Autonomous Rendezvous & Docking of Spacecraft with Non-Cooperative Satellite

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Tushar Sial
ID No. 2019A3TS0215P

Under the supervision of:

Dr. Debasish Ghose
Guidance, Control & Decision Systems Lab (GCDSL)

Department of Aerospace Engineering
INDIAN INSTITUTE OF SCIENCE, BENGALURU

December 2022

Declaration of Authorship

I, Tushar Sial, declare that this Undergraduate Thesis titled, 'Autonomous Rendezvous & Docking of Spacecraft with Non-Cooperative Satellite' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 27th December, 2022

Certificate

This is to certify that the thesis entitled, "*Autonomous Rendezvous & Docking of Spacecraft with Non-Cooperative Satellite*" and submitted by Tushar SIAL ID No. 2019A3TS0215P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor
Dr. Debasish GHOSE
Professor,
Indian Institute of Science, Bengaluru
Date:

"The Only Way to Lose is to Give up"

Tushar

INDIAN INSTITUTE OF SCIENCE, BENGALURU

Abstract

Bachelor of Engineering (Hons.)

Autonomous Rendezvous & Docking of Spacecraft with Non-Cooperative Satellite

by Tushar SIAL

Autonomous rendezvous and docking (AR&D) research is a crucial area in the advancement of space technology. AR&D provides the foundation for many space operations, including space debris removal, in-orbit assembly, refueling, and satellite maintenance. In this study, we provide guidance algorithms for a thrust-vectored spacecraft to rendezvous with an uncooperative tumbling satellite. Mainly two control schemes have been proposed for the relative translational motion: Linearized CW equations with an LQR control algorithm and Koopman operator-based LQR control algorithm for rendezvous operations in the close-by and long-range. For docking, a PN-based guidance algorithm has been designed. An LQR-based attitude controller has also been designed which comes into play during close proximity operations. Various simulation results have been plotted and discussed subsequently to demonstrate the performance and reliability of the suggested controllers. . . .

Acknowledgements

I want to express my gratitude to my thesis supervisor, Prof. Debasish Ghose, from the Department of Aerospace Engineering at the Indian Institute of Science in Bengaluru, for his inspiration and direction. Additionally, I want to thank my co-supervisor, Professor G. Sai Sesha Chalapathi, from the Electrical and Electronics Department of BITS Pilani, Pilani Campus, for his encouragement and assistance throughout the course of this research. Finally, I want to express my gratitude to my family and friends for their unwavering support over the years. . . .

Contents

Declaration of Authorship **i**

Certificate **ii**

Abstract **iv**

Acknowledgements **v**

Contents **vi**

List of Figures **viii**

Abbreviations **x**

1 Introduction **1**

2 Theory & Essential Mathematical Tools **4**

2.1 Frames of References 4

2.1.1 Earth-Centered Inertial (ECI) Frame 4

2.1.2 LVLH Frame 5

2.1.3 Body-Fixed Frame 6

2.2 Transformation between different Frames 7

2.2.1 Preliminaries 7

2.2.2 State Transformation Equations 7

2.2.3 Rotation Matrix and its Derivative] 8

2.2.4 Transformation from LVLH to Body-Fixed Frame 10

2.3 Relative Motion Dynamics 11

2.3.1 Continous-time nonlinear model 12

2.3.2 Discrete-time nonlinear model 13

2.3.3 Clohessy-Wilshire (CW) Equations 13

2.3.4 Modified CW Equations 14

2.4 Koopman Operator 15

2.4.1 Brief overview 15

2.4.2 Lifted Dynamics for Rendezvous Operation 16

2.4.3 Data Driven Approach for computing G_{koop} and H_{koop} 17
2.5 Attitude Equations of Motion 18
2.5.1 Attitude Kinematics Equations 18
2.5.2 Attitude Dynamics Equations 19

3 Guidance Algorithm Design **21**
3.1 LQR Control 21
3.1.1 Linear CW with LQR 22
3.1.2 Koopman with LQR Control 23
3.1.3 Attitude Controller 24
3.2 Docking Plan 24
3.2.1 PN Guidance Algorithm 26

4 Simulation Results **28**
4.0.1 Relative Translational Motion 29
4.0.2 Attitude Control 31
4.0.3 Docking Algorithm 32

5 Conclusion **34**

A MATLAB Codes **36**

Bibliography **51**

List of Figures

1.1 Kessler's Syndrome 1

1.2 On-orbit Servicing 2

2.1 Geocentric Equatorial System, IJK 5

2.2 Orbital Frame 6

2.3 Body Frame 6

2.4 Vector Diagram: Chaser and Target Satellites 8

2.5 ECI and LVLH Coordinate system for satellite rendezvous 11

3.1 Koopman with LQR Controller Design 23

3.2 Docking Path. Path 1 is chosen over Path 2 in order to avoid collision with the solar panel and safely dock with the target satellite 25

3.3 Fly By maneuver 25

3.4 Case 1: When the docking port vector is not rotating 26

3.5 Case 2: When the docking port vector is rotating with a constant angular velocity about some axis as in the case of a tumbling satellite 26

3.6 Motion model of Case 1 (in LVLH frame) 27

4.1 LQR Position Control Block Diagram 29

4.2 Chaser Satellite's position and velocity graphs in LVLH frame for the case of Linear CW with LQR 29

4.3 Chaser Satellite's position and velocity graphs in LVLH frame for the case of Koopman with LQR 30

4.4 Control Thrusts for the case of Linear CW with LQR 30

4.5 Control Thrusts for the case of Koopman with LQR 30

4.6 Control Thrusts for the case of Koopman with LQR 31

4.7 Attitude angle (Euler angles) and Angular velocities errors vs time plots for Attitude Controller 31

4.8 Control Torques for Attitude Controller 31

4.9 Trajectory of Chaser Satellite during docking operation for Case 1 32

4.10 Chaser Satellite's position and velocity graphs in LVLH frame during docking operation for Case 1 32

4.11 Control Thrusts during docking operation for Case 1 33

Abbreviations

AR&D	**A**utonomous **R**endezvous and **D**ocking
ECI	**E**arth **C**entered **I**nertial
LVLH	**L**ocal **V**ertical **L**ocal **H**orizontal
HCW	**H**ill **C**lohessey **W**iltshire
LQR	**L**inear **Q**uadratic **R**egulator
COM	**C**enter **o**f **M**ass
DMD	**D**ynamic **M**ode **D**ecomposition
PN	**P**roportional **N**agivation
LOS	**L**ine **O**f **S**ight

This thesis is dedicated to all those who have helped me succeed in my academic career.

Chapter 1

Introduction

Under a critical density of debris orbiting the Earth, it is possible for a single collision to cascade into an exponentially increasing chain of collisions. The Kessler Syndrome, first proposed by Donald J. Kessler in 1978, has the potential to result in a debris belt, blocking some orbits or preventing mankind from entering space at all. This problem is being addressed by current projects such as NASA's Restore L, the ESA's Clean Space Initiative, and DARPA's Robotic Servicing of Geosynchronous Satellites. Specifically, research into autonomous robotic spacecraft has been of increasing interest. Indeed, with a growing debris population, robots that can perform routine tasks autonomously (such as simple repairs/maintenance, refueling, or debris removal) may become increasingly essential for actively preventing disaster.

FIGURE 1.1: Kessler's Syndrome

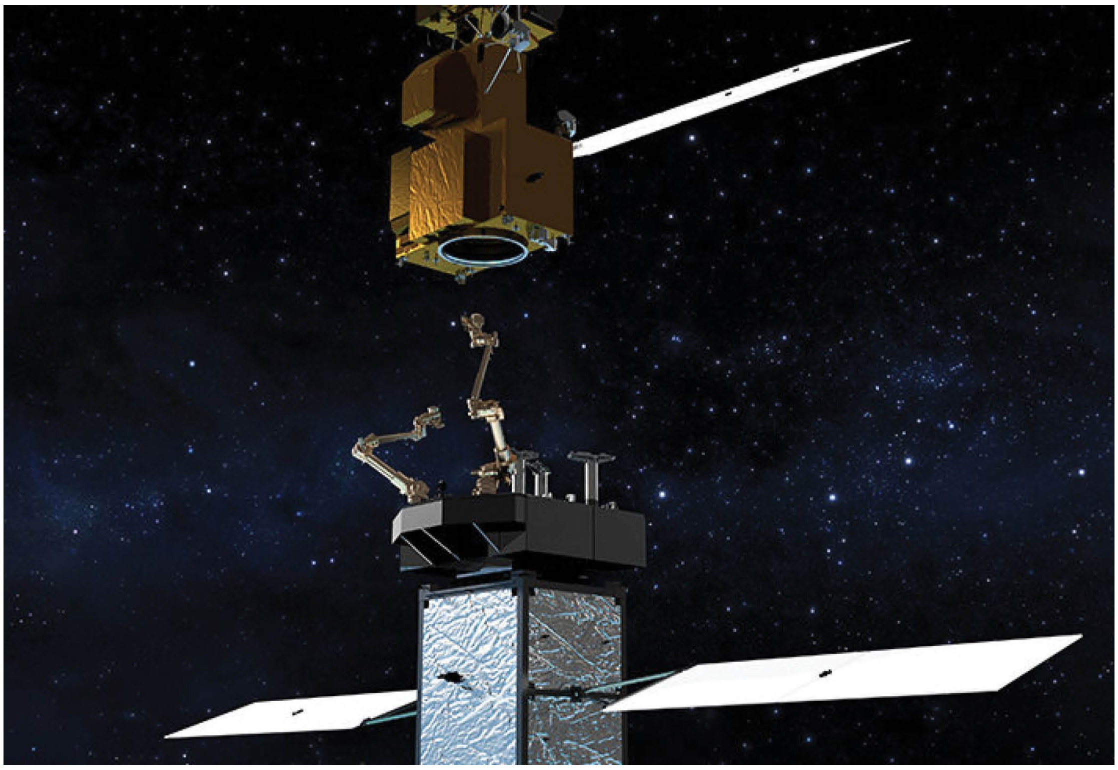

FIGURE 1.2: On-orbit Servicing

The development of space transportation technologies has made numerous extraterrestrial activities possible. Restocking the target spacecraft's material resources is important to support an operation for a long time. The autonomous on-orbit servicing of satellites is a fascinating application of space robotics. In this context, several autonomous missions to cooperative and/or uncooperative targets (which in general can be tumbling and orbiting the Earth elliptically/circularly) will require rendezvous and docking procedures.

The chaser spacecraft must be brought close to the destination spacecraft through rendezvous operations. The main stages of a rendezvous mission are typically the Launch & Guided phase, Homing phase (Terminal Guidance for Rendezvous), Orbital Rendezvous, Final approach/ proximity operations, and Docking.

The chaser spacecraft is launched from the ground (direct launch to rendezvous), from a parking orbit, or from an intermediate orbit, which starts the first phase, Launch & Guided (orbital rendezvous). The following phase is the homing phase, often known as the "approach phase" or "final phase." For closed-loop navigation to bring the chaser spacecraft from a distance of roughly 10–100 km to around 1 km (far-field rendezvous) from the target spacecraft, sensor acquisition is done during this phase. The phase known as the Final Approach comes next. Due to potential "path limitations" or "fly-around" issues that may arise as the chaser spacecraft approaches the target from a distance of a few meters, this particular phase is assigned a different identity. To prevent colliding with projecting appendages, delicate panels, etc. of the target spacecraft, a

”Path restricted” technique is required. Gaining a docking corridor that is at a substantial angle to the approach trajectory may be the goal of the ”fly-around.” The last phase, docking, differs from the approach phase in the following ways. The two spacecraft’s attitudes must be matched while simultaneously bringing their relative location and velocity to a standstill using control forces and torques.

In general, a guiding function aims to steer a spaceship from an initial pose to a final state that mitigates external disruptions. Depending on the objective of the flight, the guidance must be able to shorten either the aircraft’s duration or its fuel consumption. Based on the chaser’s relative position and velocity in relation to the target, the guidance will determine the accelerations necessary to track and attain the final conditions, the time remaining to accomplish the maneuver, and fuel minimization.

Making sure that the spacecraft maintains the proper trajectory and desirable attitude is the aim of the control function. The actuators receive the controlled force or torque produced by the control law in order to carry out this function. For near-field and final-approach rendezvous operations, various control strategies have been put forth.

The project’s motivation is to design a guidance scheme for a Thrust-vectored Satellite (Chaser Satellite) in order for it to rendezvous & dock with another Satellite (Target Satellite) that is orbiting in an ellipse pattern. In the first part, two different schemes have been considered for the Rendezvous operation. In the first scheme, Linearized CW equations along with the LQR control method have been used. In the second scheme, the nonlinear rendezvous equations are lifted into a higher dimension using the Koopman operator, where they behave linearly and then an LQR-based control method has been designed. In the second part, a close-range maneuver & docking algorithm is developed. During this phase, apart from the relative, position, and velocity, orientation also has to be taken into account. In order to achieve this, inspiration is taken from the Impact-Guidance algorithm which you conventionally use for missile guidance for intercepting operations. For attitude matching, again LQR control method has been considered. This is followed by simulating the results of the proposed controllers to illustrate their effectiveness and then investigating the robustness of the control schemes to changes in the parameters of the system, (like J2 perturbations, solar pressure, etc).

The thesis follows the following format: The reference frames taken into consideration and the transformation matrices to transfer from one frame to another are described in Chapter 2. Then the attitude dynamics and kinematic equations, together with a quick overview of the Koopman Operator and data-driven methods, are presented. Impact-Guidance Control Theory and the Koopman-based LQR approach are discussed in Chapter 3. The outcomes of the simulated case study are presented in Chapter 4. Chapter 5 offers the conclusions and recommendations for future work.

Chapter 2

Theory & Essential Mathematical Tools

This chapter covers the required mathematical components needed to execute the rendezvous maneuver and the proximity operations. The Clohessy-Wiltshire (linearized) equations for the relative motion between the chaser and the target are briefly defined, as are the coordinate systems employed. Next, the continuous-time and discrete-time nonlinear state space models for satellite rendezvous are introduced. In addition, a brief overview of the data-driven approaches and the Koopman operator has been given.

