DESIGN, DEVELOPMENT, AND TESTING OF EMBEDDED COMPUTING ON AFIT's CONTROL & AUTONOMY SPACE PROXIMITY ROBOT (CASpR)

THESIS

Collin Gwaltney, 2d Lt, USSF

AFIT-ENY-MS-21-D-067

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-21-D-067

DESIGN, DEVELOPMENT, AND TESTING OF EMBEDDED COMPUTING ON AFIT'S CONTROL & AUTONOMY SPACE PROXIMITY ROBOT (CASpR)

THESIS

Presented to the Faculty
Department of Aeronautics and Astronautics
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Aeronautical Engineering

Collin Gwaltney, B.S.M.E.
2d Lt, USSF

December 23, 2021

AFIT-ENY-MS-21-D-067

DESIGN, DEVELOPMENT, AND TESTING OF EMBEDDED COMPUTING ON AFIT'S CONTROL & AUTONOMY SPACE PROXIMITY ROBOT (CASpR)

THESIS

Collin Gwaltney, B.S.M.E.
2d Lt, USSF

Committee Membership:

Maj Costantinos Zagaris, Ph.D
Chair

Lt Col Bryan Little, Ph.D
Member

Lt Col David Curtis, Ph.D
Member

AFIT-ENY-MS-21-D-067

Abstract

Ground-based Rendezvous and Proximity Operations (RPO) testbeds provide critical testing of algorithms and equipment prior to full deployment of a system. This thesis reviews RPO algorithm testbeds and discusses the development of the Control and Autonomy Space proximity Robot (CASpR) kinematic testbed housed at the Air Force Institute of Technology (AFIT). CASpR operates on a rail system to propagate the trajectories of two satellites using the Hill-Clohessy-Wiltshire (HCW) Equations of Motion (EOMs). In this study, the implementation of a Jetson TX2i as an onboard flight computer is discussed and accomplished on one of the two simulated satellites. Each hardware component used in the process of adding embedded computing as well as the software and paths of communication are all discussed in detail. Three primary test scenarios are developed and simulated both in Matlab as well as on CASpR to begin characterizing the testbed and understanding its capabilities and limitations.

Table of Contents

Page

List of Figures

List of Tables

DESIGN, DEVELOPMENT, AND TESTING OF EMBEDDED COMPUTING ON AFIT'S CONTROL & AUTONOMY SPACE PROXIMITY ROBOT (CASpR)

I. Introduction

1.1 Motivation

The importance of Rendezvous and Proximity Operations (RPO) as a field of study is growing rapidly as space becomes increasingly competitive, interactive, and collaborative. According to the 2020 Defense Space Strategy,"The Department [of Defense] is rapidly transforming its approach to space from a support function to a warfighting domain in order to achieve our desired conditions and strategic objectives over the next 10 years in the face of identified threats, challenges, and opportunities"[20]. As space develops as a warfighting domain, the exigence of developing force multipliers is increasing rapidly.

The study of RPO allows researchers to improve Guidance, Navigation, and Control (GNC) algorithms on-board spacecraft. This in turn shifts the burden of commanding actuation from ground stations to the satellite's own on-board computers. According to DiMauro et al., "high on-board autonomy allows us to:

1. lighten the ground stations' and control centers' operational loads, reducing the overall mission costs
2. increase the science return by reducing the inactivity period due to the coverage problems or large communication delays
3. facilitate the response to contingencies/ failures, therefore increasing the mission robustness and safety

4. guarantee mission flexibility/adaptability, allowing a prompt response to the environment changes" [21]

By developing simulations, ground-based testbeds, and space-based test setups, the degree to which autonomy is used in space may be increased, leading to the outcomes outlined by DiMauro et al. Virtual simulations alone do not allow the testing of hardware components. On the other end of the spectrum, space-based tests offer true testing conditions and higher quality test results; however, these tests require huge upfront commitments. As is explored in this study, many additional and often unexpected results come from implementing hardware into a test scenario. These results provide valuable information which helps to bridge the gap between virtual simulation and reality. For a research institution such as Air Force Institute of Technology (AFIT), a ground-based testbed is the ideal middle ground for developing, testing, and iterating with GNC algorithms.

1.2 Background

Satellite RPO is a field of study which requires precision and reaction times which are faster than can be provided manually through ground stations. Several scientific Formation Flying (FF) missions have been conducted over the years to prove that satellites are capable of managing their states relative to other satellites. The Project for On-Board Autonomy-3 (PROBA-3) mission is sponsored by the European Space Agency (ESA) and set to launch in 2023 [22]. The goal of this mission requires highly precise control to enable formation flying within 1mm relative position. Because so many satellite mission sets require RPO, whether cooperative FF or uncooperative, it is an area of great interest and study.

GNC testbeds provide means of testing satellite control algorithms as well as some hardware components. A variety of ground-based testbeds are discussed in Section

2.2. One testbed is developed by Wyatt Harris at AFIT [23]. This testbed is called Control and Autonomy Space proximity Robot (CASpR) and is discussed in detail throughout this study, including its structure, software, and other development.

1.3 Problem Statement

AFIT is interested in implementing embedded computing onto ground-based RPO testbeds. Additionally, AFIT intends to continue to develop and utilize the CASpR testbed to conduct Hardware in the Loop (HIL) testing for RPO scenarios. The focus of this study then is on integrating embedded computing onto the CASpR system. This will be accomplished by placing a Jetson TX2i onboard with battery power and remotely accessing the control algorithm through a separate desktop. This allows the CASpR system to become increasingly stand-alone and provides a way forward for moving from a solid structure to a more dynamic system.

1.4 Research Focus

This research is focused on integrating embedded computing onboard the CASpR testbed to develop a more rigorous testing environment which includes hardware and computers carried onboard the simulated satellite. This allows each simulated satellite to carry its own computational load and develop a way forward for expanding the operational capabilities of CASpR. This may include adding additional simulated satellites to the test scenarios and removing the constraints of the rail system which CASpR currently uses for locomotion in favor of alternate actuation methods.

1.5 Research Objectives

The primary goal of this research is to develop a method by which embedded computing may be cost-effectively implemented onto a gantry-style RPO hardware-in-the-loop testbed. Additionally, this research endeavors to begin the work of characterizing the CASpR testbed. Additional improvements to the testbed and its underlying code are made through the course of this study, including the addition of guidance and control functions. The specific objectives in this thesis are as follows:

1. The testbed must provide computing power sufficient to perform RPO control algorithms.
2. The testbed's sensor stack must be able to carry both the computer, an external power source, and additional sensors such as cameras.
3. The testbed's power supply shall provide ample power for an hour of continuous operation.
4. The sensor stack must be easily interchangable such that other stacks may be used onboard CASpR.

1.6 Document Overview

Chapter II includes background information on the use of a wide array of ground-based GNC testbeds and classifies them based on Degrees of Freedom (DOFs), surface type, simulator type, and whether the testbeds are kinematic or dynamic. Dynamic and kinematic testbeds are explored in greater depth before delving into some of the most common ways to model relative motion of satellites. The Nonlinear Equations of Relative Motion (NERMs) are developed as well as the Linearized Equations of Relative Motion. Simplified models including Hill-Clohessy-Wiltshire (HCW), Tschauner-Hempel, and Yamanaka Ankersen along with their assumptions and preferred use cases are explored. Chapter II ends by exploring common RPO maneuvers such as targeting and common trajectories between chief and deputy satellites.

Chapter III proceeds by determining the methodology which is used to develop CASpR to meet the research goals outlined in Chapter I. First, the structure of CASpR is discussed and the potential errors introduced by its deflection. A series of equations for determining the deflection of CASpR are developed. Next, the hardware used to build CASpR including the computers, communication systems, actuators, power supply, sensors, and the stack along with their specifications are discussed. The software including C++, Python, MobaXTerm, and MATrix LABoratory (MATLAB) along with how they are used in conjunction with the hardware are explored. The system is then analyzed in terms of both time and position scaling. Finally, a series of tests are developed which begin the characterization of CASpR.

II. Background and Literature Review

This chapter discusses Rendezvous and Proximity Operations (RPO) testbeds, their purpose, and classifications. Additionally, various types of relative motion models are discussed. First the Nonlinear Equations of Relative Motion (NERMs) and their wide applicability to modeling relative motion are considered. Then, linearized models and their respective use cases, assumptions, and limitations are examined.

2.1 GNC Testbeds

Guidance, Navigation, and Control (GNC) algorithms require verification and validation. Virtual simulations are often used as a first step in this process, but these simulations lack the ability to accurately model the system's hardware components. Physical testbeds provide a method for testing GNC algorithms with Hardware in the Loop (HIL). Hardware may have delays, noise, uncertainty, response times, and other attributes which are not fully captured in a software-based model. For this reason, HIL testbeds are often built and utilized as the next step of development.

Testbeds can be space-based such as Prototype Research Instruments and Space Mission technology Advancement (PRISMA) [24] which tests GNC algorithms for formation flying as well as rendezvous maneuvers. Being space-based allows the testbed to simulate very little, as it is already testing in the operational environment. This in turn leads to higher fidelity results.

The disadvantages of a space-based testbeds are the long lead time requirement and high upfront cost. Ground-based testbeds offer a far cheaper and more rapid testing process, but often require sacrificing some fidelity of the results. In most cases, universities and other research organizations determine that ground-based systems provide adequate fidelity to verify and validate their research.

Testbeds conducting RPO research operate on the assumption that two or more satellites are in close proximity. Qualitatively, this means that the distance between the chief and each deputy is much less than the distance between the chief and the body about which it is orbiting. With this assumption, the chief satellite is traveling along a straight path rather than along the surface of an ellipsoid.

2.2 Classification of Ground-Based Testbeds

Ground-based testbeds are developed to meet an array of different requirements. Dynamic testbeds are more general than kinematic. Kinematic testbeds simulate the motion of the satellite only. Dynamic testbeds on the other hand simulate both the motion of the satellite as well as the propulsion systems which propagate the motion.

The type of testbed (dynamic or kinematic) is dictated by the needs of the research being conducted. Based on the literature, most institutions seem to prefer dynamic testbeds rather than kinematic. Testbeds rely on a variety of surfaces as well, with dynamic testbeds generally using glass, granite, or epoxy surfaces while kinematic testbeds generally utilize either a gantry-style rail system or any flat surface.

One of the most important aspects of a testbed is the Degrees of Freedom (DOFs) offered by its design. Two horizontal translational Degree of Freedom (DOF) are ordinarily simple to attain, while a third (vertical) DOF is slightly more difficult to attain for a kinematic system and much more difficult to attain for a dynamic system.

The rotational DOFs are often more difficult to achieve. Rotation about the vertical axis is generally the first rotational DOF. The two rotational DOFs about the horizontal axes require the addition of bearings which increases the complexity of the system. Because of this, the jump from one to three rotational DOFs is normally made with the addition of only one physical component. As a result, testbeds with two rotational DOFs are uncommon.

Several ground-based testbeds can be found in the literature. The key attributes of each are distilled into Table 1.

Table 1: Testbed Classification[2][3][13][14][1][15][16][17][18]

Group/ University	Name	Simulator Type	Surface	Trans. DOF	Rot. DOF	Kinematic/ Dynamic
Università La Sapienza	PINOCCHIO	Air bearing	Glass	2	3	Dynamic
Embry-Riddle	N/A	Gantry	Rail	3	1	Kinematic
NPS	POSEIDYN	Air bearing	Granite	2	1	Dynamic
UF	ADAMUS	Air bearing	Epoxy	3	3	Dynamic
Kirtland AFB	CANS	Rollers	Floor	2	1	Kinematic
Georgia Tech	ASTROS	Air bearing	Epoxy	2	3	Dynamic
Florida Tech	ORION	Air bearing	Epoxy	3	3	Dynamic*
Stanford	TRON	Robotic Arm	Rail	3	3	Kinematic

*Indicates the system has a single kinematic DOF.

2.2.1 Dynamic Testbeds

Dynamic systems require that the system can move freely without being inhibited by forces which would not be present in the operational environment. Most of these dynamic testbeds operate with a system of air bearings. The air bearings allow the system to glide across a flat surface without friction forces perturbing the motion. These air bearings result in two translational degrees of freedom.

A third translational DOF can be added, but these introduce a new level of complexity to the system. Because the third DOF is normally oriented vertically, gravity on the testbed must be taken into account. Because of this, the third translational DOF is generally kinematic rather than dynamic. However, some researchers have succeeded in producing true 6-DOF ground-based testbeds such as the ADvanced Autonomous MUltiple Spacecraft (ADAMUS) testbed built by the University of Florida[15]. This is attained by counter-weighting the system so that the sensor stack is neutrally buoyant. The ADAMUS system is shown in Figure 1.

Dynamic systems are often fitted with cold gas thrusters similar to those which

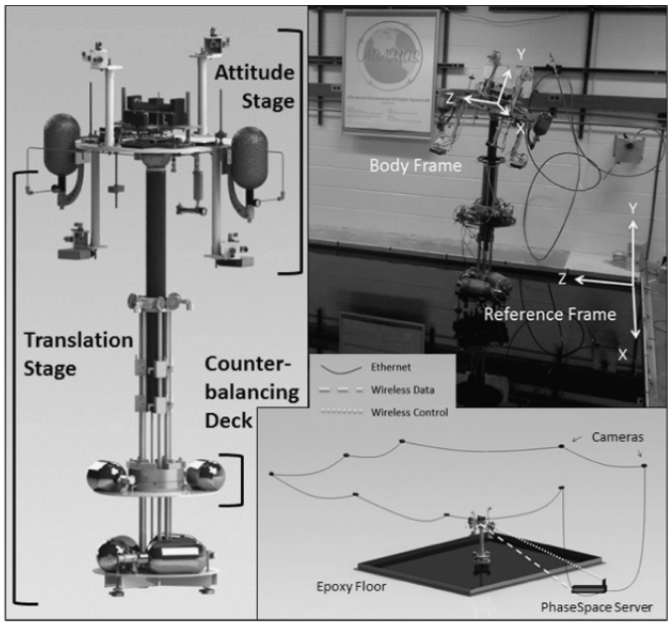


Figure 1: ADAMUS Dynamic Testbed [1]

will be used on the satellite. These provide the propulsion of the testbed in the translational degrees of freedom. The capabilities of a testbed are increased with a propulsion system in the loop. Robens, et al. were able to test limit cycle controllers using the University of Florida's ORION testbed[25].

Dynamic testbeds require that the surface they operate on is both flat and smooth. Because of these constraints, there are three primary materials which are used as surfaces. These materials and their advantages and disadvantages are provided in table 2 from Rybus et al. [26].

Table 2: Common surface materials for planar air-bearing microgravity simulators

	Surface material	Surface size	Surface roughness	Surface flatness	Surface levelling
1	Granite	Small to medium	Low	High	Adjustable legs, possibility of surface tilting to simulate low-g conditions
2	Glass	Small	Low	High	Adjustable legs, possibility of surface tilting to simulate low-g conditions
3	Granite	Medium to large	Moderate	Moderate	Levelled surface obtained during construction process, no adjustable legs

Spherical air bearings are also used for the satellite mount. This allows the simulated satellite's sensors and structure to move freely in three angular degrees of freedom. Reaction wheels and Control Moment Gyroscopes (CMGs) are used on the testbed to control the angular motion of the payload, as they would control the satellite in a space environment. These provide the moments responsible for rotating the structure in accordance with the GNC system's commanded rotations.

Space-based testing of spacecraft attitude systems is expensive and can lead to failure in a short period of time. Earth-based testing can be used instead to decrease cost and decrease time required to test spacecraft attitude systems. Building a testbed allows testing of control algorithms as well as software and hardware which will be used in the spacecraft. In [27], a spherical air bearing creates a 3D pendulum to serve as the testbed.

Although dynamic testbeds can be useful, the models developed through them are not perfect. One limitation or assumption of most air bearing testbeds currently in use is that the center of mass must be coincident with the center of rotation. This simulates a low gravity environment which may not hold up in Low Earth Orbit (LEO) when gravity gradient torque is applied.

2.2.2 Kinematic Testbeds

Kinematic systems require only that the system can move the proper direction and speed. The testbeds must be built such that the system can perform the requested maneuvers at some scaled size within the boundaries of the testbed. Additionally, they must be built so that realistic speeds or some scaling of realistic speeds requested are physically capable of being met. Kinematic testbeds are focused on the shape of the motion rather than the forces which cause a satellite to move in a spaceflight environment.

Kinematic testbeds do not require the use of thrusters which may be used on a spacecraft. As a result, the solution space for feasible designs of kinematic testbeds is enlarged and their designs can vary greatly. Some testbeds like the Cooperative Autonomous Networked Systems (CANS) testbed at Kirtland Air Force Base (AFB) are able to operate on any flat surface [2]. The CANS testbed, shown in Figure 2 operates with three omnidirectional roller wheel assemblies oriented at 60 degrees from each other. This allows the system to translate in the xy-plane as well as rotate about the vertical z axis. This gives the system three degrees of freedom.

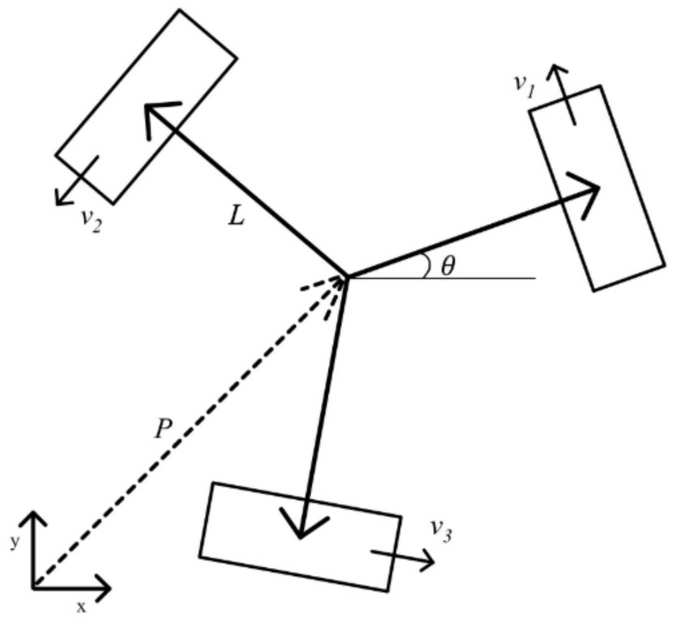


Figure 2: Diagram of CANS Lab ground based robot motor geometry [2]

Other testbeds such as one built by Harris at Embry-Riddle University [3] operate on a gantry-style rail system as shown in Figure 3. This style of testbed allows for three translational DOF to be achieved. The platform also has a rotational DOF about the z axis.

These kinematic testbeds do not use propulsion systems that are present on operational satellites such as cold gas thrusters, reaction wheels, or CMGs. Instead, motors, generally stepper motors, are used to actuate the system.

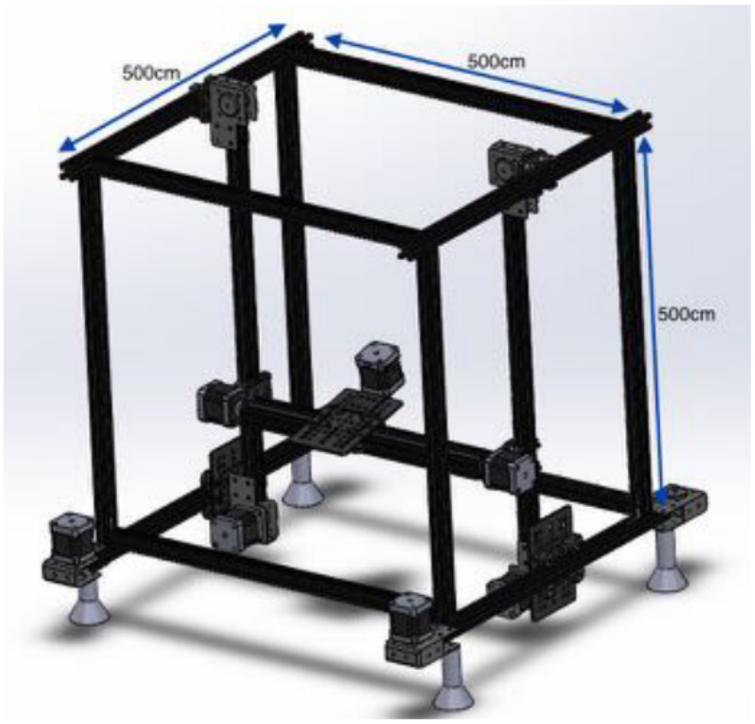


Figure 3: CAD Model of Gantry-Style Testbed [3]

2.3 Modeling Relative Motion

Satellite orbits are often defined using orbital elements such as the Classical Orbital Elements (COEs) with the reference frame being attached to the planetary body about which the satellite is primarily moving. This reference frame is generally considered to be inertial in most applications.

When conducting RPO however, an inertial reference frame does not provide a good scale for comparison between chief and deputy satellites when the satellites are relatively close together. Instead, relative motion models are used to model and compare the position and velocity of the deputy with respect to the chief. Often, the Local-Vertical-Local-Horizontal (LVLH) reference frame is used, where the origin is defined by the chief. The unit vector $\hat{i}$ is given by the direction from the center of the planetary mass toward the center of the chief. $\hat{k}$ is normal to the fundamental plane and positive in the direction of the chief's angular momentum. $\hat{j}$ is subsequently defined by the cross product of $\hat{k}$ with $\hat{i}$ as shown in Figure 4.

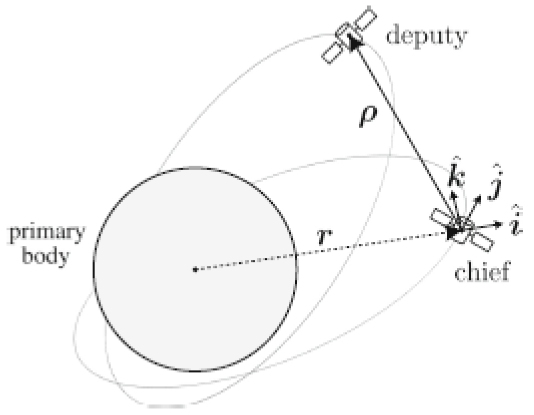


Figure 4: Relative Motion [4]

Using the LVLH reference frame, the focus is shifted from the locations of the satellites relative to the earth to the location of the deputy relative to the chief. This provides a better context for conducting RPO.

2.3.1 Nonlinear Equations of Relative Motion (NERMs)

The NERMs model the unperturbed motion of two satellites' trajectories about a larger body, the two body problem. These two satellites are commonly referred to as the chief and deputy satellites. For models assuming unperturbed motion, the only force present is that of the massive body's (usually Earth) gravity. The NERMs make no assumptions except that the motion is Keplerian. The NERMs can be modeled in any coordinate frame, but for the purposes of this paper, they are given in the LVLH frame by equations 1 - 5 from [28].

$$\ddot{x} - 2\dot{\theta}_0\dot{y} - \ddot{\theta}_0 y - \dot{\theta}_0^2 x = -\frac{\mu(r_0 + x)}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} + \frac{\mu}{r_0^2} \tag{1}$$

$$\ddot{y} + 2\dot{\theta}_0\dot{x} + \ddot{\theta}_0 x - \dot{\theta}_0^2 y = -\frac{\mu y}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} \tag{2}$$

$$\ddot{z} = -\frac{\mu z}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} \tag{3}$$

$$\ddot{r}_0 = r_0\dot{\theta}_0^2 - \frac{\mu}{r_0^2} \tag{4}$$

$$\ddot{\theta}_0 = -\frac{2\dot{r}_0\dot{\theta}_0}{r_0} \tag{5}$$

x, y, and z are the relative position of the deputy spacecraft in relation to the chief in the LVLH reference frame. Auxiliary equations 6 and 7 can be used to fully determine the initial state vector of the NERMs. With these equations, the state of the deputy spacecraft in relation to the chief can be determined by using a numerical integrator such as ODE45 in MATrix LABoratory (MATLAB).

$$\dot{r} = \sqrt{2(\varepsilon + \frac{\mu}{r_0}) - \frac{h_0^2}{r_0^2}} \tag{6}$$

$$\dot{\theta}_0 = \frac{h_0}{r_0^2} \tag{7}$$

Importantly, perturbations and control forces can easily be accounted for in the NERMs by simple addition. The external differential perturbations are given by Equation 8 while the differential control forces are given by Equation 9

$$\mathbf{d} = [d_x, d_y, d_z]^T \tag{8}$$

$$\mathbf{u} = [u_x, u_y, u_z]^T \tag{9}$$

The NERMs with external perturbations and control forces are shown in Equations 10 - 12.

$$\ddot{x} - 2\dot{\theta}_0\dot{y} - \ddot{\theta}_0 y - \dot{\theta}_0^2 x = -\frac{\mu(r_0 + x)}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} + \frac{\mu}{r_0^2} + d_x + u_x \tag{10}$$

$$\ddot{y} + 2\dot{\theta}_0\dot{x} + \ddot{\theta}_0 x - \dot{\theta}_0^2 y = -\frac{\mu y}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} + d_y + u_y \tag{11}$$

$$\ddot{z} = -\frac{\mu z}{[(r_0 + x)^2 + y^2 + z^2]^{\frac{3}{2}}} + d_z + u_z \tag{12}$$

2.3.2 Linearized Equations of Relative Motion

Several different linearized equations of relative motion can be derived, based on the differing assumptions incorporated into the model. Some of these models are discussed in the subsections below. Higher-order approximate models can also be derived for unperturbed motion, but are not explored in this paper.

