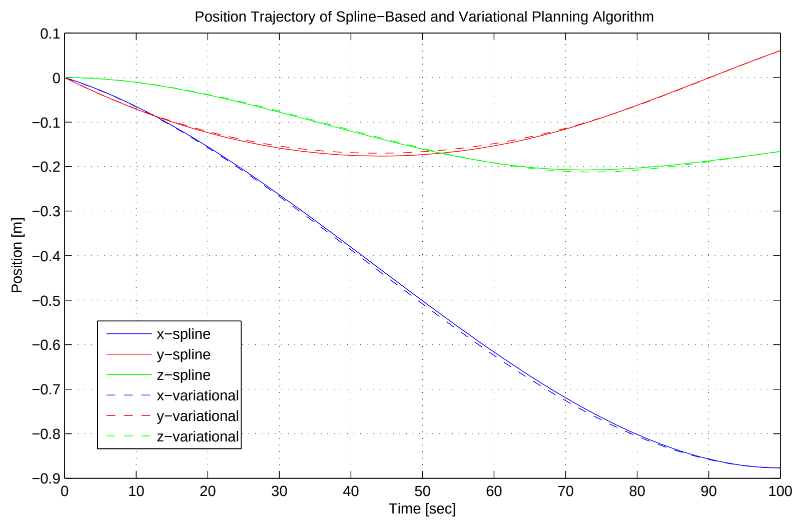
The full 3D trajectory from both planners is shown in Figure 3-27 and the obstacle spheres. Both trajectories have a smooth path that stay out of the buffer zone of the obstacle sphere. The green curve shows the path of the coning DP alignment position.

This scenario is a second validation of the planners being able to handle spherical obstacle constraints. It shows that the spline-based algorithm found another trajectory that is very close to the variational method, but is significantly smaller in computation time. The scenario also shows the ability to account for non-stationary

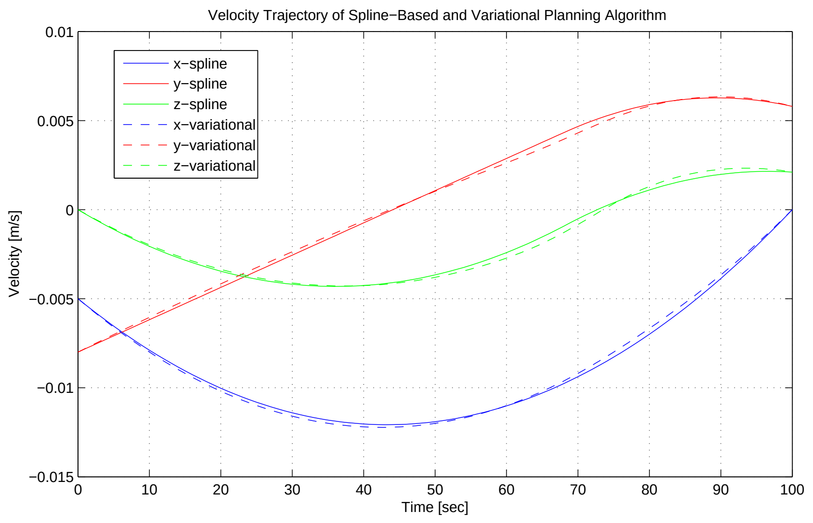
Table 3.8: Energy Cost and Computation Time for Fixed Coning Target Facing Backwards Scenario.

	Variational Technique Planning	Spline-Based Algorithm
Total Energy Cost:	$8.513 \times 10^{-5}\ J$	$8.385 \times 10^{-5}\ J$
Energy Difference:		1.5%
Computation Time:	5.348 seconds	0.397 seconds
Times Faster:		13.5x

boundary conditions. The undesirable need to readjust the weight parameters is noticed once again. A summary of the comparison studies is discussed in the next section with important conclusion to take into account for the practical use of these planning algorithms

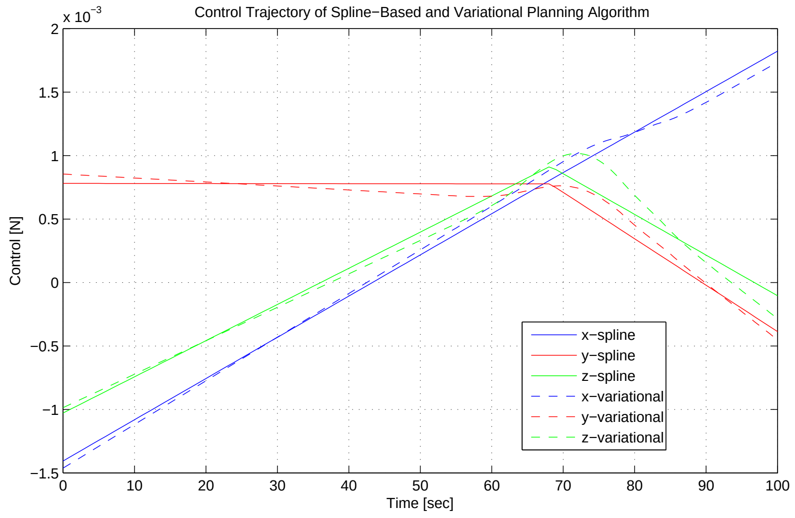


(a) Position Trajectory of Docking to Fixed Coning Target Facing Backwards

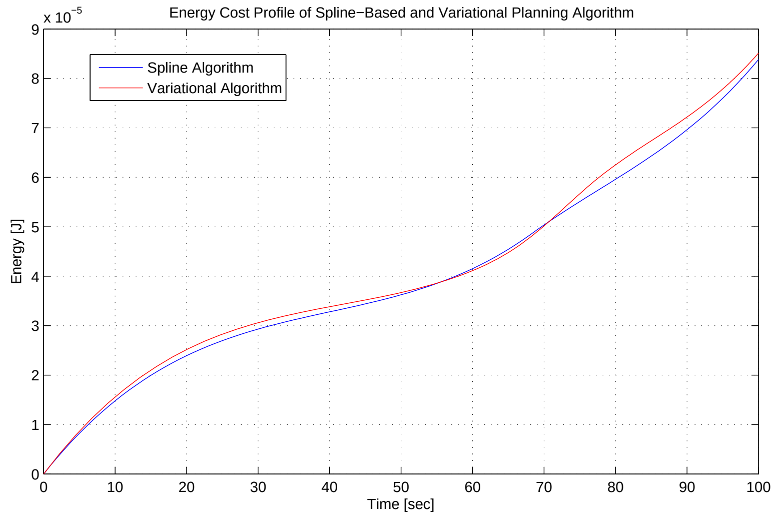


(b) Velocity Trajectory of Docking to Fixed Coning Target Facing Backwards

Figure 3-25: State Trajectory of Docking to Fixed Coning Target Facing Backwards



(a) Control Input of Docking to Fixed Coning Target Facing Backwards



(b) Energy Profile of Docking to Fixed Coning Target Facing Backwards

Figure 3-26: Control and Energy Profile for Docking to Fixed Coning Target Facing Backwards

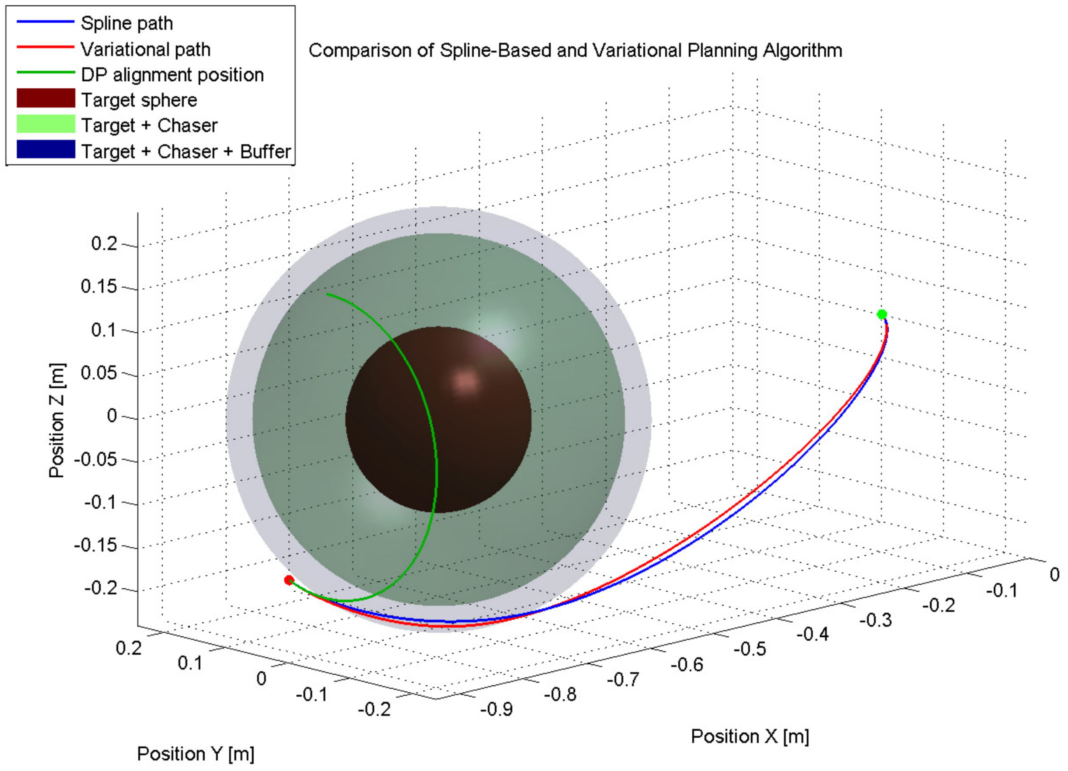


Figure 3-27: 3D Trajectory of Docking to Fixed Coning Target Facing Backwards

3.5.4 Comparison Summary

The two trajectory planning algorithms are compared first to study the sub-optimality of the spline-based algorithm. The new algorithm determined reasonable trajectories for the three docking scenarios studied. The first trajectory calculated for the docking scenario to a fixed target facing forward showed that the algorithm finds a fully energy optimal trajectory as the variational method. The other two trajectories performed equivalently as good or better. Through the comparison study, some observations were found in regards to the benchmark planner.

Even though the variational planner is supposed to provide more optimal solutions, it has significant deficiencies to practical use and implementation. One issue is the much larger computational power required to solve the HBVP compared to the competing algorithm. The practical usage issue is with the necessity to carefully select the weights in the cost functional. If not done so, then the computed trajectory has sporadic behavior and may collide with the obstacle. Such a trail and error approach to find the right weights is not desired for an autonomous docking system. The spline-based algorithm does not require any specially tuned weight, just physically defined dimensions of the chaser and target spacecraft spherical obstacles.

3.6 Summary

This chapter first introduced a general formulation of optimal planning for a continues system with differential constraints. Then it was detailed specifically with docking dynamics and boundary conditions of a tumbling satellite. A calculus of variation technique was introduced for trajectory planning and the obstacle clearance cost functional was presented. This method showed undesirable characteristic for usage and hardware implementation, but was a good tool at studying the performance of a the new spline-based planning algorithm. The two planner were compared to conclude satisfactory performance of the new planner.

Chapter 4

Trajectory Tracking

In this chapter, several tracking controllers are developed for linear systems in pursuit to achieve improved tracking performance from previous PD/PID controllers. Only discrete controllers are developed as real systems such as spacecraft exhibit real-world digital control. Also, realistic scenarios have a maximum limit on the control input $\mathbf{u}_{max}$ and thus controllers are pursued to not saturate the actuators under nominal operations. First, the traditional PD/PID controller used previously in the *GN&C modules* is briefly described. Next, several discrete optimal controllers are developed. The first is the standard full-state feedback discrete Linear-Quadratic-Regulator (LQR) controller that exhibits a PD-type form. In order to improve the steady state error and tighter tracking performance, a discrete servo-LQR controller is developed that integrates the state error and has a PID-type form. The final controller tries to combine the best properties of both LQR controllers into a phase-plane controller that switches between the discrete LQR and servo-LQR controller based on a decision strategy from the state error. Lastly, the controllers are compared in simulations with their properties being observed.

The tracking controllers are required to as precisely track the position state trajectory provided by the path planner, which is only dependent on the translational dynamics of the spacecraft. Therefore, the system dynamics for the controllers to act onto are three decoupled double integrators from Eq. (3.15),

$$\ddot{\mathbf{r}} = \frac{1}{m}\mathbf{f}$$

where m is the mass of the spacecraft, $\mathbf{r} \in \Re^3$ is the position of the spacecraft, and $\mathbf{f} \in \Re^3$ is the control input of the total force applied to the system $\mathbf{u} = \mathbf{f}$. The rest of the controllers are developed to control this continues-time second-order system. The controllers are applied to the satellite SPHERES, which has a mass value of $4.3kg$. Also, the discrete sampling period for position control is set to two seconds $\Delta t = 2$. With the given sampling period and a known maximum thrust value for SPHERES, the maximum limit on the discrete control input is:

$$u_{k,max} = 0.096N \tag{4.1}$$

To sum it together, the performance of each controller is studied for a double integrator dynamics of mass $m = 4.3kg$, a sampling period (control cycle) of $\Delta t = 2sec$, and maximum discrete control input $u_{k,max} = 0.096N$. Obviously, if a very large step input is provided for the closed loop controller and dynamic system, the corresponding control input would surpass the maximum limit $u_{k,max}$. Therefore, the controller performance study is performed under nominal operations. This refers to a step or a sinusoidal input for the SPHERES application of,

$$r(t) = 0.2 \tag{4.2}$$

$$r(t) = 0.3sin(0.0873t) \tag{4.3}$$

where the sinusoidal input corresponds to a $5°/\sec$ angular rate. The step response of each controller provides a general understanding of the controller performance while the sinusoidal input resembles a possible nonlinear trajectory the path planner may provide for obstacle avoidance. Therefore, studying the tracking performance of the sinusoidal trajectory provides a general insight on the controllers ability to track other nonlinear trajectories of similar bandwidth (curvature).

There are several performance characteristics desired by the tracking controllers for docking scenarios. One highly desired feature is a very low overshoot response. The reason for this can be easily shown by observing the operation of the **Inline Approach** phase discussed in Section 2.3.1 on page 39 and depicted in Figure 2-8 on page 40. In this maneuver, the chaser spacecraft closes in the distance towards the target spacecraft with an inline approach along the docking port axis. Here, a tracking controller is used to follow a state trajectory generated by the path planner to move the chaser spacecraft very close to the docking port face of the target. It is observed that if the tracking controller exhibits high overshoot, the chaser spacecraft may surpass the final position of the trajectory and collide with the target spacecraft. Clearly this is undesirable and a low overshoot characteristic of a tracking controller is required.

Let's define a proposed maximum limit on the percent overshoot. Given a spherical encapsulation of the traveling spacecraft (chaser) with radius R and the smallest relative distance along the state trajectory to the obstacle surface being R_c, then the proposed maximum deviation from the state trajectory is 25% of $(R_c - R)$. The subtraction of the chaser spacecraft radius is because the state trajectory defines the movement of the chaser spacecraft centroid and so $(R_c - R)$ is the face-to-face distance between the two objects. This also defines another desired performance characteristic, the maximum error in tracking of a state trajectory,

$$e_{max} = 0.25(R_c - R) \tag{4.4}$$

where e_{max} defines the maximum error for trajectory tracking. The percent overshoot (PO) is found by taking into account the maximum error e_{max} and the nominal step input Eq. (4.2) as:

$$PO_{max} = 100\frac{e_{max}}{r(t)} \tag{4.5}$$

Figure 4-1 depicts the maximum error in trajectory tracking dependent on $(R_c - R)$.