2.1 Frames of References

To fully characterize the relative motion of a spacecraft and its attitude around the Earth, three primary coordinate frames of reference are needed:

- Earth-Centered Inertial Frame
- LVLH Frame
- Body-fixed Frame

2.1.1 Earth-Centered Inertial (ECI) Frame

The Earth's center of mass is where the F_{IJK} system is born. Because it is usually believed that the Earth is a sphere with a homogeneous distribution of mass, the origin of the frame is the geometric center of the planet. This frame is also often referred to as IJK. The Capricornus constellation is indicated by the I axis. When the Earth crosses the vernal equinox, this

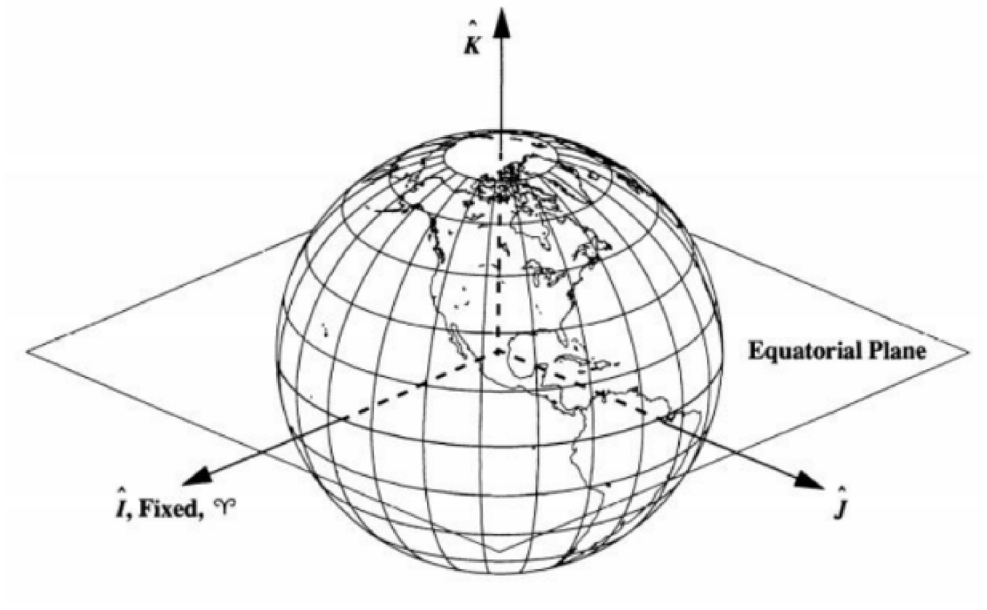

Figure 2.1: Geocentric Equatorial System, IJK

orientation is described as the direction pointing from the center of the Earth to the center of the Sun. Figure 2.1 depicts the J axis and I axis as being in the equatorial plane. The right-handed coordinates are instead completed by the K axis, which is orthogonal to the equatorial plane and points in the general direction of the North Pole. It is customary to use this system to describe how satellites orbit the Earth because it is thought to be fixed in space. The ECI is not actually inertial because the Earth is moving as well. But this assumption can is fairly valid if we were to treat small objects in orbit around it.

2.1.2 LVLH Frame

The Local-Vertical/Local-Horizontal, F_{LVLH}, is frequently used to characterize the relative motion of a chaser spacecraft with respect to a target satellite that is simultaneously orbiting the Earth. The system is anchored to the target center of mass, O_{LVLH}. The spacecraft is not regarded as a 3D body, but rather as a point mass. The z_{LVLH} axis is here thought to be pointed toward the Earth, but since there is no set convention, it can also be found in the literature with the opposite verse. The x_{LVLH} is always perpendicular to the z_{LVLH} and perpendicular to the local orbit track and parallel to the spacecraft's component of velocity. x_{LVLH} and z_{LVLH} were both located on the intended orbit plane. The right-handed rule finally finds the y_{LVLH} as a result, and it is pointed away from the orbit plane.

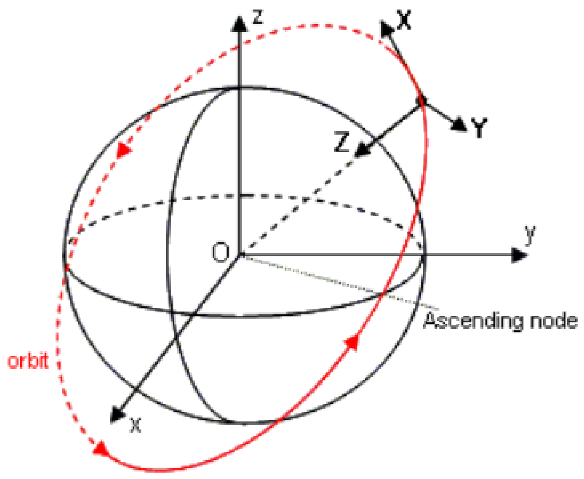

FIGURE 2.2: Orbital Frame

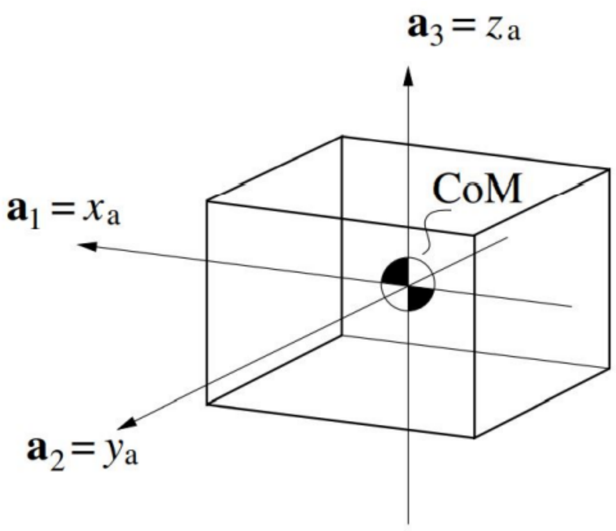

FIGURE 2.3: Body Frame

2.1.3 Body-Fixed Frame

A spacecraft's absolute attitude and rotations about its mass are commonly described by the body frame, or F_B. This frame is anchored to the O_B, or center of mass. It is important to note that a spacecraft's center of mass typically fluctuates throughout a mission; the primary cause being the loss of propellant during a maneuver, which results in a mass reduction. Additional secondary causes include the slosh of propellant, the deployment of solar panels, or the letting go of a tether used to stabilize gravity gradients. We ignore this effect in this case for the sake of simplicity. The body frame axes' directions aren't always well-defined. It is typical to think of x_B aligned with the direction of the docking axis at the conclusion of the rendezvous, which in this case is the same as the x_{LVLH}. The other two axes, y_B and z_B are situated in the same plane and are perpendicular to the unit vector of x_B. In relation to the orbital frame, this coordinate system is likewise totally mobile.

2.2 Transformation between different Frames

The coordinate frames of relevance were discussed in the previous paragraph. In many situations, it is helpful to describe how a body moves or behaves in a frame of reference other than the one it has been specified in. As a result, transformation matrices are employed to transfer data between different reference frames.

2.2.1 Preliminaries

For a unit vector $\hat{u}$, the time derivative is given by:

$$\frac{d}{dt}(\hat{\mathbf{u}}) = \frac{d}{dt}\left(\frac{\mathbf{u}}{u}\right) \tag{2.1}$$

$$= \frac{\dot{\mathbf{u}}}{u} - \frac{\dot{u}}{u^2}\mathbf{u} \tag{2.2}$$

Recalling that a vector's dot product with its derivative is given by

$$\mathbf{u} \cdot \dot{\mathbf{u}} = u\dot{u} \tag{2.3}$$

Combining Equations 2 & 3 and removing $\dot{\mathbf{u}}$, we get:

$$\frac{d}{dt}(\hat{\mathbf{u}}) = \frac{1}{u}[\dot{\mathbf{u}} - (\hat{\mathbf{u}} \cdot \dot{\mathbf{u}})\hat{\mathbf{u}}] \tag{2.4}$$

2.2.2 State Transformation Equations

With reference to Figure 2.4, we define 2 satellites namely the Chaser and Target. The Target satellite's state is used to define a rotating frame at that particular satellite. The central body serves as the center of an initial inertial frame. Set the following two relative location and velocity vectors as your starting point:

$$\mathbf{r}_{12}^I = \mathbf{r}_2^I - \mathbf{r}_1^I \tag{2.5}$$

$$\dot{\mathbf{r}}_{12}^I = \dot{\mathbf{r}}_2^I - \dot{\mathbf{r}}_1^I \tag{2.6}$$

The primary body is the inertial frame for vectors with a superscript I, and the intended target satellite is the rotating frame for vectors with a superscript R.

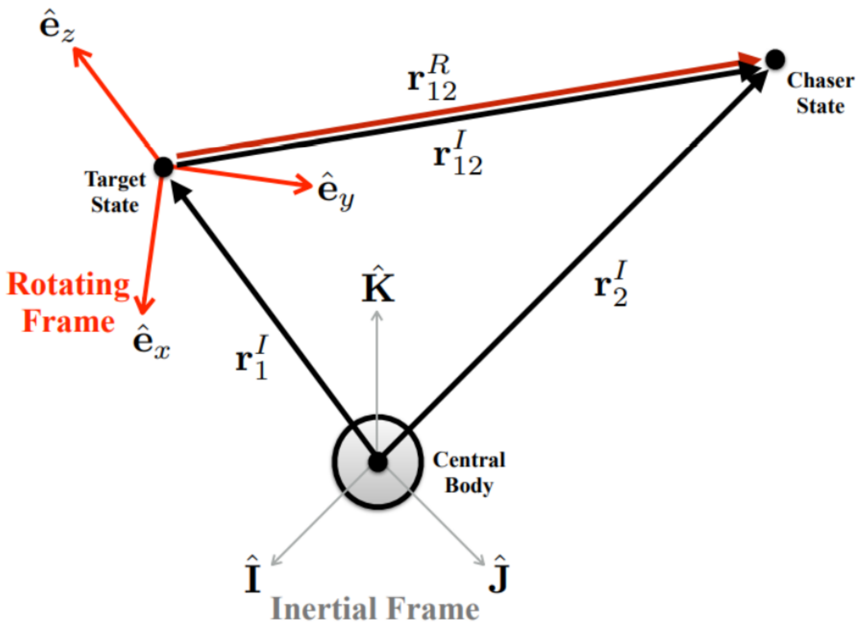

FIGURE 2.4: Vector Diagram: Chaser and Target Satellites

From the inertial frame to the rotating frame, the position and velocity are transformed using the equations below:

$$\mathbf{r}_{12}^R = [\mathbf{C}]\mathbf{r}_{12}^I \tag{2.7}$$

$$\dot{\mathbf{r}}_{12}^R = [\dot{\mathbf{C}}]\mathbf{r}_{12}^I + [\mathbf{C}]\dot{\mathbf{r}}_{12}^I \tag{2.8}$$

The 3×3 rotation matrix from the inertial to the rotating frame is represented by the matrix $[\mathbf{C}]$, and its derivative is represented by $[\dot{\mathbf{C}}]$. Equation 7 is first multiplied by $[\mathbf{C}]^T$, which is equal to $[\mathbf{C}]^{-1}$, to produce the inverse transformation (from the rotating frame to the inertial frame) and then differentiated:

$$\mathbf{r}_{12}^I = [\mathbf{C}]^T\mathbf{r}_{12}^R \tag{2.9}$$

$$\dot{\mathbf{r}}_{12}^I = [\dot{\mathbf{C}}]^T\mathbf{r}_{12}^R + [\mathbf{C}]^T\dot{\mathbf{r}}_{12}^R \tag{2.10}$$

2.2.3 Rotation Matrix and its Derivative]

The rotating frame is defined by three basis vectors, which are $[\hat{\mathbf{e}}_{\mathbf{x}}$, $\hat{\mathbf{e}}_{\mathbf{y}}$, and $\hat{\mathbf{e}}_{\mathbf{z}}]$. Where:

$$\hat{\mathbf{e}}_x = \hat{e}_{x_i}\hat{\mathbf{I}} + \hat{e}_{x_j}\hat{\mathbf{J}} + \hat{e}_{x_k}\hat{\mathbf{K}} \tag{2.11}$$

$$\hat{\mathbf{e}}_y = \hat{e}_{y_i}\hat{\mathbf{I}} + \hat{e}_{y_j}\hat{\mathbf{J}} + \hat{e}_{y_k}\hat{\mathbf{K}} \tag{2.12}$$

$$\hat{\mathbf{e}}_z = \hat{e}_{z_i}\hat{\mathbf{I}} + \hat{e}_{z_j}\hat{\mathbf{J}} + \hat{e}_{z_k}\hat{\mathbf{K}} \tag{2.13}$$

The transformation matrix $[\mathbf{C}]$ is thus:

$$[\mathbf{C}] = \begin{bmatrix} (\hat{\mathbf{e}}_\mathbf{x} \cdot \hat{\mathbf{I}}) & (\hat{\mathbf{e}}_\mathbf{x} \cdot \hat{\mathbf{J}}) & (\hat{\mathbf{e}}_\mathbf{x} \cdot \hat{\mathbf{K}}) \\ (\hat{\mathbf{e}}_\mathbf{y} \cdot \hat{\mathbf{I}}) & (\hat{\mathbf{e}}_\mathbf{y} \cdot \hat{\mathbf{I}}) & (\hat{\mathbf{e}}_\mathbf{y} \cdot \hat{\mathbf{I}}) \\ (\hat{\mathbf{e}}_\mathbf{z} \cdot \hat{\mathbf{I}}) & (\hat{\mathbf{e}}_\mathbf{z} \cdot \hat{\mathbf{I}}) & (\hat{\mathbf{e}}_\mathbf{z} \cdot \hat{\mathbf{I}}) \end{bmatrix} = \begin{bmatrix} \hat{e}_{x_i} & \hat{e}_{x_j} & \hat{e}_{x_k} \\ \hat{e}_{y_i} & \hat{e}_{y_j} & \hat{e}_{y_k} \\ \hat{e}_{z_i} & \hat{e}_{z_j} & \hat{e}_{z_k} \end{bmatrix} \tag{2.14}$$

And the transformation matrix derivative is:

$$[\dot{\mathbf{C}}] = \frac{d}{dt}[\mathbf{C}] \tag{2.15}$$

This calls for the time derivatives of the three basis vectors. We'll omit the superscript and subscript for the target satellite's state variables in the inertial frame for simplicity's sake, and define:

$$\mathbf{r} \equiv \mathbf{r}_1^I \tag{2.16}$$

$$\mathbf{v} = \dot{\mathbf{r}} \equiv \mathbf{v}_1^I \tag{2.17}$$

$$\mathbf{a} = \ddot{\mathbf{r}} = \mathbf{f}(t, \mathbf{r}, \dot{\mathbf{r}}) \tag{2.18}$$

$$\mathbf{h} = \mathbf{r}_1^I \times \mathbf{v}_1^I \tag{2.19}$$

where $\mathbf{r}$ denotes the target's position, $\mathbf{v}$ denotes its speed, $\mathbf{a}$ denotes its acceleration, and $\mathbf{h}$ denotes its particular angular momentum. All are described in the inertial reference frame with respect to the central body. (as shown in Figure 2.4).