2.3.3 Clohessy-Wiltshire

The Hill-Clohessy-Wiltshire (HCW) relative motion model [29] [30] is a set of first-order Linear Time-Invariant (LTI) equations which relate the states of a pair of chief and deputy spacecraft. The HCW equations are derived from the NERMs by making a series of assumptions. The three assumptions made to develop the HCW equations are below.

1. The only force present is Two Body Problem (2BP) gravity.

2. The distance between the two satellites is very small relative to the radius of the chief's orbit.

3. The chief satellite's orbit is circular.

Because of these assumptions, the HCW equations are appropriate for many applications, particularly LEO and geostationary orbits and are often used as a starting point for understanding the dynamics of spacecraft performing RPO or otherwise Formation Flying (FF) in close proximity. The second assumption is the same as the assumption made earlier for RPO testbeds, showing that the HCW equations may be sufficiently accurate to be used in testbed applications.

The assumptions serve to simplify the model into a set of Ordinary Differential Equations (ODEs) with a closed form solution where the state vector is given by Equation 13.

$$\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \tag{13}$$

The system is defined by Equation 14 where the system matrix is given by A in Equation 15.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) \tag{14}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \tag{15}$$

The HCW equations may then be formulated in terms of the state transition matrix, given by $e^{A(t-t_0)}$, as shown in equation 16.

$$\mathbf{x}(t) = e^{A(t-t_0)}\mathbf{x}(t_0) \tag{16}$$

Because the system is LTI, solving for the transition matrix at time can be accomplished by setting $t_0 = 0$. The transition matrix is then solved for, resulting in the solution given by equation 17, where $c_{nt} = cos(nt)$ and $s_{nt} = sin(nt)$.

$$\mathbf{x}(t) = \begin{bmatrix} 4-3c_{nt} & 0 & 0 & \frac{s_{nt}}{n} & \frac{2}{n}-\frac{2c_{nt}}{n} & 0 \\ -6nt+6s_{nt} & 1 & 0 & \frac{-2}{n}+\frac{2c_{nt}}{n} & \frac{4s_{nt}}{n}-3t & 0 \\ 0 & 0 & c_{nt} & 0 & 0 & \frac{s_{nt}}{n} \\ 3ns_{nt} & 0 & 0 & c_{nt} & 2s_{nt} & 0 \\ -6n+6nc_{nt} & 0 & 0 & -2s_{nt} & -3+4c_{nt} & 0 \\ 0 & 0 & -ns_{nt} & 0 & 0 & c_{nt} \end{bmatrix} \mathbf{x}(t_0) \tag{17}$$

This result is the closed form solution of the HCW equations given in the LVLH reference frame. From this solution, the relative state of the deputy with respect to the chief can easily be determined at any time.

An example of HCW being used in a practical application is discussed by Flores and Abad [31]. The HCW equations are used in determining the glidepath of the deputy spacecraft when conducting proximity rendezvous for autonomous capturing and docking.

The HCW equations can also be used to understand the natural motion of the spacecraft. The state-space equation given by Equation 17 can be broken down to its components. The only equation which contains a time dependent and non-harmonic term is in the y position, given by Equation 18.

$$y(t) = \frac{2}{n}\dot{x_0}cos(nt) + \left(6x_0 + \frac{4}{n}\dot{y_0}\right) sin(nt) + y_0 - \frac{2}{n}\dot{x_0} - (6nx_0 + 3\dot{y_0})t \qquad (18)$$

The time dependent term is often called the drift term. This is what determines whether the spacecraft being modeled will drift relative to the chief over a period of time. In order to force the deputy to have no drift relative to the chief, the drift term may be forced to be zero by setting the parenthetical to zero as shown in Equation 19.

$$(6nx_0 + 3\dot{y_0}) = 0 \qquad (19)$$

The velocity in the y direction may then be solved in terms of only the relative position in the x direction as well as the mean motion as shown in Equation 20

$$\dot{y_0} = -2nx_0 \qquad (20)$$

When this constraint is met, the deputy circumnavigates the chief over a single orbital period. This is commonly called Natural Motion Circumnavigation (NMC). When viewed in the x-y plane, an NMC takes the shape of a 2x1 ellipse as shown in Figure 5.

2.3.4 Tschauner-Hempel

Unlike the HCW Equations of Motion (EOMs), the EOMs proposed by Tschauner and Hempel [32] do not require that the chief's orbit be circular. The Tschauner-Hempel (TH) EOMs are a simplification of the NERMs. The simplification is accomplished by assuming that the distance between the deputy and chief is much smaller than the distance between the center of the Earth and the chief. This simplification

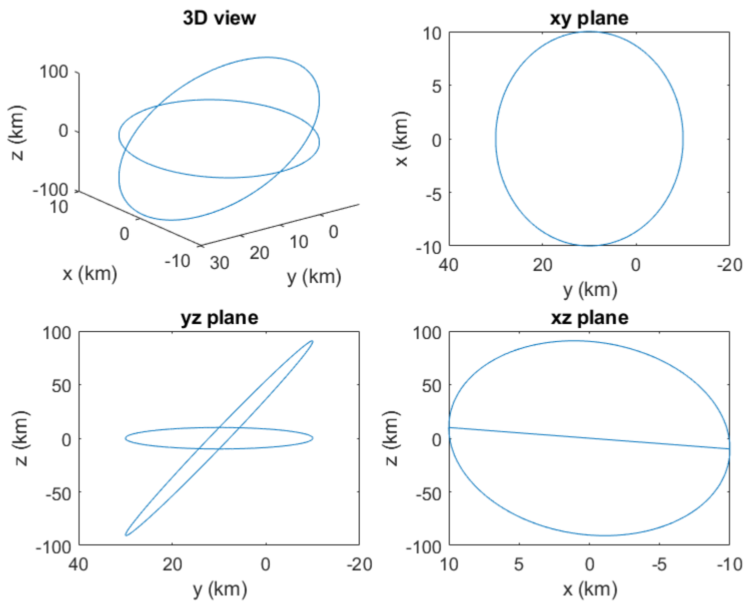


Figure 5: Two NMCs in Multiple Views

reduces the NERMs to the equations given by 21 - 23.

$$\ddot{x} - 2\dot{f}\dot{y} - \ddot{f}y - \dot{f}^2 x - 2\mu x \left(\frac{\dot{f}}{h}\right)^{3/2} = a_x \tag{21}$$

$$\ddot{y} + 2\dot{f}\dot{x} + \ddot{f}x - \dot{f}^2 y + \mu y \left(\frac{\dot{f}}{h}\right)^{3/2} = a_y \tag{22}$$

$$\ddot{z} + \mu z \left(\frac{\dot{f}}{h}\right)^{3/2} = a_z \tag{23}$$

The coordinates are then scaled with respect to the chief's radial distance as shown in Equation 24.

$$\bar{x} = \frac{x}{r_c} \tag{24}$$

This results in a Linear Time-Variant (LTV) system. In order to make the equations invariant with respect to the independent variable, the independent variable in the equations can be changed from time to true anomaly. This modification of the equations is given by Equations 25 and 26 as derived by [28].

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = k \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{25}$$

$$k = 1 + ecosf \tag{26}$$

These equations are further simplified by making the assumption that there are no external forces exhibited on the bodies. The final EOMs proposed by Tschauner and Hempel can be written as shown in Equations 27-29 as derived by [28].

$$\tilde{x}'' = \frac{3}{k}\bar{x} + 2\bar{y}' \tag{27}$$

$$\tilde{y}'' = -2\bar{x}' \tag{28}$$

$$\tilde{z}'' = -\bar{z}' \tag{29}$$

The primes indicate that the differentiation is with respect to the true anomaly rather than time. Ultimately, the simple TH EOMs assume only that no external forces are present and the distance between the chief and deputy is small compared to the distance from the earth to the chief. This small set of assumptions means that the TH EOMs are widely applicable across many scenarios. It is important to note that if the eccentricity of the chief is also assumed to be negligible, these equations reduce to the HCW EOMs. Unfortunately, many of the early solutions to the TH EOMs were complex or included singularities at apogee and perigee which made them impractical to use.

2.3.5 Yamanaka Ankersen

A solution to the TH EOMs has been developed by Yamanaka and Ankersen [33]. A State Transition Matrix (STM) was developed which is a singularity-free solution to the differential equations proposed by Tschauner and Hempel. This set of equations makes no further assumptions beyond those made in the TH EOMs.

The key breakthrough to this solution for linear propagation of unperturbed relative motion was identifying the singularity-free integral given by Equation 30 [28].

$$I = \int_{f_0}^{f} \frac{1}{k(f)^2} df = \frac{\mu^2}{h^3}(t - t_0) \tag{30}$$

The Yamanaka-Ankersen solution to the TH model, given by Equations 27-29, is shown in Equations 31 and 32 [28].

$$\bar{X}(f) = \phi(f, f_0)\bar{X}(f_0) \tag{31}$$

$$\phi(f, f_0) = \phi(f)\phi^{-1}(f_0) \tag{32}$$

The STM is given by Equations 33 and 34, where $c = kcos(f)$ and $s = ksin(f)$ and $\eta = \sqrt{1 - e^2}$ [28].

$$\phi(f) = \begin{bmatrix} s & c & 2-3esI & 0 & 0 & 0 \\ s' & c' & -3e(s'I+\frac{s}{k^2}) & 0 & 0 & 0 \\ c(1+\frac{1}{k}) & -s(1+\frac{1}{k}) & -3k^2 I & 1 & 0 & 0 \\ -2s & e-2c & -3(1-2esI) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & cosf & sinf \\ 0 & 0 & 0 & 0 & -sinf & cosf \end{bmatrix} \tag{33}$$

$$\phi^-1(f(0)) = \frac{1}{\eta^2} \times \begin{bmatrix} -3s\frac{k+e^2}{k^2} & c-2e & 0 & -s\frac{k+1}{k} & 0 & 0 \\ -3(e+\frac{c}{k}) & -s & 0 & -(c\frac{k+1}{k}+e) & 0 & 0 \\ 3k-\eta^2 & es & 0 & k^2 & 0 & 0 \\ -3es\frac{k+1}{k^2} & -2+ec & \eta^2 & -es\frac{k+1}{k} & 0 & 0 \\ 0 & 0 & 0 & 0 & \eta^2 cosf & -\eta^2 sinf \\ 0 & 0 & 0 & 0 & \eta^2 sinf & \eta^2 cosf \end{bmatrix} \tag{34}$$

In this model, the state is given in the form $\bar{\mathbf{x}}(t) = [x, \bar{x}', y, \bar{y}', z, \bar{z}]^T$. An advantage of this STM over the HCW STM is that circularity of the orbit is not required. An eccentricity as high as 0.7 was tested and the results validated[33]. Although at high eccentricities, the NMC has the tendency to diverge from the standard 2x1 ellipse which is seen in the HCW solution. The advantage of this STM compared to many other solutions to the TH EOMs is that it is free of singularities and much simpler compared to others which are also singularity free.

The TH EOMs and Yamanaka-Ankersen (YA) STM assume that there are no outside forces. Because of this, they cannot be used for scenarios which require continuous thrusting maneuvers. Impulsive maneuvers however, can be accomplished by adding the ΔV of the impulsive maneuver to the velocity of the spacecraft at the time of the burn. This can be done once or many times as deemed necessary for the scenario.

The YA STM has been used in real-world scenarios including the Canadian Advance Nanosatellite eXperiment (CanX)-4 and 5 [34]. These missions were a formation-flying demonstration which tested the relative position and velocity estimates given by both the YA STM and the HCW EOMs in a near circular orbit. Nanosatellites such as this the CanX-4 and 5 may be more restricted than most in terms of computing capability which makes a simple STM such as those given by HCW or YA appealing solutions. For eccentric orbits, the generality, simplicity, and non-singularity of the STM proposed by Yamanaka and Ankersen makes it a front runner when considering linear EOMs.

2.4 Rendezvous and Proximity Operations

The HCW EOMs are commonly used in conducting RPO. Variations of several common relative trajectories are often considered in RPO. Some of these frequently chosen trajectories and their characteristics are discussed in this section. Because the z position and velocity are decoupled from the x and y, all trajectories are plotted in the x/y plane. Additionally, targeting procedures allowing a deputy to move into a desired relative state are discussed.

2.4.1 Targeting

Targeting allows a deputy satellite to move into a desired state relative to a chief satellite. For targeting to take place, two burns are required. The first burn allows the deputy spacecraft to intercept with a desired relative position. If no second burn is completed, this is termed an intercept maneuver rather than targeting. The second maneuver determines the satellite's Δv to reach a desired velocity, allowing the deputy to rendezvous with the chief at a small relative velocity.

Targeting utilizes the HCW STM by sub-partitioning the large 6x6 matrix into four smaller 3x3 submatrices. These submatrices are given by Equation 35.

$$\boldsymbol{\Phi} = \left[\begin{array}{c|c} \boldsymbol{\Phi}_{rr} & \boldsymbol{\Phi}_{rv} \\ \hline \boldsymbol{\Phi}_{vr} & \boldsymbol{\Phi}_{vv} \end{array}\right] \tag{35}$$

The position and velocity states at any time are given by Equations 36 and 37.

$$\mathbf{r}(t) = \boldsymbol{\Phi}_{rr}\mathbf{r}_0 + \boldsymbol{\Phi}_{rv}\mathbf{v}_0 \tag{36}$$

$$\mathbf{v}(t) = \boldsymbol{\Phi}_{vr}\mathbf{r}_0 + \boldsymbol{\Phi}_{vv}\mathbf{v}_0 \tag{37}$$

If a target position is given by r_{des} at time t, a burn can be applied at time t_0. The equation of r_{des} is given by Equation 38.

$$\mathbf{r}_{des}(t) = \boldsymbol{\Phi}_{rr}\mathbf{r}_0 + \boldsymbol{\Phi}_{rv}(\mathbf{v}_0 + \boldsymbol{\Delta}\mathbf{v}_1) \tag{38}$$

Solving for Δv_1 then gives:

$$\boldsymbol{\Delta}\mathbf{v}_1 = \boldsymbol{\Phi}_{rv}^{-1}(\mathbf{r}_{des} - \boldsymbol{\Phi}_{rr}\mathbf{r}_0) - \mathbf{v}_0 \tag{39}$$

This assumes that Φ_{rv} is an invertible matrix. The velocity at time t, when the spacecraft has reached the desired position, is given by Equation 40.

$$\mathbf{v}(t) = \boldsymbol{\Phi}_{vr}\mathbf{r}_0 + \boldsymbol{\Phi}_{vv}(\mathbf{v}_0 + \boldsymbol{\Delta}\mathbf{v}_1) \tag{40}$$

Substituting in Equation 39 for Δv_1 yields:

$$\mathbf{v}(t) = \boldsymbol{\Phi}_{vr}\mathbf{r}_0 + \boldsymbol{\Phi}_{vv}\boldsymbol{\Phi}_{rv}^{-1}(\mathbf{r}_{des} - \boldsymbol{\Phi}_{rr}\mathbf{r}_0) \tag{41}$$

To achieve a desired velocity, Equation 41 is algebraically manipulated to solve for $\mathbf{\Delta v}_2$ where $\mathbf{v}_{des} = \mathbf{v}(t) + \mathbf{\Delta v}_2$. This relationship is shown in Equation 42.

$$\mathbf{\Delta v}_2 = \mathbf{v}_{des} - \mathbf{\Phi}_{vr}\mathbf{r}_0 - \mathbf{\Phi}_{vv}\mathbf{\Phi}_{rv}^{-1}(\mathbf{r}_{des} - \mathbf{\Phi}_{rr}\mathbf{r}_0) \tag{42}$$

Equations 39 and 42 are the resulting equations which are used for targeting in proximity operations modeled using the HCW dynamics. Given the spacecraft's state relative to a chief's reference frame, both relative position and velocity of the deputy spacecraft may be controlled for in these two impulsive burns. An example of targeting starting at the origin and achieving a 10 by 20 km NMC is shown in Figure 6.

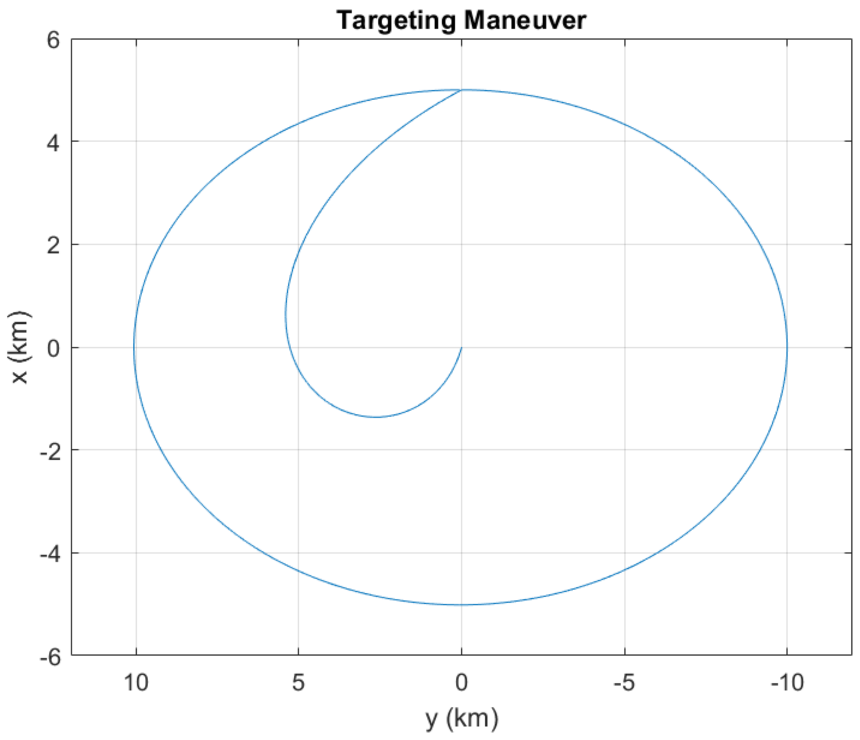


Figure 6: Targeting Maneuver

2.4.2 Common Trajectories

The most commonly used trajectories in RPO with a single deputy and chief spacecraft are discussed in this section. Control and Autonomy Space proximity Robot (CASpR) is programmed such that the relative state of the deputy at the beginning of a scenario is the zero vector. Therefore, this is used as a common starting point for the deputy when discussing maneuvers and how to enter shaped trajectories. Likewise, for the purpose of this study, the chief's position is by definition the zero vector so the two spacecraft are coincident.

2.4.2.1 Trajectory Considerations

For all orbital trajectories which are modeled using the HCW EOMs, the drift term is one of the most important aspects of developing the trajectory. Special care must be taken to ensure in most cases that no drift is present in the system, although some trajectories depend on drift. Targeting for example, allows drift to occur which aids in the positioning of the spacecraft. NMCs however do not have drift. If nulling the drift term is neglected in developing a NMC trajectory, the orbit will not be closed as shown in Figure 7

Additionally, it is important to note that in the HCW EOMs, the z component is decoupled from the x and y. Because of this, this study generally focuses on motion in the x-y plane, understanding that there may also be motion present along the z axis.

2.4.2.2 Non-centered NMC

A NMC trajectory is simple to enter into with any initial state by satisfying the no drift constraint. Any motion which satisfies the drift constraint given by Equation 19 will result in a trajectory which is harmonic in nature with a period equivalent

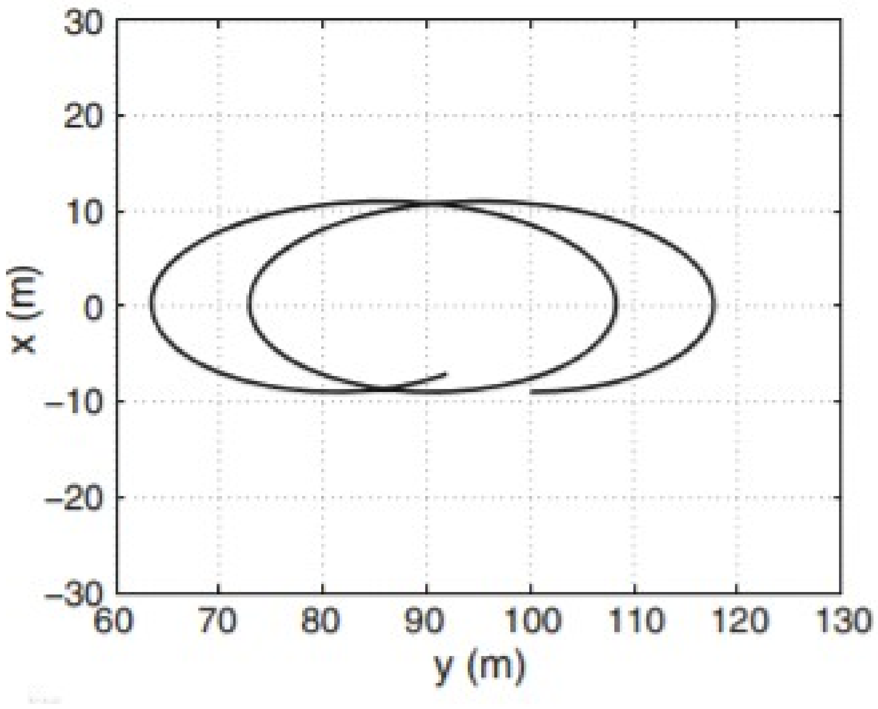


Figure 7: Secular Drift[5]

to the orbital period of the chief satellite. As a result, the trajectory will continue to intersect with the deputy's initial position. This effect can be achieved through an impulsive change in either the x or z velocity, although only an impulsive change in the x velocity will result in a NMC. This NMC is simple and reached through a single impulsive burn, but it is not centered on the initial relative position of the deputy. A non-centered NMC in which the impulsive burn in the x direction is given by 0.0054049km/s has a major axis of 20km and is shown in Figure 8.

2.4.2.3 Centered NMC

A NMC which is centered on the chief's relative position can be entered into in a similar fashion with two burns. An infinite combination of burns can be used to result in a NMC. One of the simplest methods is described here. The first burn targets a purely radial (x) position relative to the chief. When the deputy reaches the desired

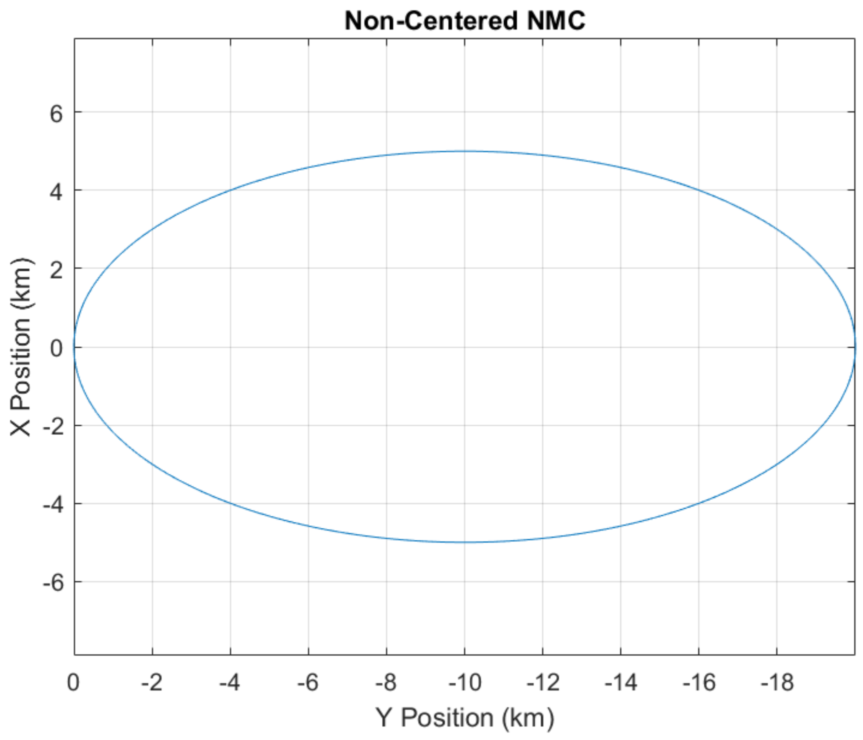


Figure 8: Non-Centered NMC

position, it has maintained alignment with the chief in the in-track (y) direction. Because of this, a NMC trajectory from this position will result in a NMC which is centered on the chief. In order to enter the desired trajectory, the second impulsive burn is calculated such that the desired resulting velocity satisfies the drift term of the HCW EOMs, that is $\dot{y}_{des} = \dot{y}_0 = -2nx_0$. By forcing this drift term to zero and giving no additional thrust in the x direction, a NMC which is centered on the chief is achieved as shown in Figure 9.