The closest distance along a trajectory to the obstacle for the docking architecture

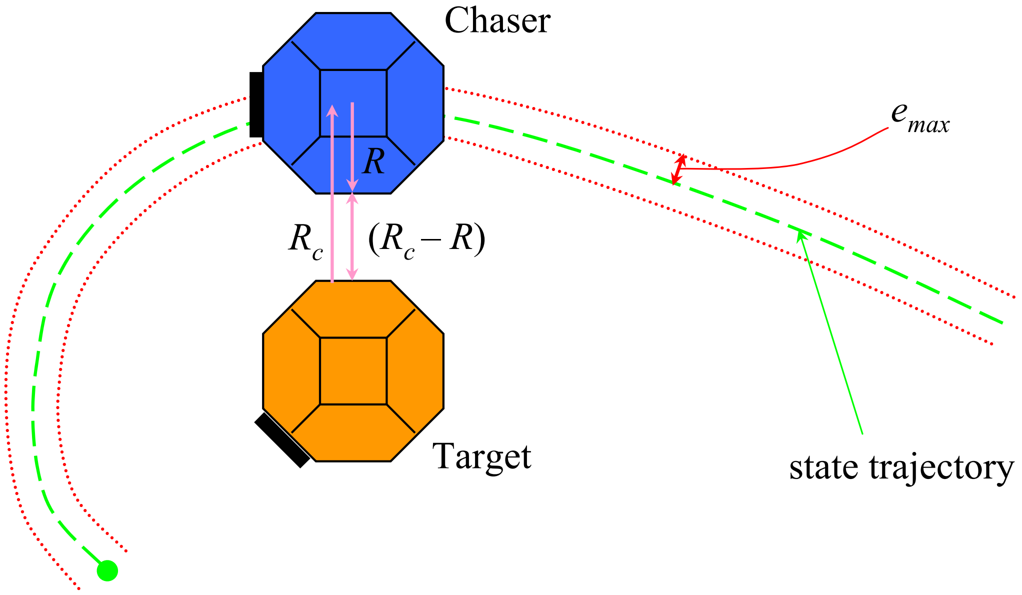


Figure 4-1: Maximum Tracking Error Dependent On The Closest Distance To Obstacle

established in Section 2.3.1 is chosen to be the *berthing* position. For the SPHERES application, the berthing position is defined to be $(R_c - R) = 0.04m$ in front of the target spacecraft docking port. Therefore, the maximum error in trajectory tracking and the percent overshoot from equations (4.2), (4.4) and (4.5) for SPHERES is:

$$e_{max} = 0.01m \tag{4.6}$$

$$PO_{max} = 5\% \tag{4.7}$$

Several tracking controllers will now be developed and discussed with regards to their state response and control effort.

4.1 PD/PID Controllers

The PID-type controllers used previously in the control GN&C module was summarized in Section 2.2.1 on page 27. The controller is restated from Eq. (2.5),

$$\mathbf{f} = \begin{bmatrix} K_P\tilde{r}_x + K_I\int \tilde{r}_x dt + K_D\tilde{v}_x \\ K_P\tilde{r}_y + K_I\int \tilde{r}_y dt + K_D\tilde{v}_y \\ K_P\tilde{r}_z + K_I\int \tilde{r}_z dt + K_D\tilde{v}_z \end{bmatrix}$$

where $\tilde{r}_i$ represents the position error of the i^{th} axis from the desired reference state $\mathbf{x}_d$. The variables K_p, K_i, and K_d are the proportional, integral, and derivative gains chosen by the user. By setting the integral gain to zero $K_I = 0$, Eq. (2.5) becomes a PD controller. The main difficulty with the PID controllers is the proper selection of the gains. A standard approach for gain selection of a second-order system for a PD controller is [20],

$$K_P = m\omega_n^2 \tag{4.8}$$

$$K_D = m\left(2\zeta\omega_n\right) \tag{4.9}$$

and PID controller,

$$K_P = m\left(\omega_n^2 + \frac{2\zeta\omega_n}{\tau}\right) \tag{4.10}$$

$$K_I = m\left(\frac{\omega_n^2}{\tau}\right) \tag{4.11}$$

$$K_D = m\left(2\zeta\omega_n + \frac{1}{\tau}\right) \tag{4.12}$$

where ω_n is the natural frequency, ζ is the damping ratio, and τ is the integration time constant for the PID controller.

With the use of equations (4.8) to (4.12), the user has only the performance tuning variables ω_n, ζ, τ to develop a tracking controller for desired specifications. There is also no direct approach to weigh the cost of expelling control effort of the system. Therefore, the PID controllers provide limited ability for the user to weigh between the response performance and amount of control effort provided. However, they do provide the user an intuitive idea of the response from adjusting each gain.

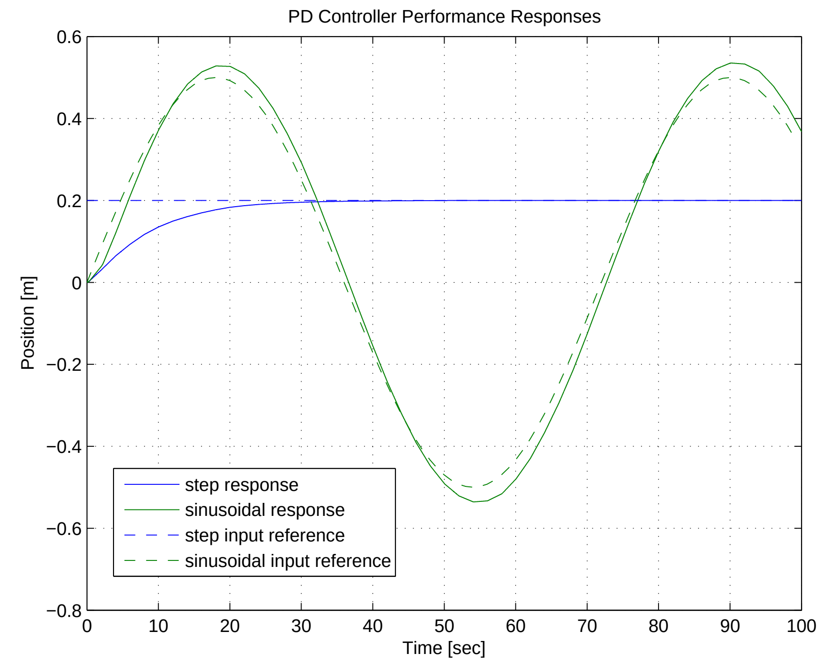
The PID controllers have been previously developed and tuned for SPHERES [7]. The gains chosen are summarized in Table 4.1:

Table 4.1: PD/PID Controller Gains Selection.

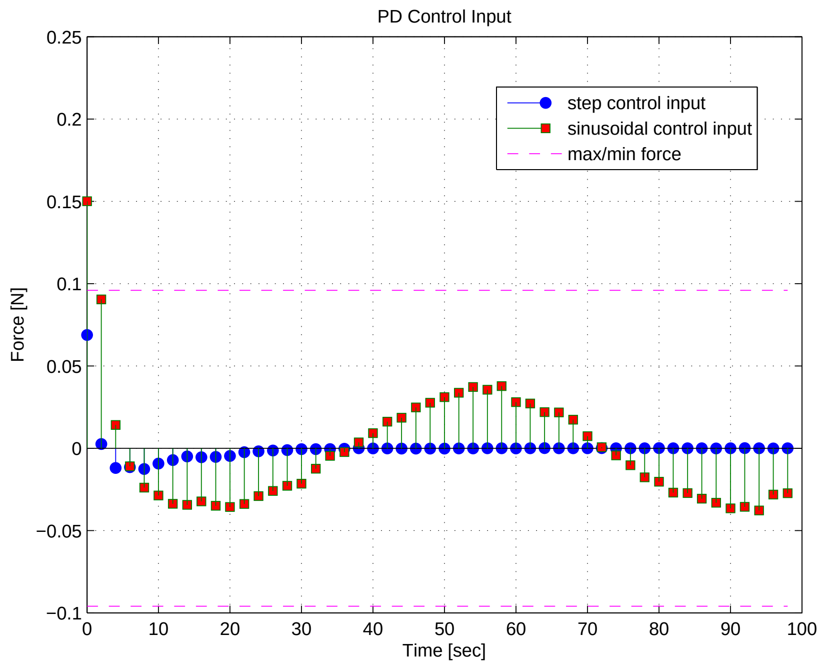
Controller	Gain Term	Value
PD Controller	K_P	0.172
	K_D	1.720
PID Controller	K_P	0.258
	K_I	0.0086
	K_D	1.935

The performance of the PD controller for a step and sinusoidal input is shown in Figure 4-2(a) and its corresponding control input in Figure 4-2(b). The response of the step input shows the PD controller behaving with a 0% overshoot and a settling time t_s of 31 seconds. When observing the sinusoidal trajectory tracking, only the performance at tracking the path after 1/4 of the period (1/4 = 18 sec) is studied for the maximum tracking error. The first 1/4 period time is let for spacecraft acquire the trajectory and the rest is regarded as precise following. The sinusoidal input tracking shows a maximum error of $0.048m$. As a result, the performance of the PD controller satisfies one of the two requirements in equations (4.6) and (4.7), the percent overshoot. The maximum tracking error is about $5x$ greater than the maximum limit set at $e_{max} = 0.01m$. The control input profile in Figure 4-2(b) shows force values within the bounds of saturation besides the first thrust for the sinusoidal input. This is acceptable as the sinusoidal tracking performance is mainly observed after 18 seconds of tracking.

PID controllers integrate the state error and are good at improving the steady-state error. This should also provide better trajectory tracking performance. However, there is generally an undesirable characteristic of PID controllers in which they exhibit higher percent overshoot at the expense of better tracking. The responses of the PID controller with gains selected from Table 4.1 are shown in Figure 4-3(a) and



(a) PD Performance Response



(b) PD Control Input

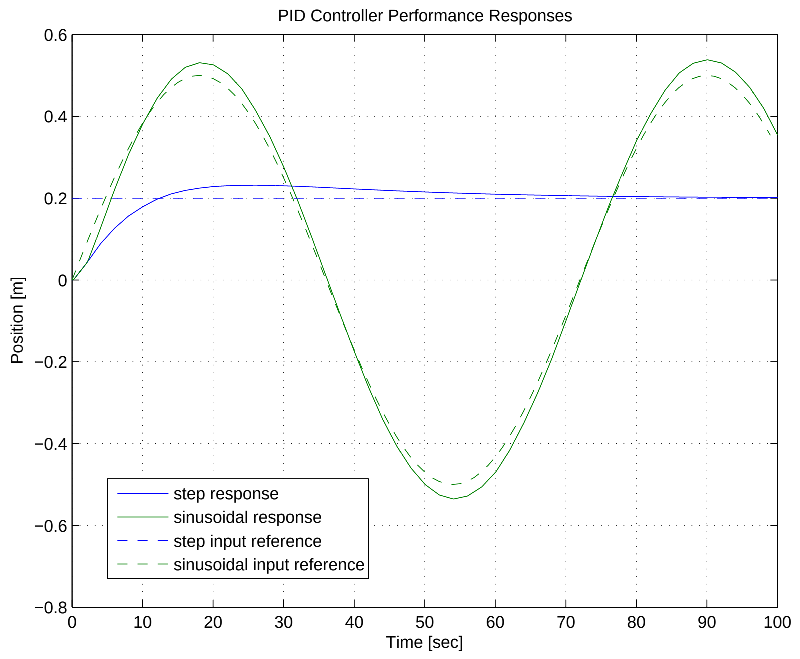
Figure 4-2: PD Controller Performance

the corresponding control input in Figure 4-3(b). The percent overshot for the step response is 16%, which is $3x$ greater than the limit of $PO_{max} = 5\%$. The settling time is rather large, 80 seconds, but the tracking performance has improved over the PD controller. The maximum error at tracking the sinusoidal input is $0.039m$. Even though the tracking performance has improved, it is still $4x$ greater than the maximum limit. The control profile in Figure 4-3(b) shows bounded discrete control inputs below saturation for most of the executions. The PID controller showed slight improvement in tracking as expected, but increased the percent overshoot beyond the allowable limit. The performance of both the PD and PID controllers is summarized in Table 4.2.

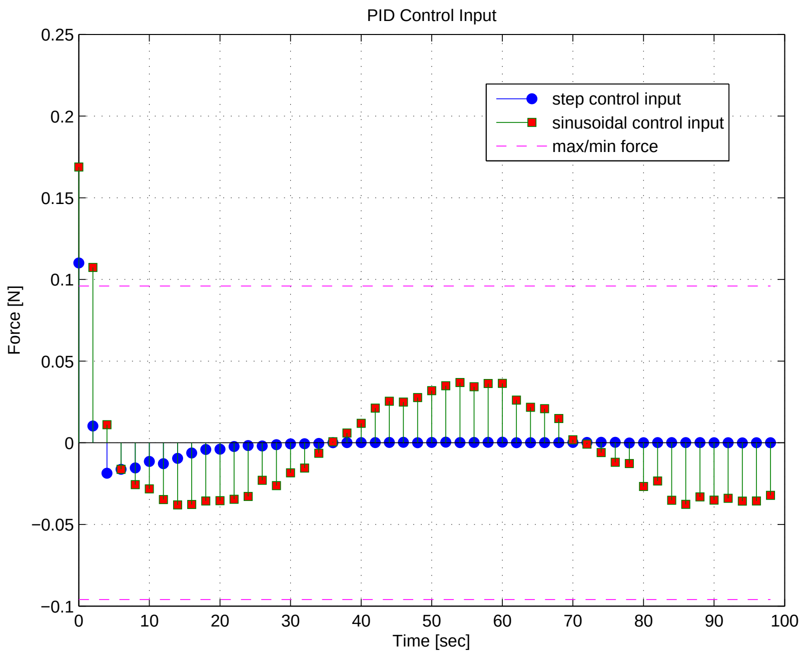
Table 4.2: PD/PID Controller Performance Summary.

Controller	Performance Term	Value
PD Controller	PO	0%
	e	$0.048m$
	t_s	$31sec$
PID Controller	PO	16%
	e	$0.039m$
	t_s	$80sec$

The PD/PID controllers are the most widely used controllers and may be sufficient for relaxed needs, but the pursuit here is to determine a high-performance controller for tracking a state trajectory. Neither the PD nor PID controller satisfied the required performance specifications for spacecraft docking scenarios. Optimal controllers are developed in the next sections for the pursuit to find a tracking controller satisfying the performance requirements.



(a) PID Performance Response



(b) PID Control Input

Figure 4-3: PID Controller Performance

4.2 LQR Controller

The LQR control algorithms are the most basic optimal controllers for linear dynamic systems. The details of the controller in application to spacecraft docking is now discussed.

Since the spacecraft translational system dynamics are three decoupled double integrators, the controllers are also decoupled along each axis. Thus, only one controller for a single axis is developed and its copy is used for the other two axis. The double integrator continues system dynamics in state-space form is,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{4.13}$$

where $\mathbf{A}$ and $\mathbf{B}$ are:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \tag{4.14}$$

Since the controllers are required to be discrete with a sampling period of Δt, the system dynamics are discretized to,

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k \tag{4.15}$$

with $\mathbf{A}_d$ and $\mathbf{B}_d$ being time-invariant discretized dynamic matrices to first-order,

$$\begin{aligned} \mathbf{A}_d &= \mathcal{I} + \mathbf{A}\Delta t \\ \mathbf{B}_d &= \mathbf{B}\Delta t \end{aligned} \tag{4.16}$$

where $\mathcal{I} \in \Re^{2x2}$ is the identity matrix. The problem formulation of the LQR controller is to determine the control inputs minimizing the discrete cost functional,

$$J = \frac{1}{2}\mathbf{x}_N^T\boldsymbol{\mathcal{H}}\mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left[\mathbf{x}_k^T\boldsymbol{\mathcal{Q}}\mathbf{x}_k + \mathbf{u}_k^T\boldsymbol{\mathcal{R}}\mathbf{u}_k\right] \tag{4.17}$$

while satisfying the system dynamics constraint of Eq. (4.15). The state and control input weighting matrices are subject to be symmetric-positive-definite (SPD):

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{Q}}^T \geq 0, \quad \boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{R}}^T \geq 0, \quad \boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}^T \geq 0 \tag{4.18}$$

The SPD requirement ensures that the extremal control law is a minimizing control input of the form:

$$u_k = \mathbf{K}_{ss}\mathbf{x}_k \tag{4.19}$$

where the steady-state feedback gain matrix $\mathbf{K}_{ss}$ is computed as,

$$\mathbf{K}_{ss} = -\left(\boldsymbol{\mathcal{R}} + \mathbf{B}_d^T\mathbf{P}_{ss}\mathbf{B}_d\right)^{-1}\mathbf{B}_d^T\mathbf{P}_{ss}\mathbf{A}_d \tag{4.20}$$

The optimal control law is found for an infinite-horizon LQR controller. This results in the $\mathbf{P}$ term tending to a constant solution when $N \to \infty$ in the general discrete Riccati equation [6] with $\mathbf{P}_{ss} \geq 0$. Thus, the terminal cost is defined to be zero $\boldsymbol{\mathcal{H}} = \mathbf{0}$ and $\mathbf{P}_{ss}$ is the solution to the Algebraic Riccati Equation:

$$\mathbf{P}_{ss} = \boldsymbol{\mathcal{Q}} + \mathbf{A}_d^T\left[\mathbf{P}_{ss} - \mathbf{P}_{ss}\mathbf{B}_d\left(\boldsymbol{\mathcal{R}} + \mathbf{B}_d^T\mathbf{P}_{ss}\mathbf{B}_d\right)^{-1}\mathbf{B}_d^T\mathbf{P}_{ss}\right]\mathbf{A}_d \tag{4.21}$$

The feedback gain matrix $\mathbf{K}_{ss}$ is computed once for the system and used throughout the control executions. Also by observing Eq. (4.19), the controller is identical to the PD controller.