The basis vector definitions for the LVLH frame are:

$$\hat{\mathbf{e}}_z = -\hat{\mathbf{r}} \tag{2.20}$$

$$\hat{\mathbf{e}}_y = -\hat{\mathbf{h}} \tag{2.21}$$

$$\hat{\mathbf{e}}_x = \hat{\mathbf{e}_z} \times \hat{\mathbf{e}}_y \tag{2.22}$$

The derivative of the $\hat{\mathbf{e}}_\mathbf{z}$ vector according to Equation 4 is:

$$\frac{d}{dt}(\hat{\mathbf{e}}_z) = \frac{d}{dt}(-\hat{\mathbf{r}}) \tag{2.23}$$

$$= -\frac{1}{r}[\mathbf{v} - (\hat{\mathbf{r}} \cdot \mathbf{v})\hat{\mathbf{r}}] \tag{2.24}$$

The derivative of the $\hat{\mathbf{e}}_\mathbf{y}$ vector, again using Equation 4, is:

$$\frac{d}{dt}(\hat{\mathbf{e}}_y) = \frac{d}{dt}(-\hat{\mathbf{h}}) \tag{2.25}$$

$$= -\frac{1}{h}[\dot{\mathbf{h}} - (\hat{\mathbf{h}} \cdot \dot{\mathbf{h}})\hat{\mathbf{h}}] \tag{2.26}$$

Where:

$$\dot{\mathbf{h}} = \frac{d}{dt}(\mathbf{r} \times \mathbf{v}) \tag{2.27}$$

$$= \left[\frac{d}{dt}(\mathbf{r}) \times \mathbf{v}\right] + \left[\mathbf{r} \times \frac{d}{dt}(\mathbf{v})\right] \tag{2.28}$$

$$= \mathbf{r} \times \mathbf{a} \tag{2.29}$$

Equation (2.26) yields Equation (2.27) for a force field with $\mathbf{r}||\mathbf{a}$ (such as two-body motion), where $\dot{\mathbf{h}} = 0$.

$$\frac{d}{dt}(\hat{\mathbf{e}}_y) = \mathbf{0} \tag{2.30}$$

The $\hat{\mathbf{e}}_\mathbf{x}$ vector's derivative is as follows:

$$\frac{d}{dt}(\hat{\mathbf{e}}_x) = \frac{d}{dt}(\hat{\mathbf{e}}_y \times \hat{\mathbf{e}}_z) \tag{2.31}$$

$$= \left[\frac{d}{dt}(\hat{\mathbf{e}}_y \times \hat{\mathbf{e}}_z\right] + \left[\hat{\mathbf{e}}_y \times \frac{d}{dt}(\hat{\mathbf{e}}_z)\right] \tag{2.32}$$

Which contains already-derived quantities. Equation (2.32) is reduced to the following for two-body motion:

$$\frac{d}{dt}(\hat{\mathbf{e}}_x) = -\hat{\mathbf{h}} \times \frac{d}{dt}(\hat{\mathbf{e}}_z) \tag{2.33}$$

2.2.4 Transformation from LVLH to Body-Fixed Frame

It is crucial to note that while there are many other combinations of Euler's angles ϕ, θ, ψ that can be used to create rotation matrices and move between different frames, There is a distinct specification of the orientation between the two frames. Because rotations do not commute, the sequence of rotations is crucial. The resulting matrix rotation is then given by:

$$L^B_{LVLH} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} -\sin\theta & 0 & \cos\theta \\ 0 & 1 & 0 \\ \cos\theta & 0 & \sin\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.34}$$

Consequently, the reference transformation's expression will be as follows:

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} -\sin\theta & 0 & \cos\theta \\ 0 & 1 & 0 \\ \cos\theta & 0 & \sin\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{LVLH} \\ y_{LVLH} \\ z_{LVLH} \end{bmatrix} \tag{2.35}$$

2.3 Relative Motion Dynamics

In a typical rendezvous mission, the movement of the chaser satellite in relation to a target satellite in an elliptical or circular orbit can be explained by a series of autonomous nonlinear differential equations. We provide continuous-time and discrete-time models for the rendezvous operation of satellites in this section after briefly discussing the governing equations.

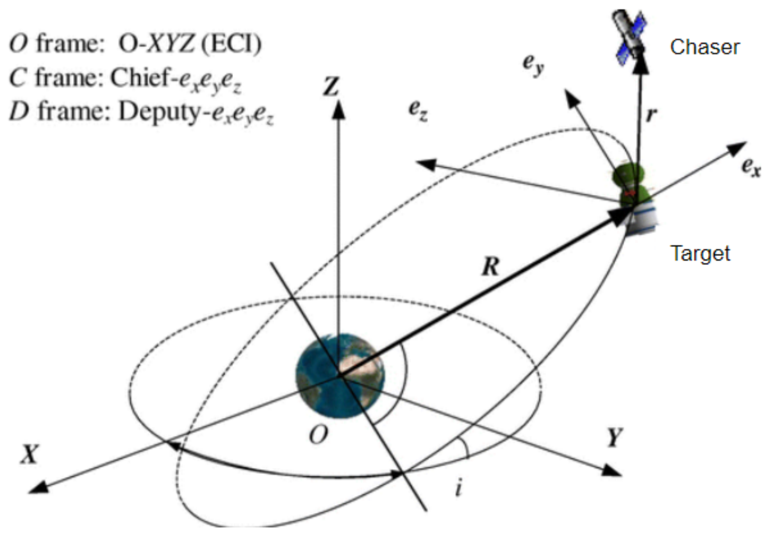

FIGURE 2.5: ECI and LVLH Coordinate system for satellite rendezvous

Suppose the target satellite is orbiting in an elliptical pattern. (the initial Kepler elements are known). As shown in Fig. 2.5, the origin of the LVLH coordinate system e$_x$-e$_y$-e$_z$ is fixed at the target spacecraft's center of mass, and the e$_y$ axis is perpendicular to the orbital plane e$_x$-e$_z$. The chaser satellite's relative motion in the LVLH frame can be described by the nonlinear equation shown below:

$$\frac{d^2\mathbf{r}}{dt^2} = -\mu\left(\frac{\mathbf{R}+\mathbf{r}}{|\mathbf{R}+\mathbf{r}|^3} - \frac{\mathbf{R}}{|\mathbf{R}|^3}\right) + \mathbf{u} \tag{2.36}$$

Where u is the control input (acceleration resulting from the chaser satellite's thrust forces), μ is the gravity constant, R is the relative position vector between the target satellite and the planet's center of gravity, and the target satellite's vector to the chaser satellite is shown by the symbol r.

2.3.1 Continous-time nonlinear model

Eq. (2.36) can be represented as the following while using the symbol $\mathbf{r_c} = [x_c \; y_c \; z_c]^T$:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 2\omega\dot{z} + \dot{\omega}z + \omega^2 x - \frac{\mu x}{|\mathbf{R}+\mathbf{r}|^3} \\ -\frac{\mu y}{|\mathbf{R}+\mathbf{r}|^3} \\ \omega^2 z - 2\omega\dot{x} - \dot{\omega}x - \mu\left(\frac{z-R}{|\mathbf{R}+\mathbf{r}|^3} + \frac{1}{R^2}\right) \end{bmatrix} + \mathbf{u} \tag{2.37}$$

where ω is the orbital velocity of the rotating coordinate system, $r := |r|$, $R := |R|$, $|R+r|^2 := x^2 + y^2 + (z-R)^2$. Let h be the orbital angular momentum of the target satellite. Then, $R^2\omega = h =$ constant. Let v the true anomaly, e $\in$ [0, 1) be the eccentricity of the target orbit, $\rho = 1 + e\cos v$, and

$$k = \mu / h^{\frac{3}{2}} = \text{constant}. \tag{2.38}$$

The orbital rate ω satisfies

$$\omega = h/R^2 = k^2\rho^2 \tag{2.39}$$

The true anomaly v and the eccentric anomaly E fulfill the following equations:

$$\sin(E) = \frac{\sqrt{1-e^2}\sin(\nu)}{1+e\cos(\nu)}, \; \cos(E) = \frac{e+\cos(\nu)}{1+e\cos(\nu)} \tag{2.40}$$

Additionally, the recognized Kepler's equation is satisfied by E and time t:

$$t = \frac{T_o}{2\pi}(E - e\sin(E)) \tag{2.41}$$

where T_o denotes the orbit's time frame. The nonlinear dynamics from (2.37) can be recast in the following state space form:

$$\dot{\mathbf{x}}_c = \mathbf{f}(\mathbf{x}_c, \mathbf{u}) \tag{2.42}$$

in which $x_c = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$. The positions and velocities of the chaser satellite in relation to the target satellite in the LVLH frame are represented by the vectors $[x\ y\ z]^T$ and $[\dot{x}\ \dot{y}\ \dot{z}]^T$, respectively.

2.3.2 Discrete-time nonlinear model

The continuous-time nonlinear dynamical system provided by Eq. (7) is discretized into a discrete-time nonlinear dynamical system using the classical fourth-order Runga Kutta discretization technique as follows:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \frac{T}{6}(\mathbf{k}_{|1} + 2\mathbf{k}_{|2} + 2\mathbf{k}_{|3} + \mathbf{k}_{|4}) \tag{2.43}$$

where $k \in [0, N_c - 1]_d$, $t_k = \frac{t_f}{N_c}k = Tk$, t_f is the final time, $T > 0$ is the sampling period, $k|1$, $k|2$, $k|3$, and $k|4$ are given as follows:

$$\mathbf{k}_{|1} = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \tag{2.44}$$

$$\mathbf{k}_{|2} = \mathbf{f}\left(\mathbf{x}(k) + \frac{T}{2}\mathbf{k}_{|1}, \mathbf{u}(k)\right) \tag{2.45}$$

$$\mathbf{k}_{|3} = \mathbf{f}\left(\mathbf{x}(k) + \frac{T}{2}\mathbf{k}_{|2}, \mathbf{u}(k)\right) \tag{2.46}$$

$$\mathbf{k}_{|4} = \mathbf{f}\left(\mathbf{x}(k) + T\mathbf{k}_{|3}, \mathbf{u}(k)\right) \tag{2.47}$$

The discrete-time system's state x and the continuous-time system's state x_c are connected as follows: $x_c(t_k) \approx x(k)$. It is possible to represent the discrete nonlinear satellite rendezvous from Eq. (2.43) in the following short form:

$$\mathbf{x}(k+1) = \mathbf{h}(\mathbf{x}(k), \mathbf{u}(k)) \tag{2.48}$$

2.3.3 Clohessy-Wilshire (CW) Equations

The linearized equations of motion of two satellites in close orbits are explained by the Clohessy-Wiltshire equations. The CW equations are true when namely that the orbit of the target satellite is nearly circular and the distance between the chaser satellite and the target satellite is very large compared to the target satellite's distance from the center of the Earth. The three straightforward differential equations that make up the CW equations can be resolved analytically. In actual use, these solutions are frequently utilized for rendezvous in a circular orbit.

The CW equations of motion (applying the assumptions to equation (2.37)) can be used to explain how a chaser satellite moves with respect to a target satellite that is in a circular trajectory around an imagined point mass.

$$\ddot{x} = 3\omega^2 x + 2\omega\dot{y} \tag{2.49}$$

$$\ddot{y} = -2\omega\dot{x} \tag{2.50}$$

$$\ddot{z} = -\omega^2 z \tag{2.51}$$

Where ω is the orbital rate $\left(\omega = \sqrt{\frac{\mu}{a_{tar}^3}}\right)$ and a_{tar} is the semi-major axis of the target orbit

The HCW equations are provided in the state-space representation below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3\omega^2 & 0 & 0 & 0 & 2\omega & 0 \\ 0 & 0 & 0 & -2\omega & 0 & 0 \\ 0 & 0 & -\omega^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.52}$$

2.3.4 Modified CW Equations

In **Section 2.3.3**, Clohessy-Wiltshire equations do not include J_2 perturbations. The following CW equations include the effects of J_2 perturbations, which is being referred to as the modified CW equations:

$$\ddot{x} - 2\omega c\dot{y} - (5c^2 - 2)\omega^2 x = 0 \tag{2.53}$$

$$\ddot{y} + 2\omega c x = 0 \tag{2.54}$$

$$\ddot{z} + (3c^2 - 2)\omega^2 z = 0 \tag{2.55}$$

Where, J_2 = Second harmonic due to Earth oblateness effect; $c = \sqrt{1+s}$; $s = \frac{3J_2Re^2}{8r_{ref}^2}(1+3\cos 2i)$; i = inclination angle, r_{ref} = radius of reference orbit.

The equations in the state-space representation are given by:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ (5c-2)\omega^2 & 0 & 0 & 0 & 2\omega c & 0\\ 0 & 0 & 0 & -2\omega c & 0 & 0\\ 0 & 0 & -(3c-2)\omega^2 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ \dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} \tag{2.56}$$

2.4 Koopman Operator

It has always been appealing to consider linearizing nonlinear systems without approximation. But during the last decade, there has been a growing interest in the Koopman operator architecture. In that work, it was shown that the eigenfunctions of the Koopman operator reveal important global topological properties of the underlying nonlinear system, such as ergodic and periodic partitions. Those findings largely concentrated on conservative measure-preserving systems that induce a unitary operator in l^2 space. In nonconservative systems, where the Koopman operator is not unitary, they were further expanded. This allowed for nonlinear control theory applications and suggested that the Koopman technique is especially well suited to offer a novel method for non-conservative systems.