2.4.2.4 Leader-Follower

A leader-follower orbit is another common trajectory type. In this scenario, the chief and deputy spacecraft are on the same orbit about earth with the exception of the point in the orbit at a given time. For many spacecraft, this is a difference in true anomaly. For a chief and deputy in close proximity, this means that there is an offset

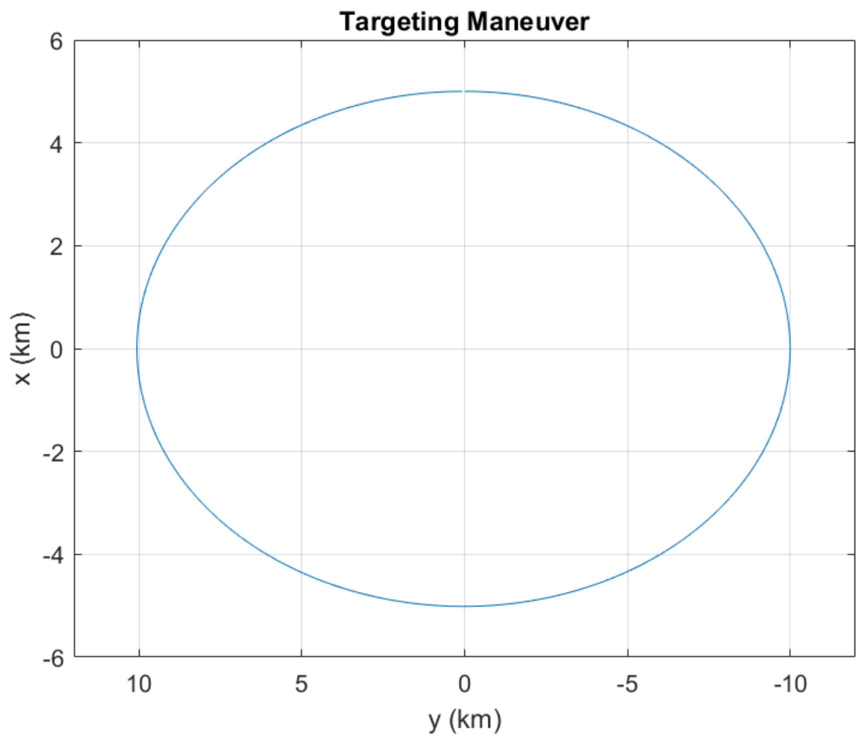


Figure 9: Centered NMC

in the in-track direction and no offset in the radial direction. The deputy satellite is static relative to the chief and can be thought of as "hovering" at a constant relative position rather than orbiting the chief. A leader-follower formation can be reached by targeting a position on the in-track axis with zero relative velocity. A similar "hovering" maneuver can be conducted off of the in-track axis but requires constant thrusting to maintain the relative position. An example of a leader-follower orbit is shown in Figure 10.

2.4.2.5 Teardrop

A teardrop trajectory, unlike the other trajectories discussed, by definition has a non-zero drift term. Because of this, the HCW EOMs are only valid for a period of time while the satellites are close unless additional thrusting is used to keep the satellites close. In a teardrop trajectory, the deputy generally moves around the chief

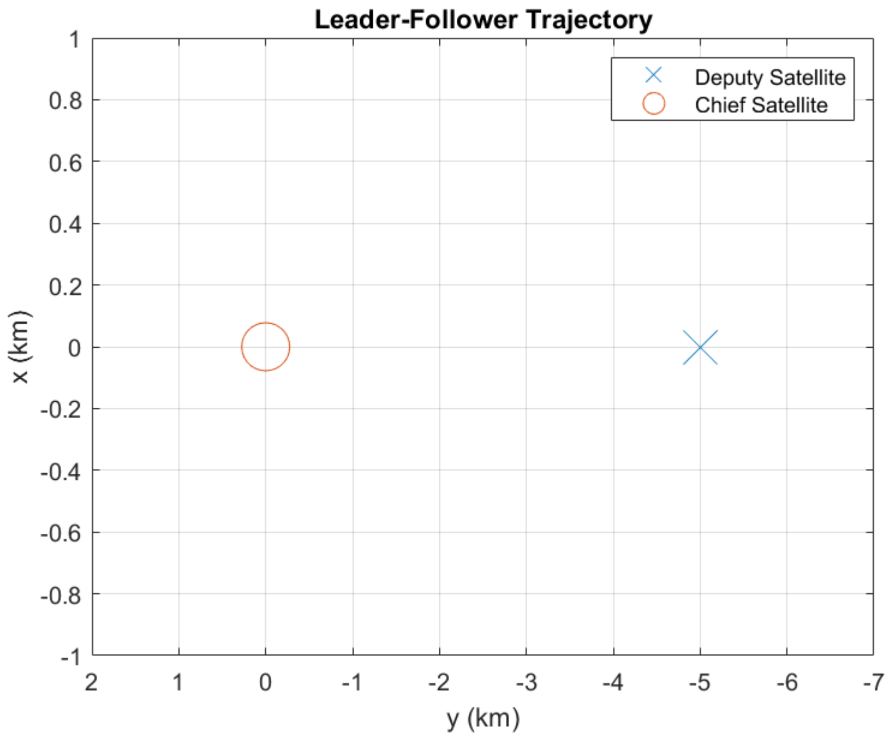


Figure 10: Leader-Follower Trajectory

for one pass and would drift past the chief without additional burns. Instead of allowing natural motion to drift the deputy away however, the deputy can perform a thrusting maneuver at the crossover point to re-enter into the previous state. The trajectory crosses over itself in the x-y plane. At this point, only the velocity must be changed to re-enter the teardrop. This allows the deputy to circumnavigate the chief in less than one orbital period; however, the cost of additional burns may deter mission planners from using this trajectory. An example of a teardrop trajectory is shown in Figure 11.

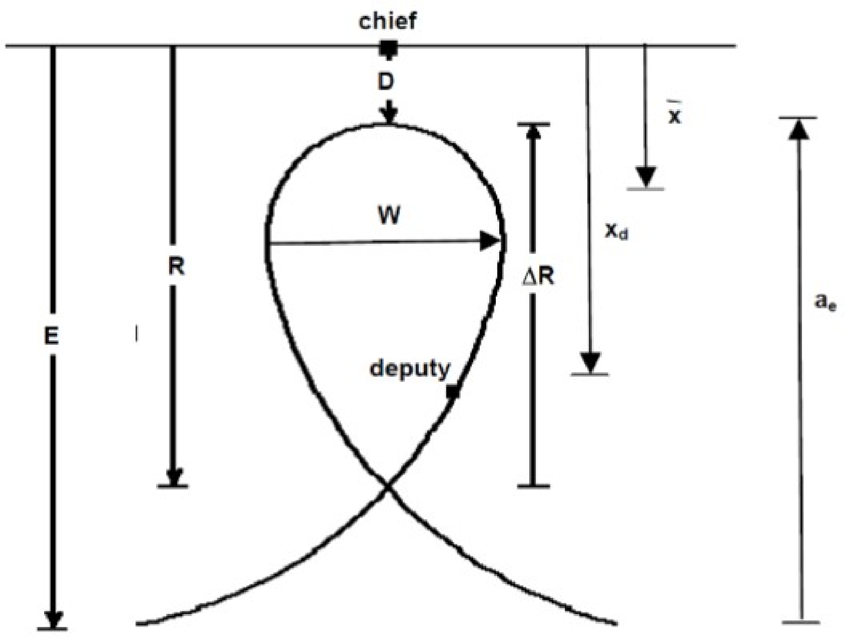


Figure 11: Teardrop Trajectory[6]

2.5 Summary

GNC hardware-in-the-loop testbeds provide a bridge for testing between virtually simulated satellites and operational satellites in a space environment. This intermediate step of testing provides both faster testing of systems and monetary savings to the program sponsors.

A variety of testbeds have been built by universities and research institutions. Those testbeds are divided into dynamic and kinematic testbeds. Dynamic testbeds test the amount of force and torque input into the system and track the geometry of the trajectory. This allows the testing of actuators, sensors, and GNC algortihms. Kinematic testbeds test the motion of the simulated satellite and allow the testing of only sensors and GNC algorithms.

Relative motion models provide a simplified method for comparing the trajectories of two or more satellites. The NERMs provide a mathematical set of equations which accurately model the relative motion of a deputy with no simplifications or assumptions. Assuming the distance between the deputy and chief satellites is small compared to the chief's semi-major axis, the relative motion can be modeled as linear.

Multiple linear models have been developed over the years. The most commonly used of which is the HCW equations. The HCW equations assume perfectly circular orbits. This assumption is valid for many orbits, most commonly in the LEO regime. For the CASpR system, HCW is used, although other models may be considered in the future. Other models such as the Yamanaka-Ankersen solution to the Tschauer-Hempel EOMs may be used to model elliptical chief orbits.

III. Methodology

Chapter II provided background information on Rendezvous and Proximity Operations (RPO) testbeds built by a variety of organizations. Chapter III discusses the following:

- Designs used to build the Air Force Institute of Technology (AFIT) Control and Autonomy Space proximity Robot (CASpR) testbed.
- Physical deflection of the horizontal tubing.
- Hardware used in the development of CASpR.
- Software used in the development of CASpR.
- Options for scaling both time and position in testing.

The first section focuses on the designs which were drawn on to build the primary structure of the CASpR testbed. The second section discusses a methodology by which the physical system's deflection is determined given a variety of parameters. The third section covers the hardware components used in the development of CASpR. The fourth and final section describes the software used by CASpR and how it is implemented to conduct RPO testing.

3.1 CASpR Structure

CASpR's primary structure is based on a design of the Mostly Printed Computer Numerical Control (MPCNC) machine described by V1 Engineering[7]. This MPCNC machine operates as a gantry-style system which operates primarily in two horizontal axes. These are referred to as the x and y axes. The z axis is defined as vertical, with the positive z direction being up in the standard configuration. The z position is driven by a motor turning a lead screw. Due to this design, motion in the z axis is more limited than the other two axes. In addition to the three translational Degrees of Freedom (DOFs), there is a single rotational Degree of Freedom (DOF) about the z axis. The design is shown in Figure 12.

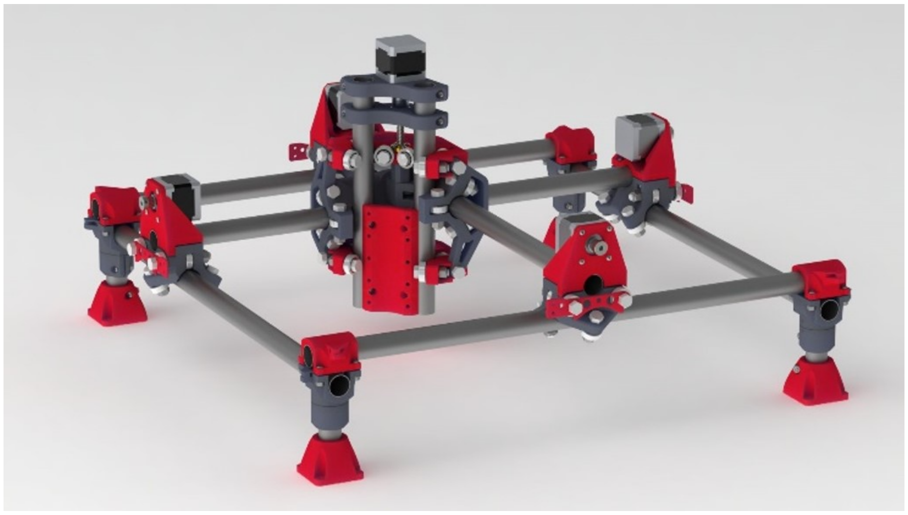


Figure 12: V1 Engineering MPCNC Machine[7]

The design is intended to be simple to build with a large number of parts being open-source 3D-printable parts. All other hardware components for this design can be found in the V1 Engineering guide[7].

In order to build CASpR, this design is scaled up to a 10' x 10' x 6' cage. This cage has two gantry-style Computer Numerical Control (CNC) machines mounted one on top of the other. The upper machine is inverted such that its positive z axis acts in the opposite direction of the standard MPCNC machine. This cage provides the basic structure of CASpR as shown in Figure 13.

3.1.0.1 CASpR Orientation and Conventions

CASpR operates primarily in the horizontal plane. This is because motion is more limited in the vertical direction. Motion in the vertical direction is driven by

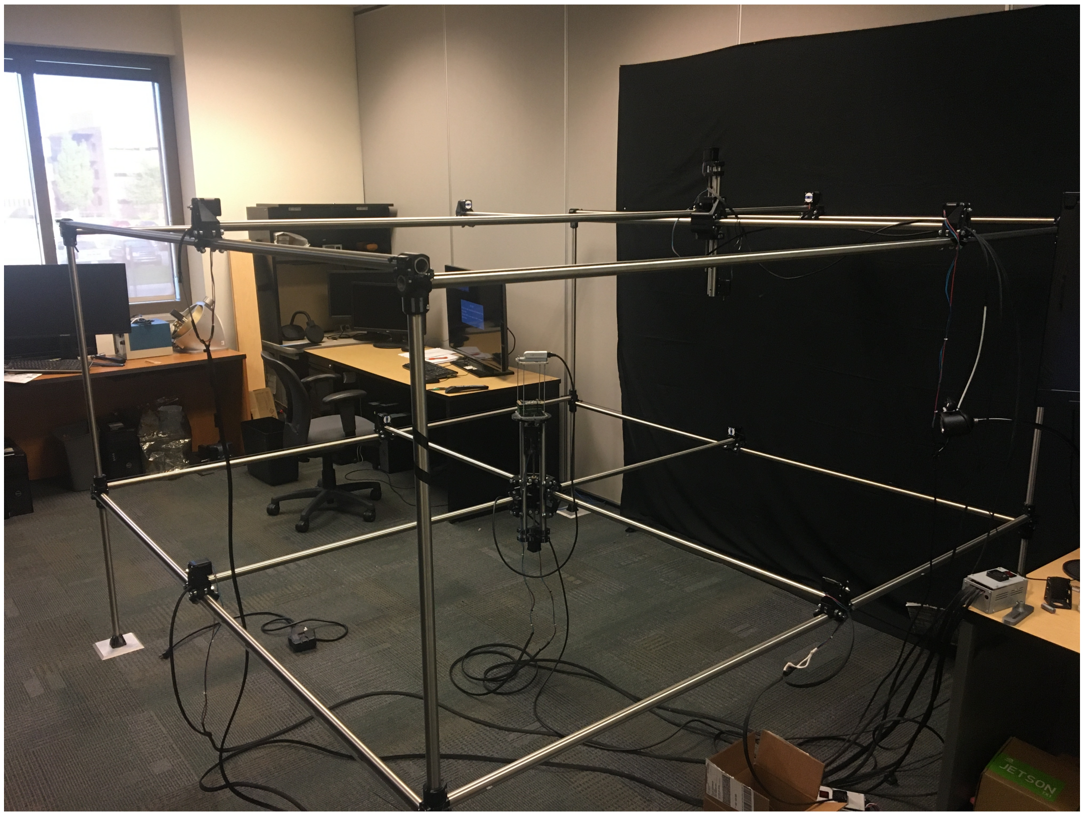


Figure 13: CASpR Structure

a stepper motor attached to a lead screw while the horizontal axes are driven by stepper motors attached to pulleys and belts. This means that motion in the vertical direction is slower and more restricted. Because the z state is decoupled from the x and y states, the vertical axis is used as the z axis with up being positive z. From this, a right-handed convention is used to define a coordinate system with x pointing toward the black backdrop. The coordinate system and general room layout are shown in Figure 14.

3.1.1 CASpR Deflection

The purpose of CASpR is to model relative motion of two Formation Flying (FF) satellites. In order to accomplish this, CASpR simulates trajectories using 4 DOFs. An issue may arise using this setup where the rods which provide the structure of

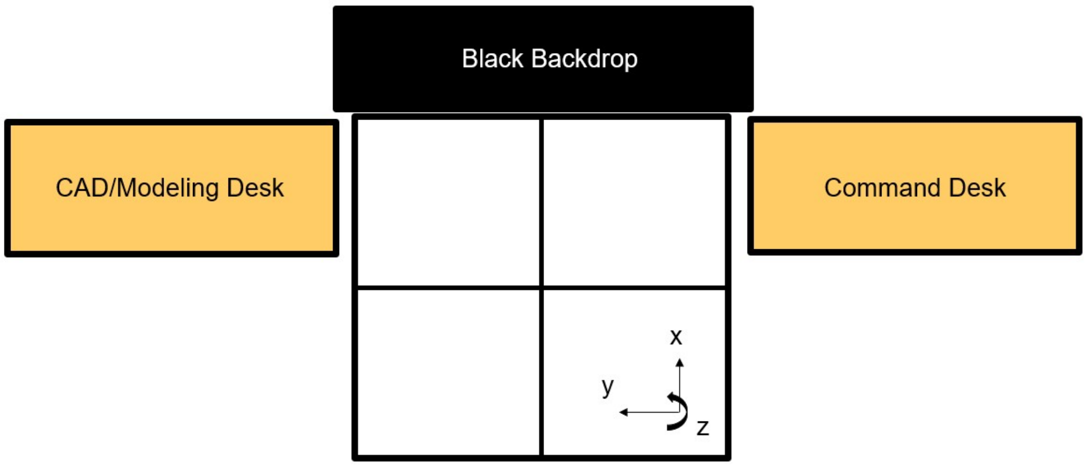


Figure 14: Coordinate System and Room Layout

CASpR are not sufficiently straight to simulate the desired horizontal motion. This is because as CASpR is scaled up in size from another model, the deflection in the horizontal rods increases due to the longer moment arm. The deflection raises the question of what degree of accuracy CASpR can provide.

The goal of these calculations is to find the total deflection in the system at the center of the stack, where the sensors sit. The pipe naming convention is shown in Figure 15.

The steps to find the total deflection are outlined below.

1. Assume the deflections in pipes 1 and 2 are equal at the center.
2. Find the relation between the forces on pipe 1 and pipe 2.
3. Find the reaction forces at the edges of pipe 1 and pipe 2.
4. Apply reaction forces at the edges of pipe 1 and pipe 2 as loads on pipes 3,4,5, and 6.
5. Find deflection on pipes 3,4,5, and 6.
6. Find deflection on pipes 1 and 2.

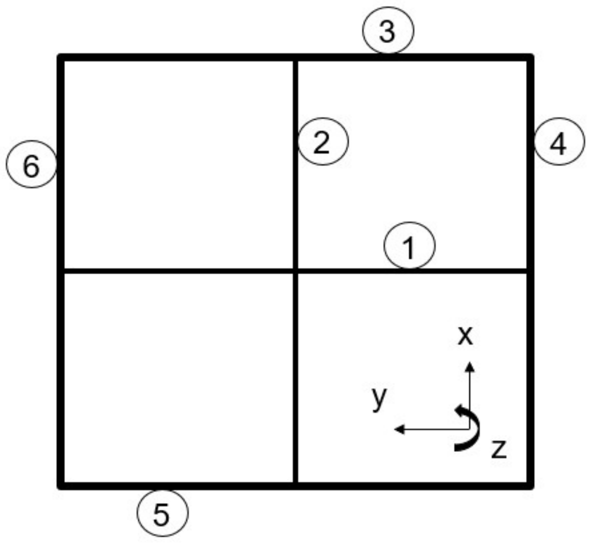


Figure 15: Pipe Naming Convention

3.1.1.1 Deflection Calculations

The 3D printed brackets which hold the pipes are rigid enough that an assumption is made to use the deflection and reaction equations for a fixed-fixed loading scenario as shown in Figure 16.

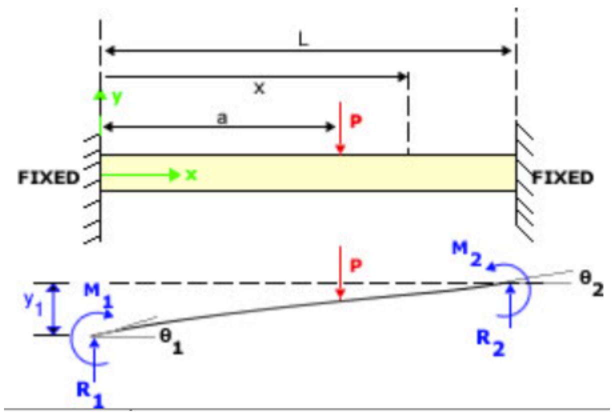


Figure 16: Fixed-Fixed Reaction [8]

The reaction forces on each of the fixed supports are given by Equations 43 and 44, where L is the length of the pipe, P is the load, a is the location of the load along the x axis, and x is the point along the x axis where analysis is being conducted.

$$R_1 = \frac{P}{L^3}(L-a)^2(L+2a) \tag{43}$$

$$R_2 = \frac{Pa^2}{L^3}(3L-2a) \tag{44}$$

The moment on the first support is given by Equation 45.

$$M_1 = \frac{-Pa}{L^2}(L-a) \tag{45}$$

Finally, the total deflection in a pipe is given by Equation 46.

$$\delta = \delta_{initial} + \theta_1 x + \frac{M_1 x^2}{2EI} + \frac{R_1 x^3}{6EI} - \frac{P}{6EI}(x-a)^3 \tag{46}$$

For a fixed-fixed loading scenario, $\theta_1 = 0$. Additionally, $\delta_{initial}$ is defined as the deflection at the edge of the beam. In a fixed-fixed loading scenario, $\delta_{initial} = 0$.

The material used for pipe structure of CASpR is a 304 stainless steel with outer diameter of 1in and inner diameter of 0.87in. This results in the material properties given in Table 3.

Modulus of Elasticity (E)	Area Moment of Inertia (I)
28000 ksi	0.0210 in^4

Table 3: Material Properties [19]

Equations are simplified by assuming that the point of interest is at the load. In terms of the variables given, this means that x = a. This assumption simplifies Equation 46 to Equation 47.

$$\delta = \frac{M_1 x^2}{2EI} + \frac{R_1 x^3}{6EI} \tag{47}$$

When considering the deflection of the two inner pipes, the total deflection is more difficult to compute. This is because the inner pipes "fixed" supports are not perfectly fixed. For example, as the load moves, pipes 4 and 6 may not deflect the same amount, resulting in the fixed supports of pipe 1 lying at different heights. This difference is assumed to be small enough that the fixed-fixed loading condition holds, but large enough to require calculating it in the total deflection.

The total deflection of pipe 1 is the sum of three deflections given by Equation 48.

$$\delta_{t1} = \delta_{i1} + \delta_{p1} + \delta_{o1} \tag{48}$$

δ_{i1} is the initial deflection of one of the outer pipes. For the purposes of calculating the total deflection in pipe 1, it is equivalent to δ_4. δ_{p1} is used to account for the difference in deflection of the two outer pipes and is proportional to that difference. δ_{p1} is given by Equation 49.

$$\delta_{p1} = \frac{\delta_6 - \delta_4}{x} \tag{49}$$

δ_{o1} is the deflection of the inner pipe using the fixed-fixed loading assumption, given by Equation 47. Figure 17 graphically shows the deflection of pipe 1 as the three deflections are summed.

Given the position of the CASpR system, the length of each side, and the total force exerted on system by the sensor stack, the deflection, moments, and reaction forces of each of the pipes can be fully defined in terms of variables P_1 and P_2, the forces exerted on pipes 1 and 2.

Once the system is fully defined in this way, all that remains is to solve for the forces on pipes 1 and 2. This is accomplished by realizing two simple constraints. The first is that the total force is simply the summation of the two forces. This is

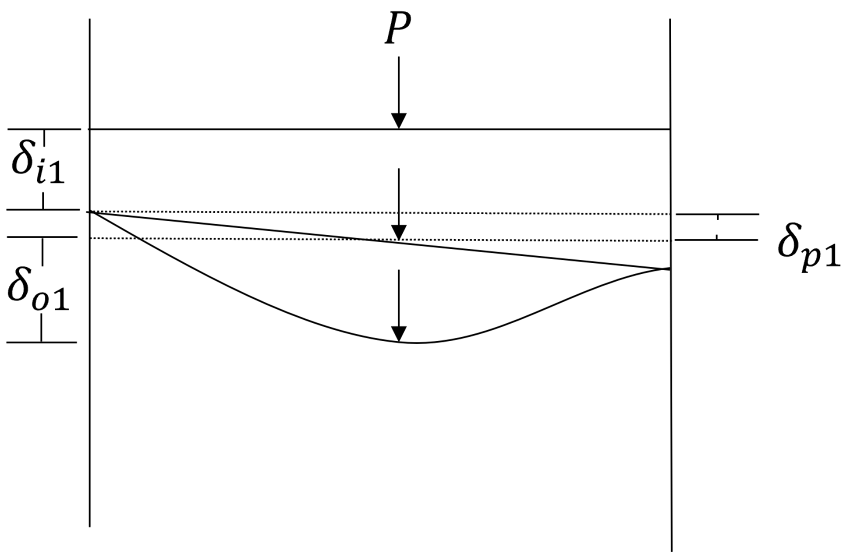


Figure 17: Components of Total Deflection

shown in Equation 50.

$$P_{total} = P_1 + P_2 \tag{50}$$

The second constraint is that the total deflection realized by pipe 1 is equivalent to that of pipe 2 as shown in Equation 51

$$\delta_{total} = \delta_{t1} = \delta_{t2} \tag{51}$$

With these constraints the system can be fully defined without variables and results for the total deflection of the system found.

3.1.1.2 Physical Measurement

Because the CASpR system is an already made physical system, physical measurements can be taken to determine whether the deflections calculated by the above equations and MATrix LABoratory (MATLAB) code are accurate. In order to measure the deflection accurately, taut wire is attached to the edges of the mechanism.

The distance between the wire and the top of the stainless steel pipe is measured at the edges of the pipe and again at the point where the load is applied. By ensuring that the two edge measurements are equal and that the wire is adequately tightened, the deflection is easily measured. By adding the deflections of the outer and connected inner pipe, the total deflection can easily be found. Measurements between the wire and pipe are taken using calipers.