This leaves the user to tune the weighting matrices for trading off the performance of the system $\boldsymbol{\mathcal{Q}}$ and the cost of control effort $\boldsymbol{\mathcal{R}}$. Once the weighting matrices are selected for the desired specification, the LQR controller provides the optimal PD gains instead of the user directly choosing the gain values. This allows improved control of the overall performance of the system and provides the user more sensible tuning variables. However, deciding the weighting matrices is also an art and there exists a standard starting point in the selection process, Bryson's rule [2]. The rule forms diagonal weighting matrices in the following form:

$$\boldsymbol{\mathcal{Q}} = \begin{bmatrix} \frac{\alpha_1^2}{x_{1,max}^2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{\alpha_N^2}{x_{N,max}^2} \end{bmatrix}, \quad \boldsymbol{\mathcal{R}} = \rho \begin{bmatrix} \frac{\beta_1^2}{u_{1,max}^2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{\beta_N^2}{u_{N,max}^2} \end{bmatrix} \tag{4.22}$$

where $\sum_i^N \alpha_i = 1$ and $\sum_i^N \beta_i = 1$. The terms $x_{i,max}^2$ and $u_{i,max}^2$ define the maximum acceptable values of the specified state and control input. Bryson's rules scales the weightings appropriately and then only the tuning variable ρ is used to trade-off between performance response and control effort. The rule generally provides good results, but it is usually the starting point for the user to iteratively tune the weighting matrices.

After initially tuning the weighting matrices using Bryson's rule and then performing further refinement, the optimal gain matrix for the SPHERES application is found to be:

$$\mathbf{K}_{ss} = [0.2195 \quad 1.4217] \tag{4.23}$$

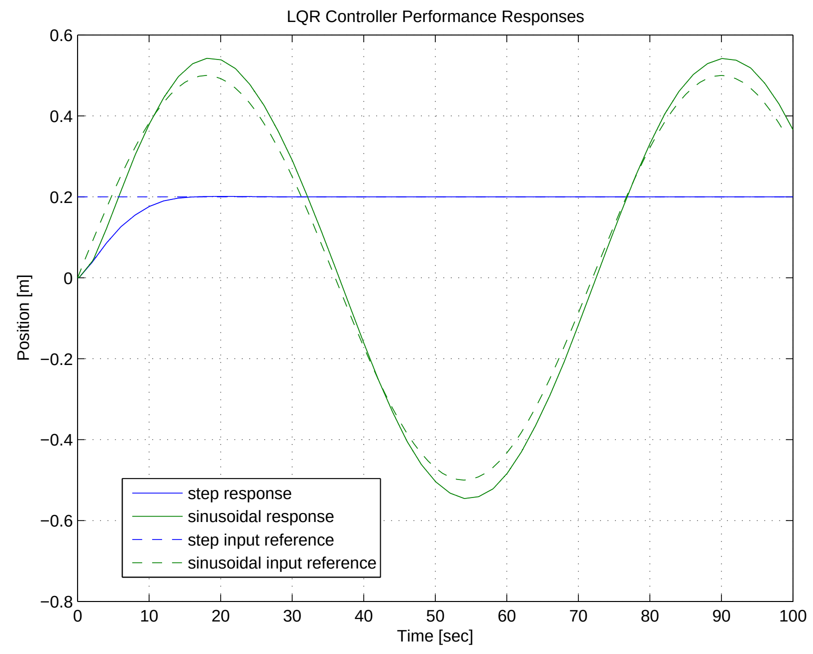
The optimal gains in Eq. (4.23) show a slightly higher proportional gain $\mathbf{K}_{ss}(1) = 0.2195$ and lower derivative gain $\mathbf{K}_{ss}(2) = 1.4217$ compared to the PD controller gains in Table 4.1, $K_P = 0.172, K_D = 1.72$. The response performance of the LQR controller is shown in Figure 4-4(a) along with the control profile in Figure 4-4(b). The step response shows a very small percent overshoot $PO = 0.5\%$ as is expected from a PD-type controller. While tracking the sinusoidal input, the LQR controller had a maximum error of $0.053m$. Both the percent overshoot and tracking error increased slightly compared to the PD controller. However, the settling time for the step input improved significantly by $2x$ to 14 seconds compared to 31 seconds. A summary of the performance values of the LQR controller are shown in Table 4.3. The control profile in Figure 4-4(b) shows thrust values below the saturation limit instead of that first thrust. During precise tracking of the inputs, the control input stayed below half the maximum limit.

The LQR controller satisfied the same requirements the PD controller achieved,

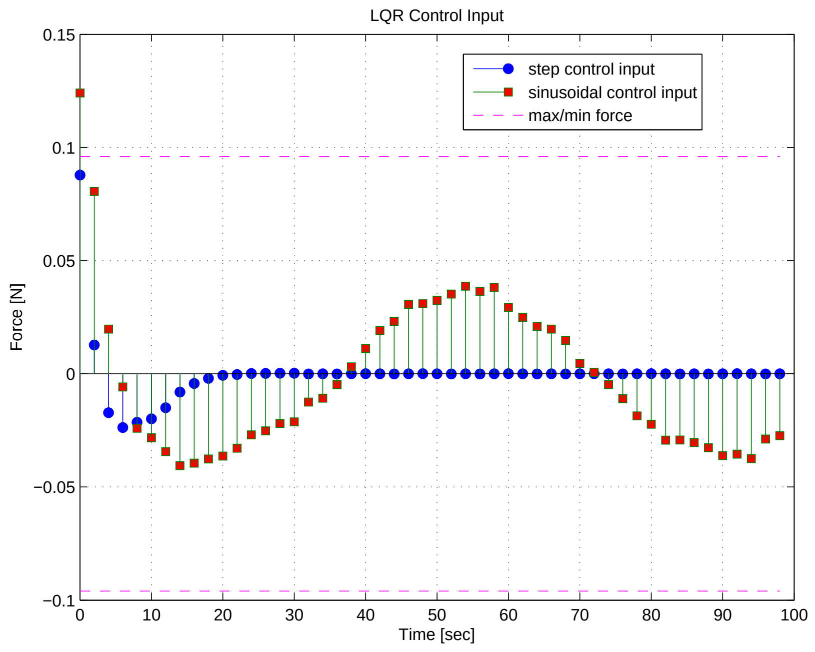
Table 4.3: LQR Controller Performance Summary.

Controller	Performance Term	Value
LQR Controller	PO	0.5%
	e	$0.053m$
	t_s	$14sec$

that being only the low percent overshoot. The necessary tracking error requirement $e_{max} = 0.01m$ has not been satisfied by any of the controllers developed thus far, PD/PID and LQR controller. The best tracking accuracy has been achieved by the PID controller $e = 0.039m$ due to its error integration capability. Next, the LQR controller is expanded to add the feature of error integration in order to improve tracking performance.



(a) LQR Performance Response



(b) LQR Control Input

Figure 4-4: LQR Controller Performance

4.3 Servo-LQR Controller

The servo-LQR controller is an extension to the previously discussed LQR controller. The discrete LQR controller in Section 4.2 is a PD-type controller. To improve the performance of the steady-state error and thus tighter tracking performance, a state error integration term is added to the optimal controller. The servo-LQR is also referred to as an integral LQR controller and is of the PID form. Instead of letting the user tune how much to integrate the state-error, the servo-LQR controller is formed to optimize for the integration gain. Therefore, the system dynamics from Eq. (4.14) is expanded to include a third state x_3 that integrates the first state x_1:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \\ 0 \end{bmatrix} \tag{4.24}$$

The discrete dynamics are found in the same manner with Eq. (4.16) on Eq. (4.24). The servo-LQR control law is found again through the use of equations (4.19), (4.20) and (4.21) with the gain matrix being $\mathbf{K}_{ss} \in \Re^3$, one dimension higher due to the additional state. Also, the weighting matrices $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{R}}$ are increased in size with the additional term $\frac{\alpha_3^2}{x_{3,max}^2}$ penalizing the maximum integration of state x_1 by the use of Bryson's rule. Again, after tuning the weights by starting off with Bryson's rule and then refining iteratively trough trail and error, the servo-LQR gain matrix for the SPHERES application is found to be,

$$\mathbf{K}_{ss} = [0.6191 \quad 2.1948 \quad 0.08129] \tag{4.25}$$

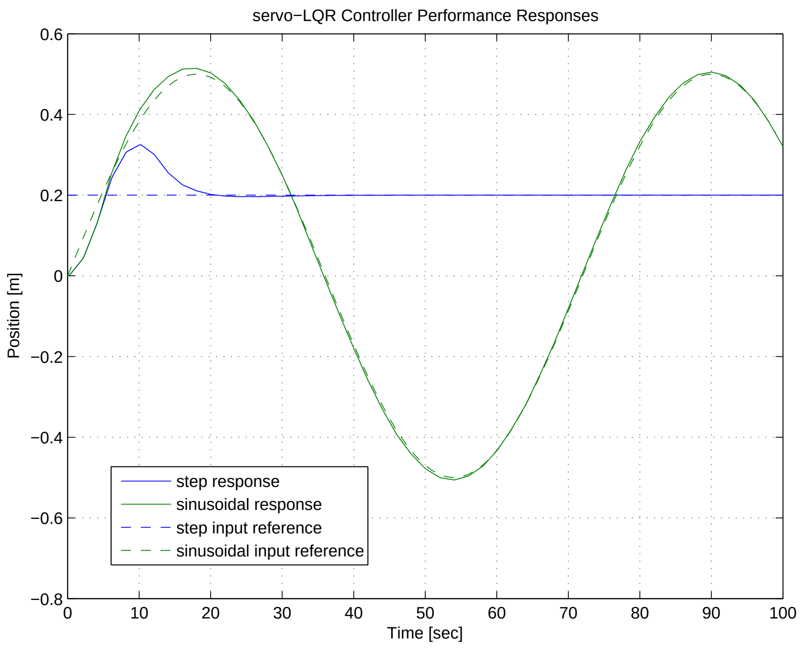
where $K_P = \mathbf{K}_{ss}(1)$, $K_D = \mathbf{K}_{ss}(2)$, and $K_I = \mathbf{K}_{ss}(3)$. The optimal gains in Eq. (4.25) are compared to the PID controller gains from Table 4.1. All of the servo-LQR controller gains are higher than that of the PID controller and consequently will expect faster response for the reference inputs. The only worry for higher gains is the possibility of high overshoot and a highly oscillatory response.

The servo-LQR controller responses and control profile are shown in Figures 4-5(a) and 4-5(b). By observing the step input response in Figure 4-5(a), it is noticed that the controller speeds up towards the step input of $0.2m$ quicker than any of the previous controllers. The response reaches $0.2m$ in 5 seconds for the first time, but then overshoots by $PO = 62\%$ and achieves a settling time of 20 seconds. This is a very reactive controller. The huge overshoot is the worst of all the controllers and is undesirable for the use of tracking trajectories of docking scenarios. However, the error tracking of $0.0142m$ has improved as expected from the addition of the error integration capability. Unfortunately, servo-LQR controller does not satisfy any of the performance specifications for docking, percent overshoot and tracking error. It does achieve the smallest tracking error and is very close to the requirement of $0.01m$. A performance summary of the controller is shown in Table 4.4. The control profile in Figure 4-5(b) shows the high thrusting in the first ten seconds of the highly responsive controller. Ater the controller acquires the sinusoidal trajectory, the control input performs at nominal values that do not saturate the thrusters.

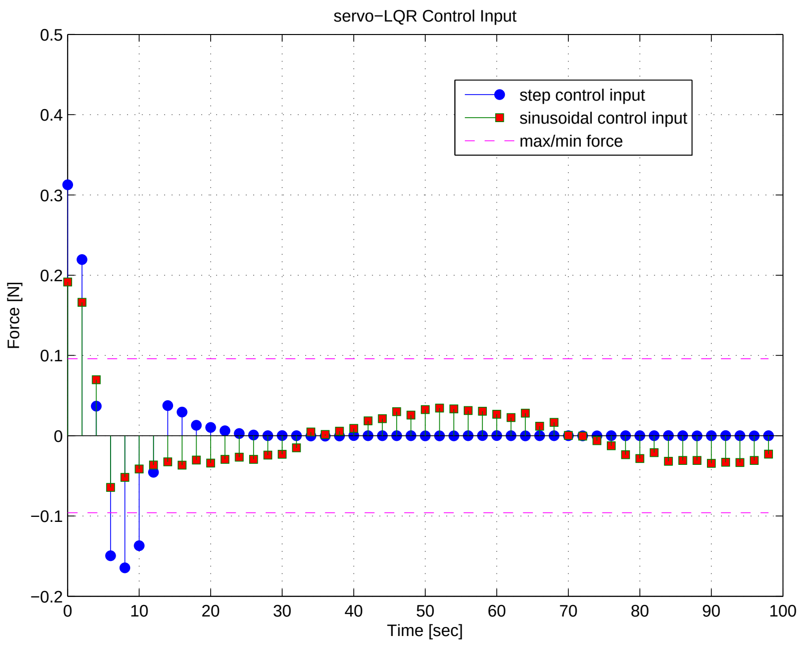
Table 4.4: Servo-LQR Controller Performance Summary.

Controller	Performance Term	Value
Servo-LQR Controller	PO	62.0%
	e	$0.014m$
	t_s	$20sec$

There are currently four controllers that have been studied, two of the PD-type (PD and LQR controllers) and two of the PID-type (PID and servo-LQR controllers). Both of the PD-type controllers achieve the specified requirement for the percent overshoot $PO_{max} = 5\%$. However, they have the lowest performance in tracking nonlinear trajectories and do not satisfy the requirement $e_{max} = 0.01m$. On the other hand, the PID-type controllers attained the best performance in the tracking error, but showed unaccaptable values for the percent overshoot $PO_{PID} = 16\%$ and $PO_{servo-LQR} = 62\%$. It seems that the ideal controller would combine the best performance characteristics of each of the PD and PID type controllers in order to achieve the tight constraints defined for docking scenarios. Such a controller is developed in the next section.



(a) Servo-LQR Performance Response



(b) Servo-LQR Control Input

Figure 4-5: Servo-LQR Controller Performance

4.4 Phase-Plane LQR Controller

There is still a search for a controller that will satisfy the performance requirements of $PO_{max} = 5\%$ and $e_{max} = 0.01m$. Each of the previous controllers exhibit positive traits and certain negative characteristics that made them unsatisfactory for a tracking controller. In this section, a control law is developed that attempts to blend the best performance characteristics of the LQR and servo-LQR without keeping their negative traits.

The goal of the tracking controllers is to attain a nonlinear trajectory and follow it precisely. The step input demonstrates the controllers ability to acquire a trajectory while the actually sinusoidal input shows how precisely a controller can track the trajectory. The performance measure for the step input is the percent overshoot and for the sinusoidal input is the tracking error. During the step input system, the spacecraft is initially separated by $0.2m$. On the other hand, for the sinusoidal input system the initial separation is near zero. Since the LQR controllers advantage is in its low percent overshoot for the step input and the servo-LQR controller is best at minimizing tracking error for the sinusoidal reference, it would be best to use each controller during their best performing setup. As the servo-LQR tracks a nonlinear trajectory the best once it has acquired it, the controller should be applied when the spacecraft is very near the trajectory. Meanwhile, the LQR controller is best suited for any larger separations from the reference input. This methodology leads us to the phase-plane LQR controller, where the decision logic between using the LQR or servo-LQR controller is dependent on the state error.