2.4.1 Brief overview

Bernard O. Koopman showed in 1931 that an infinite-dimensional operator of linear nature on a Hilbert space of observable functions of the system's state can be used to depict a nonlinear dynamical system. This operator, called the Koopman operator, is linear, and the behavior of a nonlinear system is completely characterized by its spectrum decomposition. The Koopman operator K is an infinite dimensional linear operator that works on a group of observable functions $s = [s1, s2, ..., s_{N_k}]^T$ where $s_i : R^n \to R$. These functions evolve linearly over time. The Koopman operator $K : F \to F$ is defined as follows:

$$(K\mathbf{s})\mathbf{x}(k) = \mathbf{s}(\mathbf{f}(\mathbf{x}(k))) = \mathbf{s}(\mathbf{x}(k+1)) \tag{2.57}$$

where F is a set of invariant functions that the Koopman operator can act on (commonly referred to as observables). In contrast to dynamics that are linearized around a particular linearization point and become inaccurate away from this point, the Koopman operator explains the evolution of the observables of a nonlinear system with complete accuracy across the state space. It is possible to formulate the observable function s(x) as:

$$\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), s_2(\mathbf{x}), ...s_{N_k}(\mathbf{x})]^T \tag{2.58}$$

where $s_j(x) : R^n \to R$, for all $j \in \{1, \;.\;\;.\;\;.\;\;, P_k\}$, $P_k >> n$. The state l, which is frequently referred to as the lifted state can be expressed in the following form:

$$\mathbf{l}(k) = \mathbf{s}(\mathbf{x}(k)) \tag{2.59}$$

2.4.2 Lifted Dynamics for Rendezvous Operation

The basic steps for approximating the nonlinear rendezvous equation with a higher dimensional linear state space model utilizing the Koopman operator are presented in this section. Consider the discrete-time nonlinear rendezvous equation shown in Eq (2.48) under **Section 2.3.2**. By using the following linear lifted state space model, we aim to approximate Eq. (2.48):

$$\mathbf{l}(k+1) = G_{koop}\mathbf{l}(k) + H_{koop}\mathbf{u}(k) \tag{2.60}$$

where P_k is the dimension of the lifted state $l(k)$, $G_{koop} \in R^{P_k \times P_k}$, $H_{koop} \in R^{P_k \times m}$, $l(k) \in R^{P_k}$, $u(k) \in R^m$ and $k \in [0, N_c - 1]_d$. The initial condition l_0 is given by

$$\mathbf{l}_0 = \mathbf{s}(\mathbf{x}_0) = [s_1(\mathbf{x}_0), s_2(\mathbf{x}_0), ...s_{P_k}(\mathbf{x}_0)]^T \tag{2.61}$$

where the initial condition $x_0 = x(0)$ for the initial discrete nonlinear model in Eq (2.48). The following expression represents the final state of the lifted dynamics indicated in the above equation:

$$l(N) = G_{koop}^N l_0 + \sum_{\tau=0}^{N-1} G_{koop}^{N-\tau-1} H_{koop}\mathbf{u}(\tau) \tag{2.62}$$

where $x(N)$ represents the final state of the initial discrete-time nonlinear equation and $l(N) = s(x(N))$. The terminal state can be compactly rewritten as follows:

$$l(N) = R_{N_{koop}}\mathbf{u}_{koop} + Q_{koop} \tag{2.63}$$

where $u_{koop} \in R^{N_m}$, $R_{N_{koop}} \in R^{P_k \times N_m}$, and $Q_{koop} \in R^{P_k}$ are defined as:

$$\mathbf{u}_{koop} := [\mathbf{u}(0)^T, \mathbf{u}(1)^T, ...\mathbf{u}(N-1)^T]^T \tag{2.64}$$

$$\mathbf{R}_{N_{koop}} := [G_{koop}^{N-1} H_{koop}, ..., H_{koop}] \tag{2.65}$$

$$\mathbf{Q}_{koop} := G_{koop}^{N} \mathbf{l}_0 \tag{2.66}$$

2.4.3 Data Driven Approach for computing G_{koop} and H_{koop}

The Koopman operator has been approximated in finite dimensions in the past, although with mixed results. Extended Dynamic mode decomposition uses the best-fit linear model to advance the spatial data in lifted dimensions from one time to the next in an effort to approximate the Koopman operator.

The matrices G_{koop} and H_{koop} from the preceding section are now approximated using the Extended DMD method with Control. In order to do this, a group of arbitrary control inputs and random initial states x_0 are selected, with the entries coming from a uniform distribution $\left[-\frac{1}{2}, \frac{1}{2}\right]$. These arbitrary control inputs are successively applied to Eq. (2.48). Let the control input u(k) be applied to take the state of the chaser satellite from $x(k)$ to $x(k+1)$. In this manner, the matrices M, U, and N are built with the states $M = [x(0), ..., x(e)]$ and their associated control inputs stored as $U = [u(0), ..., u(e)]$; and let N be equal $[x(1), ..., x(e+1)]$ where $(e+1)$ is the total number of data points. The matrix N can be expressed as:

$$\mathbf{N} = \mathbf{f}(\mathbf{M}, \mathbf{U}) \tag{2.67}$$

The matrices G$_{koop}$ and H$_{koop}$ are obtained by solving the following least squares optimization problem; with the known variables M, N, and U:

$$\min_{G_{koop}, H_{koop}} ||\mathbf{N}_{lift} - \mathbf{G}_{koop}\mathbf{M}_{lift} - \mathbf{H}_{koop}\mathbf{U}||_F \tag{2.68}$$

where,

$$\mathbf{M}_{lift} = [\mathbf{s}(x(0)), ..., \mathbf{s}(x(e))] \tag{2.69}$$

$$\mathbf{N}_{lift} = [\mathbf{s}(x(1)), ...\mathbf{s}(x(e+1))] \tag{2.70}$$

with

$$\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), ...s_{N_k}(\mathbf{x})]^T \tag{2.71}$$

being a set of nonlinear observable functions given in the form $s_j(\mathbf{x})$ for every $j \in \{1,..., N_k\}$. The following provides the analytical answer to the above optimization problem:

$$[G_{koop}, H_{koop}] = \mathbf{N}_{lift}[\mathbf{M}_{lift}, \mathbf{U}]^{\dagger} \tag{2.72}$$

2.5 Attitude Equations of Motion

2.5.1 Attitude Kinematics Equations

Quaternions provide the foundation for the explanation of spacecraft kinematics. To avoid singularities and reduce the computing cost of time integration, it is very convenient to compute angular rotations using quaternions rather than Euler's angles. A 4 × 1 matrix is a typical way to define quaternions:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.73}$$

where $[q_1\ q_2\ q_3]^T$ and $[q_0]$ are vectorial and scalar parts, respectively. The unit quaternion has the following requirements, which is a property of quaternions:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{2.74}$$

Moreover, it can also be expressed as $R_T R = 1$, which denotes a rotation. The spacecraft's kinematics can thus be described as

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.75}$$

The aforementioned equation can be compactly rewritten as,

$$\dot{q} = \frac{1}{2}\Omega(\omega)q \tag{2.76}$$

Quaternions can be used to define a rotation matrix between the LVLH frame and the body-fixed frame, as was demonstrated with Euler's angles.

$$L^B_{LVLH} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 + q_2q_0) \\ 2(q_1q_2 + q_3q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_1q_0) \\ 2(q_1q_3 - q_2q_0) & 2(q_2q_3 + q_1q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.77}$$

Quaternions are not visually intuitive, so in order to better grasp the rotation, they are typically transformed back to Euler's angles at the end.

$$\phi = \arctan\left(\frac{2(q_0q_1 + q_2q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right) \tag{2.78}$$

$$\theta = \arcsin(-2(q_1q_3 + q_0q_2)) \tag{2.79}$$

$$\psi = \arctan\left(\frac{2(q_1q_2 + q_0q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right) \tag{2.80}$$

2.5.2 Attitude Dynamics Equations

The rigid body equation of motion, which defines attitude dynamics, is given by,

$$\dot{h}_{tot} = N_e - \omega \times h_{tot} \tag{2.81}$$

The total angular momentum of the satellite's N spinning sections is represented by the quantity $htot = \sum_{j=1}^{N}(J_j\omega j)$. The previous equation is also referred to as Euler's equation. ω is the satellite's angular velocity, and N_e is the total amount of external torques acting on it. Every quantity is expressed in terms of the body reference frame. The inertia matrix of the satellite, J_s, is typically a 3×3 matrix and is represented by the following formula:

$$J_s = \begin{bmatrix} J_x & -J_{xy} & -J_{xz} \\ J_{yx} & J_y & -J_{yz} \\ -J_{zx} & -J_{zy} & J_z \end{bmatrix} \tag{2.82}$$

J_{xy}, J_{xz}, and J_{yz} are the inertia's products, and J_x, J_y, and J_z are its moments in the symmetric matrix above. Since it is believed to be a straightforward situation where the body frame axes coincide with the major axis of inertia, it can be condensed as follows.

$$J_s = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{2.83}$$

Consequently, Euler's equation can be expanded and rewritten as

$$\dot{\omega} = I_s^{-1} N_e - I_s^{-1}(\omega \times I_s \omega) \tag{2.84}$$

It is typical to recast the equation with a skew matrix, S(), in place of the cross-product,

$$S(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.85}$$

Euler's equation can be rearranged and written as

$$\dot{\omega} = I_s^{-1} N_e - I_s^{-1}(S(\omega) I_s \omega) \tag{2.86}$$

Chapter 3

Guidance Algorithm Design

The design of the control system and guidance algorithm for the rendezvous operations of satellites is covered in this section. Two Control schemes have been proposed here:

1) Use the CW equations to describe the linearized rendezvous equations of two satellites in proximity orbits and then use an LQR Controller with the objective of minimizing fuel consumption.

2) Use Koopman Based Approach to lift the nonlinear dynamics into a higher-dimensional space over which it exhibits linear-system behavior and then uses an LQR Controller with the objective of minimizing fuel consumption.

Apart from this, an Attitude controller has been designed using an LQR-based control algorithm for matching the chaser satellite's orientation with that of the target satellite. The final part of this section covers the guidance algorithm proposed for Satellite Docking for which the PN-based impact guidance algorithm has been designed.

3.1 LQR Control

In general, a control law's goal is to direct the dynamic system toward the desired end state by regulating the input signal. A spacecraft's orientation can be adjusted to shield it from external disturbances or transported from an initial condition to a final state along a planned trajectory with the aid of a control function. The linear quadratic regulator (LQR) control is an algorithm that is well-known. A non-linear system must first be linearized since this control may only be used for linear systems. Consider a linear, continuous-time, and controllable system and the performance criteria as described in the equations given below:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{3.1}$$

$$\mathbf{J} = \int_0^\infty [\mathbf{u}^T R\mathbf{u} + \mathbf{x}^T Q\mathbf{x}]dt \tag{3.2}$$

Where, in the cost function, Q and R, which represent the weighting matrices, should, respectively, be positive-semi-definite and positive definite. The aim of the LQR control problem is to determine a function that minimizes the cost function and penalizes the states x and the control effort u, by using the controller design parameters Q and R respectively. The LQR controller has the following form:

$$u(t) = -Kx(t) \tag{3.3}$$

$$K = -R^{-1}B^T P \tag{3.4}$$

Where P is determined by the following equation's positive (symmetric) semi-definite solution, also known as the Ricatti equation.

$$0 = PA + A^T P - Q + PBR^-1B^T P \tag{3.5}$$

If the pair (A, B) can be controlled and (Q, A) can be detected, the problem can be solved. MATLAB can be used to solve the Ricatti equation and calculate K. The Q and R matrices are initially selected. To produce a decent response, R is then kept constant while Q is changed. In this study, an LQR control has been designed to move a spacecraft along a predetermined trajectory from its initial position and velocity to its destination.

3.1.1 Linear CW with LQR

The chaser's movement in relation to the target is described using the Hill equations. The LQR control approach can be employed with no problems because they are linear. Using control inputs, the state space equation (from section 2.3.4) is given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ (5c-2)\omega^2 & 0 & 0 & 0 & 2\omega c & 0 \\ 0 & 0 & 0 & -2\omega c & 0 & 0 \\ 0 & 0 & -(3c-2)\omega^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{3.6}$$

We may determine the values of A and B and then design an LQR-based control system for position control by comparing the aforementioned equation with the form $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$. The separation between the target satellite and the chaser satellite will be impacted by J_2 disturbances. In order to keep $[x\ y\ z]$ constant, three control inputs are introduced.

3.1.2 Koopman with LQR Control

In this instance, the chaser satellite's movement wrt the target satellite is defined by the nonlinear relative motions of equations.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 2\omega\dot{z} + \dot{\omega}z + \omega^2 x - \frac{\mu x}{|\mathbf{R}+\mathbf{r}|^3} \\ -\frac{\mu y}{|\mathbf{R}+\mathbf{r}|^3} \\ \omega^2 z - 2\omega\dot{x} - \dot{\omega}x - \mu\left(\frac{z-R}{|\mathbf{R}+\mathbf{r}|^3} + \frac{1}{R^2}\right) \end{bmatrix} + \mathbf{u} \tag{3.7}$$

By increasing the state space of the earlier system to a higher dimensional across which it accepts a linear state space model representation but is inherently infinite-dimensional, one can use the Koopman operator to transform a finite-dimensional nonlinear system into a linear system.

$$\mathbf{l}(k+1) = G_{koop}\mathbf{l}(k) + H_{koop}\mathbf{u}(k) \tag{3.8}$$

Using the Extended DMD + Control method, an approximate solution for G_{koop} and H_{koop} matrices are found (refer to section 2.4.3). The new state space system in the lifted dimensions is linear and thus, LQR control can now be used without any discrepancies. Note that here, the discrete form of LQR will have to be used to generate the required control sequences $u(k)$.

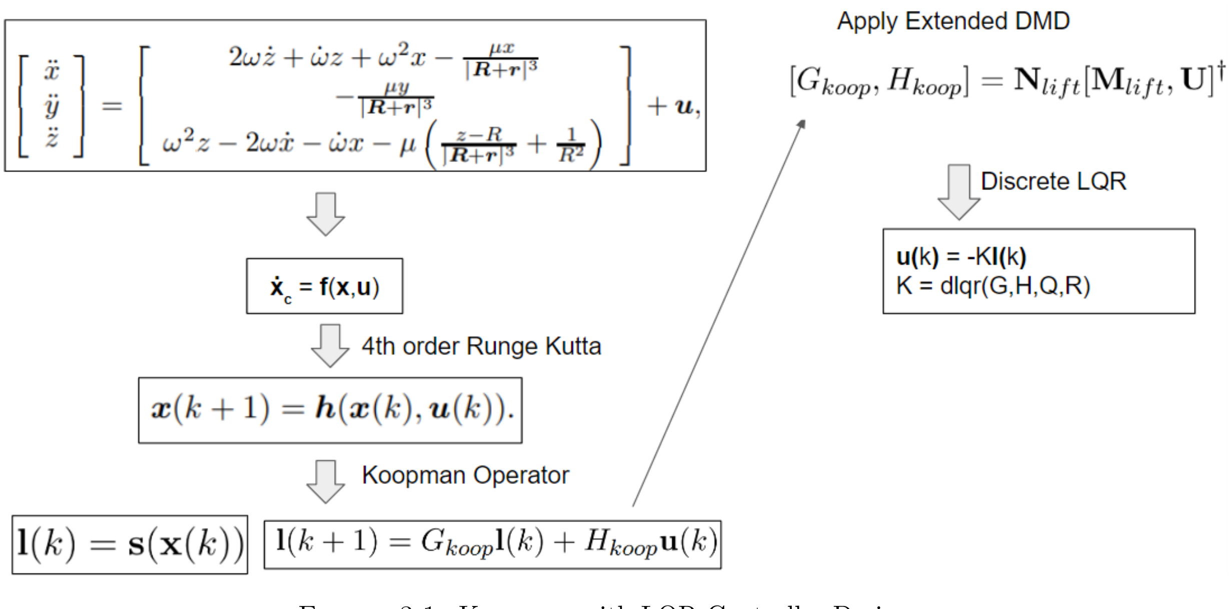
$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 2\omega\dot{z} + \dot{\omega}z + \omega^2 x - \frac{\mu x}{|\boldsymbol{R}+\boldsymbol{r}|^3} \\ -\frac{\mu y}{|\boldsymbol{R}+\boldsymbol{r}|^3} \\ \omega^2 z - 2\omega\dot{x} - \dot{\omega}x - \mu\left(\frac{z-R}{|\boldsymbol{R}+\boldsymbol{r}|^3} + \frac{1}{R^2}\right) \end{bmatrix} + \boldsymbol{u},$
$\dot{\mathbf{x}}_c = \mathbf{f}(\mathbf{x},\mathbf{u})$
4th order Runge Kutta
$\boldsymbol{x}(k+1) = \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k)).$
Koopman Operator
$\mathbf{l}(k) = \mathbf{s}(\mathbf{x}(k))$
$\mathbf{l}(k+1) = G_{koop}\mathbf{l}(k) + H_{koop}\mathbf{u}(k)$
Apply Extended DMD
$[G_{koop}, H_{koop}] = \mathbf{N}_{lift}[\mathbf{M}_{lift}, \mathbf{U}]^\dagger$
Discrete LQR
u(k) = -Kl(k)
K = dlqr(G,H,Q,R)