3.2 Hardware

Many different components are used in the development of CASpR. This section discusses all of these hardware components including communication systems, computers, actuators, power supplies, and sensors.

3.2.1 Computation

A standard desktop computer running a Windows Operating System (OS) is used as the primary computing source for running the software required for CASpR to operate. This desktop is termed the flight computer, as it models the computing system which will be used on-board prospective satellites. The role of the flight computer is to run the software which records the spacecraft's position and velocity data and subsequently implements a control algorithm based on the spacecraft's state.

This setup with the desktop computer however, does not allow the testing and verification of a flight worthy computing module on-board the simulated spacecraft. For the purpose of a flight worthy computer, the Jetson TX2i is utilized[35]. When the Jetson is used in lieu of the desktop, the Jetson becomes the flight computer.

3.2.1.1 Arduinos

CASpR is controlled by two Low-Level Controllers (LLCs), one for each of the two simulated satellites. The role of the Low-Level Controller (LLC) is to translate force commands input to the satellite into rotation of the stepper motors. Additionally, the LLCs propagate the Hill-Clohessy-Wiltshire (HCW) dynamics while a simulation is being run. Arduino Megas are used as the computing unit for the LLCs. In order to control the actuation of each machine's stepper motors, a RepRap Arduino Mega Pololu Shield (RAMPS) board is used. The RAMPS board and Arduino are shown in Figure 18.

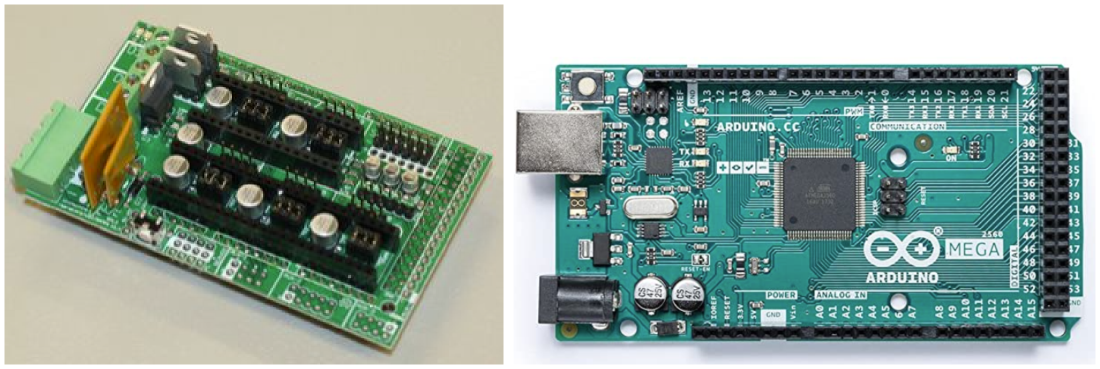


Figure 18: RAMPS Board[9] and Arduino[10]

3.2.1.2 Jetson TX2i

The Jetson TX2i runs on a Linux Ubuntu OS rather than Windows OS. The TX2i communicates through a standard 128 pin connector. The Elroy Carrier is used as an interface through which High-Definition Multimedia Interface (HDMI), Universal Serial Bus (USB), power, and other connections can be established to a Personal Computer (PC). The combination of the Jetson TX2i and the Elroy Carrier Board are shown in Figure 19.

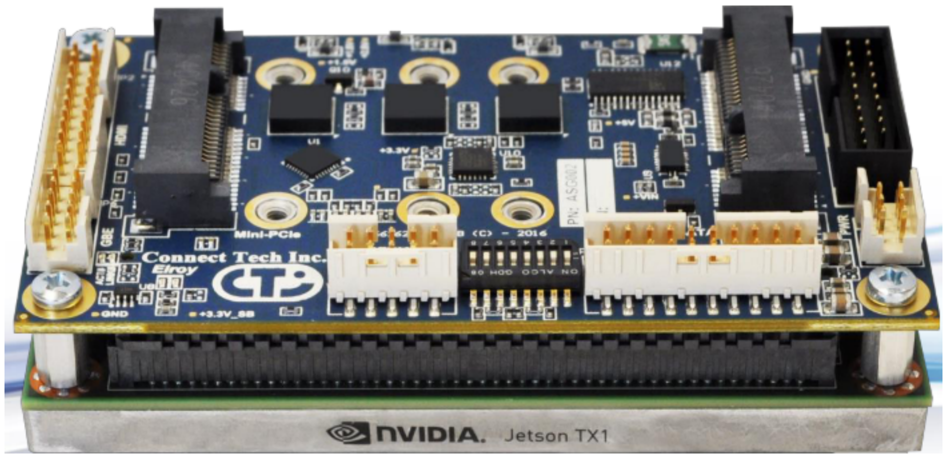


Figure 19: Jetson and Elroy Carrier Board[11]

3.2.2 Communications

Prior to the start of this research, the communications for the CASpR system were run through entirely wired connections as shown in Figure 20.

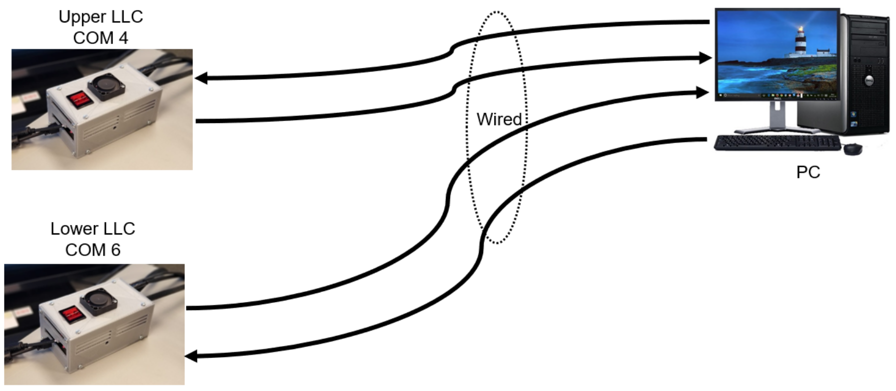


Figure 20: CASpR Old Communications Diagram

The PC, also known as as the flight computer is used to run the driver code for the program. From the flight computer, force commands are sent to the LLCs. The LLCs combine the commanded forces with the forces given by the Keplerian motion of the satellite. The LLCs then send position commands through wired connections to the four stepper motors in order to actuate CASpR.

This research focuses on integrating embedded computing onto the CASpR platform. This is accomplished with a Jetson TX2i and a pair of Digi XBees. The PC remotely accesses the Jetson TX2i through a Secure Shell (SSH) connection. The Jetson then becomes the flight computer. The PC can be any computer which can utilize a SSH connection. The Jetson TX2i then communicates with the LLCs through a pair of Digi XBees[36], a transmitter connected to the TX2i and a receiver connected to the LLC. These XBees operate at a Baud rate of 115200 bits per second. For the time being, the lower CASpR simulated satellite is the only one which is setup to operate from the embedded flight computer; however,the upper LLC can be controlled from the onboard flight computer in the same way. In the early stages of this research actuator position information is fed back into the PC via a wired connection in this updated model which is shown in Figure 21.

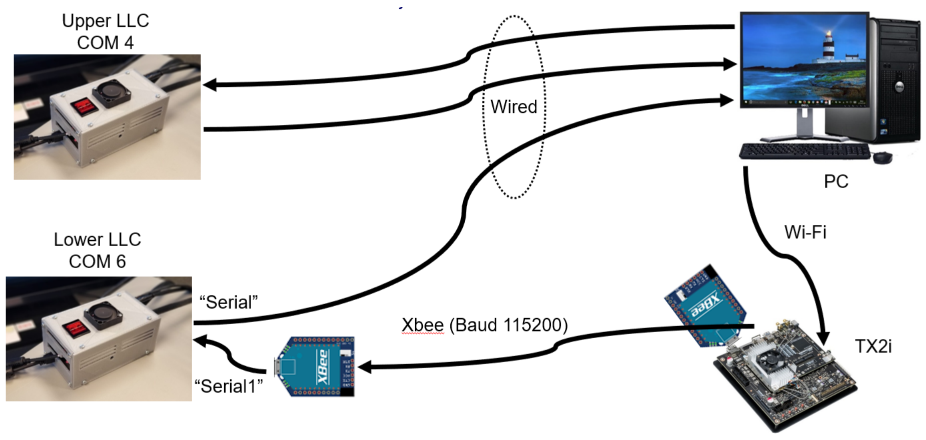


Figure 21: CASpR Communications Diagram

Through the completion of this study another iteration of communications is developed which does not require the PC to store actuator position data. Instead, this data is stored onboard the flight computer. A router is used to set up a "CASpR" network. This router is hard-wired to the PC. The Jetson is connected to the network remotely so that a secure connection is created on the closed network.

The effect is that the flight computer carries more of the mission's computational load and serves as a better test of the system's capabilites. Additionally, this removes all CASpR connections from the PC, so any computer could take its place without modifying the CASpR's hardware. Only the router is required to be physically connected to the PC. Position data remains easily accessible to the user through MobaXterm's Graphical User Interface (GUI). The new communications diagram for the lower LLC is shown in Figure 22.

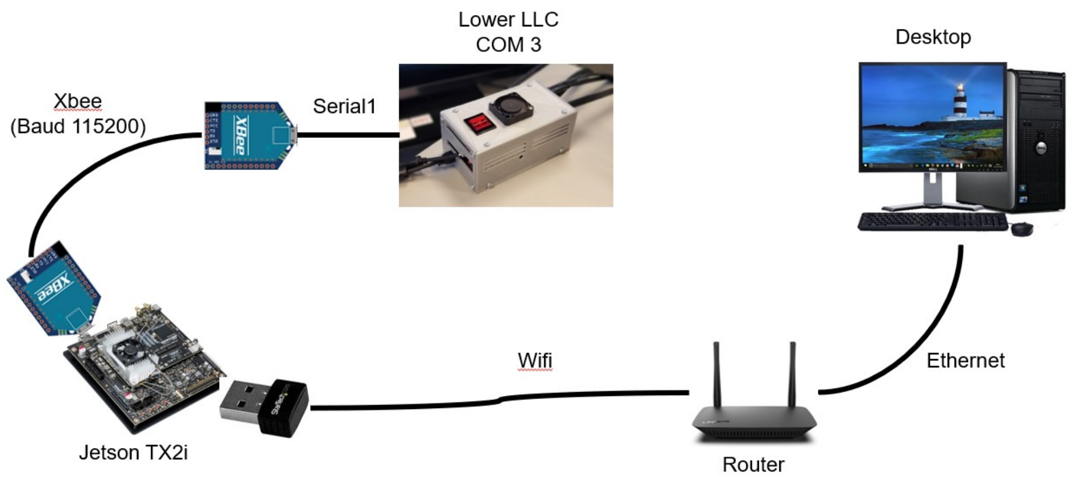


Figure 22: New Communications Diagram

3.2.3 Actuators

The actuators used onboard CASpR are STEPPERONLINE Nema 17 stepper motors. Stepper motors are used for their precision in rotational position. In the Nema 17 stepper motor there are 200 steps per revolution of the motor. This leads to a precision which, though better than conventional motors, is less than may be desired for a testbed with CASpR's requirements. In order to combat this, microstepping is used. Microstepping allows half-steps, quarter-steps, eighth-steps, or sixteenth-steps to be used rather than full steps. This is accomplished by using the A4988 stepper motor driver board and modifying the jumpers on the RAMPS board. The jumpers are labeled as 1 and the A4988 board is labeled as 2 in Figure 23.

Because there are only five positions for the A4988 board, only five stepper mo-

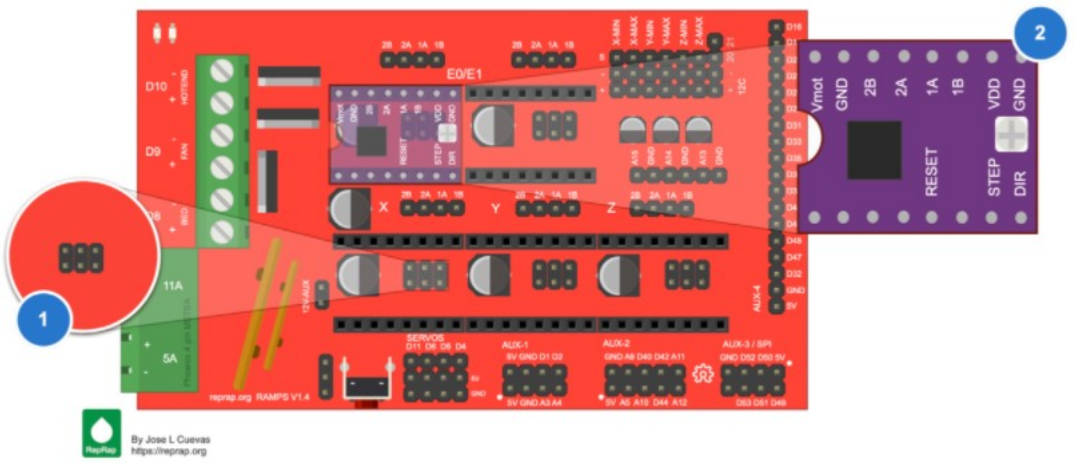
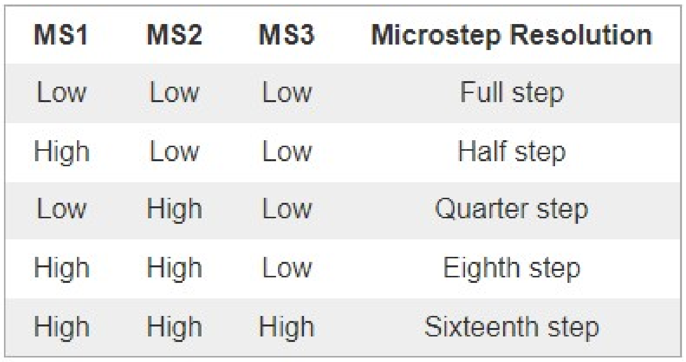


Figure 23: RAMPS Board with Jumpers and A4988 Board [9]

tors may be controlled through a single RAMPS board. This is not an issue with the current design because there are 4 DOFs for each RAMPS board; however, a 6 DOF machine with six stepper motors would require an additional RAMPS board. Although micro-stepping increases the precision of the stepper motors, the trade-off is that the maximum velocity is decreased. For the purposes of standard CASpR testbed operations, the motors which control the x and y translational positions can be set to utilize half-steps, resulting in 400 steps per revolution. The motors which control z translational and z rotational positions are set to utilize quarter-steps because of the decreased velocity requirements in these DOFs. This results in 3200 steps per revolution. The requirements for setting the jumpers is given in Figure 24.



MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Figure 24: Current Adjustment to Microstep with A4988 [12]

3.2.4 Power Supply

Power for CASpR is supplied through multiple sources. Most of the power flowing into CASpR is supplied through standard 120-volt household electricity via the desktop computer. When the LLCs are commanded through a wired connection, power is supplied via the desktop. For the LLCs using XBee enabled wireless communication, power is supplied directly from household electricity.

The Jetson TX2i is used as a flight computer embedded on the simulated spacecraft. If household electricity were used for the flight computer, rotation about the z axis may result in wire wrap from the power cord, hindering the simulated spacecraft's motion. In order to allow the simulated spacecraft to move freely without wire wrap, a battery is used instead. For the purposes of this study, a 4S Lithium Polymer (LiPo) battery is utilized. This is because the Elroy Carrier Board and Jetson TX2i operate with a desired 12 volt input but can range from 9-19.6V[35]. A 4S battery contains four cells which provide a nominal voltage of 3.7V each for a total nominal voltage in a 4S battery of 14.8V. The 4S battery can range in voltage from 14.8-16.8V depending on the capacity; therefore, a 4S battery will always provide an acceptable range of voltages for the TX2i.

The discharge rating is a measure of how fast a battery can be safely discharged without damaging the battery. The maximum discharge rate is found by multiplying the discharge rating by the capacity of the battery as shown in Equation 52.

$$I_{max} = C.rating \times Amp.hours \tag{52}$$

Therefore, a battery with a discharge rating of 35 and capacity of 2200mAh, has a maximum sustained load safely at 77A.

The power requirement for the Jetson TX2i is 20W. With a voltage as low as 9V, the maximum current draw for the TX2i can be calculated using Equation 53 to be

2.22A.

$$I = \frac{P}{V} \tag{53}$$

Because of this, the presupposed discharge rating of 35C is more than enough to meet the system requirements. Given the maximum current, a battery with a capacity of 2200mAh will be able to last for approximately one hour assuming the Jetson is drawing power at its maximum limit. For this reason, the Ready Made Remote Control (RMRC) battery shown in Figure 25 is used.

Figure 25: RMRC Battery

3.2.5 Sensors

RPO is often conducted with vision-based systems. Vision-based navigation onboard a deputy satellite, allows it to gain information about the chief's relative position. For the purposes of this research, a camera is assumed to be the primary sensor on the deputy spacecraft. The camera used is the Intel RealSense D435i, shown in Figure 26.

Figure 26: Intel RealSense Depth Camera D435i

Without an onboard flight computer, wire wrap is a problem for the camera and other sensors. Wire wrap is prevented by having the flight computer on board with the camera and collecting sensor data with the flight computer.

3.2.6 Stack Design

The key hardware components most closely related to the flight computer are combined onto a sensor stack which rides along the simulated satellite. The components include a the depth camera, the battery, the Jetson TX2i, heatsink, and Elroy Carrier Board as well as a USB hub, Wi-Fi adapter, and XBee. A Computer-Aided Design (CAD) model of the sensor stack is shown in two views in Figre 27.

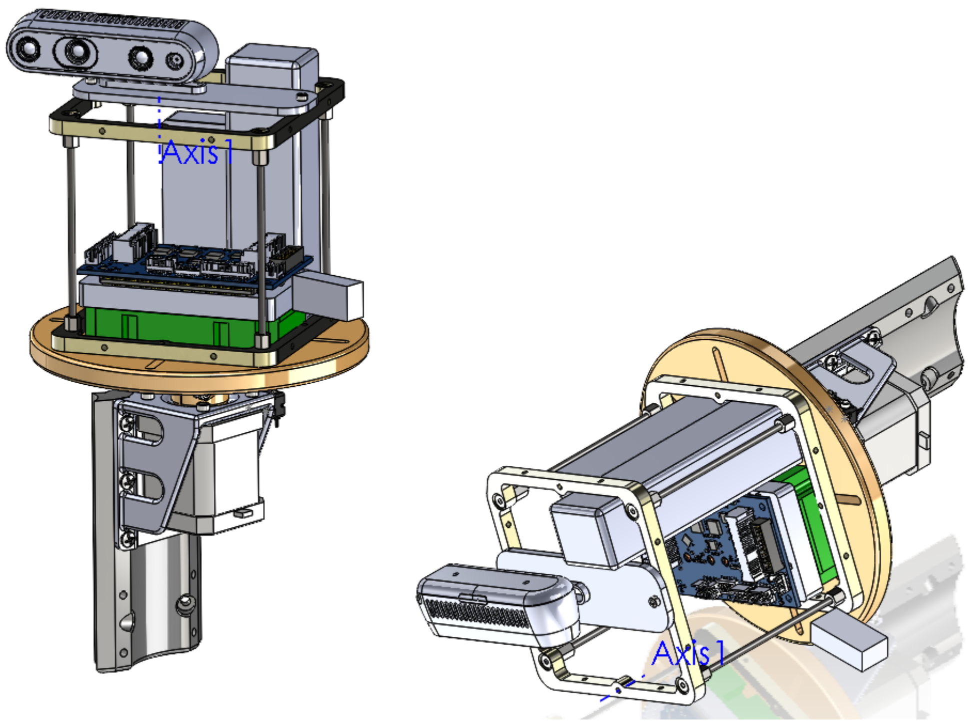


Figure 27: Dual View of Sensor Stack

3.3 Software

Several software programs are used in the development of CASpR. This section discusses all of these programs including C++, Python, MobaXTerm, and MATLAB and the function of the code developed in each. The code used is available in Appendix ??.

3.3.1 C++

C++ is used in programming the LLCs and their interactions with the actuators. The software used in writing the C++ code for this study is Platform.IO, but many different Integrated Development Environment (IDE)s including the Arduino IDE can be used.

One of the primary functions of the code in the LLCs is to propagate the HCW

Equations of Motion (EOMs). Without any commanded force, the LLC will still control the actuators to move the simulated satellite in Keplerian-restricted relative motion. This code also uses the stepper motors' rotational displacement to determine the simulated spacecraft's position relative to its starting position. Until a motion capture system is acquired, this data is generally used as the system's truth data. This truth data is passed back to the flight computer and may be viewed through the Python-developed GUI.

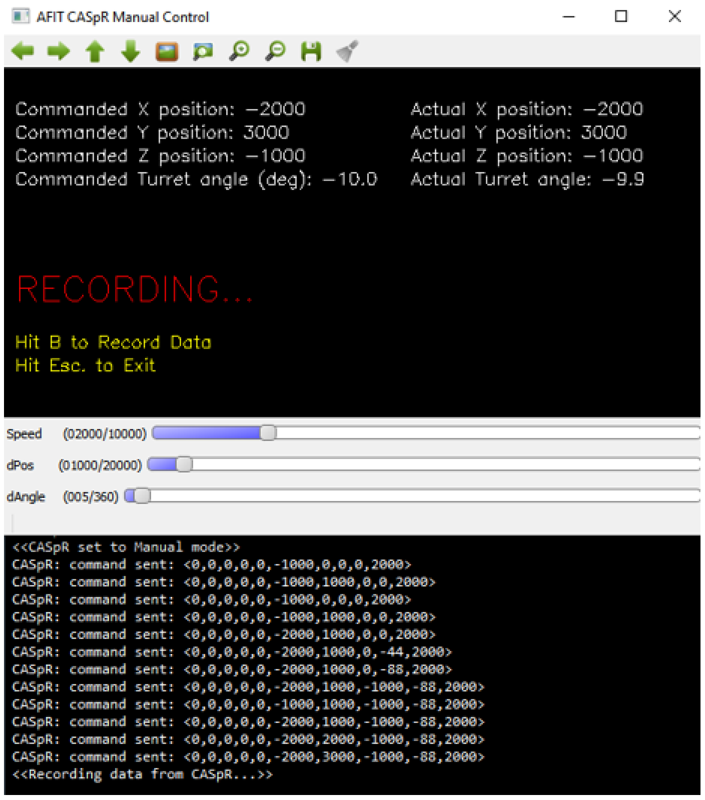
In standard testbed use cases, the user does not interact with the C++ on the LLCs. There are some instances where better understanding and modifying the C++ code may be beneficial. The initial state (position and velocity) of the spacecraft is defined in the C++ code. This initial state is hard coded as a zero vector; therefore, deputy is said have the same state as the chief spacecraft. Because of this, the state will always remain a zero vector unless some additional force input is commanded.

It may be desirable to have a non-zero initial state for some use-cases. One example of this is placing the deputy into a Natural Motion Circumnavigation (NMC) trajectory by modifying the initial velocity in the x direction. This allows the C++ code on the LLC to easily be tested. If the resulting trajectory is a 2x1 ellipse, the code is validated.

3.3.2 Python

Python is used in programming the Guidance, Navigation, and Control (GNC) controller, manually controlling CASpR, building a GUI, and sending commands to the LLCs. The Python code is run by the flight computer regardless of whether it is on-board CASpR or a desktop. For the purposes of this study, the Python code is implemented and run through Spyder, though other IDEs such as Pycharm, Visual Studio (VS) Code, and many others may be used instead.

When running the Python code, a GUI is presented to the user. The program begins in a manual control mode in which the user can control the position of the simulated spacecraft within CASpR. This GUI allows the user to modify the speed of the actuators, change in position, and change in angle for a commanded movement. This GUI is shown in Figure 28.



```
Commanded X position: −2000          Actual X position: −2000
Commanded Y position: 3000           Actual Y position: 3000
Commanded Z position: −1000          Actual Z position: −1000
Commanded Turret angle (deg): −10.0  Actual Turret angle: −9.9

RECORDING...

Hit B to Record Data
Hit Esc. to Exit
```

```
Speed   (02000/10000)
dPos    (01000/20000)
dAngle  (005/360)
```

```
<<CASpR set to Manual mode>>
CASpR: command sent: <0,0,0,0,0,-1000,0,0,0,2000>
CASpR: command sent: <0,0,0,0,0,-1000,1000,0,0,2000>
CASpR: command sent: <0,0,0,0,0,-1000,0,0,0,2000>
CASpR: command sent: <0,0,0,0,0,-1000,1000,0,0,2000>
CASpR: command sent: <0,0,0,0,0,-2000,1000,0,0,2000>
CASpR: command sent: <0,0,0,0,0,-2000,1000,0,-44,2000>
CASpR: command sent: <0,0,0,0,0,-2000,1000,0,-88,2000>
CASpR: command sent: <0,0,0,0,0,-2000,1000,-1000,-88,2000>
CASpR: command sent: <0,0,0,0,0,-1000,1000,-1000,-88,2000>
CASpR: command sent: <0,0,0,0,0,-2000,1000,-1000,-88,2000>
CASpR: command sent: <0,0,0,0,0,-2000,2000,-1000,-88,2000>
CASpR: command sent: <0,0,0,0,0,-2000,3000,-1000,-88,2000>
<<Recording data from CASpR...>>
```

Figure 28: GUI for Manual Control of CASpR

The manual control of CASpR is commanded via the keyboard. The controls are shown in Figure 29.