A phase-plane shows the error of both states in a plot where the y-axis would represent the velocity state error $\tilde{x}_{k,2}$ and the x-axis represents the position state error $\tilde{x}_{k,1}$ in the case of a double integrator dynamics at iteration k. With this tool, the control law can be designed to apply a specific controller in certain areas of the phase-plane. From the discussion of advantages of the LQR and servo-LQR controller, the servo-LQR is desired to be applied for small values of $\tilde{x}_{k,1}$. Therefore, the control law is developed to apply the servo-LQR controller for position errors of less than

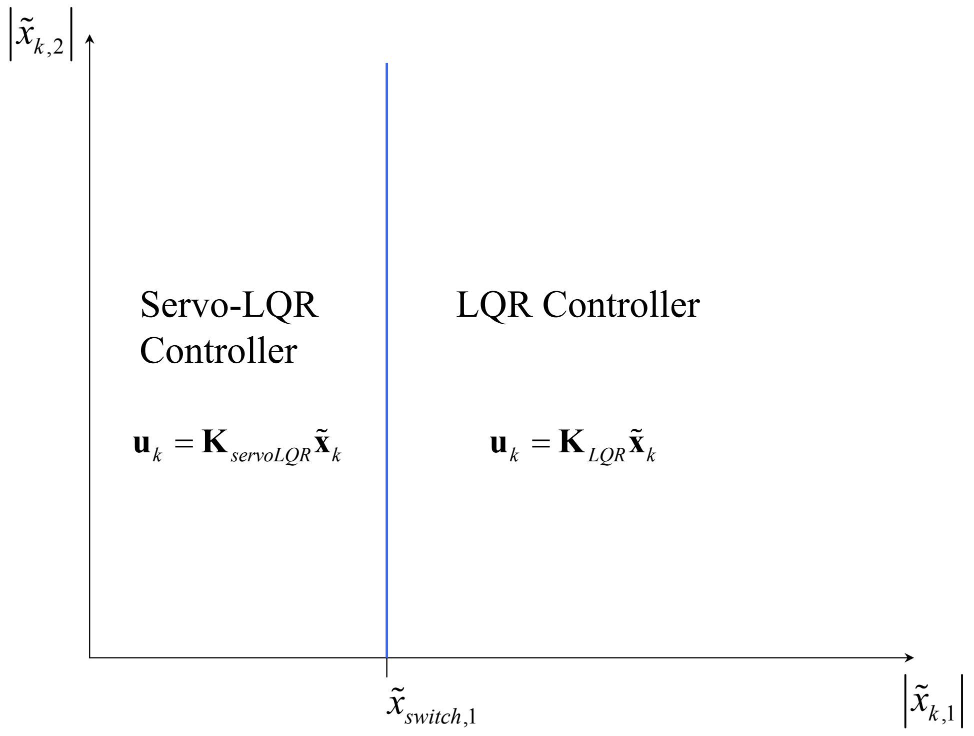

$|\tilde{x}_{k,2}|$
Servo-LQR Controller
$\mathbf{u}_k = \mathbf{K}_{servoLQR}\tilde{\mathbf{x}}_k$
LQR Controller
$\mathbf{u}_k = \mathbf{K}_{LQR}\tilde{\mathbf{x}}_k$
$\tilde{x}_{switch,1}$
$|\tilde{x}_{k,1}|$

Figure 4-6: Phase-Plane LQR Controller

$\tilde{x}_{switch,1}$,

$$u_k = \begin{cases} \mathbf{K}_{servoLQR}\tilde{\mathbf{x}}_k & \text{if } |\tilde{x}_{k,1}| \leq \tilde{x}_{switch,1} \\ \mathbf{K}_{LQR}\tilde{\mathbf{x}}_k & \text{else} \end{cases} \tag{4.26}$$

where $|\tilde{x}_{k,1}|$ represents the absolute value of the position error. A depiction of the phase-plane control law is shown in Figure 4-6. This is the simplest form of a phase-plane controller. Future work could expand the logic to account for the velocity error $\tilde{x}_{k,2}$. Generally, one could build nonlinear patches within the phase-plane to apply different controllers.

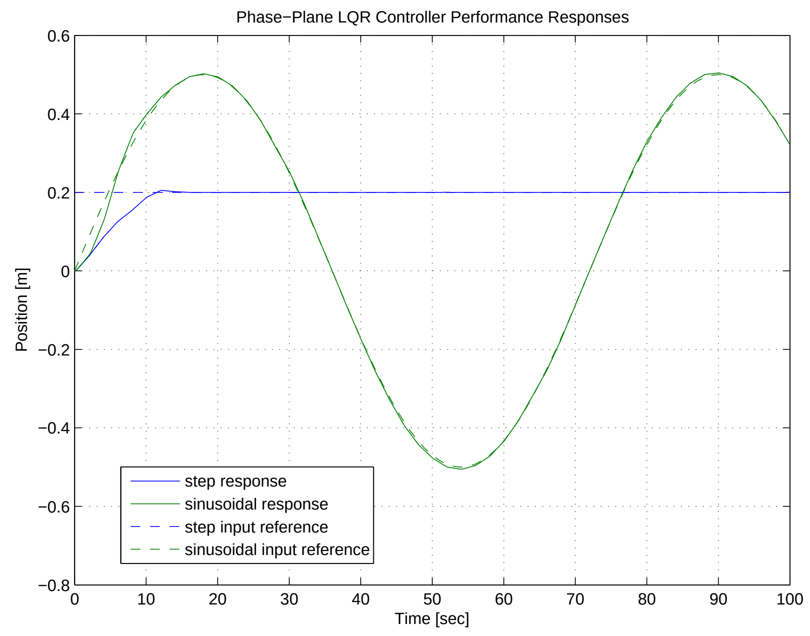
The phase-plane controller of Eq. (4.26) is further adjusted to improve performance and robustness. The refinement is made towards the error integration behavior of the switch to the servo-LQR controller. Once error integration begins, it builds up over time and keeps maximizing the control effort from this term. For example, when a step input is applied to a sole PID controller, the integration term will build up a positive force initially until the trajectory reaches a state error of zero. At this point, a trajectory generally overshoots and needs a negative force to return. However, since the integration term built up a positive force to this point, in the next control cycles, it will begin to reduce it due to the negative sign of the state error but still stay positive for a while. Even though the total control output is the addition of the proportional-integral-derivative errors multiplied with the corresponding gains, having the integral term provide a positive force when clearly a negative force is need is unnecessary. Therefore, a reset on the error integration $\tilde{\mathbf{x}}_{k,3} = 0$ is introduced to the servo-LQR of the phase-plane controller. The logic for when to reset the error integration term is defined to be when the sign in the state error switches.

In addition, there is another condition when to perform the reset. As mentioned, the phase-plane controller switches between the LQR and the error integrating servo-LQR controller. Let's assume the controller is applied to a highly nonlinear and difficult trajectory. While using the servo-LQR to tightly follow the trajectory, there may come a point when the controller may deviate from it by more than $\tilde{x}_{switch,1}$ and switch to the LQR controller. To this point, the servo-LQR has built up the error integration term and has stopped once the phase-plane controller switches to the LQR. Later on, the LQR may reacquire the trajectory with an error of less than $\tilde{x}_{switch,1}$ and switch back to the servo-LQR. Then the servo-LQR would continue on building the non-zero integration term from where it was left off previously when the initial switch was made. That previous accumulated error integration value is irrelevant to the current control of the dynamics. Therefore, the error integration term should also be reset during the switching between the LQR and servo-LQR controller. Both of the logics for resetting the error integration $\mathbf{x}_{k,3}$ is expanded to the phase-plane controller from Eq. (4.26) to:

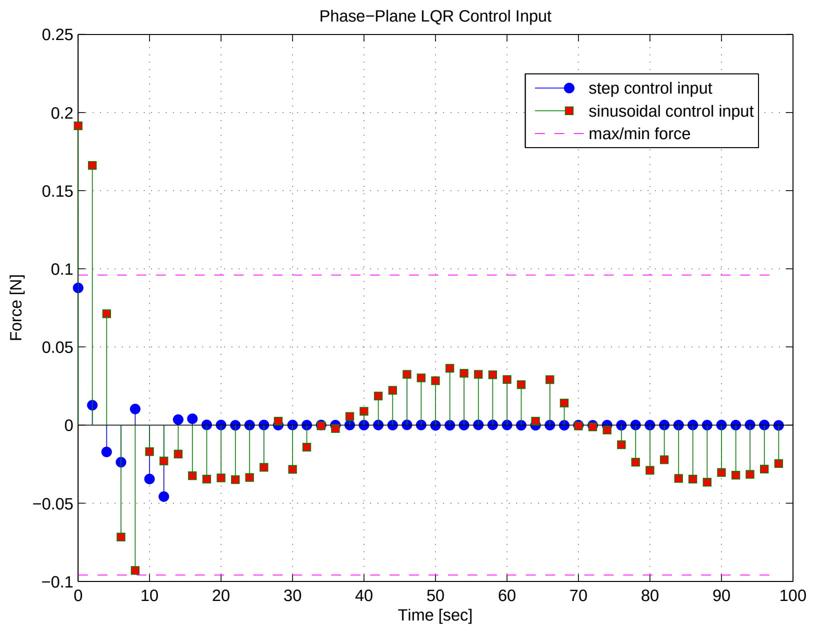
$$u_k = \begin{cases} \mathbf{K}_{servoLQR}\tilde{\mathbf{x}}_k & \text{if } |\tilde{x}_{k,1}| \le \tilde{x}_{switch,1} \\ \tilde{\mathbf{x}}_{k,3} = \begin{cases} 0 & \text{if } [sign(\tilde{\mathbf{x}}_{k,1}) \neq sign(\tilde{\mathbf{x}}_{k-1,1})] \text{ or } [\mathbf{u}_{k-1} = LQR] \\ \tilde{\mathbf{x}}_{k,3} & \text{else} \end{cases} & \text{else} \\ \mathbf{K}_{LQR}\tilde{\mathbf{x}}_k \end{cases} \tag{4.27}$$

The value of $\tilde{x}_{1,switch}$ for the SPHERES application is chosen to be $0.05m$ after several simulation studies. The phase-plane LQR controller response and control profile is shown in Figures 4-7(a) and 4-7(b). When the step input is applied, the phase-plane controller applied the LQR control during the first several seconds until the position error reached $0.05m$. To this point, the state trajectory was heading towards a response of zero percent overshoot, as is the case for PD-type controllers. Afterwards, the servo-LQR controller took over and stabilized the response to the $0.2m$ input with its ability of improving steady-state error. One worry was that the servo-LQR would provide a large overshoot once it takes over. However, as it is applied at only a $0.05m$ position error, the logic does not allow the controller enough time to integrate the error enough to provide a large overshoot. Therefore, the phase-plane LQR controller performs extremely well to the step input with a percent overshoot of 2.5% and a settling time of $12.8sec$, the lowest of any of the controllers. This performance satisfies the percent overshoot requirement. Sinusoidal trajectory tracking also shows highly satisfactory results. The phase-plane controller mainly used the servo-LQR and would jump to the LQR controller if the state trajectory deviated from the reference input by more than $0.05m$. From the good characteristic of the servo-LQR in tracking trajectories, the lowest tracking error is achieved to be $0.008m$. Finally, the search for a satisfactory controller is complete as both the maximum percent overshoot and tracking error requirements are assured by the phase-plane LQR controller. A summary of the controllers' performance is shown in Table 4.5.

The phase-plane LQR controller will be used in the GN&C control module of the architecture defined in Section 2.2.1. Due to its great response performance in



(a) Phase-Plane LQR Performance Response



(b) Phase-Plane LQR Control Input

Figure 4-7: Phase-Plane LQR Controller Performance

Table 4.5: Phase-Plane LQR Controller Performance Summary.

Controller	Performance Term	Value
Phase-Plane LQR Controller	PO	2.5%
	e	$0.0084m$
	t_s	$12.8sec$

step inputs and nonlinear trajectories, it is shown to be a robust controller for all reasonable reference inputs sent by the MVM module. The main use of the controller is to follow the nonlinear obstacle-free trajectories provided by the online path planner discussed in Section 3.4. Since this is the chosen controller, the implementation for a hardware device such as a spacecraft is shown in Algorithm 4.1. Also as the controller is a combination of the others discussed, it shows the their possible implementations as well. In addition to the terms for deciding to reset the integration error, a user-supplied flag $flag_{reset}$ is provided to allow the MVM module to have more control of the capabilities of the controller.

Algorithm 4.1: Phase-Plane LQR Controller

```
input  : $\tilde{\mathbf{x}}_k$, $K_{LQR}$, $K_{servoLQR}$, $\Delta t$, $\tilde{x}_{switch,1}$, $flag_{reset}$
output: $u_k$

if $|\tilde{x}_k(1)| \geq \tilde{x}_{switch,1}$ then
    $u_k = \mathbf{K}_{LQR}(1)\tilde{\mathbf{x}}_k(1) + \mathbf{K}_{LQR}(2)\tilde{\mathbf{x}}_k(2)$;
else
    if $[sign(\tilde{\mathbf{x}}_k(1)) \neq sign(\tilde{\mathbf{x}}_{k-1}(1))]$ $or$ $[\mathbf{u}_{k-1} = LQR]$ $or$ $[flag_{reset} = 1]$ then
        $\tilde{\mathbf{x}}_k(3) = 0$ ;
    end
    $\tilde{\mathbf{x}}_k(3) = \tilde{\mathbf{x}}_k(3) + \tilde{\mathbf{x}}_k(1)\Delta t$;
    $u_k = \mathbf{K}_{servoLQR}(1)\tilde{\mathbf{x}}_k(1) + \mathbf{K}_{servoLQR}(2)\tilde{\mathbf{x}}_k(2) + \mathbf{K}_{servoLQR}(3)\tilde{\mathbf{x}}_k(3)$;
end
return $u_k$
```

4.5 Summary

In this chapter, several discrete tracking controllers are presented in addition to the PID controllers: LQR, servo-LQR, and phase-plane LQR/servo-LQR controllers. These are for tracking a nonlinear trajectory provided by the spline-based planning algorithm. The chapter first began by defining the performance requirements for the controllers for applications of spacecraft docking. Specific values were determined for the SPHERES testbed. Next, the behavior of PD and PID controllers was observed and found to not meet the requirements. In developing the LQR controllers, methods of tuning the gains was discussed. The last controller, phase-plane LQR, attempted to combine the best characteristics of both the LQR and servo-LQR controllers. The final controller showed to meet the required tracking accuracy and low-overshoot.

Chapter 5

Simulation and Experimental Autonomous Docking

A realistic system of two spacecraft docking includes noise in both the measurement of the states from actual sensors and process noise of the control inputs from imperfect thrusters. In addition to these uncertainties, there are external disturbances applied to the spacecraft by the space environment. The most common in-space disturbances for a spacecraft in Low-Earth-Orbit (LEO) are gravity gradient, aerodynamic drag, magnetic field, and solar pressure. Therefore, a closed-loop control system is used to counteract any of these disturbances and uncertainties in order for the spacecraft to follow a desired trajectory. There are two components that will maneuver a spacecraft for docking to complex tumbling target spacecraft. First is the decision in the desired trajectory that drives the chaser from its initial state to a final state that aligns it infront of the docking port axis. This is accomplished by the spline-based trajectory planning algorithm developed in Chapter 3 and described in Algorithm 3.1. The computed path is an energy sub-optimal trajectory that avoids the target spacecraft as an obstacle. The second component consists of the trajectory tracking controller developed in Chapter 4, which executes a closed-loop control law that follows the path calculated by the planning algorithm. These two algorithms developed in this thesis are coupled together to execute the robustly designed phases of a docking mission managed by the new MVM module described in Section 2.3.1. The overall GN&C

architecture in Figure 2-7 on page 38 shows these two components working together managed by the MVM module.