FIGURE 3.1: Koopman with LQR Controller Design

3.1.3 Attitude Controller

The rotation and attitude variations of a spacecraft are described by the rigid body equations of Euler. Since they are a collection of nonlinear equations, the system must be linearized before an LQR control can be applied. Attitude control aims to guide the state vector to zero with respect to the intended ultimate state. The state-space equation for the position control is

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{3.9}$$

The state vector here is given below:

$$\mathbf{x} = \{q_1\ q_2\ q_3\ \omega_1\ \omega_2\ \omega_3\}^T \tag{3.10}$$

Only the vectorial components of the state vector, not the scalar portion, are taken into account. This can be explained by remembering that the scalar component is linearly dependent on the other components once the vectorial component is known. Instead, torques calculated by the LQR, $\mathbf{u} = -K\mathbf{x}$, are what determine the control action.

$$u = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3.11}$$

Following the linearization of Euler's equations, The input matrix, B, and the state matrix, A, can be shown as

$$A = \begin{bmatrix} 0_3 & 0_3 \\ 0.5I_3 & 0_3 \end{bmatrix} \tag{3.12}$$

$$B = \begin{bmatrix} I_s^{-1} & 0_3 \end{bmatrix} \tag{3.13}$$

3.2 Docking Plan

Once the chaser satellite is in close proximity to the target satellite, more emphasis on the accuracy of the position and velocity control is required in order to dock safely. The classical rendezvous approach makes the path between the arrival and the departure unconstrained. However, during the approach, physical contact of the chaser spacecraft with the target satellite is involved, so the ability to maneuver without collision is a fundamental requirement. In some scenarios, the large structure accessories, such as the battery panel or antenna, can be seen as obstacles to maneuvering. A possible movement in a complex environment is one that can make sure the trajectory wouldn't contact the obstacles. Thus, one of the biggest problems is to

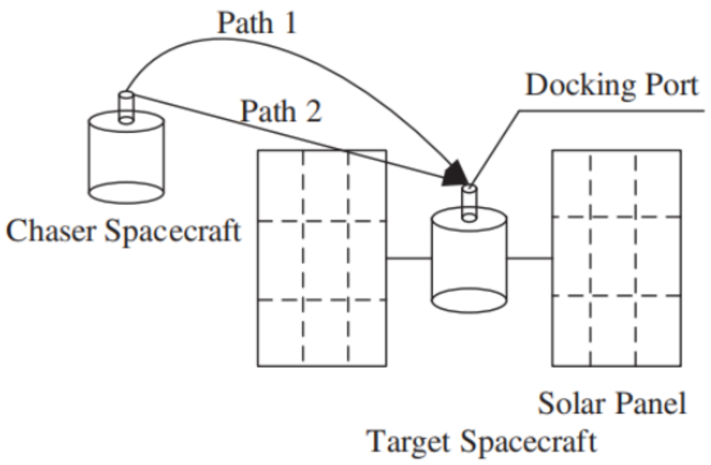

FIGURE 3.2: Docking Path. Path 1 is chosen over Path 2 in order to avoid collision with the solar panel and safely dock with the target satellite

establish a control algorithm that can guide the chaser spacecraft approaching the docking port of the target satellite, without colliding.

In a practical situation, the chaser satellite isn't aware of the position of the docking port. So it is first made to fly by the target satellite. In other words, the chaser satellite is made to move in a circular orbit around the target satellite (when observed from the target satellite or LVLH frame) in order for the sensors to locate the position and orientation of the docking port. This can be easily demonstrated by the figure given below:

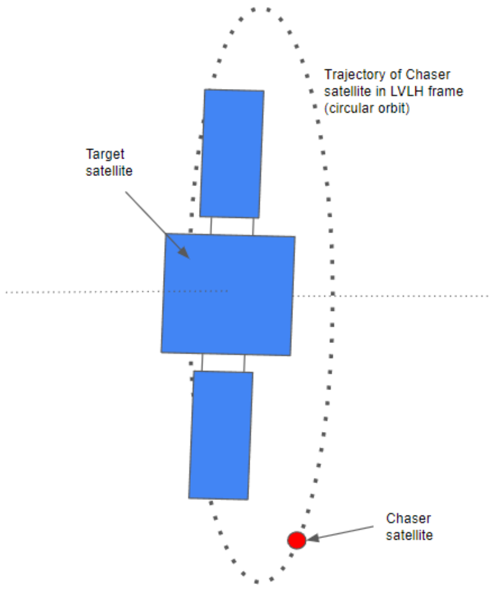
Trajectory of Chaser satellite in LVLH frame (circular orbit)
Target satellite
Chaser satellite

FIGURE 3.3: Fly By maneuver

The docking problem is first broken down into various cases. The docking port vector is normal to the area from the center of the docking port and projecting outwards. The main objective in all the cases is to align the velocity vector with the docking port vector (but opposite to that of the same) along with reducing the relative position and velocity to 0. The following figure represents all the cases.

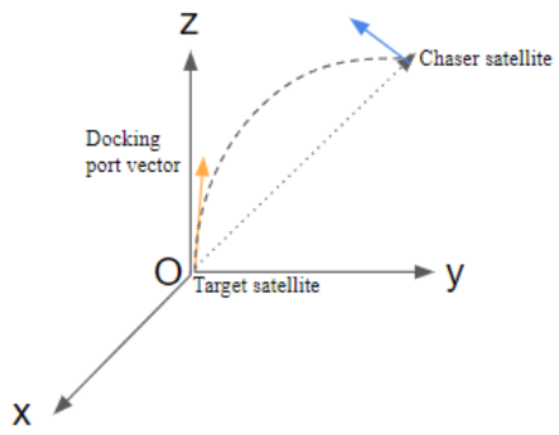

FIGURE 3.4: Case 1: When the docking port vector is not rotating

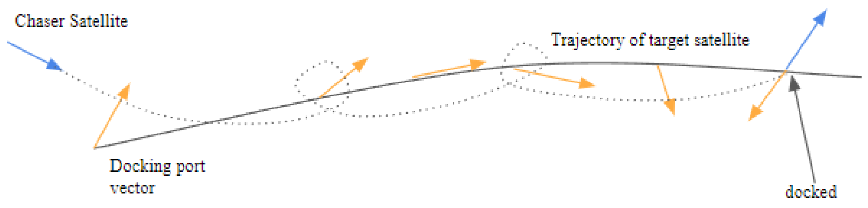

FIGURE 3.5: Case 2: When the docking port vector is rotating with a constant angular velocity about some axis as in the case of a tumbling satellite

3.2.1 PN Guidance Algorithm

As a major class of traditional guidance, proportional navigation (PN) has been very popular both in theoretical research and in practical applications. Proportional navigation is most widely used in short-range guidance. It probably had its origins among the mariners who realized that a collision was ensured if two constant velocity vessels maintained a constant relative bearing while closing in range. The first application of proportional navigation in modern air-to-air and surface-to-air missile systems can be dated back to the 1940s. Since then, proportional navigation has been the most commonly used as an empirical guidance law, due to its simplicity, effectiveness, and ease of implementation.

Consider Case 1 i.e. the simplest case when the docking port vector is not rotating. Further, for the time being, assume that the velocity vector and the docking port vector lie in the same plane i.e. Y-Z plane. Since we're considering the equations in the LVLH frame, the docking port velocity is 0. Refer to the following figure:

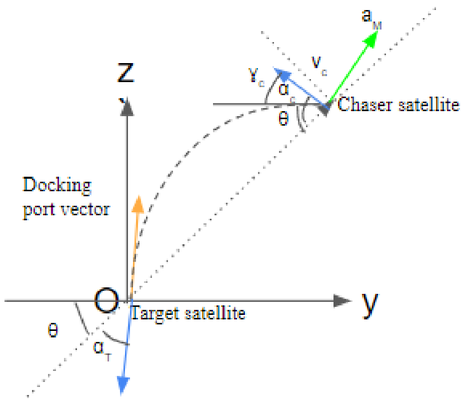

FIGURE 3.6: Motion model of Case 1 (in LVLH frame)

From the above figure, the relative kinematic equation can be given as,

$$\dot{\theta} = -\frac{V_c \sin \alpha_c}{R} \tag{3.14}$$

$$\dot{R} = -V_c \cos \alpha_c \tag{3.15}$$

$$\dot{\gamma_c} = -\frac{a_M}{V_c} \tag{3.16}$$

$$\alpha_c = \gamma_c + \theta \tag{3.17}$$

where R is the distance between the LOS of the two satellites, V$_c$ is the velocity vector of the chaser satellite, γ_c is the flight path angle, θ is the Line of Sight angle and a$_M$ represents the acceleration command perpendicular to the LOS. When Proportional-Guidance algorithm is adopted, the acceleration is shown as

$$a_M = N V_c \dot{\theta} \tag{3.18}$$

where, N is the navigation gain (constant number), $\dot{\theta}$ is the rate of the Line of Sight (LOS) angle.

Also, by geometry and using the CW equations (2.53) - (2.55),

$$a_M = \ddot{y} \sin (\alpha_c - \theta) + \ddot{z} \cos (\alpha_c - \theta) \tag{3.19}$$

$$V_c = \begin{bmatrix} 0 \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{3.20}$$

Chapter 4

Simulation Results

It is expected that the target satellite would travel in an elliptical orbit of eccentricity $e = 0.4621$, inclination $i = 0^o$, and semimajor axis, $a = 19435600$ m. At $t = 0s$, it is known that the initial position and velocity of the chaser spacecraft in the LVLH frame is $[500km\ 50km\ 20km]$ and $[2.727m/s\ -600m/s\ 0m/s]$ respectively. Also, the initial orientation (Euler angles θ, ϕ, ψ) and angular velocity of the chaser satellite are $[20^o\ 20^o\ 20^o]$ and $[1rad/s\ 1rad/s\ 1rad/s]$ respectively. For external disturbances, J_2 perturbations have been accounted for with the Second harmonic due to Earth's oblateness effect, $J_2 = 1082.64 \times 10^{-6}$. The radius of reference circular orbit, $r_{ref} = 7000$ km. The simulation is run for $t_f = \frac{T_c}{4}$, where T_c is the target orbit's rotational period. In the data-driven method, to generate the sequence of data $x(k)$; $k \in \{0....,e\}$, we sample 850 initial conditions which are taken from the uniform distribution over $[0.5, 0.5]^6$. For each sample, we apply control inputs $u(k)$ which are taken randomly from a uniform distribution over $[0.3, 0.3]^3$. The following observable functions were utilized in the simulations:

$$[s_1\ s_2\ s_3\ s_4\ s_5\ s_6] = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}] \tag{4.1}$$

$$[s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}] = \frac{[1, \dot{x}, \dot{y}, \dot{z}, x, y, z]}{(1 + x^2 + y^2 + z^2)^{\frac{3}{2}}} \tag{4.2}$$

$$[s_{14}, s_{15}, s_{16}] = \frac{[x^2\dot{x}, y^2\dot{y}, z(z - ||\mathbf{x}_0||_2)\dot{z}]}{[x^2 + y^2 + (z - ||\mathbf{x}_0||_2)^2]^{\frac{3}{2}}} \tag{4.3}$$

$$[s_{17}, s_{18}, s_{19}] = \frac{[x, y, z]}{[x^2 + y^2 + (z - ||\mathbf{x}_0||_2)^2]^{\frac{3}{2}}} \tag{4.4}$$

The complete simulation environment has been developed using MATLAB Software.

Note :

Here I've provided the link to the Simulation Video

4.0.1 Relative Translational Motion

A schematic representation of the simulation can be seen in the block diagram. The Hill equations characterise the system and define the chaser's motion in relation to the target. Therefore, this block, which serves as the system's foundation, contains the dynamic equations of motion. The guidance block's acceleration or force vector serves as the plant's input, and the output is each instant's real position and velocity of the chaser. The LQR guidance, which uses the plant's actual position and velocity as inputs, calculates the control accelerations.

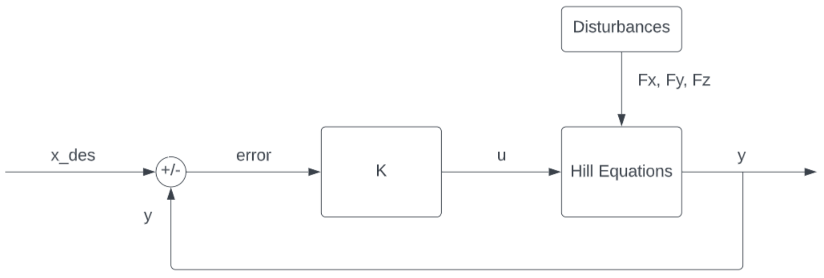

FIGURE 4.1: LQR Position Control Block Diagram

Figures 4.2 and 4.3 represent the chaser satellite's position and velocity vs time plots in the LVLH frame for the two controllers. The control inputs for the 2 controllers are computed by using the linearized and the lifted dynamics respectively as shown in Figures 4.4 and 4.5.

It is observed that both controllers worked well for near-field rendezvous operations. However, Linear CW with an LQR control scheme failed to perform in far-field rendezvous operations. However, Koopman with an LQR control scheme doesn't involve any kind of linearization and is thus, able to guide the chaser satellite to the ideally achieved states. However, one thing to be pointed out is that the control effort was found to be more in the case of the Koopman scheme as compared to Linear CW with LQR.