From the manual control setting, GNC mode can be reached by pressing G on the keyboard. In GNC mode, a vast range of research may be accomplished. GNC mode requires that the user has implemented a control algorithm of some kind within the Python code. For the purposes of this research, the file "*CASpR_Manual_Control.py*" is used as the driver code. Manual control is entered into after running the code. From

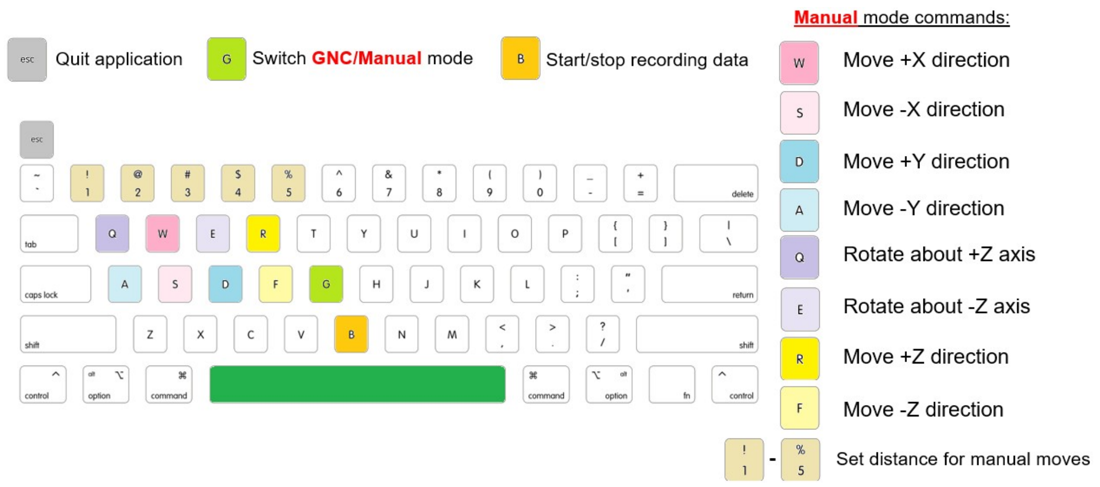


Figure 29: Keyboard Controls

here, GNC mode may be entered as described above by pressing G. The GNC mode driver code uses algorithms which may include targeting or active control to modify the spacecraft's trajectory. The output from the Python code is contained in the command string which contains forces in the x, y, and z directions as well as a torque in the z direction. This information is then sent to the LLCs which propagate the dynamics of the system.

Python is also used to record the data which is gathered throughout a simulation. The data may be collected in manual mode by pressing B on the keyboard or is automatically collected for the user in GNC mode. After the completion of the simulation, the raw data is output into a file named dataCASpR.mat. This file carries all of the information which is passed through the state string. This information includes CASpR's positions in the x, y, and z directions as well as z rotational position. The initial positions and velocities are also contained as references as well as a scale factor.

For the purposes of this study, an algorithm is developed which uses a targeting maneuver, propagation without any control, and active control implemented through a Linear Quadratic Regulator (LQR) controller. This allows the user to avoid modifying the C++ code for most use cases and instead use the targeting maneuver to place the deputy spacecraft at a desired "initial" state before using the LQR controller to maintain a desired trajectory.

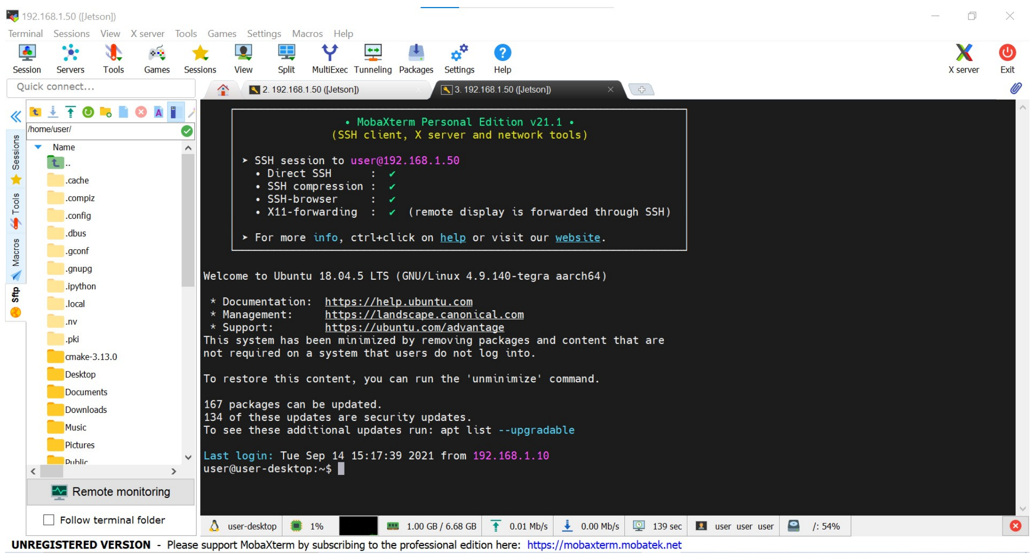
3.3.3 MobaXTerm

MobaXTerm is a Windows application which offers a wide range of remote networking tools including SSH and others. For the purposes of this study, only the SSH tool is used. SSH allows a user to remotely access a computer securely across an unsecured network. For a connection to be established, the remotely accessed computer's Private Internet Protocol version 4 (IPV4) address, username, and password must be known. This information can be stored in MobaXTerm for easy multiple instances of accessing the same computer.

Through SSH, commands may be given to the remote computer via the command line. A file tree also allows the user to easily view the files on the remotely operated computer. A program or file can easily be run through the command line. For example, CASpR's manual control file can be run through the command line with the following command, "*python3 /home/user/Documents/CASpR_Manual_Control.py*". A screenshot of the MobaXTerm interface while connected remotely to CASpR is shown in Figure 30.

3.3.4 MATLAB

MATLAB is used in reading, downsampling, and analyzing the data and results of the CASpR testbed. The data which is recorded and collected from the Python code is compiled into three separate packets. The first consists of the command strings, the second is composed of the position strings, and the last is simply all of the time steps. This data is only for the period that is in GNC mode or otherwise recorded.


192.168.1.50 ([Jetson])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. 192.168.1.50 ([Jetson])
3. 192.168.1.50 ([Jetson])
/home/user/
Name
..
.cache
.compiz
.config
.dbus
.gconf
.gnupg
.ipython
.local
.nv
.pki
cmake-3.13.0
Desktop
Documents
Downloads
Music
Pictures
Public
Remote monitoring
Follow terminal folder
user-desktop 1% 1.00 GB / 6.68 GB 0.01 Mb/s 0.00 Mb/s 139 sec user user user /: 54%

```
          • MobaXterm Personal Edition v21.1 •
        (SSH client, X server and network tools)

 ➤ SSH session to user@192.168.1.50
   • Direct SSH      :  ✔
   • SSH compression :  ✔
   • SSH-browser     :  ✔
   • X11-forwarding  :  ✔  (remote display is forwarded through SSH)

 ➤ For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.9.140-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

167 packages can be updated.
134 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Tue Sep 14 15:17:39 2021 from 192.168.1.10
user@user-desktop:~$
```

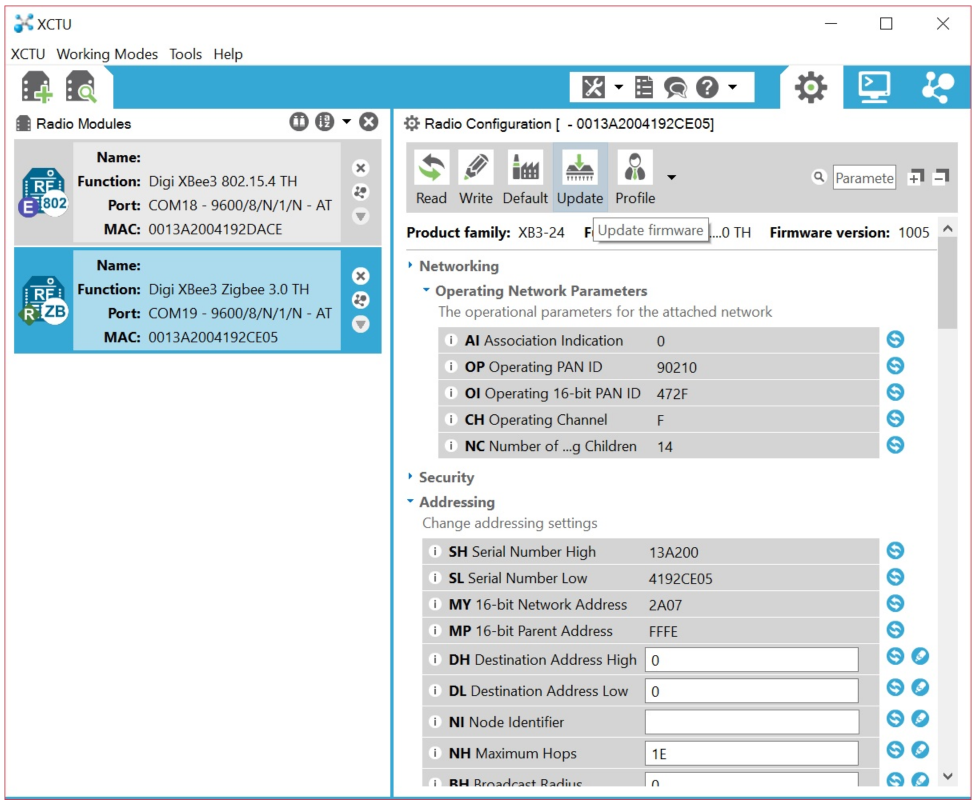
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Figure 30: MobaXTerm Interface

The data from the position packet contains: [xPosAct, yPosAct, zPosAct, thetaAct, x0, xDot0, y0, yDot0, z0, zDot0, theta0, thetaDot0, dtMicro]. This data is then used to plot the trajectory of the simulated spacecraft throughout the recorded period. For the purposes of this study, this is the method which is used to visually compare expected trajectories with actual trajectories on CASpR. Although the precision of a visual comparison is low, it is adequate for verifying the GNC algorithms.

3.3.5 XCTU

The XBee Configuration & Test Utility (XCTU) software provides the capability of configuring the XBees to communicate with each other as well as the LLCs. XCTU allows users to easily specify the baud rates at which to transmit or receive data. For this study, a baud rate of 115200 is used. This means that up to 115200 bits per second may be transmitted. XCTU also allows users to easily see which devices are connected to each comport. This is useful when switching between wired connections and the wireless connections offered by the XBees. An example of the XCTU GUI is shown in Figure 31.


XCTU
XCTU Working Modes Tools Help
Radio Modules
Radio Configuration [- 0013A2004192CE05]
Read Write Default Update Profile
Update firmware
Paramete

Name:
Function: Digi XBee3 802.15.4 TH
Port: COM18 - 9600/8/N/1/N - AT
MAC: 0013A2004192DACE

Name:
Function: Digi XBee3 Zigbee 3.0 TH
Port: COM19 - 9600/8/N/1/N - AT
MAC: 0013A2004192CE05

Product family: XB3-24 F ...0 TH Firmware version: 1005

Networking

Operating Network Parameters

The operational parameters for the attached network

AI Association Indication	0
OP Operating PAN ID	90210
OI Operating 16-bit PAN ID	472F
CH Operating Channel	F
NC Number of ...g Children	14

Security

Addressing

Change addressing settings

SH Serial Number High	13A200
SL Serial Number Low	4192CE05
MY 16-bit Network Address	2A07
MP 16-bit Parent Address	FFFE
DH Destination Address High	0
DL Destination Address Low	0
NI Node Identifier	
NH Maximum Hops	1E
BH Broadcast Radius	0

Figure 31: XCTU GUI

3.4 Scaling

CASpR as a testbed operates as a scaled model of the satellites which it simulates. This scaling is necessarily positional. Additionally, the model can be temporally scaled as well.

3.4.1 Position

The position of the model spacecraft in CASpR may be scaled differently in each axis. However, the x and y axes are scaled equally for standard operations on CASpR so that the shape in the xy plane is preserved. Because the z component is decoupled in the HCW EOMs, the scaling of it is also decoupled without much loss of understanding of the trajectory's true shape.

The stepper motors then operate at a rate of 200 steps per revolution. This can be broken down into 2 microsteps per full step for a total of 400 microsteps per revolution due to the RAMPS board. The circumference of the idler which is mounted on the stepper motor is composed of 16 teeth. A single tooth of the rubber belt has a pitch, the distance between teeth, of 2mm. The CASpR cage is 8 feet by 8 feet; however, because the sensor stack is not a point load, the workable simulation space is approximately 7 feet 1 inch by 7 feet 1 inch or 2160mm by 2160mm.

The natural scaling of CASpR with no scale factor is one microstep per kilometer of simulated motion. This can be scaled simply by using a scale factor to increase the force commands which are sent to the LLCs. Scaling is generally desirable as it allows higher resolution while keeping the simulated spacecraft within the testbed cage. As a reference for users of CASpR, there are approximately 27000 microsteps of workable area within each horizontal axis of CASpR. The calculations to determine the positional scaling are shown in Equations 54 and 55.

$$\left(\frac{2microsteps}{step}\right)\left(\frac{200steps}{rev}\right)\left(\frac{rev}{16teeth}\right)\left(\frac{teeth}{2mm}\right)=\frac{12.5microsteps}{mm} \tag{54}$$

$$\left(\frac{12.5microsteps}{mm}\right)\left(\frac{2160mm}{CASpRside}\right)=\frac{27000microsteps}{CASpRside} \tag{55}$$

3.4.2 Time

Time is not scaled for general CASpR testbed operations; however, it can be scaled for the purposes of rapid testing. The time scale can be pseudo-modified by changing the radius of the orbit which is defined in the C++ code on the LLCs. The orbit can be modified to a very small orbit. Although clearly impossible for real-world satellites, a simulated satellite can operate at any radius, even one which lies within the radius of earth. For an orbit with a desired period, the semi-major axis, which is equivalent to the radius for a circular orbit, is given by Equation 56.

$$a = \sqrt[3]{\frac{\mu}{\left(\frac{2\pi}{Period}\right)^2}} \tag{56}$$

For a thirty second period, the radius of the orbit is then approximately 208.7 kilometers. This orbital radius is commonly used in CASpR testing for rapid results.

3.5 Tests and Expected Results

A series of incremental tests are conducted in order to build and validate the CASpR system and its associated hardware and software. First, tests only considering natural motion are conducted. Next, a controller is added to burn into a NMC trajectory. These tests are accomplished in MATLAB only in this section. The tests are then accomplished on CASpR and the results documented in Chapter IV.

3.5.1 Natural Motion

The first motion test for the CASpR testbed requires the use of the LLCs C++ code only. The initial state is set within the C++ code to be a zero vector. No outside control is added and the trajectory of the deputy spacecraft is expected to remain at the origin. Although this result is expected to be uninteresting, the test serves as a baseline to validate that the dynamics on the LLCs are modeled correctly.

Next, an initial state is designed such that given Keplerian motion, the spacecraft will remain in a NMC trajectory. This test is implemented by modifying the reset() function of the C++ code. The initial state is set to be a zero vector with the exception of the velocity in the x-direction. Any initial velocity in purely the x direction will set a satellite into a NMC.

This satisfies the drift term of the HCW EOMs from Equation 19. The expected trajectory of this is a 2x1 ellipse in the x-y plane. This trajectory is not centered about the chief and will instead intercept the chief with each orbital period.

3.5.2 Burn into NMC

After validating the Keplerian model built in the C++ code only, thrusting is introduced to the system and implemented in the Python code. The first scenario with thrusting has the deputy spacecraft's trajectory starting with an initial state defined by the zero vector. The spacecraft is then commanded to instantaneously thrust in the positive x-direction. This is expected to place the spacecraft into a trajectory with the same shape as the previously discussed simulation. Rather than an input desired velocity, this test is based off of an input desired force due to the differences in the code on the LLC and flight computer. The desired force is determined onboard the flight computer and then passed to the LLC. Because the thrusting is accomplished solely in the Python code onboard the flight computer, this serves as a test for the communications between the flight computer and the LLC.

Next, because thrusters cannot in reality provide instantaneous changes in velocity, a time frame over which the control should be commanded is specified. This is accomplished by simply dividing the desired force commanded by the period of time. This period of time will likely be on the order of approximately ten seconds for most maneuvers. The non-instantaneous burn into a NMC trajectory is expected to provide results extremely similar to instantaneous, given the length of an orbital period. In practice however, due to the discrete time steps of CASpR's software loops, it is possible that the total amount of force imparted to the system may be slightly larger or smaller than the desired force which would be imparted in an instantaneous burn. This difference in total force commanded may or may not be negligible. This provides a more realistic test case than a purely virtual simulation would provide since true space operations are not instantaneous.

Often in RPO it is of interest to enter into a NMC which is centered on the chief or otherwise not intercepting the chief. This can be for the purposes of inspection, communications, or scientific research. To accomplish this, multiple burns are required. The first burn is intended to place the deputy in a specific position and the second to give the deputy a specific velocity which will allow it to enter the NMC or other trajectory. Using the targeting equations discussed in section 2.4.1, the first burn is used to set the initial position of the spacecraft in the NMC to be purely in the +X direction. The second burn is intended to set the velocity of the spacecraft so that it enters into the NMC with no drift.

With targeting, the spacecraft is given a timeframe over which it must rendezvous with the desired position. The spacecraft is then commanded to complete the second thrusting maneuver to set it into the NMC. A wait time of a single orbital period is then given to allow the trajectory of the spacecraft to propagate. The orbit is expected to close with the initial point on the NMC to validate the implementation of the targeting algorithm.

3.5.3 Active Control

After validating the model with spacecraft thrusting but no control algorithm, a control algorithm is introduced to the system and implemented in the Python code. Optimal control is implemented through a LQR controller.

For the application of the controller in testing, the system is assumed to be circular such that the HCW equations apply. A satellite following the HCW model is fully controllable with only along-track and cross-track control components. As a result of this, the number of total thrusters required is lower than a system not restricted by the HCW assumptions. The A and B matrices of the state-space HCW equations are given by equations 57 and 58.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \tag{57}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0_{3x3}} \\ \mathbf{I_{3x3}} \end{bmatrix} \tag{58}$$

An LQR controller simply utilizes a gain to control the system to a desired state. This gain is determined by solving the Riccati equation given by equation 59 for the stabilizing symmetric positive definite matrix P, where parameters Q and R are user defined weighting matrices [28].

$$\mathbf{A}^T\mathbf{P} + \mathbf{PA} + \mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} = 0 \tag{59}$$

Once P is determined, the gain for the system can also be calculated with equation 60.

$$K = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{60}$$

The error in the system is simply the difference in the current state and the desired state. The desired state in this case is given by the propagated unperturbed HCW model as calculated by the flight computer. Therefore, the error is given by equation 61 where x is the current state and x_{des} is the desired state.

$$\mathbf{e} = \mathbf{x} - \mathbf{x_{des}} \tag{61}$$

The control of the system is simply the gain multiplied by the error in the system as shown in equation 62.

$$\mathbf{u} = -K\mathbf{e} \tag{62}$$

The controllability matrix is given by equation 63

$$\mathbf{M}_c = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \mathbf{A}^3\mathbf{B}, \mathbf{A}^4\mathbf{B}, \mathbf{A}^5\mathbf{B}] \tag{63}$$

Putting this matrix into reduced row echelon form reveals that the rank of the matrix is 6, indicating that the system is fully controllable in-plane, although cross-track control is needed for full 3-dimensional controllability. Even reducing the B matrix to only allow control in the in-track direction, the system remains fully controllable.

The first controlled scenario has the deputy spacecraft's trajectory starting with an initial state defined by the zero vector and using two non-impulsive burns to enter into a NMC trajectory centered on the chief. The controller is tasked with modifying the radius of the spacecraft in a scaled version of the same NMC.

IV. Results and Analysis

Chapter III outlined the methodology which is used to develop the embedded computing capability of Control and Autonomy Space proximity Robot (CASpR). The hardware and software used in development of CASpR are reviewed in detail. Additionally, tests are developed which are used as baselines for determining the validity of CASpR as a testbed.

Chapter IV discusses the following:

1. Results of deflection simulation and testing.
2. Results of simulated trajectories.
3. Potential errors which may be encountered while using CASpR.
4. Limitations of both hardware and software.
5. Scaling a trajectory so that it fits the physical constraints of CASpR.

4.1 Deflection of CASpR

A MATrix LABoratory (MATLAB) code is developed to compute the range of possible deflections given a range of input lengths, x positions, and y positions. The driver code and its associated functions can be found in the appendix for reference. Although the code is robust enough to handle any CASpR side length and a load at any position, for the purposes of this study, the current side length of 96 inches is used. Because the structure of CASpR is square in the x-y, there is rotational symmetry about the z axis. This means that with the assumption of a square system only one quarter of the system needs to be calculated for the deflection in all four quadrants to be known.

The simulation is run in a square configuration with a side length of 96 inches and weight of 6.8 pounds. This weight is the weight of the main platform of CASpR without any sensor stack. The qualitative results at 1 foot intervals are calculated and plotted in the heatmap shown in Figure 32.

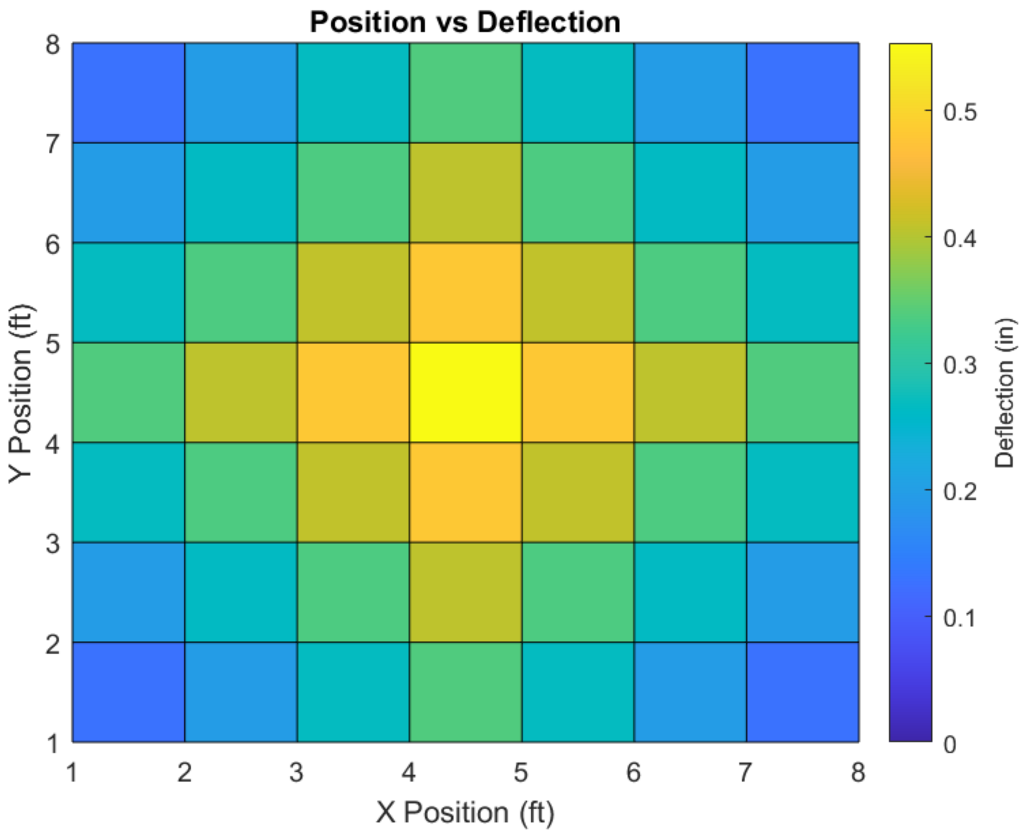


Figure 32: Heatmap of Deflections

As expected, the heatmap reveals that the peak of deflection occurs when the sensor stack is at the center of CASpR's range and the deflection decreases near the edges. The deflection at the sensor stack is notably small at the corners. The quantitative results of this deflection study are found in Table 4.

Table 4, shows that for the current system the maximum deflection is approximately 0.55 inches. This result is approximately what is seen physically in CASpR, thus lending validity to the calculations performed. This deflection is approximately

Table 4: Quantitative Results

x\y (feet)	0	1	2	3	4	5	6	7	8
0	0	0.064	0.132	0.204	0.274	0.204	0.132	0.064	0
1	0.064	0.128	0.196	0.268	0.337	0.268	0.196	0.128	0.064
2	0.132	0.196	0.264	0.336	0.406	0.336	0.264	0.196	0.132
3	0.204	0.268	0.336	0.408	0.480	0.408	0.336	0.268	0.204
4	0.274	0.337	0.406	0.480	0.553	0.480	0.406	0.337	0.274
5	0.204	0.268	0.336	0.408	0.480	0.408	0.336	0.268	0.204
6	0.132	0.196	0.264	0.336	0.406	0.336	0.264	0.196	0.132
7	0.064	0.128	0.196	0.268	0.337	0.268	0.196	0.128	0.064
8	0	0.064	0.132	0.204	0.274	0.204	0.132	0.064	0

*All deflections are in inches

0.57% of the total side length.