The new autonomous system for docking is tested in simulations for varying target tumbling scenarios. The simulation runs also show the new autonomous system ability to dock to a tumbling spacecraft from any random initial configurations. In addition, the new spline-based planning algorithm is experimentally tested on the SPHERES testbed aboard the International Space Station (ISS). The testing facility provides a true micro-gravity environment and allows the spacecraft to maneuver in all six degrees-of-freedom by the use of onboard CO_2 thrusters. As this facility is a realistic representation of an outer space environment, the algorithms tested on the SPHERES hardware increases their Technology Readiness Level (TRL) to **TRL 6** [10]. The experimental test aboard the ISS of the spline-based planning algorithm developed in this thesis is the *first time* a true path planning algorithm was successfully executed online in micro-gravity. In addition to being the first in-space online path planner, it is also the first to account for obstacle avoidance. The experimental validation has provided a large step in the advancement of space technology, specifically for docking purposes.

The simulation and experimental runs are to validate the capability of the new autonomous docking system by the inclusion of the following improvements:

- A new robust formulation of docking mission phases described in Table 2.3 on page 39 and developed in Chapter 2. Tested in both experiment and simulation.
- An upgrade of the solver module with the spline-based trajectory planning Algorithm 3.1 on page 91 developed in Chapter 3. Tested in both experiment and simulation.
- Improvement of the previous PID controller to a high-performance phase-plane LQR controller in Algorithm 4.1 on page 140 developed in Chapter 4. Tested only in simulation.

The table describing the position planning phases for docking scenarios from Table 2.3 on page 39 is shown here again in Table 5.1:

Table 5.1: MVM maneuvers for any docking scenario.

Maneuver	Controllers	Termination Conditions
1. DP Axis Alignment	Path planner & LQR tracking controllers	time limit
2. Inline Approach	Path planner & LQR tracking controllers	time limit
3. Berthing	LQR controllers	state error $<$ tol
4. Capture	Open-Loop Thrust	time limit

These are the phases that are executed in the following simulation and experimental tests. The trajectory planner is necessary for the **DP Axis Alignment** and **Inline Approach** phases. A description of the planning, control, and estimation architecture during these two phases is shown in Figure 5-1. The MVM module compiles the inputs required for the planning algorithm such as the ones in Table 3.5 on page 103 when docking to a rotating spacecraft. The trajectory is calculated once and saved. Then the MVM module feeds the state trajectory $\mathbf{x}^*(t_k)$ to the phase-plane LQR controller that determines a discrete control law executed by onboard thrusters. Sensor noise is added to the measurements $\mathbf{y}(k+1)$ while an estimation algorithm determines the best estimate of the state $\hat{\mathbf{x}}(t_{k+1})$. An Extended Kalman Filter (EKF) developed by Nolet [10] is used as the estimation algorithm. This estimated state closes the loop by feeding back to the controller that closes the error between the desired trajectory from the planner.

The simulation and experimental tests are performed on two SPHERES of identical characteristics. Therefore, the specific values for the terminal positions and size of the obstacle is appropriately defined. The SPHERES has a radius of 10.5 cm center-to-face of DP, both the spacecraft and obstacle radii are defined to be $R = 0.105$ m and $R_{obs} = 0.105$ m. The *berthing position* is defined to be 4 cm from the face of the docking port, which results in being 25 cm centroid to centroid distance between the two spacecraft, $b = 0.25$ m. The *DP alignment position* is set to be 10 cm longer than the berthing position, resulting in a 35 cm separation distance, $z = 0.35$ m. Now, consideration is made to the buffer length R_{buffer} added to the $R + R_{obs}$ spherical ob-

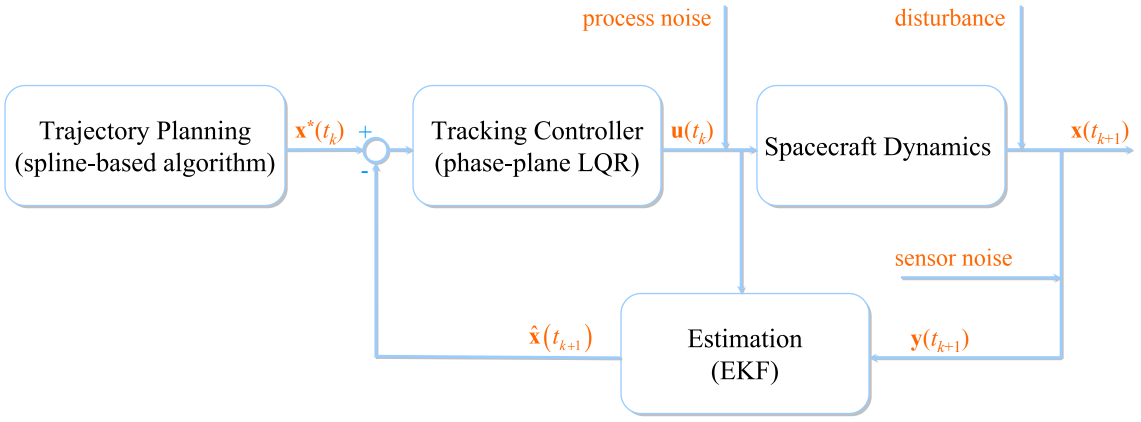


Figure 5-1: Block Diagram of Trajectory Planning, Control, and Estimation

stacle. The buffer distance is defined to have two different values for the two different phases, *DP Axis Alignment* and *Inline Approach*. The distance is chosen to maximize the size of the spherical obstacle while not encompassing the terminal positions inside it. For the *DP Axis Alignment* phase, the buffer length is set to be $R_{buffer} = 0.11$ m resulting in a spherical obstacle of $R + R_{obs} + R_{buffer} = 0.32$ m, which is 3 cm less than the DP alignment position. As for the Inline Approach phase, the buffer length is set to $R_{buffer} = 0.02$ m with a total obstacle size of $R + R_{obs} + R_{buffer} = 0.23$ m, 2 cm less than the berthing position. As a result, the planning for the first phase will consider a much larger obstacle sphere and a smaller one for the second phase. Also the final time in the path planning for the two phases is different, $t_f = 100$ seconds for DP Axis Alignment and $t_f = 30$ seconds for the Inline Approach phase. These values hold for all the simulation tests, and is summarized in Table 5.2.

First the simulation runs are discussed to study the complete fusion of the newly developed algorithms as an autonomous control system for docking to complex tumbling spacecraft.

Table 5.2: Parameters for DP Axis Alignment and Inline Approach phases for simulation tests.

	DP Axis Alignment	Inline Approach
final time	$t_f = 100sec$	$t_f = 30sec$
terminal positions	$z = 0.35m$	$b = 0.25m$
chaser radius	$R = 0.105m$	$R = 0.105m$
target radius	$R_{obs} = 0.105m$	$R_{obs} = 0.105m$
buffer distance	$R_{buffer} = 0.11m$	$R_{buffer} = 0.02m$
total obstacle	$R + R_{obs} + R_{buffer} = 0.32m$	$R + R_{obs} + R_{buffer} = 0.23m$

5.1 Simulation Docking

A simulation study is done on the complete new autonomous control system that combines the new phase sequences, trajectory planning algorithm, and phase-plane LQR controller. Simulations are generally a first step at verifying a newly developed control system. They are quick and provide the complete set of data of the dynamics. However, the simulations only account for the physics that are modeled and can provide results not nearly similar to a realistic environment. The simulation developed for testing the new algorithms is of enough fidelity to be a reasonable representation of the actual dynamics [10]. It is originally written to simulate the dynamics of the SPHERES aboard the ISS. The MATLAB simulator is built with the following features [10]:

- **Dynamics**: Double integrator translational and rigid-body attitude dynamics.
- **Uncertainties**: Process and measurement noise of characteristics extracted from experimental data of the SPHERES.
- **Control**: Controllers are executed at a discrete cycle of 0.5Hz for position and 1Hz for attitude control.
- **Estimation**: Discrete Extended Kalman Filter (EKF) at an estimation rate of 5Hz for best state estimate.

The simulations are performed on two docking scenarios that are the most complicated tumbling dynamics considered in this thesis:

Docking to Fixed Rotating Target Out-of-Plane The target spacecraft performs a steady rotational tumble where the docking port axis sweeps a plane where the chaser spacecraft is not initially located. The targets' position stays fixed through the scenario as the chaser needs to plan a path around the target spacecraft avoiding it as an obstacle.

Docking to Fixed Coning Target Facing Backwards In this scenario, the target spacecraft turns 180° to face its back to the chaser and performs a steady rotation where its rotation vector is not perpendicular to the docking port axis. This setup causes the docking port axis to sweep a cone. The chaser spacecraft needs again to maneuver about the target obstacle and reach its final state.

These two scenarios show the autonomous control system ability to dock from the most difficult initial configurations. The success of these scenarios assures in the confidence of the architecture to work on any random initial conditions.

5.1.1 Docking to Rotating Spacecraft Out-of-Plane

This scenarios attains the complexity of requiring path planning with obstacle avoidance, a scenario the previous glide-slope algorithm is unable to accomplish. The planning is performed for a tumbling target spacecraft that is rotating at a rate of 2.415 deg/sec about a random axis of rotation. However, the rate axis is assured to have the docking port axis of the target sweep a plane in which the chaser spacecraft is not initially located. This means that the chaser spacecraft needs to maneuver about the x, y, and z axis to avoid the target obstacle and reach its final state. A depiction of this scenario is shown in Figure 1-2 on page 21 in Section 1.2.

Figure 5-2 shows the calculated trajectories from the spline-based planning algorithm and the actual path followed by the spacecraft for the first two phases. The phase-plane LQR controller shows a tracking performance of 1 cm deviation from the desired paths. This error tolerance is within the requirements specified in Chapter 4 of 1 cm. The 13-element state difference between the chaser and target spacecraft is shown in Figure 5-3. The main observation is in the position and attitude difference near the end of the scenario. For the first 130 seconds, the first two phases are executed with the use of the path planner. Afterwards, the berthing position is tracked as a step input to the phase-plane LQR controller until tight constraints in tolerance to being at the berthing position and pointing straight towards the target spacecraft for DP alignment. The quaternion attitude difference shows that the chaser spacecraft reached the line-of-sight (LOS) cone described in Chapter 2 at about 110 seconds as it decided to regulate its attitude. These berthing position constraints are satisfied at 180 seconds and the chaser spacecraft executes the Capture phase as it thrusts towards the target spacecraft closing in the 4 cm distance to a physical contact. Since the spacecrafts are 10.5 cm radius, a relative distance of 21 cm results in contact. This happens at the end of the capture phase.

A 3D plot of the computed trajectories and the one actually followed is shown in Figure 5-4. The drawn obstacle sphere is that of the DP Axis Alignment phase obstacle size. Therefore, the trajectory for Inline Approach enters this obstacle as its' obstacle size is much smaller described in Table 5.2. The plot shows reasonable trajectories calculated by the spline-based planning algorithm and good following performance by the phase-plane LQR controller.

The new autonomous control system proved effective at docking to a rotating target spacecraft with satisfactory performance. In this scenario, the docking port swept a plane where the chaser is not initially located. Thus, the chaser computed a 3D trajectory twice while avoiding the obstacle. The tracking of the berthing position and execution of the controlled capture maneuver performed well.

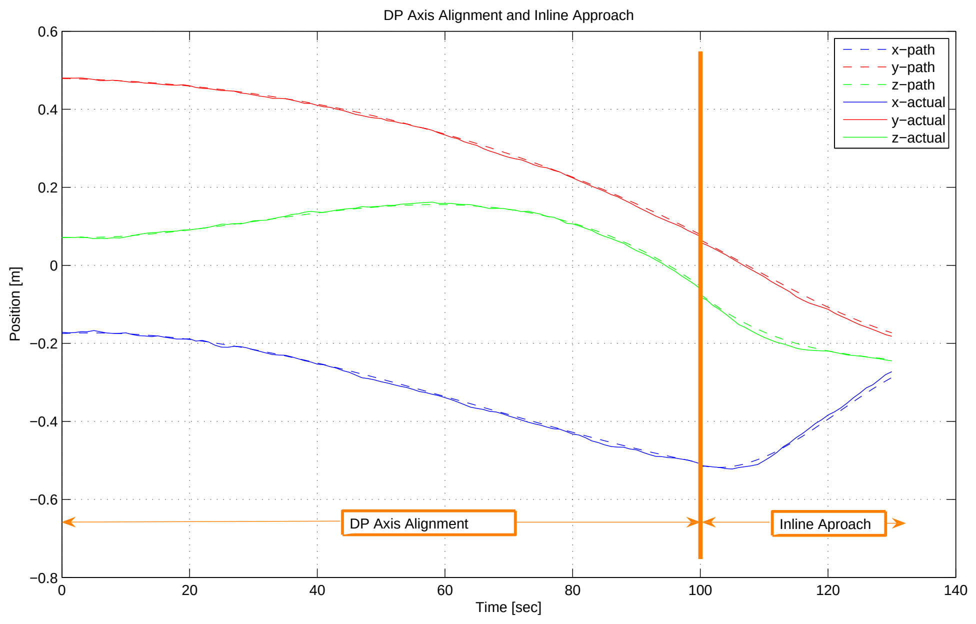


Figure 5-2: Planned and Actual State Trajectories for Docking to Rotating Target Out-of-Plane