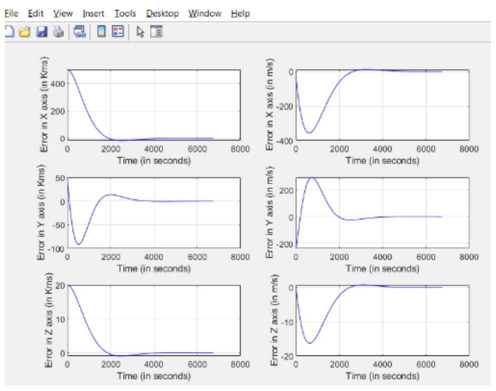

FIGURE 4.2: Chaser Satellite's position and velocity graphs in LVLH frame for the case of Linear CW with LQR

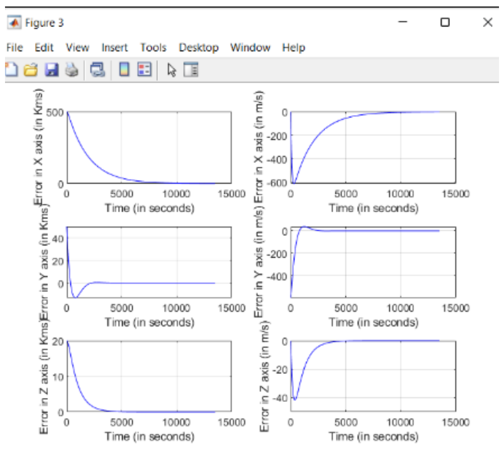

FIGURE 4.3: Chaser Satellite's position and velocity graphs in LVLH frame for the case of Koopman with LQR

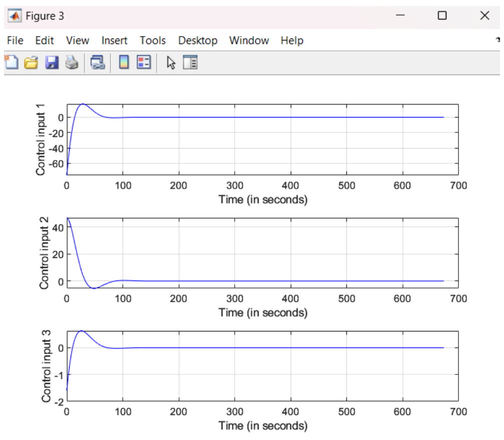

FIGURE 4.4: Control Thrusts for the case of Linear CW with LQR

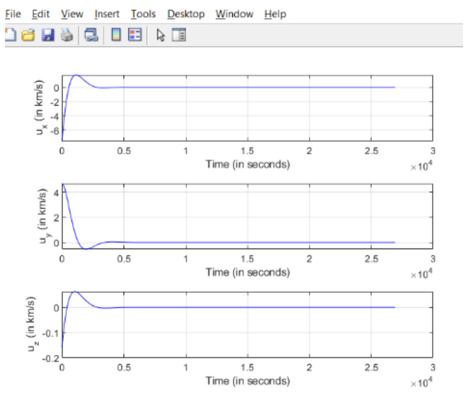

FIGURE 4.5: Control Thrusts for the case of Koopman with LQR

4.0.2 Attitude Control

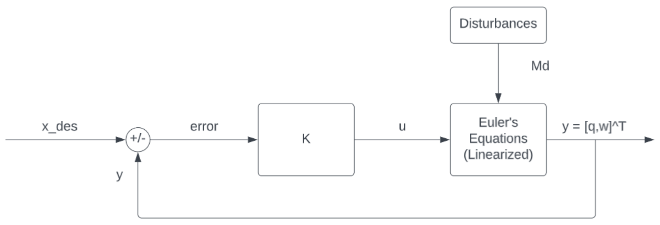

FIGURE 4.6: Control Thrusts for the case of Koopman with LQR

Once the chaser spacecraft is close to the target satellite, it is also critical to maintaining the correct attitude. Additionally, attitude control is put into place for this. The algorithm that was created is once more an LQR control. However, because the dynamic equations in this situation are not linear, it is first necessary to linearize Euler's equations before computing the gain matrix. The controller receives its input from the state vector, which is determined by the discrepancy between the actual and desired states. A control torque is produced by the controller. The torque from the outside disturbances is the plant's input, and its output is the orientation and angular rate of the chaser in the body-fixed frame. The figures provided below, show how the attitude controller functions.

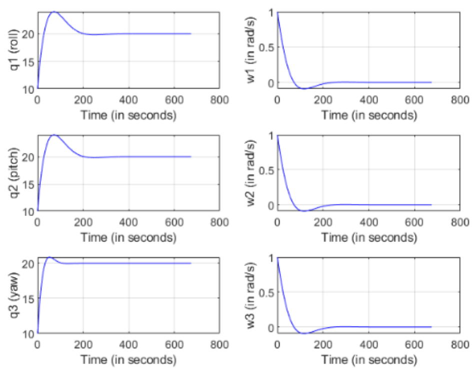

FIGURE 4.7: Attitude angle (Euler angles) and Angular velocities errors vs time plots for Attitude Controller

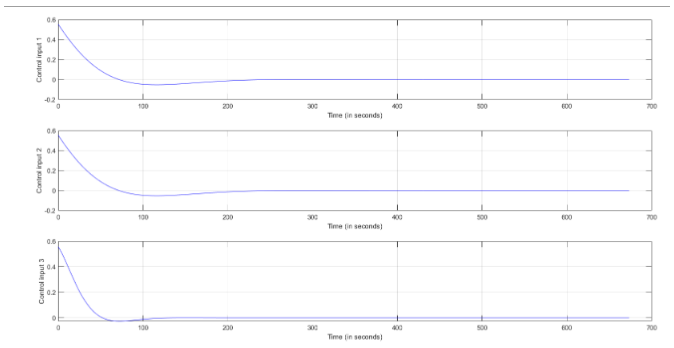

FIGURE 4.8: Control Torques for Attitude Controller

4.0.3 Docking Algorithm

Kindly note that this docking algorithm hasn't been included in the main model yet and was tested independently under different initial conditions. The reason is that this algorithm is valid only for specific case 1 (refer to section 3.2). In case 1, the docking port vector is stationary and it is assumed that both the velocity vector of the chaser satellite and the docking port vector lie in the same plane (i.e Y-Z plane).

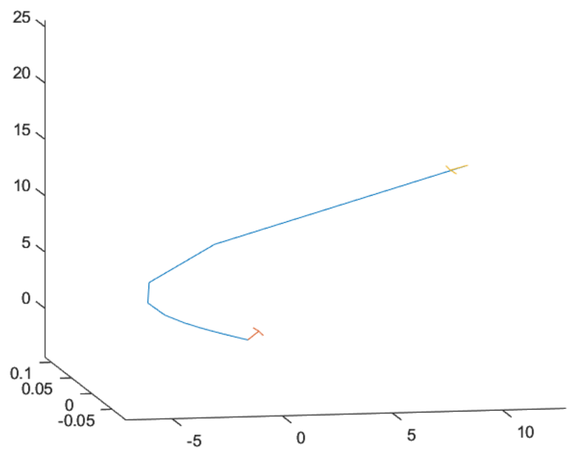

Figure 4.9: Trajectory of Chaser Satellite during docking operation for Case 1

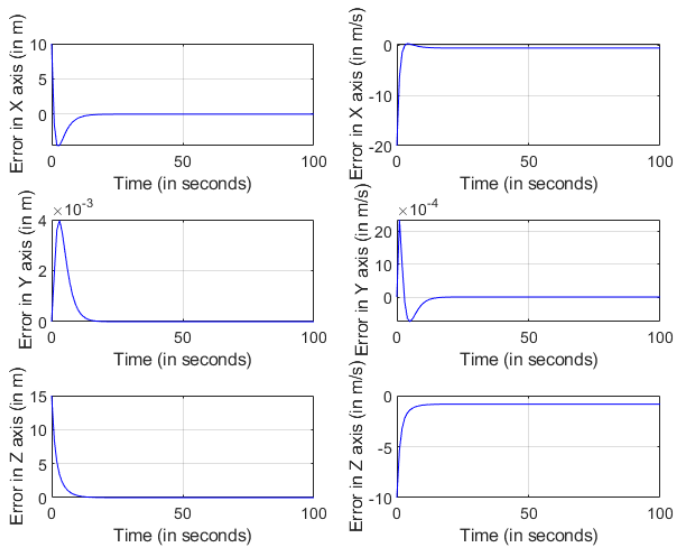

Figure 4.10: Chaser Satellite's position and velocity graphs in LVLH frame during docking operation for Case 1

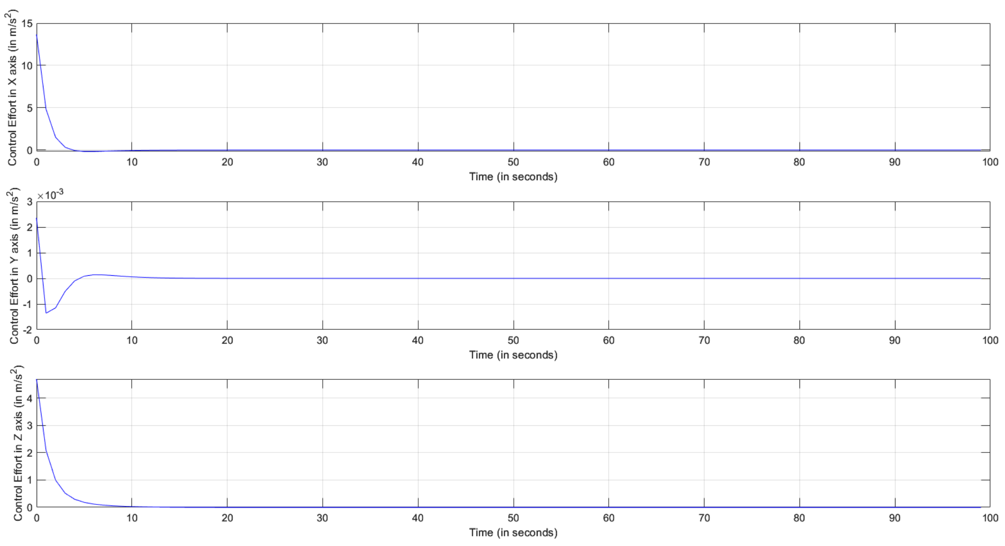

FIGURE 4.11: Control Thrusts during docking operation for Case 1

The initial position and velocity of the chaser satellite in LVLH frame is given by $[10m\ 0m\ 15m]$ and $[-20m/s\ 0m/s\ -10m/s]$ respectively. The Navigation gain is taken to be 1.2 for the simulation (arrived at by hit and trial method). The simulation is run for a total of 100s. A Proportional-Guidance based algorithm has been designed to align the approaching chaser satellite's velocity vector with the docking port vector (in opposite direction). A control thrust is produced by the controller. The performance of the proposed guidance algorithm is depicted by the above given figures (4.9 - 4.11).

Chapter 5

Conclusion

The issue of rendezvous and docking between two satellites has been investigated in this research. The objective is to create and evaluate guiding and control algorithms for various approach circumstances. To mimic the relative motion of the two satellites in the first scenario that was discussed, linearized CW equations were employed. It was investigated how satellite relative motion should be modeled under J_2 perturbations. In MATLAB, simulations are run while designing an LQR-based position controller.

However, the controller based on the Linearized CW equations proved to be ineffective in the case of far-field rendezvous operation. It is possible to utilize nonlinear control techniques to improve accuracy and resilience. They do, however, turn out to be computationally demanding. It was possible to account for the nonlinear dynamics of the spaceship rendezvous problem while still using linear control design methods, thanks to the Koopman framework used in this study. Thus, in the second case, a Koopman operator-based guidance system was implemented, which proved to be successful for both long-range and close-by rendezvous operations. When compared to when Linear CW Equations were used, the Koopman-based control technique was found to provide improved accuracy.

An LQR-based strategy was tested for attitude control in order to keep the intended orientation acting against outside disturbances. The simulations show that the state error tends to zero and that the disturbances are damped very quickly. Given that all the gains are calculated before the simulation begins, the key benefits of this strategy are its ease of adjusting and cheap computing cost.

The problem for autonomous docking of chaser spacecraft with target satellite was broken down and properly formulated. In this thesis, only case 1 was considered where the docking port vector is stationary (when observed from the LVLH frame) and the chaser spacecraft's velocity vector lies in the same plane as the docking port vector. A PN-based guidance algorithm was designed for docking the chaser satellite with the target satellite.

In the future, it could be interesting to test the performance of the controllers for trajectories other than elliptical/circular orbits for target satellites. Apart from this, other data-driven approaches (like Neural networks for Koopman embeddings, Sparse Identification, etc.) can be looked into for better approximating the A_{koop} and B_{koop} matrices as Extended DMD has an inherent issue overfitting which demands some cross-validation.

The next phase would be to design a guidance-based algorithm for maneuvering the chaser satellite from close proximity to the docking port for case 2 i.e. when the docking port vector is stationary but it doesn't necessarily lie in the same plane as the chaser spacecraft's velocity vector. For this, the 3D impact-guidance algorithm will be looked into. In the final phase, a guidance algorithm will be designed for the case where the docking port vector is rotating (as in the case of a tumbling satellite), and then also design a controller for post-docking stabilization. Once all the relevant algorithms have been designed in the simulation environment, the last step would be to set up a test bench for demonstration of the rendezvous and docking algorithms in the physical environment.