The physical testing resulted in a total deflection of 0.17 inches in the outer pipe and 0.36 inches in the inner pipe for a total deflection of 0.53 inches. Given the assumptions that the string is perfectly tight and may have some small amount of deflection and each side of the string may not be exactly the same height above the pipe, as well as the small inaccuracies inherent in measurements, 0.53 inches seems to be well within a reasonable range of the predicted deflection value. This result validates the calculations conducted.

The sensor stack used in this study weighs 1.87 pounds. An analysis of the standard CASpR system with this sensor stack yields a maximum total deflection of 0.57 inches.

Another calculation with a range of tested total loads is performed to understand the degree to which the weight of the sensor stack impacts the total deflection. The results of this calculation are shown in Figure 33.

The deflection of the system varies very little by the total load being applied at the center of CASpR. This means that the weight of new sensor stacks is fairly negligible in the deflection of the system. If deflection is determined to be too great, rather than focusing on lightening the load, emphasis should be placed on maximizing the

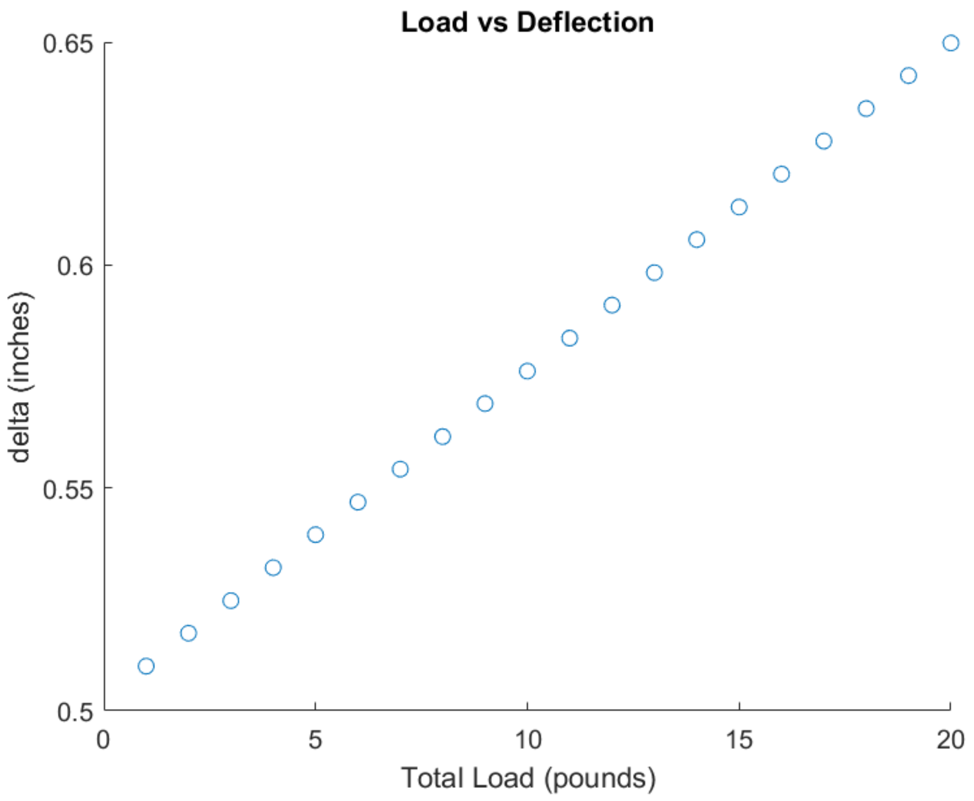


Figure 33: CASpR Total Load vs Deflection

stiffness of the joints. This may be accomplished by using tension rods inside the structural pipes.

The degree of accuracy in this system is directly a result of the deflection of the horizontal tubing which makes up CASpR. Although the accuracy required for the CASpR system is not yet known, these calculations can be completed for any CASpR-style system which may be considered in the future when the true location of CASpR is known based on a motion capture system. This means that materials, tube dimensions, and other parameters can be driven by the size and weight requirements of the system.

4.2 Sensor Stack

The sensor stack is designed to be easily exchangeable so that unique stacks can be made for individual research projects and easily swapped onto and off of CASpR. The sensor stack used for this research is shown onboard CASpR in Figure 34.

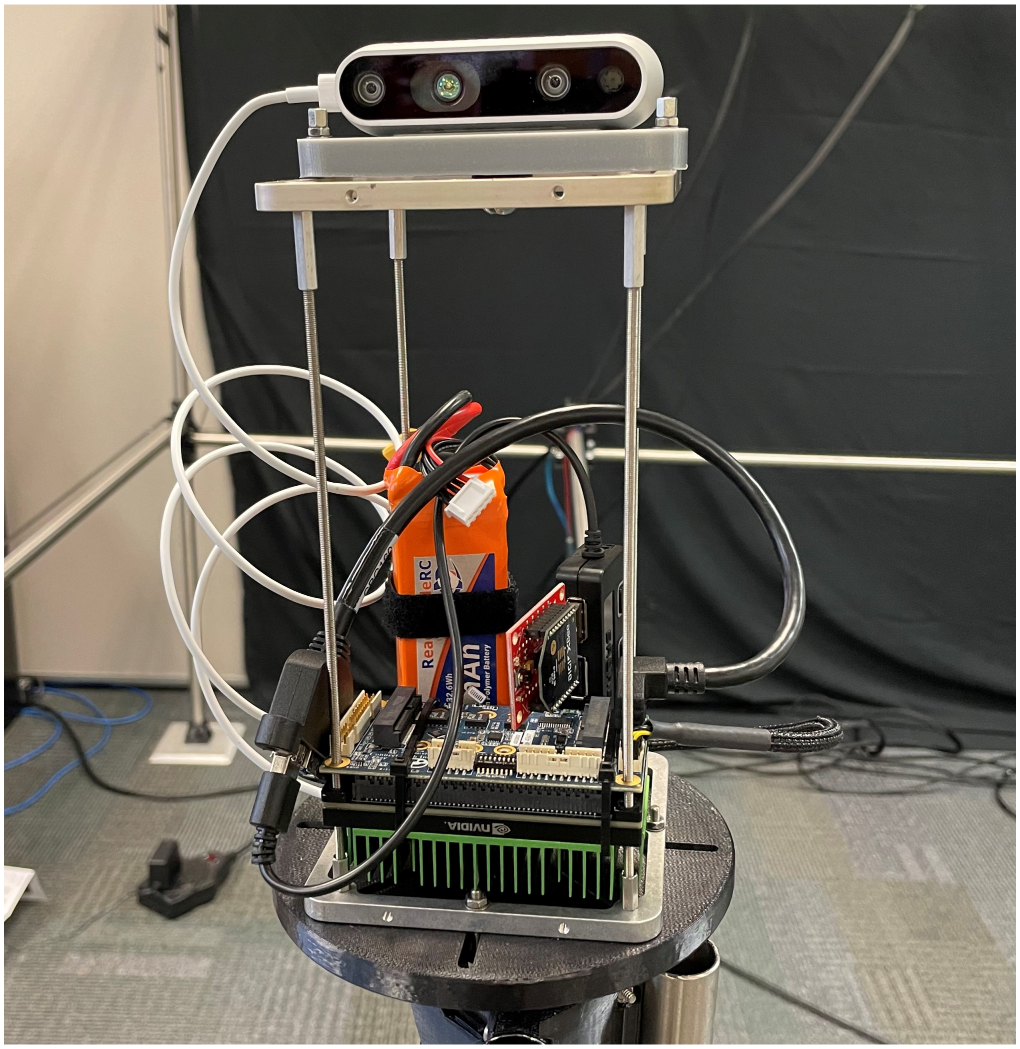


Figure 34: Attached Sensor Stack

4.3 Characterization of CASpR

CASpR may begin to be characterized by developing a series of tests and comparing the results with the results which are expected. The orbits simulated include trajectories achieved through natural motion as well as burning into Natural Motion Circumnavigation (NMC), or using active control to achieve a desired trajectory. Next, some limitations of the actuators are explored. This includes errors in position caused by issues with delay in the system as well as the pointing accuracy limitations inherent in the stepper motors. Next, software limitations including truncated positional commands. Some unexpected issues with the development and implementation of CASpR are also discussed. Finally, methods for scaling both time and position are discussed.

4.3.1 Simulated Orbits

As discussed in Section 3.5, a series of tests are conducted to validate the dynamics, targeting algorithm, and Linear Quadratic Regulator (LQR) controller developed on CASpR. The first of these tests considers a spacecraft with a zero vector initial state and no external forces. The Low-Level Controller (LLC) is then modified to provide an initial velocity in the x-direction to create a NMC. This same test is then conducted by modifying the Python code to target the NMC rather than starting in it. Next, the targeting algorithm is used to enter into a NMC which is centered on the chief, requiring two burns. Finally, active control with a LQR controller is tested by targeting into a NMC and then utilizing a LQR control to actively control into a different NMC.

The trajectories simulated on CASpR are scaled down to be completed within the 8 foot square frame. The results are then scaled back up with the same scaling factor, resulting in simple side-by-side comparisons with the expected results.

4.3.1.1 Natural Motion

The first natural motion test is the most basic. The initial state of the spacecraft is given by the zero vector. No control or external perturbations are used. The resulting trajectory is shown in Figure 35 where the starting and ending points of the trajectory are marked by a red circle and yellow "X" respectively.

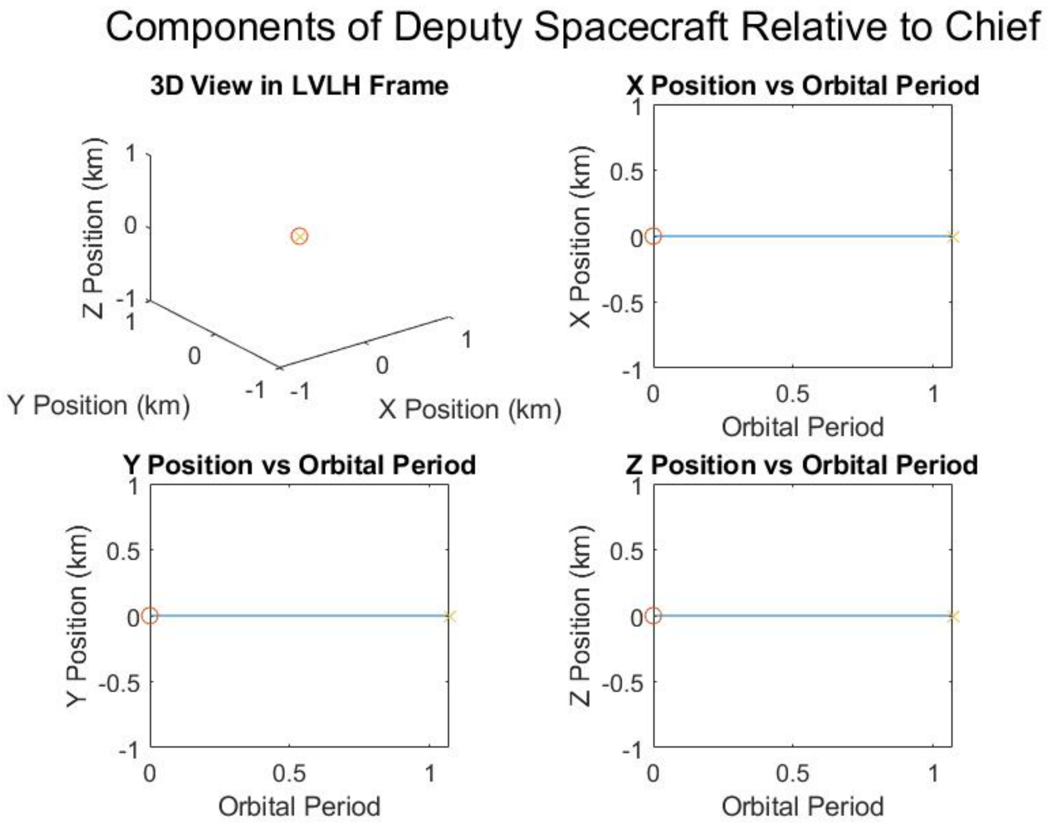


Figure 35: Coincident Chief and Deputy

As expected based on the Hill-Clohessy-Wiltshire (HCW) Equations of Motion (EOMs), the deputy satellite is coincident with the chief spacecraft for the duration of the simulation. Using the HCW assumptions, when no outside forces act on a satellite, its position relative to another satellite at the same altitude is constant. This result begins to validate the HCW dynamics model used on the Low-Level Controllers (LLCs).

Next, natural motion is used to move the deputy spacecraft into a NMC which intersects with the chief. This is accomplished by modifying the C++ so that an initial velocity in the x direction of 1.0472 km/s for an orbit with a period of 30 seconds, and an orbital semi-major axis of 208.6764866 km. The period time of 30 seconds is chosen to allow for more rapid testing since a true Low Earth Orbit (LEO) orbit takes approximately 90 minutes. This yields a NMC semi-minor axis of 5 km and a semi-major axis of 10 km. The initial state vector is given by equation 64.

$$\mathbf{X} = [0, 0, 0, 1.0472, 0, 0]^T \tag{64}$$

The expected and actual from CASpR results are shown in Figures 36 and 37 respectively.

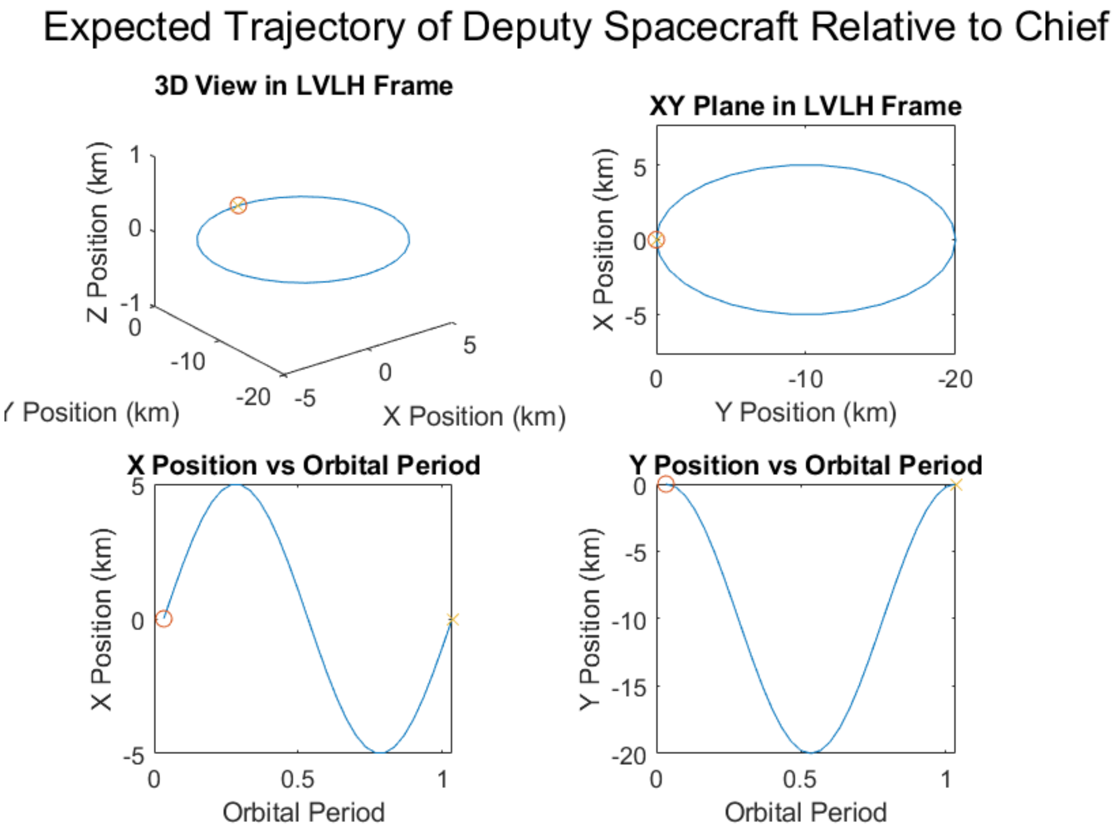


Figure 36: Expected Trajectory Given Initial X Velocity in C++

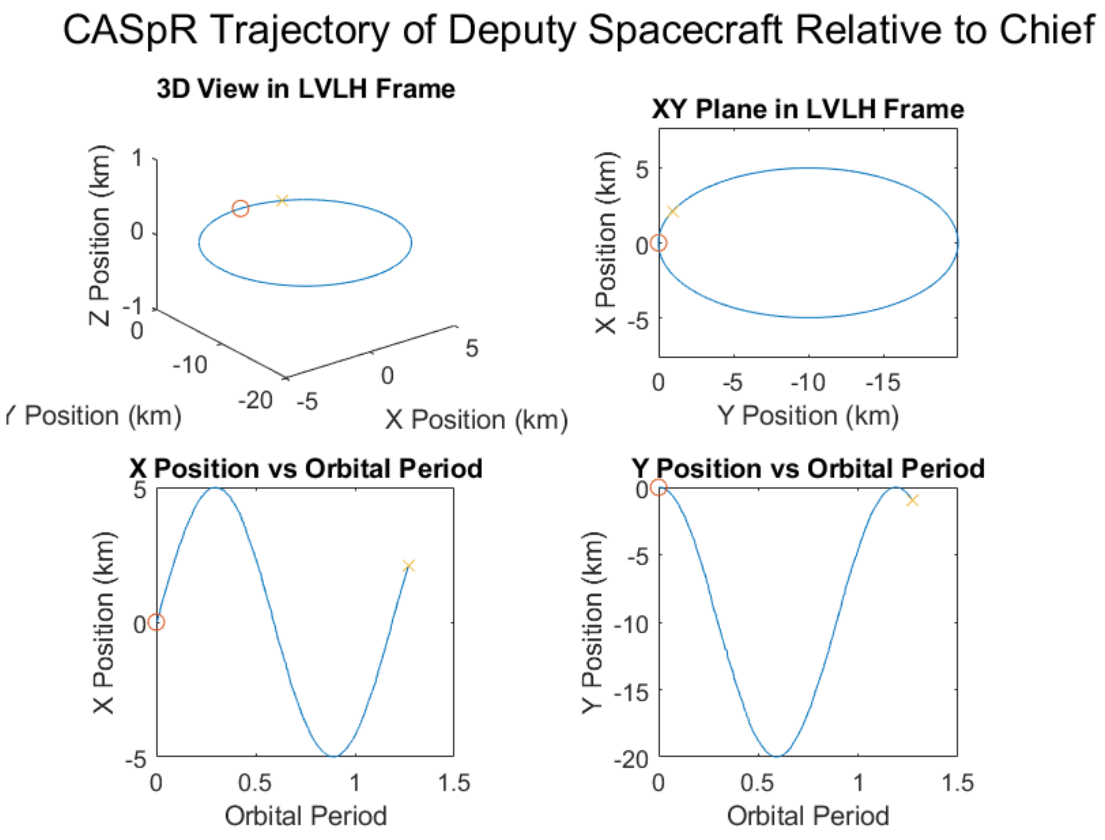


Figure 37: CASpR Trajectory Given Initial X Velocity in C++

The deputy satellite follows a NMC which is not centered on the chief, but instead intersects it. This NMC is centered on (0,10,0)km. The shape of the trajectory matches the prediction shown in Figure 37, further validating the HCW EOMs model in the CASpR LLCs. However, it is noted that the time which it takes the orbit to reach its initial position again is approximately 18.8% longer than the 30 second orbital period. This seems to indicate that there exists a timing issue in CASpR which has not previously been identified.

4.3.1.2 Burn into NMC

Next, the Jetson TX2i is introduced as a variable. A targeting algorithm is developed within the Python code on the TX2i to move the deputy satellite from the zero state vector to a desired state vector. The use of this targeting algorithm allows users to avoid modifying the C++ code when desiring a non-zero initial state.

One difficulty of developing this method with the targeting maneuvers is that the targeting maneuver equations result in ΔV; however, the LLCs only accept force commands. In order to solve this mismatch, equations 65 - 66 are used to solve for a force which can be sent as a command.

$$\mathbf{F} = \frac{d(m\mathbf{v})}{dt} \tag{65}$$

CASpR assumes that the mass of the spacecraft is constant at 1 kg. This simplifies to equation 66.

$$\mathbf{F} = \frac{d\mathbf{v}}{dt} \tag{66}$$

Therefore, the force command sent to the LLC is simply Δv divided by the period of time during which the thrusting maneuver is accomplished.

The first test of the targeting algorithm utilizes a single burn to move into a NMC which is centered on (0,10,0)km. This is the same trajectory which is previously tested with the C++ code and should result in a near initial state given approximately by Equation 67, although it will not be exact due to the non-instantaneous burn.

$$\mathbf{X} = [0, 0, 0, 1.0472, 0, 0]^T \tag{67}$$

For this test, the burn is applied over a duration of one second. The resulting trajectory is shown in Figure 38.

Interestingly, the trajectory completes a NMC as expected; however, the time to complete the NMC is greater than one period. This does not make sense with the HCW EOMs. A NMC at a certain altitude will always take the same amount of time

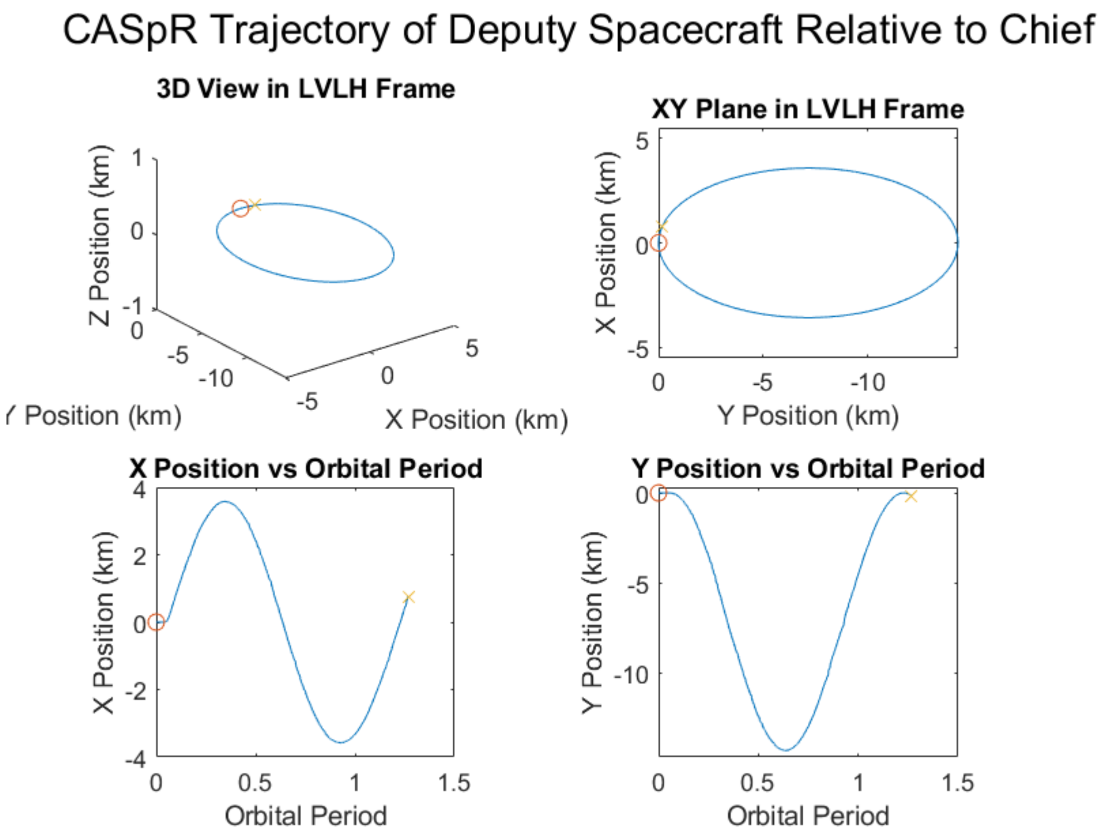


Figure 38: CASpR Trajectory Given Initial X Velocity in Python

if only Keplerian motion is considered. This result reveals that there is some small issue with the current CASpR setup which causes a timing mismatch between the C++ code on the LLC and the Python code on the flight computer.

Next, a two-burn targeting maneuver is utilized to place the deputy satellite into a NMC which is centered on the chief. The same size NMC is used with a semi-minor axis of 5 km and semi-major axis of 10 km. The targeted position of the first burn is given by the position vector (5,0,0)km. The burn occurs over a period of 1 second and the spacecraft is allowed to propagate according to the dynamics for half of an orbital period before the second burn is completed. This second burn is designed to meet the energy matching condition so that the state immediately following the second burn is given by equation 68.

$$\mathbf{X} = [5, 0, 0, 0, -2.0944, 0] \tag{68}$$

The expected trajectory and trajectory tested on CASpR are shown in Figures 39 and 40.

Based on the comparison of the targeting tests and their predicted results, the trajectory created by the targeting algorithm experiences a large amount of drift when no drift should be present. This is likely due to the unresolved timing issue.