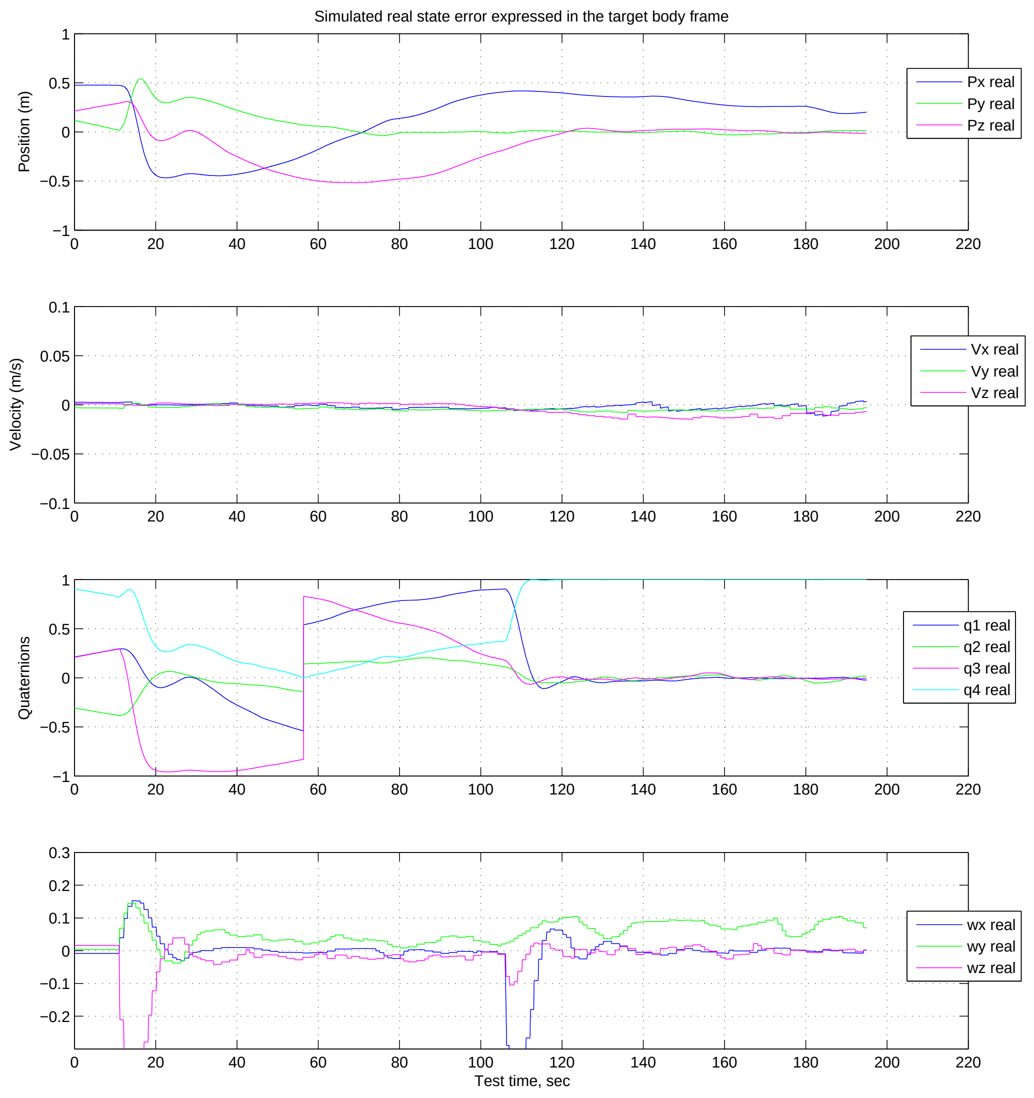


Figure 5-3: State Differences for Docking to Rotating Target Out-of-Plane

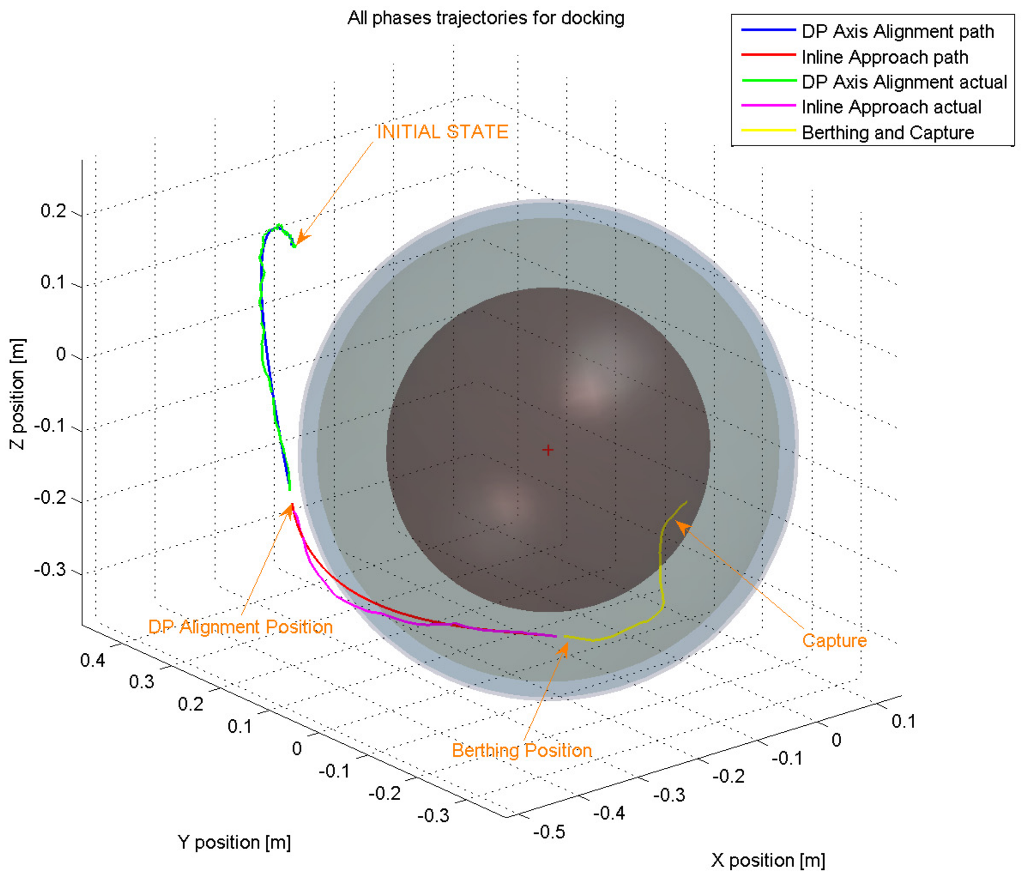


Figure 5-4: 3D State Trajectories for Docking to Rotating Target Out-of-Plane

5.1.2 Docking to Coning Spacecraft backwards

The tumbling dynamics of the target spacecraft in this scenario is still a pure rotation, but the rotation rate vector of 2.496 deg/sec is not perpendicular to the docking port axis as in the previous scenario. This causes the docking port axis to sweep a cone. In addition, the target spacecraft faces its back towards the chaser during the complete coning tumbling dynamics. Therefore, it requires the chaser to maneuver in all three x,y, and z axis to avoid the obstacle and reach the terminal position, DP alignment and berthing positions.

The position trajectories of the DP Axis Alignment and Inline Approach phases is shown in Figure 5-5. The computed trajectories from the path planner is shown as a dashed curves while the actual followed by the tracking controller are the solid curves. The accuracy in tracking for this scenario is about 2 cm, larger than then the phase-plane LQR simulations from Chapter 4. This is due from the addition of process and measurement uncertainty to the system simulation. The previous simulation study of the tracking controllers in Chapter 4 is performed on a deterministic system, no noise added. The additional uncertainties result in the larger accuracy value. Nevertheless, the buffer distance for the first phase is 11 cm from Table 5.2 so the tracking accuracy assures no collision. However, the Inline Approach phase has a buffer length of 2 cm and arises concerns. Fortunately, there is no collision in this simulation run of such a docking scenario. This emphasizes the importance in choosing the buffer distance appropriate to the tracking uncertainty.

The state difference of the two spacecraft is shown in Figure 5-6. The first two phases brought the chaser spacecraft to the berthing position at 168 seconds without collision. Then this position is tracking for a short time of 6 seconds before the Capture phase is initiated. This is how long it takes the chaser to satisfy the attitude alignment and steady control of the berthing position to a certain low tolerance satisfactory for executing the controlled contact. At about 120 seconds is when the chaser entered the LOS cone to begin adjusting its attitude to align the “docking port mechanism”.

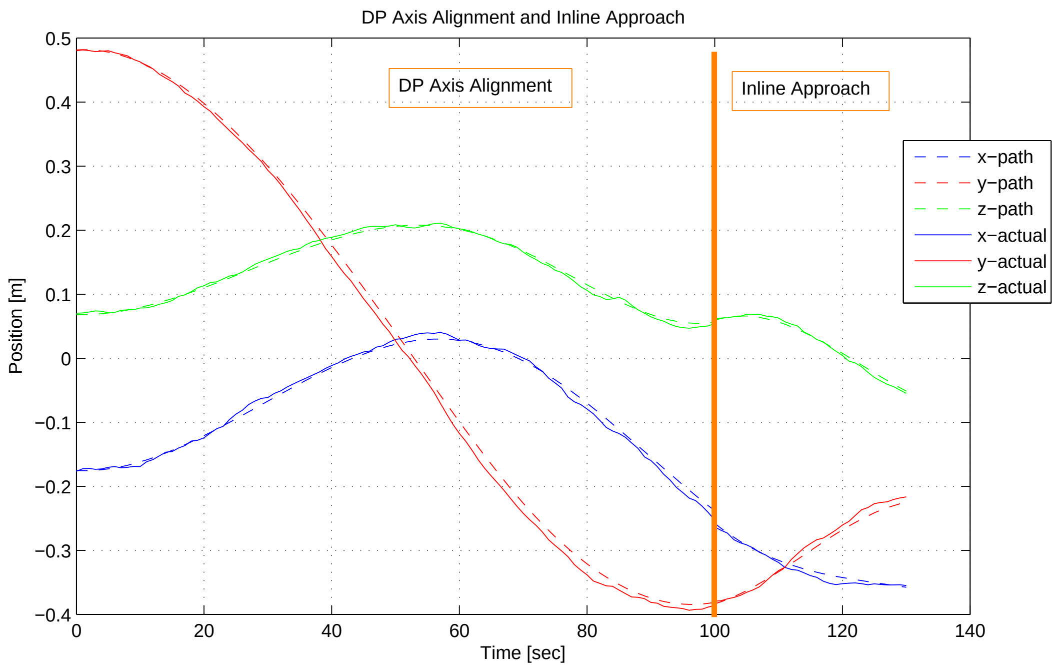


Figure 5-5: Planned and Actual State Trajectories for Docking to Coning Target Facing Backwards

Figure 5-7 shows a 3D plot of the four phases executed to achieve docking to a coning spacecraft. The DP Axis Alignment trajectory passes nicely about the spherical obstacle. The following trajectory moved the chaser closer to the docking port of the target, to within 4 cm face-to-face berthing position. At this point, it does not take long until the chaser satisfies all the constraints to capture. The dots show the coning path of the target spacecraft docking port axis.

The new planning algorithm and tracking controller managed by the improved MVM module achieved the most complex docking scenario in this thesis.

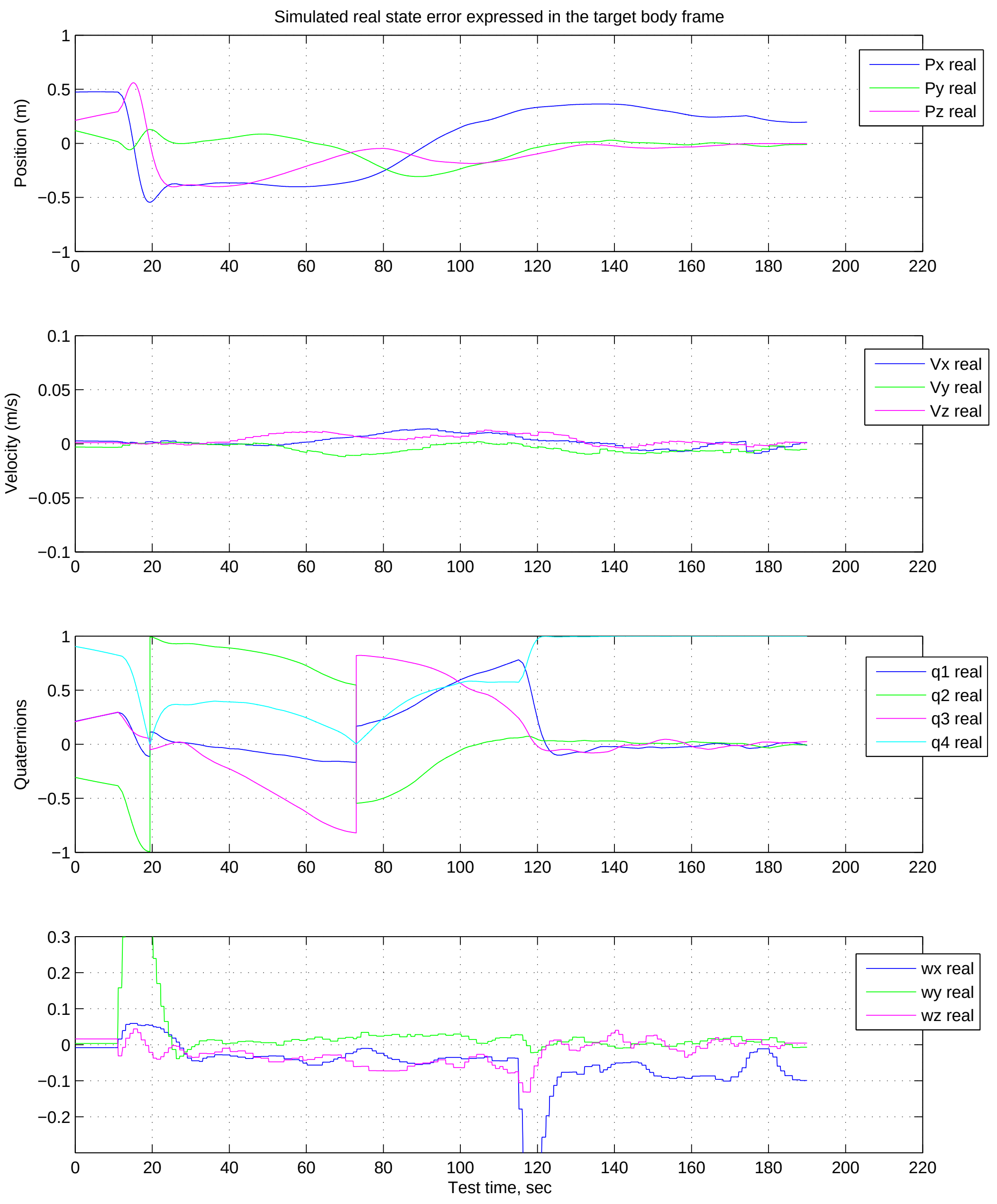


Figure 5-6: State Differences for Docking to Coning Target Facing Backwards

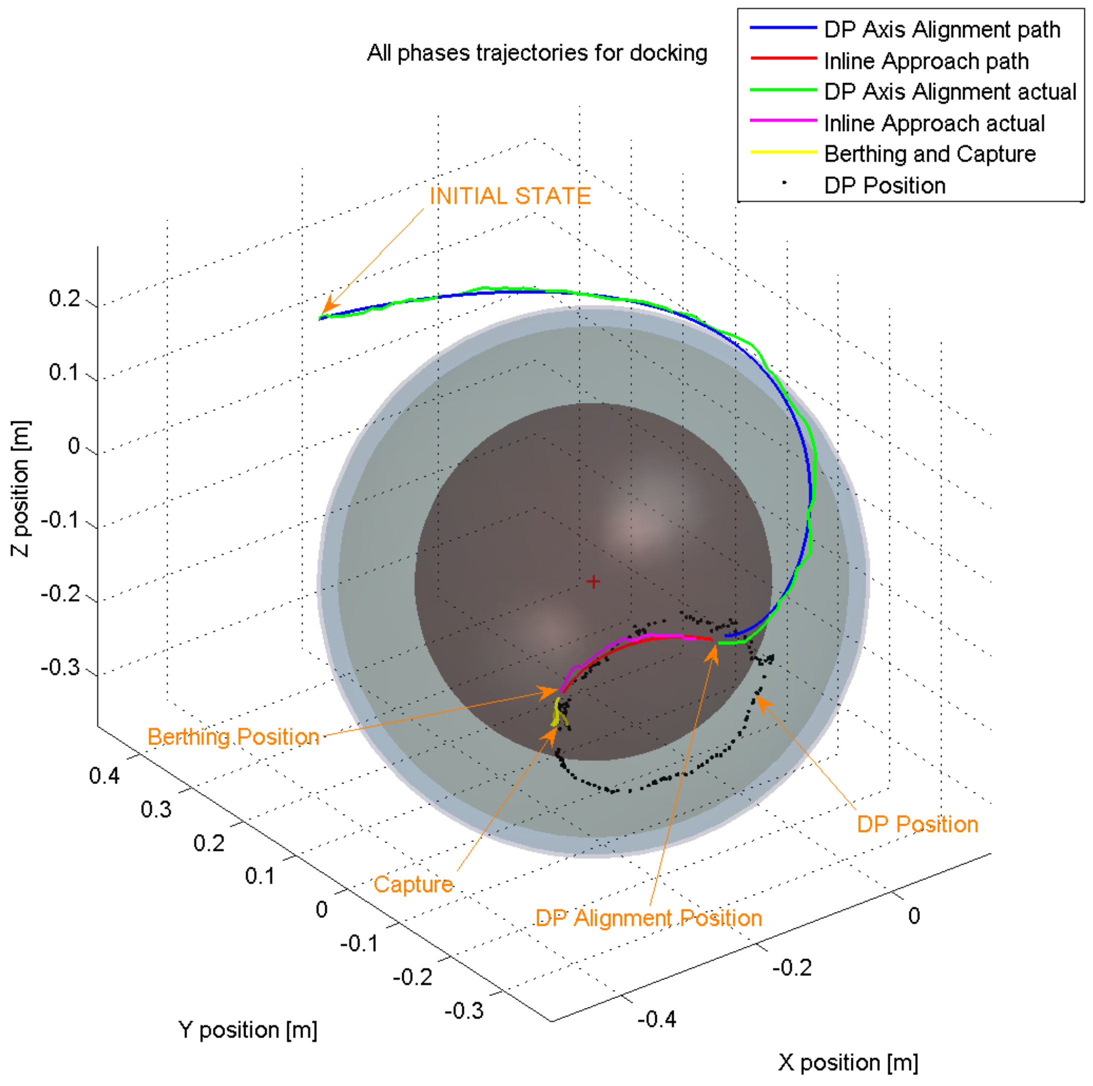
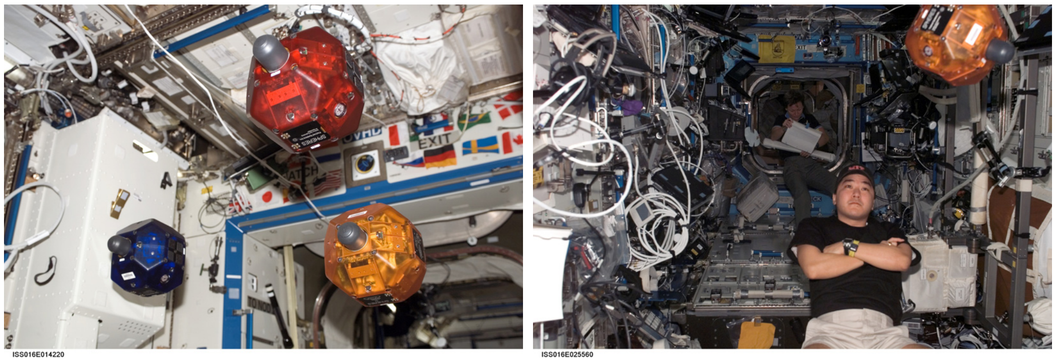


Figure 5-7: 3D State Trajectories for Docking to Coning Target Facing Backwards



(a) Three SPHERES aboard the ISS inside the US Laboratory

(b) Astronaut Daniel Tani performing an experiment using SPHERES during test session 10

Figure 5-8: SPHERES testbed aboard the ISS

5.2 Experimental Docking aboard the ISS

The SPHERES testbed was developed by the MIT Space Systems Laboratory to incrementally advance the development, validation, and maturation of control, autonomy, and estimation algorithms. It utilizes the unique space environment provided by the International Space Station (ISS) to offer full micro-gravity dynamics and controlled experimental testing. A more detailed discussion of the facility can be found in [4, 15, 10, 11]. Currently, there are three SPHERES aboard the ISS shown in Figure 5-8(a) undergoing experiments in spacecraft formation flying, autonomous docking, reconfiguration, and fragmented configurations. Figure 5-8(b) shows astronaut Daniel Tani performing an experiment using SPHERES during test session 10.