Appendix A

MATLAB Codes

Actuatual_Trajectory_in_ECI_Frame.m script

```
%%%
Actuatual_Trajectory_in_ECI_Frame.m script begins
%%%

%%% Simulation of Target and Chaser satellite trajectory in ECI frame.
disp('Simulation started')

%%% Get planet parameters
Reference_planet

%%% Initial conditions for Target Satellite

%For now we're assuming that the chaser satellite has the information about
%the trajectory of target satellite.

alt_c = 10*254*1.6*1000; % in meters
R0 = Re+alt_c;           % in meters

x0_tar = R0; % in meters
y0_tar = 0;  % in meters
z0_tar = 0;  % in meters

i_tar = 0*pi/180;                                        % Inclination of target satellite orbit

r_tar = norm([x0_tar;y0_tar;z0_tar]);                    % Semi-major axis of target satellite
semi_major_tar = R0 + 9000*1000;
vcirc_tar = sqrt(2*myu/r_tar - myu/semi_major_tar);      % orbital speed of target satellite
%vcirc_tar = sqrt(myu/r_tar);
theta_tar = atan2(y0_tar,x0_tar);

xdot0_tar = -vcirc_tar*sin(theta_tar);            % in m/s
ydot0_tar = vcirc_tar*cos(i_tar)*cos(theta_tar);  % in m/s
zdot0_tar = vcirc_tar*sin(i_tar)*cos(theta_tar);  % in m/s

% initial state vector of target satellite
```

```
stateinitial_tar = [x0_tar; y0_tar; z0_tar; xdot0_tar; ydot0_tar; zdot0_tar];

% Keplerian elements for target satellite
[a,ecc,incl,RAAN,argp,nu,truelon,arglat,lonper] = ijk2keplerian(stateinitial_tar(1:3),stateinitial_tar(

%%% Initial conditions for Chaser Satellite
alt_c = 10*254*1.6*1000; % in meters
R0 = Re + alt_c;         % in meters

x0_c = R0+500*1000;       % in meters
y0_c = 50*1000;           % in meters
z0_c = 20*1000;           % in meters

r_cha = norm([x0_c;y0_c;z0_c]);  % semi-major axis of chaser satellite
semi_major_c = R0+9000*1000;
vcirc_c = sqrt(2*myu/r_cha - myu/semi_major_c);       % orbital speed of target satellite
%vcirc_c = sqrt(myu/r_cha);
i_c = 0*pi/180;                         % inclination of chaser satellite orbit
theta_c = atan2(y0_c,x0_c);

xdot0_c = -vcirc_c*sin(theta_c);          % in m/s
ydot0_c = vcirc_c*cos(i_c)*cos(theta_c);  % in m/s
zdot0_c = vcirc_c*sin(i_c)*cos(theta_c);  % in m/s

stateinitial_c = [x0_c; y0_c; z0_c; xdot0_c; ydot0_c; zdot0_c];

%{
%%% Integrating equations of motion
options = odeset('RelTol',1e-6);
%[tout_tar,stateout_tar] = ode45(@Target_Satellite,tspan,stateinitial_tar,options);  % target satellite
[tout_c,stateout_c] = ode45(@Target_Satellite,tspan,stateinitial_c,options);
%}

%%% Conversion from ECI to LVLH frame
[C,Cdot] = ECI2LVLH(stateinitial_tar);

%%% Finding Initial conditions in LVLH frame
r_c_0 = stateinitial_c(1:3);
r_t_0 = stateinitial_tar(1:3);


%w = [0;0;vcirc_tar/semi_major_tar]; %orbital speed of target satellite
v_c_0 = stateinitial_c(4:6);
v_t_0 = stateinitial_tar(4:6);
v_ct_0_I = v_c_0 - v_t_0;


%%% Finding Relative motion equations (CW_equations)
Relative_Motion

%%Convert meters to Kilometers
stateout_tar = stateout_tar/1000;
%stateout_c = stateout_c/1000;

%%% Extract state vector of target satellite
```

```
xout_tar = stateout_tar(:,1); % in kms
yout_tar = stateout_tar(:,2); % in kms
zout_tar = stateout_tar(:,3); % in kms

xdot_out_tar = stateout_tar(:,4)*1000; % in m/s
ydot_out_tar = stateout_tar(:,5)*1000; % in m/s
zdot_out_tar = stateout_tar(:,6)*1000; % in m/s

%{
%%% Extract state vector of chaser satellite
xout_c = stateout_c(:,1);
yout_c = stateout_c(:,2);
zout_c = stateout_c(:,3);

xdot_out_c = stateout_c(:,4);
ydot_out_c = stateout_c(:,5);
zdot_out_c = stateout_c(:,6);

r_out_c = [xout_c,yout_c,zout_c];
v_out_c = [xdot_out_c,ydot_out_c,zdot_out_c];
%}


subplot(1,2,1)
plot3(xout_tar, yout_tar, zout_tar,'b-')
xlabel('X (in kms)')
ylabel('Y (in kms)')
zlabel('Z (in kms)')
title('Absolute motions of chaser and target satellites in ECI frame')
grid on
hold on
plot3(xout_ECI,yout_ECI,zout_ECI,'r-')
%plot3(xout_c,yout_c,zout_c,'r-')
%plot(xout_c, yout_c)
t = plot3(xout_tar(1),yout_tar(1),zout_tar(1),'o','MarkerFaceColor','red'); %target satellite marker
c = plot3(xout_ECI(1),yout_ECI(1),zout_ECI(1),'o','MarkerFaceColor','green'); %chaser satellite marker
%c = plot3(xout_c(1),yout_c(1),zout_c(1),'o','MarkerFaceColor','green'); %chaser satellite marker

surf(X,Y,Z,'EdgeColor','none')
legend('Target Sat Trajectory','Chaser Sat Trajectory','Target Sat position','Chaser Sat position','Ear
axis equal

subplot(1,2,2)
plot(xout_ECI/1000,yout_ECI/1000,'b-')
xlabel('X ( x 10^6 meters)')
ylabel('Y (x 10^6 meters)')
title('Absolute motion of chaser and target satellites in ECI frame (X-Y)')
grid on
hold on
%plot(xout_c/1000,yout_c/1000,'r-')
plot(xout_tar/1000,yout_tar/1000,'r-')
r = plot(xout_tar(1)/1000,yout_tar(1)/1000,'o','MarkerFaceColor','g');
r_R = plot(xout_ECI(1)/1000,yout_ECI(1)/1000,'o','MarkerFaceColor','y');
legend('Chaser Sat Trajectory', 'Target Sat Trajectory','Target Sat position','Chaser Sat position')
%axis equal
```

```
for k = 2:length(xout_ECI)
    t.XData = xout_tar(k);
    t.YData = yout_tar(k);
    t.ZData = zout_tar(k);
    c.XData = xout_ECI(k);
    c.YData = yout_ECI(k);
    c.ZData = zout_ECI(k);
    r.XData = xout_tar(k)/1000;
    r.YData = yout_tar(k)/1000;
    r_R.XData = xout_ECI(k)/1000;
    r_R.YData = yout_ECI(k)/1000;
    drawnow
end

fig2 = figure();
set(fig,'color','white')
subplot(3,2,1)
plot(tout_ren,xout_ECI-xout_tar,'b-')
xlabel('Time (in seconds)')
ylabel('Error in X axis (in Kms)')
grid on

subplot(3,2,3)
plot(tout_ren,yout_ECI-yout_tar,'b-')
xlabel('Time (in seconds)')
ylabel('Error in Y axis (in Kms)')
grid on

subplot(3,2,5)
plot(tout_ren,zout_ECI - zout_tar,'b-');
xlabel('Time (in seconds)')
ylabel('Error in Z axis (in Kms)')
grid on

subplot(3,2,2)
plot(tout_ren,xdot_out_ECI - xdot_out_tar,'b-');
xlabel('Time (in seconds)')
ylabel('Error in X axis (in m/s)')
grid on

subplot(3,2,4)
plot(tout_ren,ydot_out_ECI - ydot_out_tar,'b-');
xlabel('Time (in seconds)')

xlabel('Time (in seconds)')
ylabel('Error in Z axis (in m/s)')
grid on

%%%
Actuatual_Trajectory_in_ECI_Frame.m script ends
%%%
```

Relative_Motion.m script

```
%%% Simulation of Relative motion (Satellite Rendezvous)
disp('Relative Motion Simulation started')

%%% Initial conditions
%w = vcirc_tar/semi_major_tar;  % orbital velocity of target satellite
w = sqrt(myu/semi_major_tar^3); %orbital rate of target satellite
%w = sqrt(myu/r_tar^3);

%%% Extracting initial states of chaser satellite in rotating coordinate frame (LVLH)
x0 = r_ct_0_R(1); % in m
y0 = r_ct_0_R(2); % in m
z0 = r_ct_0_R(3); % in m

xdot0 = v_ct_0_R(1); % in m/s
ydot0 = v_ct_0_R(2); % in m/s
zdot0 = v_ct_0_R(3); % in m/s

stateinitial = [x0; y0; z0; xdot0; ydot0; zdot0]; % Initial state vector of chaser satellite in LVLH fr
%stateinitial = [10;0;10;0;0;0];
disp(stateinitial)
%%% Integrating equations of motion
options = odeset('RelTol',1e-6);
%[tout_ren,stateout_ren] = ode45(@Rendezvous,tspan,stateinitial,options);  %Relative motion equation
[tout_tar,stateout_tar] = ode45(@Target_Satellite,tspan,stateinitial_tar,options);

%Rendezvous with perturbation
Opt    = odeset('RelTol',1e-6);
%Far field rendezvous
state_ref = [100;0;100;10;1;10];
[tout_ren,stateout_ren] = ode45(@Rendezvous_with_Perturbation,4*tspan,stateinitial,Opt,w,R0,state_ref);
U = diff(stateout_ren(:,4:6));


%Safety Ellipse
stateinitial_ell = stateout_ren(end,:);
disp(stateinitial_ell);
for t = 0:0.1:10
    state_ref = [100*cos(1*t); 0*t; 100*sin(1*t); -100*sin(1*t); 0*t; 100*cos(1*t)];
    [tout_ren_ell,stateout_ren_ell] = ode45(@Rendezvous_with_Perturbation,t:0.001:t+0.1,stateinitial_el
    disp(stateout_ren_ell)
    disp('Iteration done')
    %U = diff(stateout_ren_ell(4:6));
    hold all
    plot3(stateout_ren_ell(:,1),stateout_ren_ell(:,2),stateout_ren_ell(:,3),'r--')
    %plot(tout_ren,U(:,1))
    stateinitial_ell = stateout_ren_ell(end,:);
end
%Attitude Control
stateinitial_att = [1;1;1;10;10;10];
[tout_att,state_att] = ode45(@Attitude_Control,0.1*tspan,stateinitial_att);
%U = diff(state_att(:,4:6));

%%% Time Window
N = 500;
```

```
m = 3;
tmax = 2*period*number_of_orbits;
timestep = tmax/N;
time = 0:timestep:tmax; %

%%%Converting Continuous to discrete using Runga Kutte Method
stateinitial_test = [1000;-1000;1000;3;3;-3];
N_test = 500;
tmax_test = 5000;
timestep_test = tmax_test/N;
time_test = 0:timestep_test:tmax_test;

for i = 1:length(time_test)-1
    utest(1,i) = unifrnd(-1,1);
    utest(2,i) = unifrnd(-1,1);
    utest(3,i) = unifrnd(-1,1);
end

[stateout_ren_dis_test,time_test] = RK4(timestep_test,tmax_test,stateinitial_test,utest,R0,w);

%[stateout_ren_dis,time] = RK4(timestep,tmax,stateinitial,R0,w);

%%% Koopman Transformation
stateout_koop_test = Koopman(stateout_ren_dis_test);
%stateout_koop = Koopman(stateout_ren_dis);

%%% Naive DMD
[G, Akoop, Bkoop] = DMD(stateout_ren_dis_test,utest);
%Z = DMD(stateout_koop,time); %output state vector in LVLH frame after DMD

%%% Calculating Terminal state vector in Lifted Dimension
[r,c] = size(Bkoop);


%
beta_koop = (Akoop^N)*stateout_koop_test(:,1);

%%% IRLS Algorithm for finding optimal control inputs
%ukoop = IRLS(N,m,CN_koop,beta_koop);

%%% Constructing Final output state vector
%time_test = 0:timestep:tmax_test-1;

%Q = 100*[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0;0 0 0 1000 0 0; 0 0 0 0 1000 0; 0 0 0 0 0 1000;];
%R  = 10^7*eye(3);

Q = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0;0 0 0 500000 0 0; 0 0 0 0 10000 0; 0 0 0 0 0 100000;];
R  = 5*10^8*eye(3);

K = dlqr(Akoop,Bkoop,Q,R);

Z = zeros(length(stateout_ren_dis_test(:,1)),length(time));
Z(:,1) = stateinitial;

Ukoop = diff(Z(:,4:6));
```

```
%plot(time,Z(1,:));
%plot(time_test(1:end-1),utest(3,:));

%%% Extracting output state vector in rotating frame (LVLH)

xout = stateout_ren(:,1);
yout = stateout_ren(:,2);
zout = stateout_ren(:,3);

xdot_out = stateout_ren(:,4);
ydot_out = stateout_ren(:,5);
zdot_out = stateout_ren(:,6);

%{
xout = stateout_ren_dis(1,:);
yout = stateout_ren_dis(2,:);
zout = stateout_ren_dis(3,:);

xdot_out = stateout_ren_dis(4,:);
ydot_out = stateout_ren_dis(5,:);
zdot_out = stateout_ren_dis(6,:);
%}

xout_tar = stateout_tar(:,1);
yout_tar = stateout_tar(:,2);
zout_tar = stateout_tar(:,3);

xdot_out_tar = stateout_tar(:,4);
ydot_out_tar = stateout_tar(:,5);
zdot_out_tar = stateout_tar(:,6);


%%% Converting from LVLH frame to ECI frame
r_out_LVLH = [xout,yout,zout];
%r_out_LVLH = [xout;yout;zout];
r_out_tar = [xout_tar,yout_tar,zout_tar];

v_out_LVLH = [xdot_out,ydot_out,zdot_out];
v_out_tar = [xdot_out_tar,ydot_out_tar,zdot_out_tar];

[r_out_ECI,v_out_ECI] = LVLH2ECI(C,Cdot,r_out_LVLH,r_out_tar,v_out_LVLH,v_out_tar);

%%% Extracting output position vector of chaser satellite in ECI frame
xout_ECI = r_out_ECI(:,1)/1000; %in Kms
yout_ECI = r_out_ECI(:,2)/1000; %in Kms
zout_ECI = r_out_ECI(:,3)/1000; %in Kms

xdot_out_ECI = v_out_ECI(:,1); %in m/s
ydot_out_ECI = v_out_ECI(:,2); %in m/s
zdot_out_ECI = v_out_ECI(:,3); %in m/s
```

Reference_Planet.m script

```
% This function will define all the parameters of the planet

J2 = 1082.64e-6;     %Second Harmonic due to Earth oblateness effect
Re = 6.3716e6;       % Radius of Earth
M = 5.972e24;        % Mass of Earth
G = 6.67e-11;
myu = G*M;           % Gravitational Constant
```

Target_Satellite.m script

```
function dstatedt = Target_Satellite(t,state)
%{
This function returns the velocity and acceleration vectors for Target
Satellite trajectory in E-C-I frame.
%}


%%% Intertia parameters
mass = 2.6;    % in kgs

Reference_planet

r_vector = state(1:3);
mod_r = norm(r);
r_hat = r/rnorm;
F_grav = -(G*m*M/(mod_r)^2)*r_hat;
%%% Dynamics
F = Fgrav;
accel = F/m;

fstate = [v; a];
```