4.3.1.3 Active Control

After validating the targeting maneuvers, active control is tested. First, the same targeting maneuver is used to place the spacecraft into a NMC given by Equation 69.

$$\mathbf{X} = [5, 0, 0, 0, -2.0944, 0]^T \tag{69}$$

This NMC is not the desired trajectory of the spacecraft. Instead, the desired trajectory is in the same shape, but scaled such that the semi-major axis length is decreased by 20%. A LQR controller is developed to control the spacecraft to move into a desired state which is a NMC given by 70.

$$\mathbf{X} = [4, 0, 0, 0, -1.6755, 0]^T \tag{70}$$

This difference in the initial trajectory and the desired trajectory provides a test for the controller. The Q and R matrices are given by Equations 71 and 72.

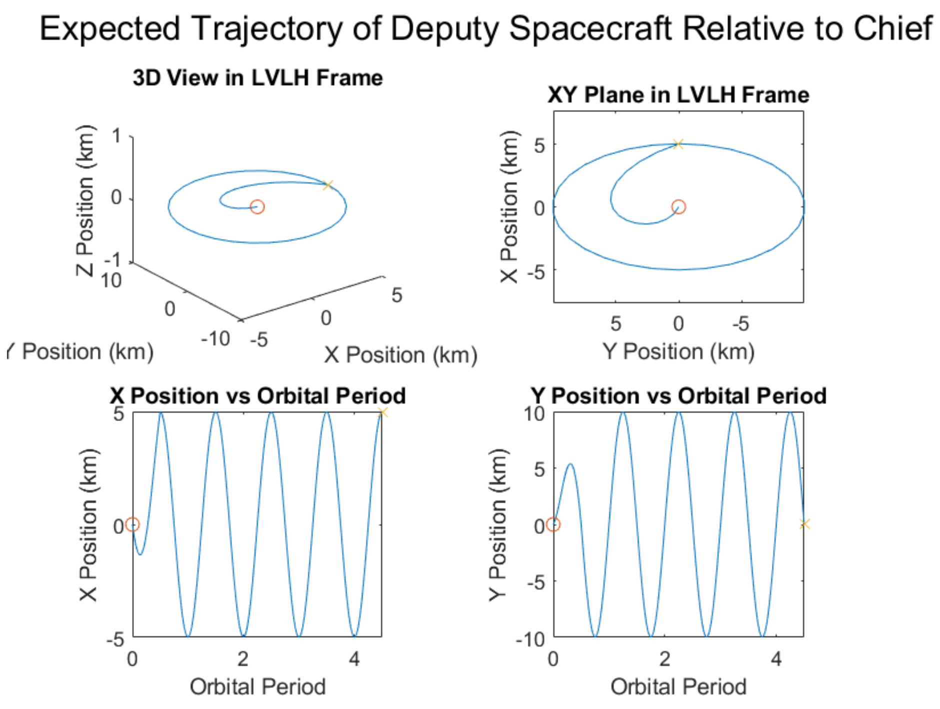


Figure 39: Targeting Expected Trajectory

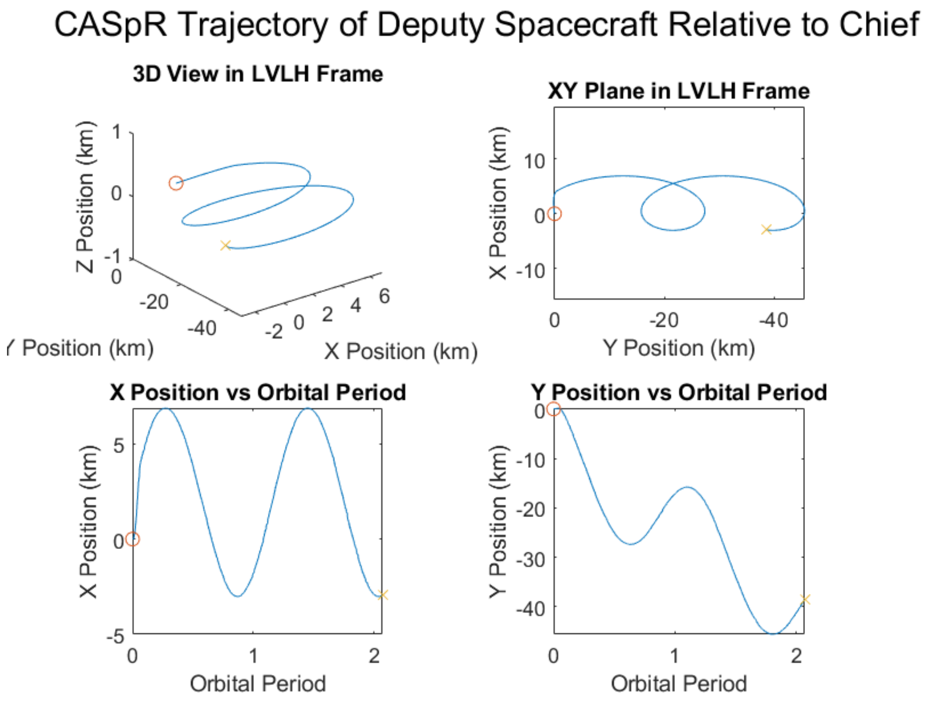


Figure 40: Targeting CASpR Trajectory

$$Q = 1e-6 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{71}$$

$$R = 1e+4 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{72}$$

The expected and tested controlled trajectories are shown in Figures 41 and 42.

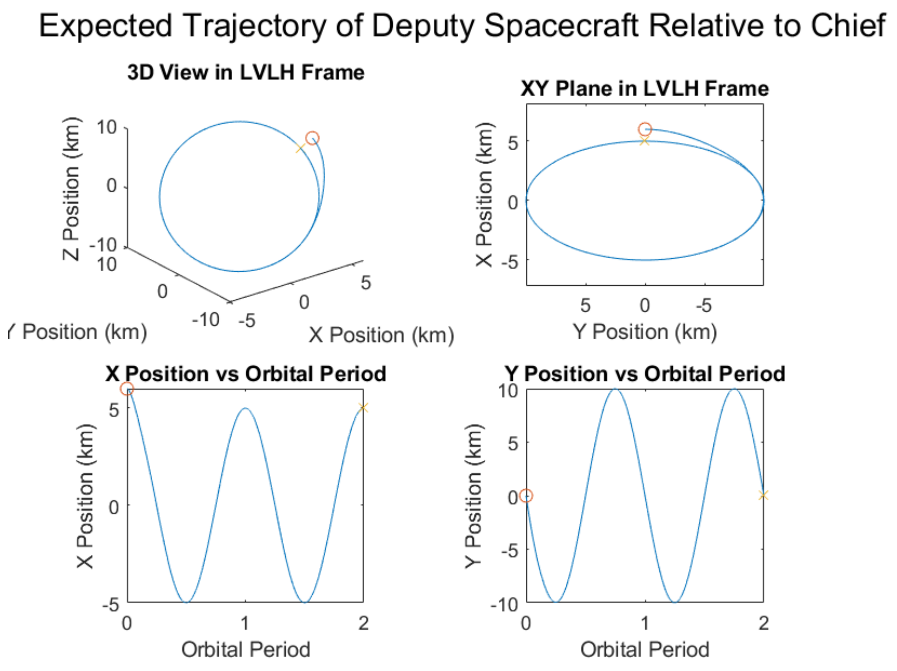


Figure 41: Expected LQR Controlled Trajectory

Because of the timing issue which became apparent in the single burn into a NMC tests, it is suspected that the same issue causes the controller to fail. The controller

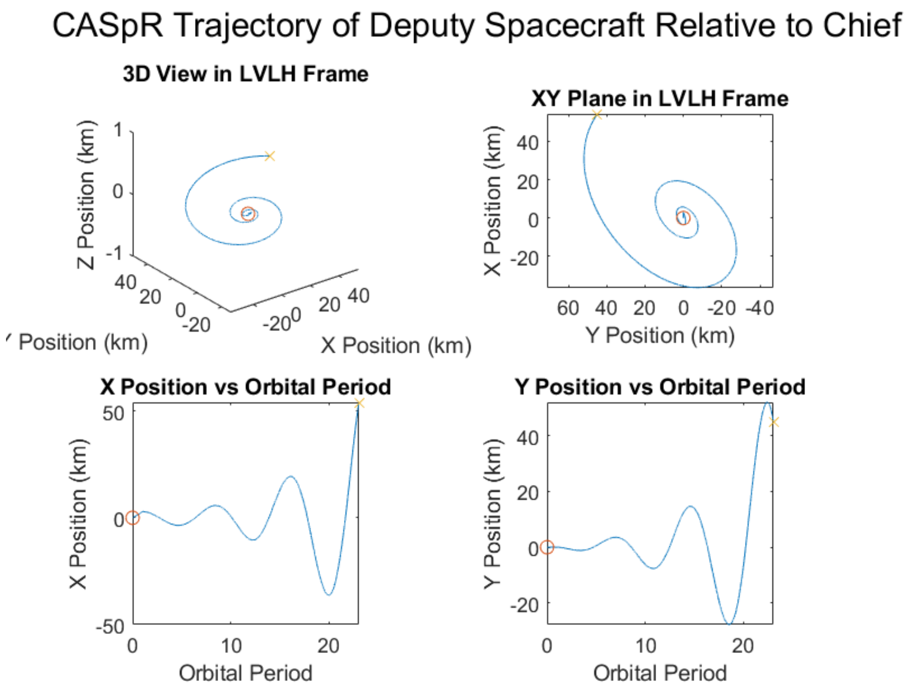


Figure 42: CASpR LQR Controlled Trajectory

used in this section attempts to rendezvous with a NMC. Because of this moving target, the controller algorithm is dependent on time. As a result, the timing issue causes this style of controller to fail.

4.3.2 Limitations of Actuation

Stepper motors, though more precise than brushed or brushless Direct Current (DC) motors, are still limited in their resolution. The resolution can be increased by microstepping as discussed in section 3.4.1. An unfortunate side effect of microstepping however is that additional resolution comes at the cost of torque. In order to avoid slipping, the stepper motor must provide adequate torque to move the load. Because of this constraint, microstepping on CASpR is limited to two microsteps per full step. Other issues such as physically undersized orbits, and positional error are discussed in the following sections.

4.3.2.1 Error in Position Data

After completing one full period in a NMC, the trajectory should realign with the initial position. For a trajectory beginning at the origin, after one period of Keplerian motion, the spacecraft should return to the origin based on the HCW EOMs. A test is completed on CASpR during which the spacecraft is placed into a LEO trajectory and scaled such that most of the available space within CASpR is used. The C++ code on the LLC is modified to give an initial velocity in the x direction. Data is collected for slightly longer than one orbital period. The resulting trajectory is plotted in Figure 43.

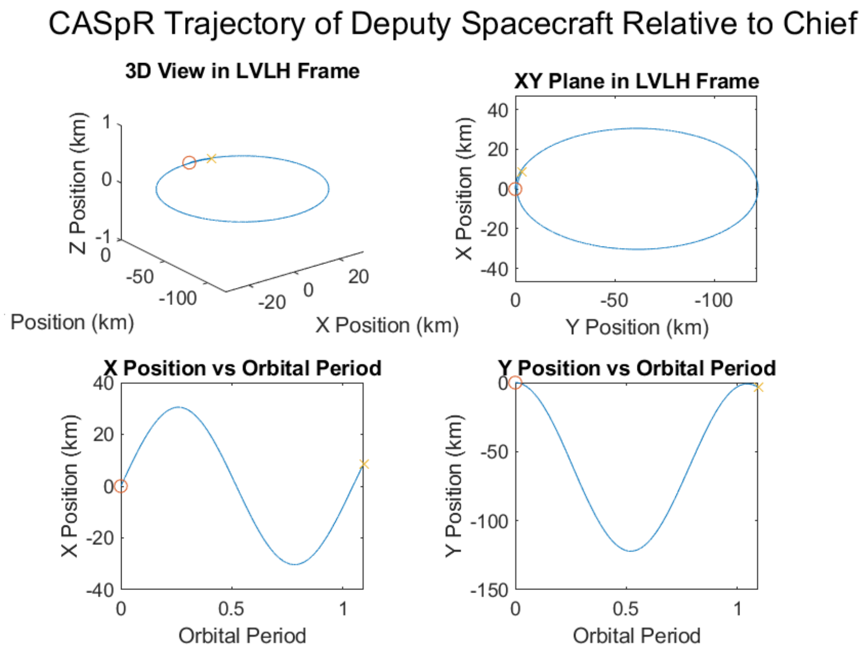


Figure 43: Uncentered LEO NMC

It is visually apparent that the trajectory is not perfectly closed as it would be under ideal conditions. After what should be one orbital period, the state vector is given by [-12.0313, 0.0307, -5.8314, 0.0267, 0, 0]. This large difference is due to the timing error. The true orbital period is determined instead by matching the period to when x = 0. This occurs at t = 6075 seconds rather than 5677 seconds. The difference in the HCW and simulated x/y position is given by [0.000658,0.8918] km. As a percent of full scale, this yields an accuracy of [1.08x10^{-5}%,7.3x10^{-3}%]. This means that the actual position follows the desired position accurately; however, the desired position has begun to drift slightly from intersecting with the origin. For this large NMC, the difference between the origin and the position after one true orbit is 0.9 km for a percent of full scale difference of 0.737%.

Keeping in mind that the natural scaling CASpR is 1 microstep to 1 km, the results can be expected to be within 1 km of the HCW propagated trajectory. The x error for the entire trajectory is shown in Figure 44.

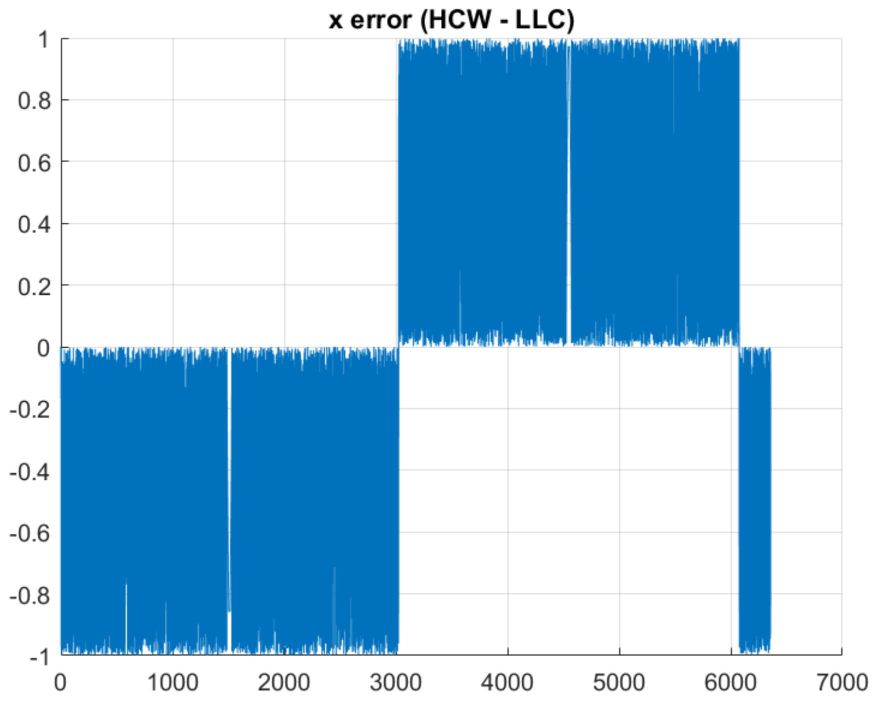


Figure 44: Error in X position

Because the error is never greater than 1 or less than -1, the trajectory accurately tracks the trajectory of the propagated HCW EOMs within 1 kilometer.

4.3.2.2 Undersized Orbit

The CASpR system is limited in its physical simulation by the stepper motors used. CASpR is driven by stepper motors which are micro-stepped such that the smallest possible motion is one-half of the standard step size as stated in 3.4.1. Because of this, orbits tested on CASpR must be scaled carefully. An orbit which is too large will run over into the boundary box created by CASpR's structure while an orbit which is too small will result in a low resolution model. An example of an undersized orbit is shown if Figure 45.

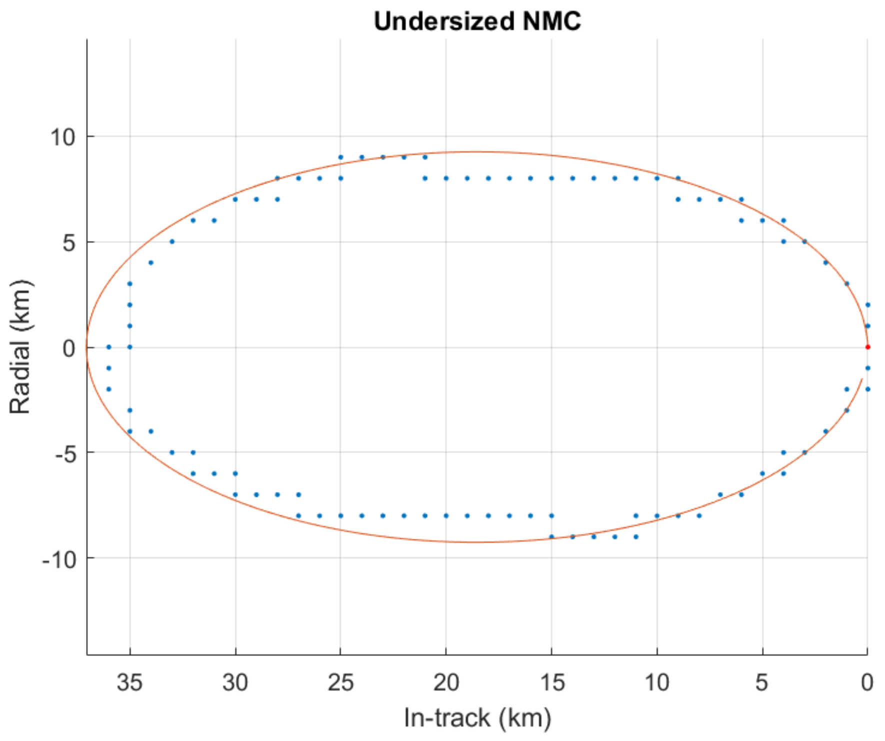


Figure 45: Undersized NMC

The red line represents the desired state of the spacecraft given by the HCW EOMs while the blue points represent the actual position of the simulated spacecraft as given by the stepper motor commands from the LLC. The simulated satellite roughly follows the desired state; however, it seems to lag behind the desired trajectory at points and skip ahead at others. The roughness of resolution is due to the size of the orbit within CASpR compared to the size of available microsteps. This trajectory is only 37 microsteps across - the equivalent of approximately 3 mm.

4.3.3 Software Limitations

4.3.3.1 Truncation

In addition to the hardware issues which arise from undersizing an orbit, software issues are also present. It may be noted that in Figure 45, the spacecraft is not always at the microstep closest to the desired state. This is at least in part due to truncation. Because the stepper motors cannot accept partial microsteps, a microstep is only commanded when the truncated desired state changes. This means that although rounding would provide a slightly higher fidelity model, the current system will experience small amounts of error due to truncation.

4.3.3.2 Standby Mode

An unexpected result which was found while testing is an issue arising from the flight computer going to sleep. Before integrating the Jetson TX2i, the flight computer used is a desktop. Because the period of the LEO trajectory is over 1.5 hours, the desktop has ample time to go into a sleep or standby state. When this occurs, the stepper motors unexpectedly jump to a different position. This jump seems to occur at the time that the computer goes to sleep. The trajectory and jump are shown in Figure 46.

Interestingly, the jump appears to move the satellite to another place on the orbit and continue propagating; however, when viewing in person, this seems to be a false result. While the simulated spacecraft does suddenly jump, it does not appear to jump to a place on the desired trajectory. Instead, the satellite jumps to a random

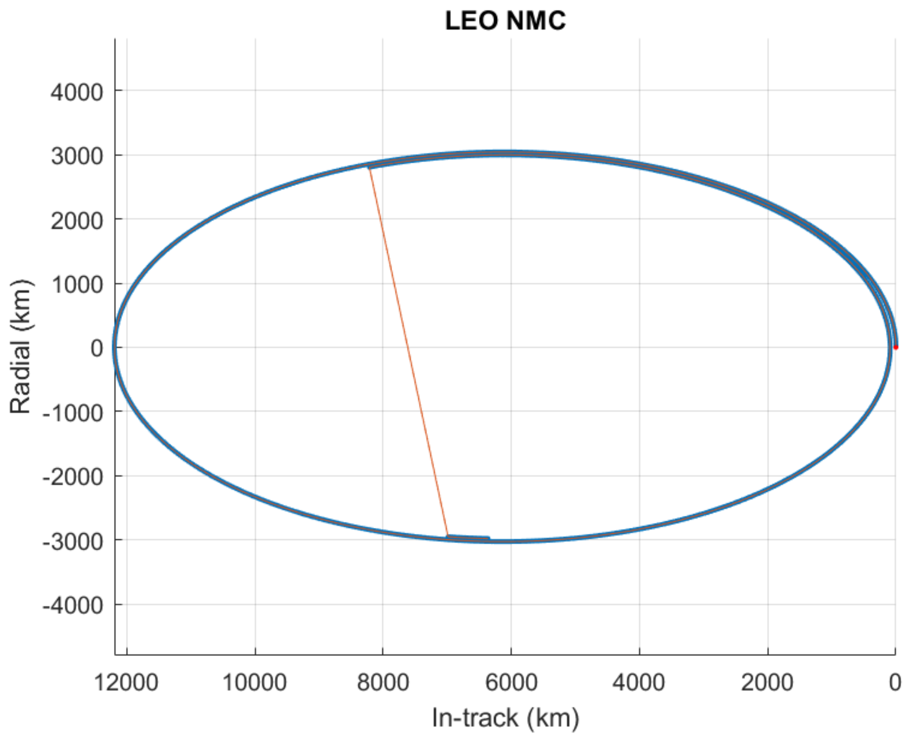


Figure 46: Jumping Error for LEO NMC

point in 3D space, but continues moving. For reasons unknown at this time, the sleeping computer interrupts the communications loop between the LLC and flight computer and causes the jump. Because the LLCs control the stepper motors and do not go into a standby mode, the orbit continues to propagate without understanding the change in position caused by the jump. When the computer is woken back up, the communications loop is no longer interrupted so the position data can once again be received by the flight computer and recorded.

This issue is solved simply by setting the flight computer to not go to sleep in a time frame during which a simulation is being run.

4.3.3.3 Dropped Data

Throughout the implementation of wireless communication through the Digi XBees, occasional tests failed due to errors which occurred after several seconds of running.

These errors occurred seemingly at random with multiple runs of the same test resulting in errors over 10 seconds apart.

The errors indicate that more data is being returned from the LLC in the position packet than is expected. Inspection of the data returned reveals that two data packets are inadvertently combined into one. The first of the two data packets holds almost all of the information it should; however, the last character or few characters are missing. Most importantly, the end of line character is missing. This causes the Python code not to recognize the data as a complete packet. The result is that more data is added until the end of line character from a new packet appears.

This error occurs when insufficient time is given for CASpR to complete one iteration through the main loop. The error is resolved by modifying the C++ code such that the variable dtOutput is increased to a satisfactory point. dtOutput is the variable which controls how often data is sent between the LLC and the flight computer. This solution reduces the frequency at which data is returned and may result in a very slight decrease in the fidelity of the model.

Although the error is resolved in this research, other research may require a higher computational load and run into the same error again.

4.3.4 Scaling

Because of the errors which occur due to truncation as well as the limited microstepping size, proper scaling is important when simulating trajectories on CASpR. A trajectory which is not properly scaled to the system will result in lower resolution models. Two trajectories, one unscaled and one scaled are provided in Figures 47 and 48.