The testbed SPHERES has a face-to-face distance of 0.21 m while its largest diameter is of 0.25 m. It uses twelve cold-gas thrusters positioned around the spacecraft to maneuver in all six degrees of freedom; thus providing translational and attitude control to the system. Each thruster provides a maximum force of 0.12 N resulting in a maximum axial force of 0.24 N and 0.012 Nm of torque. The metrology system consists of ultrasound times-of-flights from five beacons enclosing a testing volume of 1.4 m x 0.9 m x 1.2 m and three internal gyroscopes. This data feeds to the estimation algorithm to provide real-time state data at 5Hz. In regards to autonomous docking experiments, the SPHERES testbed does not have any mechanical docking ports attached, but instead, a velcro system is used for the "locking mechanism." Still, the location of the velcro face is referred to as the docking port face of the spacecraft.

The first successful test of an online path planner in-space was performed during the 10th testing session of the SPHERES testbed aboard the ISS on December 12, 2007. In addition to the standard path planning, the algorithm also accounted for obstacle avoidance. The planner is the spline-based planning algorithm developed in this thesis in Chapter 3. The experiment consisted of autonomous docking to a fixed non-tumbling target spacecraft that has its back facing towards the chaser. This requires the chaser spacecraft to plan a trajectory around the target considering it as an obstacle to get in front of the docking port. The results of this experiment are discussed in the next section.

5.2.1 Docking to Fixed Non-Tumbling Spacecraft Facing Backwards

This experiment demonstrates the new spline-based planning algorithm and improved MVM module with the four mission phases. The new phase-plane LQR controller is not implemented in this experiment and the standard PID controller from Section 4.1 on page 120 is used for all trajectory tracking. The position control law is executed at a rate of 0.5 Hz while the attitude controller runs twice as fast at 1 Hz. Also, only the position planning consisting of the four phases from Section 2.3.1 on page 39 is demonstrated. The attitude planning with the line-of-sight (LOS) cone is not implemented. In this experiment, the attitude planning for chaser is **Point to Target** for the first two phases, DP Axis Alignment and Inline Approach, and then **Regulate Attitude** for the Berthing and Capture phases.

Before the four phases are executed, there are two maneuvers that need to be performed in order to get the SPHERES ready for the docking scenario. At the beginning of any experiment with SPHERES that require full state estimation, the onboard EKF algorithm is given 15 seconds to converge to its best estimate of the state while the spacecrafts are freely drifting. Afterwards, a maneuver is added for each spacecraft to acquire the desired initial configuration for the experiment to begin. At this point, there is full state estimation at 5Hz and the spacecrafts are oriented to the correct initial conditions for the specified docking scenario.

As in the simulation from the previous section, the DP alignment position is defined to be 0.35 m relative separation while the berthing position is 0.25 m. However, the obstacle size for the DP Axis Alignment phase is increased to 0.34 m, and the berthing obstacle stays the same at 0.23 m. The final time for the path planning in the Inline Approach is reduced to 20 seconds. The parameters used for the experiment tested docking scenario are summarized in Table 5.3.

Table 5.3: Parameters for DP Axis Alignment and Inline Approach phases for experimental test.

	DP Axis Alignment	Inline Approach
final time	$t_f = 100sec$	$t_f = 20sec$
terminal positions	$z = 0.35m$	$b = 0.25m$
chaser radius	$R = 0.105m$	$R = 0.105m$
target radius	$R_{obs} = 0.105m$	$R_{obs} = 0.105m$
buffer distance	$R_{buffer} = 0.13m$	$R_{buffer} = 0.02m$
total obstacle	$R + R_{obs} + R_{buffer} = 0.34m$	$R + R_{obs} + R_{buffer} = 0.23m$

Figure 5-9 shows the full global state (position, velocity, attitude, and angular rates) of the chaser and Figure 5-10 for the target spacecraft during the test. Figure 5-11 shows a 3D plot of the online calculated paths (dashed curves) and the actual path (solid) followed by the chaser spacecraft. The plotted red sphere within Figure 5-11 is that of the target satellite while the larger purple transparent sphere represents the "obstacle" sphere the chaser planned to avoid. Lastly, Figure 5-12 shows the state differences relative to the target spacecraft body frame. The docking scenario is decomposed into two preliminary maneuver and the four phases for the chaser spacecraft:

1. **Extended Kalman Filter (EKF) Convergence:** The SPHERES satellites are let to drift without actuation while the EKF converges to the proper states

(position, velocity, attitude, attitude rates) for 15 seconds.

2. **Initial Configuration:** The chaser spacecraft orients its attitude to point towards the target spacecraft and waits until the target turns to face its back to the chaser and points the docking port (DP) away. This maneuver setup the initial configuration of both spacecraft to begin the docking scenario and lasts 23 seconds as seen in Figure 5-9.

3. **DP Axis Alignment:** The chaser spacecraft calculates a path online from its current position to 14 cm in front of the targets' docking port face while considering the target as an obstacle so no collision occurs. The path is planned and executed for 100 seconds. Then another 10 seconds is added for the chaser to stay at the DP alignment position. This allows the chaser to be aligned with the docking port axis of the target and prepares to move closer inline to the docking port face. The chaser keeps its attitude pointing towards the target throughout the whole trajectory following. By observing Figure 5-11, the calculated trajectory is feasible and a success. It avoids the target spacecraft, ends 14 cm in front of the docking port, and has a continues and smooth form for execution. However, the trajectory following had very weak performance using the PID controller. There is a maximum deviation of 10 cm from the planned trajectory. Fortunately, the path deviation is away from the obstacle. Also, the buffer distance for this maneuver is set to 13 cm, so no collision would occur if the actual trajectory deviated towards the obstacle. This test emphasized the importance to develop improved tracking controller such as the phase-plane LQR control law. Unfortunately, this controller has not been tested aboard the ISS, but is currently implemented for the next test session.

4. **Inline Approach:** The chaser calculates a second path online to move close to within 4 cm from the target's docking port face. As the chaser did not follow the first path perfectly, it ended in a different location other than the 14 cm in front of the DP, but at 20 cm from the target's face. Therefore, planning of a second path is wise and the calculated trajectory as seen in Figure 5-11 is

feasible and a success. Here, the PID controller is used again to the follow the trajectory. One characteristic of PID controllers is having a higher overshoot than PD-type controller. As a result, when the chaser is following the trajectory to 4 cm within the DP face, it overshot 2 cm further. These results show how dangerous overshooting is at such close proximity. The chaser spacecraft keeps pointing towards the target during this phase.

5. **Berthing:** The chaser spacecraft maintains a distance 4 cm in front of the targets' DP face until the position and state error are below a tight tolerance before executing a controlled capture thrust. The tight position and attitude constraints are for alignment of the two docking ports. The satisfaction of these constraints is seen on the first plot of Figure 5-12, as the relative position in y and z directions are near zero as x is along the docking port axis. The chaser also performs a roll for *Regulate Attitude* to align the DP faces. This phase lasts for 25 seconds.

6. **Capture:** The chaser spacecraft performs a closed-loop thrust to "capture", velcro contact. This is performed by providing a step input to the chaser to move 1 cm into the target spacecraft using the PID controller. There is a 12 second time out of this phase. This is sufficient time for the spacecraft to move in 4 cm. This resulted in a controlled physical contact at 215 seconds.

This experimental test using SPHERES aboard the ISS successfully validated the first online path planner in micro-gravity , the spline-based planner in Algorithm 3.1 on page 91. The planning algorithm achieved in calculating an energy sub-optimal trajectory online that avoids a spherical obstacle. After this experiment, the spline-based planning algorithm could be considered to have matured to TRL 6. The PID tracking controllers showed a poor performance and emphasized the necessity to have a low-overshooting and tight tracking controller. Such a controller is developed in Chapter 4, but has not yet been implemented and tested on the SPHERES testbed aboard the ISS. The experiment also validated the new sequence of phases in position planning for docking missions.

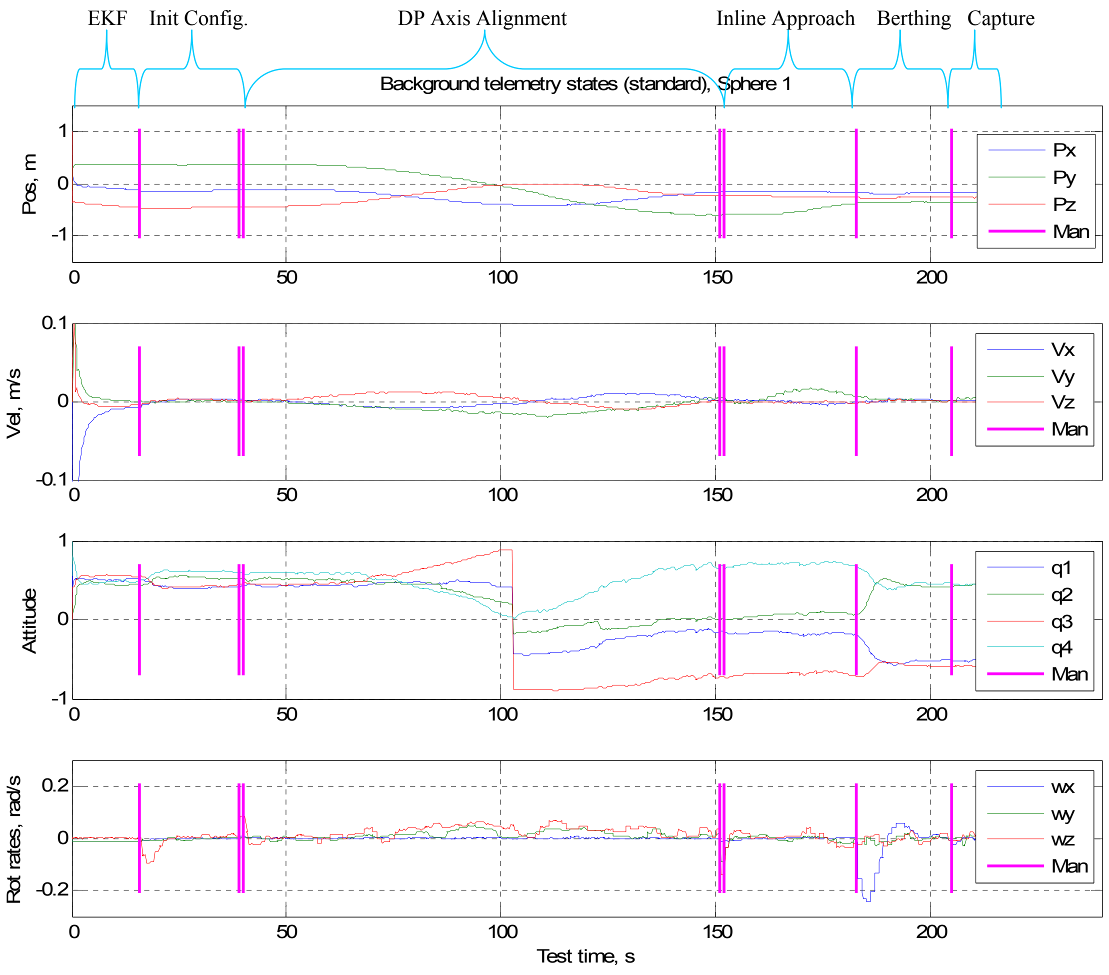


Figure 5-9: State Estimates of Chaser Spacecraft from Experimental Test of Docking to Fixed Non-Tumbling Spacecraft Facing Backwards

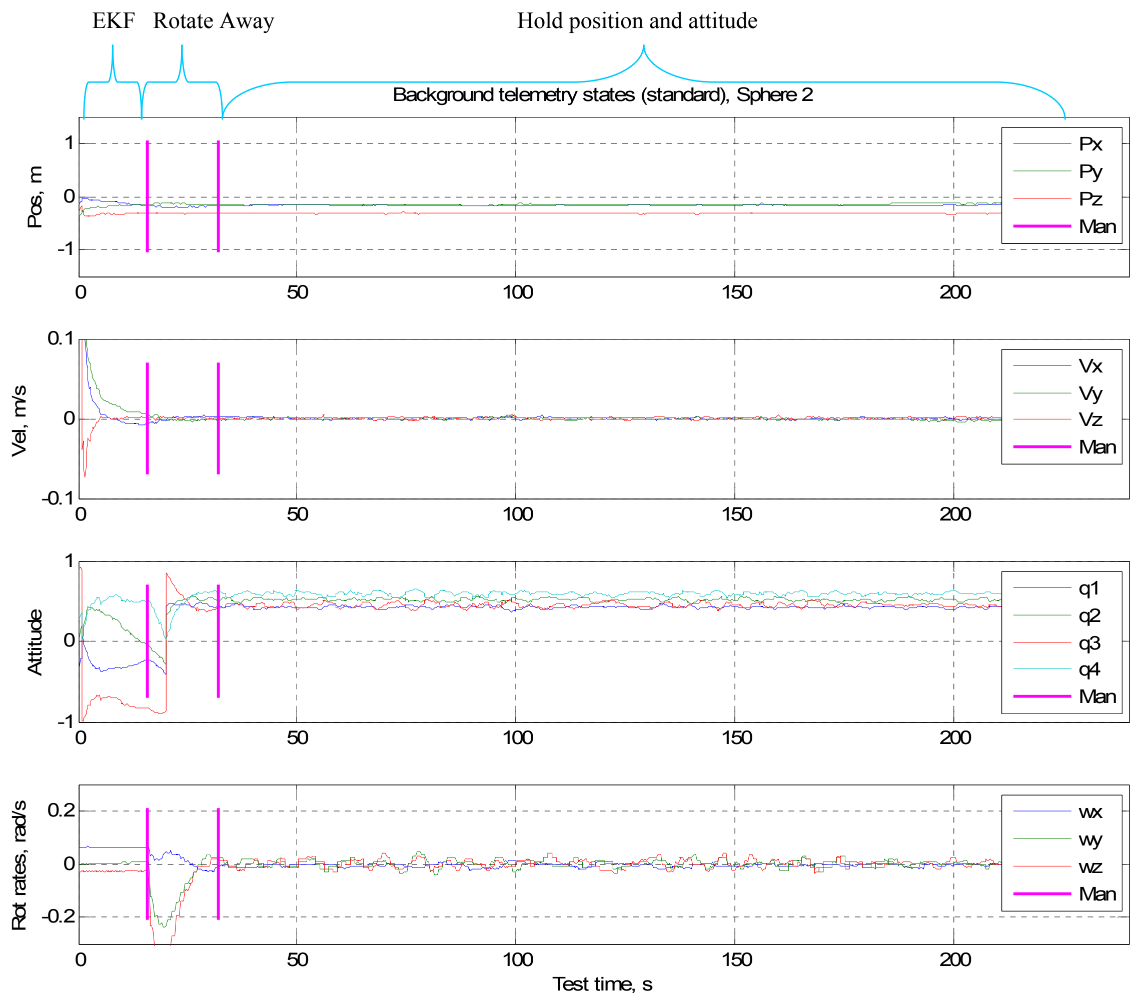


Figure 5-10: State Estimates of Target Spacecraft from Experimental Test of Docking to Fixed Non-Tumbling Spacecraft Facing Backwards