ECI2LVLH.m script

```
function [C, Cdot] = ECI2LVLH(state_out)

r = state_out(1:3); % position vector in ECI frame
rnorm = norm(r); % magnitude of position vector
r_hat = r/rnorm; % unit position vector

v = state_out(4:6); % velocity vector in ECI frame
vnorm = norm(v); % magnitude of velocity vector
v_hat = v/vnorm; % unit velocity vector

h = cross(r,v); % angular momentum vector in ECI frame
hnorm = norm(h);
h_hat = h/hnorm;

C(1,:) = ex_hat;
C(2,:) = ey_hat;
C(3,:) = ez_hat;

Cdot(1,:) = ex_dot;
Cdot(2,:) = ey_dot;
Cdot(3,:) = ez_dot;
```

LVLH2ECI.m script

```
function [r_out_ECI, v_out_ECI] = LVLH2ECI(C,Cdot,r_LVLH,r_ref,v_LVLH,v_ref)
[m,n] = size(r_LVLH);

for i = 1:m
    r_out_ECI_tilda = (C')*((r_LVLH(i,:))');
    r_out_ECI(i,:) = (r_out_ECI_tilda + (r_ref(i,:))')';

    v_out_ECI_tilda = (C')*((v_LVLH(i,:))' - Cdot*r_out_ECI_tilda);
    v_out_ECI(i,:) = (v_out_ECI_tilda + (v_ref(i,:))')';

end
```

uhat_d.m script

```
function uhatd = uhat_d(u,ud)
uhat = u/norm(u);
uhatd = (ud - dot(uhat,ud)*uhat)/norm(u);
```

Rendezvous.m script

```
function dstate = Rendezvous(time,state_vec)

x_c = state_vec(1);
y_c = state_vec(2);
z_c = state_vec(3);

%%%Gravity Model
Reference_planet
w = 0.000592034043275745;

%%% Dynamics
%x_ddot = 2*w*state(5) + w*w*(R0+x) - myu*(R0+x)/(norm([R0+x; y; z])^(3/2));
%y_ddot = -2*w*state(4) + w*w*y - myu*y/(norm([(R0+x); y; z])^(3/2));
%z_ddot = -myu*z/(norm([(R0+x); y; z])^(3/2));

%x_ddot = 2*w*state(5) + 3*w*w*x;
%y_ddot = -2*w*state(4);
%z_ddot = -w*w*z;
%accel = [x_ddot; y_ddot; z_ddot];

A = [0 0 0 1 0 0;0 0 0 0 1 0; 0 0 0 0 0 1; 3*w*w 0 0 0 2*w 0; 0 0 0 -2*w 0 0; 0 0 -w*w 0 0 0];
B = [0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
K = [0.001 0 0 0.3 0 0; 0 0.01 0 0 7 0; 0 0 0.01 0 0 7];

U = [0;0;0];
Xdot = A*state + B*U;

%%% Return derivatives vector
%dstate = [vel; accel];
```

```
dstate = Xdot;
```

Rendezvous_with_Perturbation.m script

```
function dstate = Rendezvous_with_Perturbation(t,state,w,R0,state_ref)

% This function implements the CW equations and outputs state vector in
% LVLH frame. J2 perturbations have also been included.

i = 0*pi/180;
x_c = state_vec(1);
y_c = state_vec(2);
z_c = state_vec(3);

%%%Gravity Model
Reference_planet

%%% Dynamics (
%x_ddot = 2*w*state(5) + w*w*(R0+x) - myu*(R0+x)/(norm([R0+x; y; z])^(3/2));
%y_ddot = -2*w*state(4) + w*w*y - myu*y/(norm([(R0+x); y; z])^(3/2));
%z_ddot = -myu*z/(norm([(R0+x); y; z])^(3/2));

s = (3*J2*Re*Re/(8*R0*R0))*(1+3*cos(2*i));
c = sqrt(1+s);

%%% Dynamics with Controller
% LQR Controller has been implemented.

A = [0 0 0 1 0 0;0 0 0 0 1 0; 0 0 0 0 0 1; (5*c-2)*w*w 0 0 0 2*w*c 0; 0 0 0 -2*w*c 0 0; 0 0 -(3*c-2)*w*
B = [0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
%K = [0.001 0 0 0.3 0 0; 0 0.01 0 0 7 0; 0 0 0.01 0 0 7];

Q = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0;0 0 0 100 0 0; 0 0 0 0 100 0; 0 0 0 0 0 100];
%Q = 10*eye(6);
R = 10^9*[100 0 0; 0 100 0; 0 0 100];
%R = [100 0 0; 0 10 0; 0 0 10^2];
K = lqr(A,B,Q,R);
U = -K*(state - state_ref);
%U = [0;0;0];


Xdot = A*(state - state_ref)+ B*U;
hold all
plot(t,U)

%%% Return derivatives vector
%accel = [x_ddot;y_ddot;z_ddot];
%dstate = [vel; accel];
dstate = Xdot;
```

RK4.m script

```
function [stateout,time] = RK4(timestep,tmax,stateinitial_nl,u,R0,w)
time = 0:timestep:tmax;
```

```
n = length(stateinitial_nl);
stateout = zeros(n,length(time)-1);
stateout(:,1) = stateinitial_nl;

%B = [0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
%K = [1 0 0 0 0 0; 0 0.1 0 0 0 0; 0 0 0 0 0 0];

for j = 1:length(time)-1
    k1 = Derivatives(stateout(:,i),u(:,i),R0,w);
    k2 = Derivatives(stateout(:,i)+k1*timestep/2, u(:,i),R0,w);
    k3 = Derivatives(stateout(:,i)+k2*timestep/2,u(:,i),R0,w);
    k4 = Derivatives(stateout(:,i)+k3*timestep,u(:,i),R0,w);
    kRk4 = (1/6)*(k1+2*k2+2*k3+k4);

    %U(:,i) = -K*stateout(:,i);
    stateout(:,i+1) = stateout(:,i)+kRk4*timestep;

end
```

Derivatives.m script

```
function dstatedt = Derivatives(state,U,R0,w)

i = 0*pi/180;

%%%Gravity Model
Reference_planet

s = (3*J2*Re*Re/(8*R0*R0))*(1+3*cos(2*i));
c = sqrt(1+s);

%%% Dynamics with Controller

% A simple statefeedback controller has been used for now. This controller
% will be upgraded to include LQR.

A = [0 0 0 1 0 0;0 0 0 0 1 0; 0 0 0 0 0 1; (5*c-2)*w*w 0 0 0 2*w*c 0; 0 0 0 -2*w*c 0 0; 0 0 -(3*c-2)*w*
B = [0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];

%K = [0.001 0 0 0.3 0 0; 0 0.01 0 0 7 0; 0 0 0.01 0 0 7];
%U = -K*state;

Xdot = A*state + B*U;

%%% Return derivatives vector
dstatedt = Xdot;
```

DMD.m script

```
function [G, A, B] = DMD(stateout_ren_dis_test,U,time)

%This function performs DMD on the discretized system.

X = stateout_ren_dis_test(:,1:end-1);
```

```
Y = stateout_ren_dis_test(:,2:end);

%A = Y*pinv(X);
%B = eye(6);

G = Y*pinv([X;U]);

A = G(:,1:6);
B = G(:,7:end);
```

Attitude_Control.m script

```
function [dstate,U] = Attitude_Control(t,state)

% This function is used for attitude control of the satellite. Here the
% input is given in terms of quaternions.

%initial conditions
q1 = state(4);
q2 = state(5);
q3 = state(6);

w1 = state(1);
w2 = state(2);
w3 = state(3);

Jxx = 25;
Jyy = 25;
Jzz = 5;
I = [Ixx 0 0; 0 Iyy 0; 0 0 Izz];
%%%Gravity Model
Reference_planet

%%%Dynamics with Controller
state_ref = [0; 0; 0; 20; 20; 20];

%LQR control algorithm has been implemented
U = -K*(state - state_ref);

Xdot = A*(state - state_ref)+ B*U;

%Returning derivatives vector
dstate = Xdot;
```

Docking_Main_Trajectory.m

```
    % Planet Parameters
Reference_planet

% Docking port vector
d = [2;0;3]; % Docking port location from COG of target satellite
d_hat = d/norm(d); % unit vector of docking port.

semi_major_tar = 19435600;
```

```
w = sqrt(myu/semi_major_tar^3);
R0 = 10435600;

% Chaser Satellite Relative Position
x0 = 10;
y0 = 0;
z0 = 15;

xdot0 = -20;
ydot0 = 0;
zdot0 = -10;

v = [xdot0;ydot0;zdot0];
v_hat = v/norm(v);

% Relative Dynamics
%Rendezvous with perturbation

Opt    = odeset('RelTol',1e-6);

%Near field rendezvous
state_ref = [0;0;0;-d_hat];
state_initial = [x0;y0;z0;xdot0;ydot0;zdot0];
tspan = 0:1:100;
[tout_ren,stateout_ren] = ode45(@Docking,tspan,state_initial,Opt,w,R0,state_ref);
U = diff(stateout_ren(:,4:6));

% Extracting State output vector
xout = stateout_ren(:,1);
yout = stateout_ren(:,2);
zout = stateout_ren(:,3);

xdot_out = stateout_ren(:,4);
ydot_out = stateout_ren(:,5);
zdot_out = stateout_ren(:,6);

% plots
fig1 = figure();
set(fig1,'color','white');

subplot(3,2,1)
plot(tout_ren,xout,'b-')
xlabel('Time (in seconds)')
ylabel('Error in X axis (in m)')
grid on

subplot(3,2,3)
plot(tout_ren,yout,'b-')
xlabel('Time (in seconds)')
ylabel('Error in Y axis (in m)')
grid on

subplot(3,2,5)
plot(tout_ren,zout,'b-');
xlabel('Time (in seconds)')
```

```
ylabel('Error in Z axis (in m)')
grid on

subplot(3,2,2)
plot(tout_ren,xdot_out,'b-');
xlabel('Time (in seconds)')
ylabel('Error in X axis (in m/s)')
grid on

subplot(3,2,4)
plot(tout_ren,ydot_out,'b-');
xlabel('Time (in seconds)')
ylabel('Error in Y axis (in m/s)')
grid on

subplot(3,2,6)
plot(tout_ren,zdot_out,'b-');
xlabel('Time (in seconds)')
ylabel('Error in Z axis (in m/s)')
grid on

fig2 = figure();
set(fig1,'color','white');

subplot(3,1,1)
plot(tout_ren(1:end-1),U(:,1),'b-')
xlabel('Time (in seconds)')
ylabel('Control Effort in X axis (in m/s^2)')
grid on

subplot(3,1,2)
plot(tout_ren(1:end-1),U(:,2),'b-')
xlabel('Time (in seconds)')
ylabel('Control Effort in Y axis (in m/s^2)')
grid on

subplot(3,1,3)
plot(tout_ren(1:end-1),U(:,3),'b-');
xlabel('Time (in seconds)')
ylabel('Control Effort in Z axis (in m/s^2)')
grid on


fig3 = figure();
set(fig2, 'color', 'white')

hold all
final = stateout_ren(end,1:3)/norm(stateout_ren(end,1:3));
plot3(xout,yout,zout)
quiver3(0,0,0,d_hat(1),d_hat(2),d_hat(3))
quiver3(x0,y0,z0,v_hat(1),v_hat(2),v_hat(3))
axis([-10 10 -0.1 0.1 -15 15])
%quiver3(0,0,0,final(1),final(2),final(3))
```

Docking.m

```
     function dstate = Rendezvous_with_Perturbation(t,state,w,R0,state_ref)

% This function implements the CW equations and outputs state vector in
% LVLH frame. J2 perturbations have also been included.

i = 0*pi/180;

%%%Gravity Model
Reference_planet

s = (3*J2*Re*Re/(8*R0*R0))*(1+3*cos(2*i));
c = sqrt(1+s);

%%% Dynamics with Controller

% LQR Controller has been implemented.

A = [0 0 0 1 0 0;0 0 0 0 1 0; 0 0 0 0 0 1; (5*c-2)*w*w 0 0 0 2*w*c 0; 0 0 0 -2*w*c 0 0; 0 0 -(3*c-2)*w*
B = [0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
Q = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0;0 0 0 10 0 0; 0 0 0 0 10 0; 0 0 0 0 0 10];
R = 10*[1 0 0; 0 1 0; 0 0 1];

K = lqr(A,B,Q,R);
U = -K*(state - state_ref);

Xdot = A*(state - state_ref)+ B*U;

%{
% Impact angle guidance
vx = state(4);
vy = state(5);
vz = state(6);

v = norm([vx;vy;vz]);
theta = atan2(vz,vx);
alpha_chaser = atan2()
theta_dot = -v*sin(alpha_chaser - theta)/R;
a = N*v*theta_dot;
%}

%%% Return derivatives vector
dstate = Xdot;
```

Bibliography

[1] Koji Yamanaka, Finn Ankersen, "New State Transition Matrix for Relative Motion on an Arbitrary Elliptical Orbit" *Journal of Guidance, Control, and Dynamics.* Vol. 25, No. 1, January–February 2002.

[2] Arya B.R., Laila Beebi M., Johnson Y, "Satellite Tracking Control under J_2 Perturbations" *International Journal of Scientific & Engineering Research* Volume 7 Issue 4, April 2016.

[3] M. T. Walsh and M. A. Peck, "A general approach for calculating far-field orbital rendezvous maneuvers," *AIAA Guidance, Navigation, and Control Conference* 2017, p. 1730.

[4] C. D. Karlgaard, "Robust rendezvous navigation in elliptical orbit" *Journal of Guidance, Control, and Dynamics* vol. 29, no. 2, pp. 495– 499, 2006.

[5] Y. Yao, R. Xie, and F. He, "Flyaround orbit design for autonomous rendezvous based on relative orbit elements" *Journal of Guidance, Control, and Dynamics* vol. 33, no. 5, pp. 1687–1692, 2010.

[6] I. Lopez and C. R. Mclnnes, "Autonomous rendezvous using artificial potential function guidance" *Journal of Guidance, Control, and Dynamics* vol. 18, no. 2, pp. 237–241, 1995.

[7] V. Zinage and E. Bakolas, "Far-field minimum-fuel spacecraft rendezvous using Koopman operator and l 2/l 1 optimization" *2021 American Control Conference (ACC),* 2021, pp. 2992–299.

[8] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control" *SIAM Journal on Applied Dynamical Systems* vol. 15, no. 1, pp. 142–161, 2016.

[9] Jeffrey Hough and Steve Ulrich, "Lyapunov Vector Fields for Thrust-Limited Spacecraft Docking with an Elliptically-Orbiting Uncooperative Tumbling Target" *AIAA SciTech Forum* 6-10 January 2020, Orlando, FL.

[10] Zhanyuan Jiang, Jianquan Ge, Qiangqiang Xu, and Tao Yang, "Impact Time Control Cooperative Guidance Law Design Based on Modified Proportional Navigation" *aerospace* 2021, 8, 231.