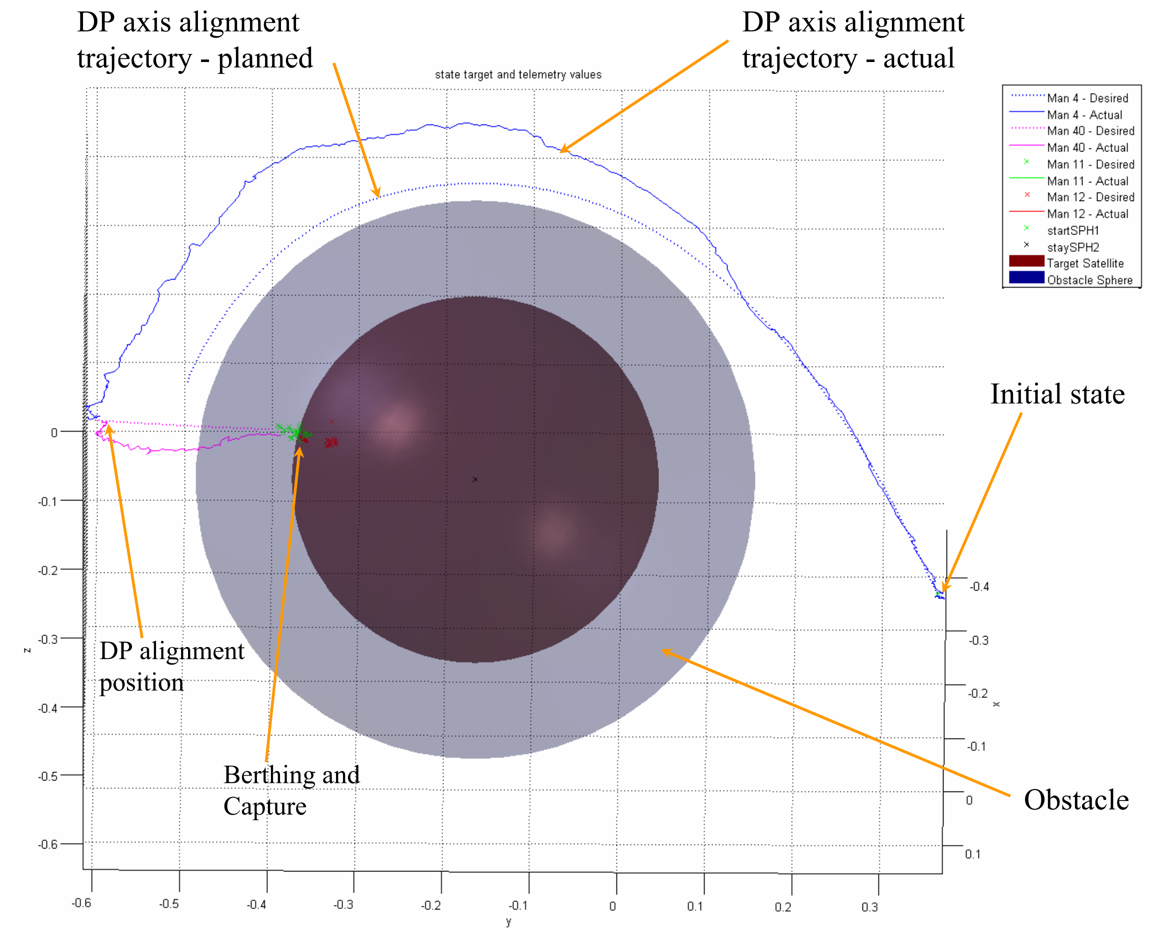


Figure 5-11: 3D Plot of Computed and Actual Trajectories of Chaser Spacecraft from Experimental Test of Docking to Fixed Non-Tumbling Spacecraft Facing Backwards

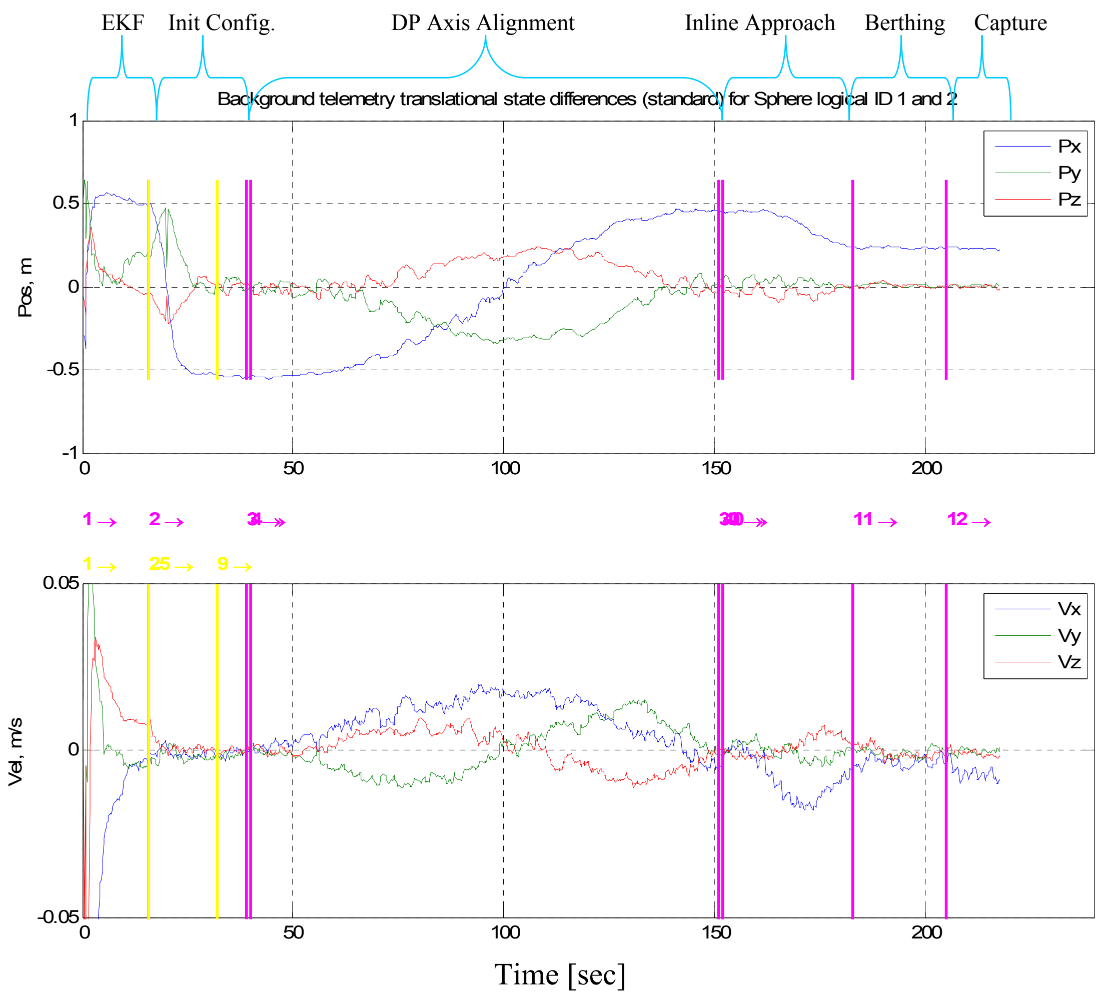


Figure 5-12: State Differences between Both Spacecraft from Experimental Test of Docking to Fixed Non-Tumbling Spacecraft Facing Backwards

5.3 Summary

This chapter validated the new autonomous GN&C architecture and its algorithms, spline-based trajectory planning algorithm and phase-plane LQR controller. Two simulations were performed of the most complicated docking scenarios. The combined path planner and tracking controllers achieved a successful dock of the two scenarios. In order to verify the planning algorithm in a realistic space environment, an experimental test was performed on the SPHERES testbed aboard the ISS. The test successfully demonstrated the first in-space online path planner in a docking scenario of a fixed non-tumbling target spacecraft facing backwards. This brought the spline-based planning algorithm to TRL 6 and significantly advanced this dates space technology for autonomous docking.

Chapter 6

Conclusions and Recommendations

6.1 Thesis Summary

This thesis presented a GN&C architecture and algorithms that attempt to supersede the traditional method of docking with an autonomous onboard solution solution that requires little or no human-in-the-loop supervision. First, there was a discussion of docking scenarios for tumbling target spacecraft to define the problem statement in Chapter 1. Then a GN&C architecture is presented that consists of several modules performing its unique functions in Chapter 2. It was determined that the capability of the architecture is dependent on the algorithms employed in each of the modules. The better each algorithm performs its required function, the better the overall system architecture performs. This was a motivation to develop a new trajectory planning algorithm that accounts for obstacles in Chapter 3. Without a planner with obstacle avoidance, docking to a tumbling spacecraft is not achievable by the previous glideslope algorithm from any initial configuration. Two planner were developed. One that was to be implemented into the GN&C architecture, the spline-based algorithm 3.1, and another to validate the planning results of the algorithm, the variational technique to optimal planning. The comparison showed that the sub-optimal trajectories were close to the benchmark planner and satisfactory for the application of two spacecraft docking. Then in order to track the planned trajectory in a closed-loop system that handles noises and disturbances, several LQR tracking controller were developed in Chapter 4. The performance of each controller was studied and the final phase-plane LQR controller tried to put together the best characteristics of both the LQR and servo-LQR controllers into Algorithm 4.1. Then the spline-based algorithm and the phase-plane LQR controller is coupled together to execute the robustly designed phases of a docking mission managed by the new MVM module described in Section 2.3.1. The complete autonomous control system was tested in simulation and experiment on the SPHERES testbed aboard the ISS. The simulations demonstrated successful docking of the two scenarios: *Docking to Fixed Rotating Target Out-of-Plane* and *Docking to Fixed Coning Target Facing Backwards*. These are the most complicated docking scenarios considered in this thesis. The experimental test performed the first in-space online path planning algorithm to a docking scenario of a fixed non-tumbling target spacecraft facing backwards. The chaser managed to calculate a feasible trajectory online and execute it using a PID controller. This brought the spline-based planning algorithm to possible TRL level 6 and significantly advanced todays space technology for autonomous docking.

6.2 Issues and Recommendations

The issues and recommendations are considered for expanding the complexity of the docking scenarios, improving the new planning algorithm and tracking controller:

- The most complicated tumbling dynamics of the target considered in this thesis is when the spacecraft is performing a steady rotation about some axis, assuming symmetric inertia tensor. Further work should account for asymmetric inertias which lead to nutation dynamics of the docking port. The current architecture and algorithm are in a form to handle these dynamics as the state propagator considers non-trivial inertias. However, future work would need to test these algorithms in such docking scenarios.

- The planning algorithm has shown to perform well in hardware, but there are several issues that still need to be addressed for implementation on an actual spacecraft. This issue is with knowing the state of the target spacecraft at the pre-defined final time t_f. The current method simply propagates the initial state of the target with translational and attitude dynamics model in a deterministic manner. However, the problem here is with knowing how accurate that initial state is. Realistically, the measured state would be provided by hardware sensor and would thus consist of having some level of noise. Propagating the noisy state into the future increases your uncertainty in that state the longer you propagate. It is a similar case in ballistic projectiles. If you know the initial position and velocity to a certain accuracy, how accurate can you predict where it will be in the future. Therefore, the issue of planning under uncertainty arises. Possible solutions by still using the current planner would be to plan ahead only to the time where the level of uncertainty is acceptable. Determining this level of uncertainty for spacecraft docking is also a good topic to study. For example, if we know the final attitude of the spacecraft to within 100 degrees, then if we re-plan, one would get trajectories all over the place that are impossible to follow and might collide with the target. Another possible future work is to extend the path planner to multiple moving obstacles. Currently it is implemented to handle a single non-moving obstacle, as this is sufficient enough for just two spacecraft docking. However, many other mission require in-space assembly and reconfiguration, so multiple spacecraft would be flying around.

- Further work can be performed on the phase-plane LQR controller by adding the velocity state error in the control logic. Currently, only the position error is considered when switching between the LQR and servo-LQR controllers. Considering the velocity error could have the phase-plane controller not apply the servo-LQR below the defined position error if the velocity error is too large. This would increase the robustness of the controller to more extreme dynamics. Further work would be to test this controller in hardware on the SPHERES testbed aboard the ISS to show an improvement to the current PID controller.

Bibliography

[1] Wigbert Fehse. *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, Cambridge, United Kingdom, 2003.

[2] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, Upper Saddle River, NJ, 2002.

[3] Carl Glen Henshaw. *A Variational Technique for Spacecraft Trajectory Planning*. PhD thesis, University of Maryland College Park, 2003.

[4] Mark Hilstad. A multi-vehicle testbed and interface framework for the development and verifcation of separated spacecraft control algorithms. Master's thesis, Massachusetts Institute of Technology, 2002.

[5] Geoffrey Huntington. *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control Problems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[6] Donald E. Kirk. *Optimal Control Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1970.

[7] E. M. Kong, M. O. Hilstad, S. Nolet, and D. W. Miller. Development and verification of algorithms for spacecraft formation flight using the spheres testbed: Application to tpf. *SPIE*, 5491:308–319, 2004.

[8] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, Cambridge, United Kingdom, 2006.

[9] Michael F. Machula and Gurpartap S. Sandhoo. Rendezvous and docking for space exploration. *1st Space Exploration Conference: Continuing the Voyage of Discovery*, 2005.

[10] Simon Nolet. *Development of Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite*. PhD thesis, Massachusetts Institute of Technology, 2007.

[11] A. S. Otero, A. Chen, D. W. Miller, and M. Hilstand. Spheres: Development of an iss laboratory for formation fligh and docking research. *IEEE Aerospace Conference Proceedings*, 25:59–73, 2002.

[12] Michael A. Paluszek and Stephanie J. Thomas. Generalized 3d spacecraft proximity path planning using a*. *Infotech at Aerospace*, 2005.

[13] Richard Quintero, Raymond C. Montgomery, and Jr. Peter Tchoryk. Autonomous rendezvous and docking scenarios for the development of guidelines and standards. *AIAA Space Program and Technologies Conference and Exhibit*, 1993.

[14] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints. *Journal of Guidance control and dynamics*, 4:755–764, 2002.

[15] Alvar Saenz-Otero. *Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station*. PhD thesis, Massachusetts Institute of Technology, 2005.

[16] Lawrence F. Shampine, Jacek Kierzenka, and Mark W. Reichelt. Solving boundary value problems for ordinary differential equations in matlab with bvp4c. *The MathWorks, Inc. tutorial*, 2000.

[17] Marcel J. Sidi. *Spacecraft Dynamics and Control, A Practical Engineering Approach*. Cambridge University Press, Cambridge, United Kingdom, 1997.

[18] Cornel Sultan, Sanjeev Seereram, and Raman K. Mehra. Deep space formation flying spacecraft path planning. *The International Journal of Robotics Research*, 26:405–430, 2007.

[19] W T Vetterling W H Press, S A Teukolsky. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, United Kingdom, 1993.

[20] Bong Wie. *Space Vehicle Dynamics and Control*. AIAA Education Series, Reston, VA, 1998.

[21] Douglas Zimpfer, Peter Kachmar, and Seamus Tuohy. Autonomous rendezvous, capture and in-space assembly: Past, present and future. *1st Space Exploration Conference: Continuing the Voyage of Discovery*, 2005.