A scale factor is used in the Python code to easily manipulate the size of the trajectory. For standard testing on CASpR, a microstep on the stepper motor equates

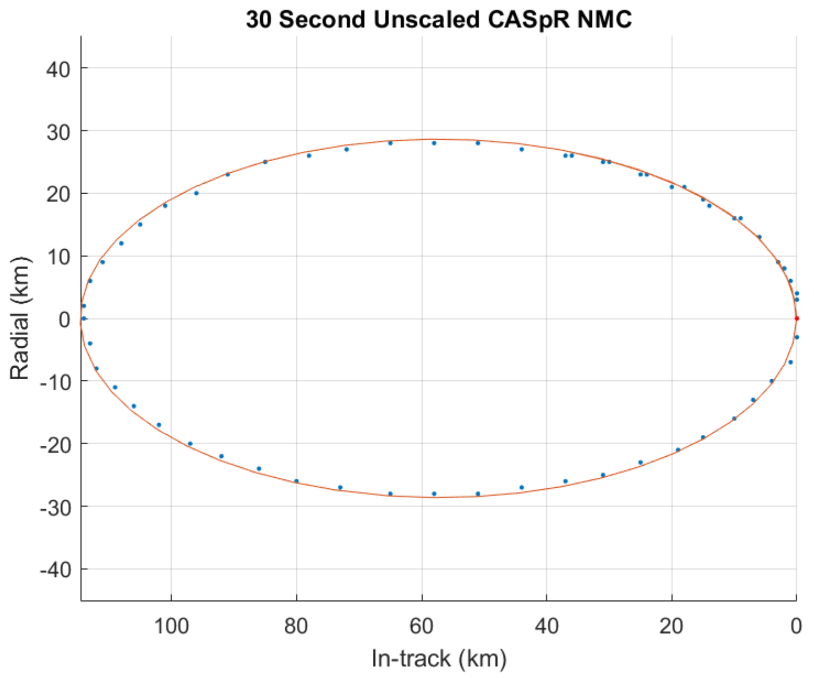


Figure 47: 30 Second Unscaled NMC Simulation

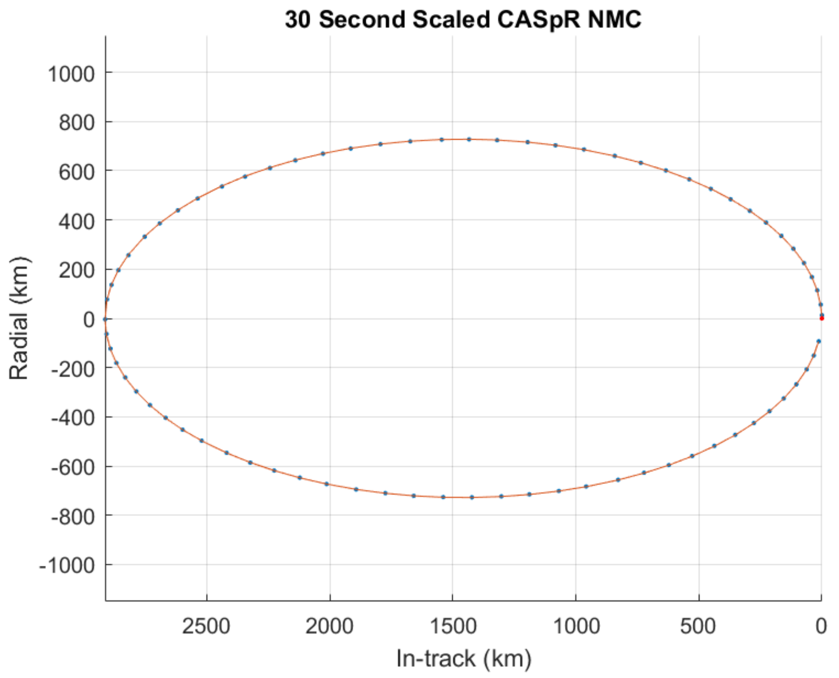


Figure 48: 30 Second Scaled NMC Simulation

to one kilometer on the simulated orbit. This can be modified in the C++ or Python codes. This is most easily accomplished by modifying the scale factor "Scalexy" in Python; however, the resulting data will also need to be modified in MATLAB to correct the scale when plotting. By determining the expected size of the orbit and knowing that the physical working area of CASpR is 27000 microsteps, the trajectory may be properly scaled to yield truncation and microstep size errors negligible.

As noted previously, it takes approximately 27000 microsteps for CASpR to traverse one side of the structure. The base software assigns one kilometer in the simulation to one microstep on CASpR. This means that a NMC with a semi-major axis of 10 km will only take up a 10x20 microstep area. For this reason, an additional scale factor is assigned to enlarge the trajectory to fill CASpR's structure.

Assuming a desired trajectory in simulation is known, a rule of thumb is developed for scaling CASpR. First, a buffer of 1000 microsteps on each side of CASpR is implemented to avoid side wall collision. This leaves 25000 microsteps of usable distance per side. Next, CASpR is positioned to begin the simulation at the center of the frame. This leaves CASpR 12500 microsteps away from the nearest side. The scale factor may then be determined by dividing 12500 by the maximum expected x or y distance that the simulated satellite would be from the origin as shown in equation 73.

$$Scalexy = \frac{12500}{xyMax(km)} \tag{73}$$

A NMC which would in space have a semi-major axis of 10 km and begins at the origin with velocity only in the x direction, has a scale factor of 625 as shown in equation 74.

$$625 = \frac{12500}{(2)(10)} \tag{74}$$

If the size of the expected trajectory is not yet known, a smaller scale factor may be used initially to avoid side wall collisions.

A visual representation of the scaling is shown in Figure 49.

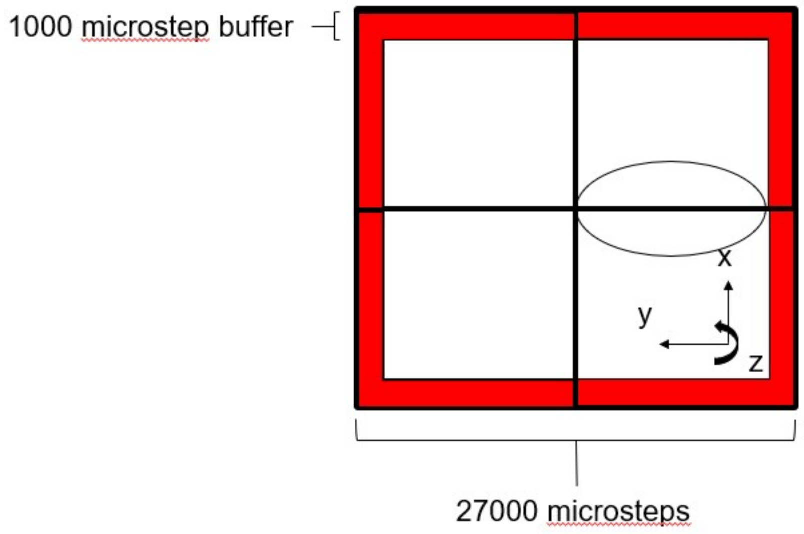


Figure 49: Development of Scale Factor

4.3.5 Refresh Rate

In a real-world scenario, satellites operate in real-time with calculations, such as using a controller, taking some small amount of time. For CASpR however, timesteps are being used to propagate the dynamics, determine the current state, determine the desired state, determine the control to reach the desired state, and apply control. Because of this, there may be some small error due to the rate at which new calculations are completed. For the tests conducted in this study, it is observed that the Python code runs through the main loop at a rate of approximately 20Hz. This rate is expected to be sufficient for the intended uses of CASpR.

V. Conclusions and Recommendations for Future Work

5.1 Summary

The Control and Autonomy Space proximity Robot (CASpR) system is an important tool for modeling the trajectories of spacecraft in relative orbits. It allows testing of controllers and hardware components as an interim between purely software-based simulations and full scale testing in the space environment. As a result of this capability, substantial amounts of time and money can be saved in developing satellites. Chapter I provides the goals of this research to include developing a method by which embedded computing may be cost-effectively implemented onto a gantry-style Rendezvous and Proximity Operations (RPO) Hardware in the Loop (HIL) testbed. The secondary goal of beginning to classify the testbed is also met in this research effort.

Chapter II provided the background of other ground-based Guidance, Navigation, and Control (GNC) testbeds, including their advantages and disadvantages. Different methods of modeling the relative motion of a spacecraft are discussed and Hill-Clohessy-Wiltshire (HCW) is chosen as the preferable Equations of Motion (EOMs) to be used for CASpR. Common trajectories and maneuvers which are used in RPO are also shown so they may be used on CASpR.

Chapter III provides a description of the construction of CASpR to include its structure and hardware as well as the key software components which are involved. A model to determine the deflection inherent in the structural design of CASpR is developed. This chapter also provides the methodology for a small range of tests used to validate that CASpR is capable of accomplishing basic RPO maneuvers. Next, the scale of the testbed along with ways to modify it are shown.

Finally, Chapter IV provides the results and analysis of the deflection and the RPO maneuvering tests. Other unintended results of interest as well as limitations of the system which arose through the planned testing of CASpR are discussed so that future users may better understand the intricacies and minutiae of the testbed.

5.2 Conclusions

A series of improvements are made to the existing CASpR system throughout this research. The most substantial of these improvements are outlined below.

- Embedded computing onboard CASpR is accomplished through use of a Jetson TX2i. This Jetson TX2i is placed onboard CASpR by integrating it onto the sensor stack. The Jetson TX2i acts as a flight computer and provides sufficient computing power to perform the desired RPO and control algorithms.
- A sensor stack is developed which carries the flight computer in addition to an external power source, communications hardware, as well as additional sensors.
- The onboard power supply is sized to provide power for over an hour of continuous RPO operations before requiring a new battery to be swapped.
- The sensor stack is built not only to carry the necessary hardware but also to be interchangeable such that alternate sensor stacks may easily be used onboard CASpR
- Additionally, an algorithm is developed and implemented for controlling a spacecraft to burn over a short period of time into a Natural Motion Circumnavigation (NMC) about a chief spacecraft and a controller is built as well.
- A timing issue is identified which results in inaccurate trajectories. The error which is thought to be associated with these timing issues appears to worsen as models grow in complexity. However, the error is present in a simulation as simple as applying an initial condition through the Low-Level Controller (LLC).

For this reason, it seems that the timing issue lies at least in part in the C++ code. Alternatively, the issue may be caused by the accumulation of delays across the more complex communications setup.

5.3 Recommendations for Future Work

- Add a Motion Capture system to improve validation rather than use "true" data from stepper motors.
- Equip both the upper and lower simulated satellites with embedded computing.
- Add pitch tilt DOF to the system.
- Understand and resolve the issue which results in apparent timing issues when attempting to complete maneuvers.
- Properly tune the control matrices Q and R in the LQR controller
- Equip CASpR with a homing feature by using microswitches which signal the edges of CASpR. This can be accomplished with microswitches connected to the RepRap Arduino Mega Pololu Shield (RAMPS) board in the Low-Level Controllers (LLCs).
- Minimize the amount of wasted data being passed back to the flight computer. Currently the packet passed contains: (mode, Fx, Fy, Fz, tauZ, xPosCmd, yPosCmd, zPosCmd, thetaCmd, speed). When in GNC mode, xPosCmd, yPosCmd, zPosCmd, and speed are unused. Likewise, in Manual mode, Fx, Fy, Fz, and tauZ are unused. Because the mode is passed in as part of the packet, the packet size may be reduced without loss of information.
- Decrease the height of the sensor stack to minimize wasted space.

5.4 Future Applications of CASpR

CASpR in its current form operates well as a ground-based kinematic testbed for two satellites. Increased operational capability can be achieved by allowing the two simulated satellites to move about each other in all directions rather than designating a single satellite as “upper” and the other as “lower”. This capability may be achieved by removing the physical constraint of the structure of the cage. Rather than moving along a rail-type system, drones with rotors may be used. This allows a much larger area to be used in addition to as many simulated satellites as may be desired. One drawback of this method however, is the shortened possible simulation time since it is difficult to keep drones flying for more than approximately 15 minutes.

Other sensors which will be implemented in the future include a vision-based system which can precisely and accurately determine the location of the simulated spacecraft within the CASpR cage. These sensors will not be onboard the simulated satellite. Instead, they will be mounted statically in the room and communicate between each other to determine the location.

Appendix A. CASpR Documentation

1.1 Setting up Static IP Address

In order to consistently establish a SSH connection between the Jetson TX2i and another computer, it is important to set up a static IP address on the Jetson. This can be accomplished with a few commands in the terminal. The terminal can easily be opened by pressing Ctrl+Alt+T. To check the current IP address, the command “ifconfig” may be used. This yields all of the Jetson’s current connections. The current IP address is found under wlan0 with the descriptor inet.

In order to change the IP address until the machine is powered off, the command “sudo ifconfig wlan0 192.168.1.101” may be used.

Setting a static IP address requires different commands. “sudo nano /etc/netplan/-1-netcfg.yaml” opens a file which stores the network interfacing information. To set a static IP address, enter the following lines (without bullets/numbering):

network:

1. version: 2
2. renderer: NetworkManager
3. ethernets:
    - eth0:
        - dhcp4: no
        - dhcp6: no
        - addresses: [192.168.1.50/24]
        - gateway4: 192.168.1.1

    - nameservers:
        - addresses: [8.8.8.8, 8.8.4.4]

- eth1:
    - dhcp4: yes
    - dhcp6: no
    - nameservers:
        - addresses: [8.8.8.8, 8.8.4.4]

- wlan0:
    - dhcp4: yes
    - dhcp6: no
    - addresses:
        - 192.168.1.101/24
    - nameservers:
        - addresses: [8.8.8.8, 4.4.4.4]

Write the amended file by pressing Ctrl+S. The system must then be rebooted for the static IP to be fully set up. This can be accomplished by using the command "sudo reboot" in the terminal. When the system reboots, the static IP address should be set.

1.2 Setting up MobaXterm

MobaXterm is used to set up a SSH connection between the Jetson TX2i and another computer. Each session in MobaXterm can be set up separately or sessions may be saved to quickly reconnect to a computer with which a connection has previously been made. To set up a connection the following steps are taken.

1. Select “Session”
2. Select “SSH”
3. In the “Remote host” box enter the IP address of the Jetson’s USB Wifi adapter
   (a) At the time of this research, the IP address used is 192.168.1.101
   (b) If the correct IP address is not known, it can be found through the Jetson TX2i’s command line. The command “ifconfig” yields all of the Jetson’s connections. The desired IP address is found under “addresses”.
   (c) It is important that the IP address be set up as static as described in 1.1.
4. Select the check box beside “Specify username”
5. Select the box with the person and key beside the drop down box
   (a) Select “New”
      i. In the “Name” text box use any name for the connection for example “Jetson”
      ii. In the “Username” text box enter the username for the Jetson TX2i. For the purposes of the Jetson used in this study, the username is “user”
      iii. In the “Password” text box enter the password for the Jetson TX2i. For the purposes of the Jetson used in this study, the username is “user”
6. In the drop down box, select the newly made Jetson profile “[Jetson]”
7. Select “Ok”. This will open the SSH connection between the two computers.

Once a session is begun, the files of the Jetson are easily accessible and viewable in the file tree on the left side of the screen. The right side of the screen is primarily the terminal. Commands may be passed to the Jetson through this terminal. To run manual control, the following command may be used

"*python3 /home/user/Documents/CASpR_Manual_Control.py*".

After running the command and starting the manual control script, the CASpR command window appears.

1.3 Understanding the Python Code

1.3.1 Timing

Several timing variables are used in the Python code. Without understanding their meanings and uses, they can easily be confused. The timing variables and their definitions are provided below.

- caspr.TimeElapsed - Time since GNC mode began
- *time_to_target*[0] - Time between waypoints in the targeting algorithm
- *thrust_duration* - Length of time across which $\Delta V1$ is applied
- *thrust2_duration* - Length of time across which $\Delta V2$ is applied
- *wait_time*[0] - Time spacecraft is allowed to drift according to HCW EOMs after $\Delta V2$ is applied

Bibliography

1. D. Gallardo, R. Bevilacqua, and R. E. Rasmussen. Advances on a 6 degrees of freedom testbed for autonomous satellites operations. In *AIAA Guidance, Navigation, and Control Conference 2011*, number August, 2011.

2. Mark Mercier, Sean Phillips, Matt Shubert, and Wenjie Dong. Terrestrial Testing of Multi-Agent, Relative Guidance, Navigation, and Control Algorithms. *2020 IEEE/ION Position, Location and Navigation Symposium, PLANS 2020*, pages 1488–1497, 2020.

3. Kristia K. Harris. *Design and Validation of a Hardware-In-The-Loop Testbed for Proximity Operations Payloads.* PhD thesis, Embry-Riddle University, 2015.

4. Giovanni Franzini and Mario Innocenti. Relative Motion Dynamics with Arbitrary Perturbations in the Local-Vertical Local-Horizon Reference Frame. *Journal of the Astronautical Sciences*, 67(1):98–112, 2020.

5. Thomas A. Lovell and David A. Spencer. Relative orbital elements formulation based upon the clohessy-wiltshire equations. *Journal of the Astronautical Sciences*, 61(4):341–366, 2014.

6. Thomas A. Lovell and D. L. Brown. Impulsive-Hover Satellite Trajectory Design for Rendezvous and Proximity Operations Missions. *American Astronautical Society*, 07(102), 2007.

7. V1 Engineering. Mostly Printed CNC machine.

8. Amesweb. fixed-beam-deflection-formula @ amesweb.info.

9. RepRap. RAMPS 1.4 low-level controller Arduino Shield.

10. Arduino. arduino-mega-2560-rev3 @ store-usa.arduino.cc.

11. Connect Tech Inc. Elroy Carrier for NVIDIA® Jetson™ TX2/TX2i.

12. Pololu Robotics and Electronics. A4988 Stepper Motor Driver Carrier.

13. Marco Sabatini, Marco Farnocchia, and Giovanni B. Palmerini. Design and tests of a frictionless 2D platform for studying space navigation and control subsystems. *IEEE Aerospace Conference Proceedings*, pages 1–12, 2012.

14. Richard Zappulla, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. *Journal of Spacecraft and Rockets*, 54(4):825–839, 2017.

15. K. Saulnier, D. Pérez, R. C. Huang, D. Gallardo, G. Tilton, and R. Bevilacqua. A six-degree-of-freedom hardware-in-the-loop simulator for small spacecraft. *Acta Astronautica*, 105(2):444–462, 2014.

16. Panagiotis Tsiotras. ASTROS: A 5DOF experimental facility for research in space proximity operations. *Advances in the Astronautical Sciences*, 151(January 2014):717–730, 2014.

17. Markus Wilde, Brian Kaplinger, Tiauw Go, Hector Gutierrez, and Daniel Kirk. ORION: A simulation environment for spacecraft formation flight, capture, and orbital robotics. *IEEE Aerospace Conference Proceedings*, 2016-June, 2016.

18. Tae Ha Park, Juergen Bosse, and Simone D'Amico. Robotic Testbed for Rendezvous and Optical Navigation: Multi-Source Calibration and Machine Learning Use Cases. In *AAS/AIAA Astrodynamics Specialist Conference*, pages 1–20, 2021.

19. Matweb. SpecificMaterial @ asm.matweb.com.

20. U.S. Department of Defense. Defense Space Strategy Summary. (June):1–10, 2020.

21. G. Di Mauro, M. Lawn, and R. Bevilacqua. Survey on guidance navigation and control requirements for spacecraft formation-flying missions. *Journal of Guidance, Control, and Dynamics*, 41(3):581–602, 2018.

22. Luis F Peñin, Yann Scoarnec, Carolina Cazorla, José Ramón Villa, Ronald Kassel, Francisco Javier Benito, Dominique Baudoux, Luigi Strippoli, Damien Galano, and Karim Mellab. Proba-3: ESA's small satellites precise formation flying mission to study the Sun's inner corona as never before. In *Small Satellite Conference*, volume 34, pages 1–9, 2020.

23. Wyatt J Harris. *Visual Navigation and Control for Spacecraft Proximity Operations with Unknown Targets.* PhD thesis, Air Force Institute of Technology, 2021.

24. Sten Berge, Björn Jakobsson, Per Bodin, Anders Edfors, and Staffan Persson. Rendezvous and formation flying experiments within the prisma in-orbit testbed. *European Space Agency, (Special Publication) ESA SP*, 2005(606):719–728, 2006.

25. Johannes Robens and Markus Wilde. Limit-Cycle Controls for Spacecraft in Formation Flight and Docking. *IEEE Aerospace Conference Proceedings*, 2021-March:1–14, 2021.

26. Tomasz Rybus and Karol Seweryn. Planar air-bearing microgravity simulators: Review of applications, existing solutions and design parameters. *Acta Astronautica*, 120:239–259, 2016.

27. Mario A. Santillo, Nalin A. Chaturvedi, N. Harris McClamroch, and Dennis S. Bernstein. 3D pendulum experimental setup for earth-based testing of the at-

titude dynamics of an orbiting spacecraft. *Proceedings of the American Control Conference*, pages 2479–2484, 2007.

28. Terry Alfriend, Srinivas Vadali, Pini Gurfil, Jonathan How, and Louis Breger. *Spacecraft Formation Flying: Dynamics, Control and Navigation.* Elsevier, 2010.

29. W. H. CLOHESSY and R. S. WILTSHIRE. Terminal Guidance System for Satellite Rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 674, 1960.

30. G. W. Hill. Researches in the Lunar Theory. *American Journal of Mathematics*, 1(1):5–26, 1878.

31. Angel Flores-Abad, Ou Ma, Khanh Pham, and Steve Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26, 2014.

32. J. Tschauner and P. Hempel. Optimale Beschleunigeungsprogramme fur das Rendezvous-Manover. *Astronautica Acta*, 10:296–307, 1964.

33. Koji Yamanaka and Finn Ankersen. New state transition matrix for relative motion on an arbitrary elliptical orbit. *Journal of Guidance, Control, and Dynamics*, 25(1):60–66, 2002.

34. Niels Henrik Roth. *Navigation and Control Design for the CanX-4/-5 Satellite Formation Flying Mission.* PhD thesis, University of Toronto, 2010.

35. Nvidia. [Jetson] DATA SHEET NVIDIA Jetson TX2 / TX2i System-on-Module Pascal GPU + ARMv8 + 8GB LPDDR4 + 32GB eMMC, 2014.

36. DIGI. XBee ® / XBee-PRO SX Radio Frequency (RF) Module User Guide.

Acronyms

2BP Two Body Problem. 15

ADAMUS ADvanced Autonomous MUltiple Spacecraft. 8

AFB Air Force Base. 11

AFIT Air Force Institute of Technology. iv, 2, 3, 33

CAD Computer-Aided Design. 49

CANS Cooperative Autonomous Networked Systems. 11

CanX Canadian Advance Nanosatellite eXperiment. 23

CASpR Control and Autonomy Space proximity Robot. iv, viii, 3, 4, 5, 26, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 63, 64, 65, 66, 67, 68, 70, 71, 72, 73, 74, 77, 78, 79, 80, 81, 83, 85, 86, 87, 88, 89, 90

CMGs Control Moment Gyroscopes. 10, 11

CNC Computer Numerical Control. 34

COEs Classical Orbital Elements. 12

DC Direct Current. 77

DOF Degree of Freedom. 7, 8, 11, 34, 46

DOFs Degrees of Freedom. 4, 7, 34, 35, 46

EOMs Equations of Motion. iv, 18, 20, 21, 22, 23, 26, 28, 29, 32, 51, 56, 58, 69, 71, 72, 78, 79, 80, 87

ESA European Space Agency. 2

FF Formation Flying. 2, 16, 35

GNC Guidance, Navigation, and Control. 1, 2, 4, 6, 10, 31, 51, 52, 53, 54, 55, 87, 89

GUI Graphical User Interface. viii, 45, 51, 52, 55

HCW Hill-Clohessy-Wiltshire. iv, 4, 15, 16, 17, 18, 20, 22, 23, 24, 25, 26, 28, 29, 32, 42, 50, 56, 58, 60, 61, 69, 71, 72, 78, 79, 80, 87

HDMI High-Definition Multimedia Interface. 42

HIL Hardware in the Loop. 3, 6, 87

IDE Integrated Development Environment. 50, 51

IPV4 Internet Protocol version 4. 54

LEO Low Earth Orbit. viii, 10, 16, 32, 70, 78, 81, 82

LiPo Lithium Polymer. 47

LLC Low-Level Controller. 42, 44, 45, 51, 59, 68, 72, 73, 78, 80, 82, 83, 88

LLCs Low-Level Controllers. 42, 43, 44, 47, 50, 51, 53, 55, 57, 58, 69, 71, 72, 82, 89

LQR Linear Quadratic Regulator. 53, 54, 60, 61, 68, 74

LTI Linear Time-Invariant. 15, 17

LTV Linear Time-Variant. 19

LVLH Local-Vertical-Local-Horizontal. 12, 13, 14, 17

MATLAB MATrix LABoratory. 5, 14, 40, 50, 54, 58, 63, 85

MPCNC Mostly Printed Computer Numerical Control. 33, 34

NERMs Nonlinear Equations of Relative Motion. 4, 6, 13, 14, 15, 18, 19, 31

NMC Natural Motion Circumnavigation. viii, 18, 22, 25, 26, 27, 28, 51, 58, 59, 60, 62, 68, 70, 71, 72, 73, 74, 76, 77, 78, 79, 82, 85, 88

ODEs Ordinary Differential Equations. 16

OS Operating System. 41, 42

PC Personal Computer. 42, 43, 44, 45

PRISMA Prototype Research Instruments and Space Mission technology Advancement. 6

PROBA-3 Project for On-Board Autonomy-3. 2

RAMPS RepRap Arduino Mega Pololu Shield. 42, 45, 46, 57, 89

RMRC Ready Made Remote Control. viii, 48

RPO Rendezvous and Proximity Operations. iv, 1, 2, 3, 4, 6, 7, 12, 13, 16, 23, 26, 33, 48, 60, 87, 88

SSH Secure Shell. 44, 54

STM State Transition Matrix. 21, 22, 23, 24

TH Tschauner-Hempel. 18, 20, 21, 22

USB Universal Serial Bus. 42

VS Visual Studio. 51

XCTU XBee Configuration & Test Utility. 55

YA Yamanaka-Ankersen. 22, 23

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 12/23/2021	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* July 2020 - December 2021

4. TITLE AND SUBTITLE Design, Development, and Testing of Embedded Computing on AFIT's Control & Autonomy Space Proximity Robot (CASpR)	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gwaltney, Collin, A, 2nd LT	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way Wright-Patterson AFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENY-MS-21-D-067
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	**10. SPONSOR/MONITOR'S ACRONYM(S)** N/A
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This thesis reviews RPO algorithm testbeds and discusses the development of the Control and Autonomy Space proximity Robot (CASpR) kinematic testbed housed at the Air Force Institute of Technology (AFIT). CASpR operates on a rail system to propagate the trajectories of two satellites using the Hill-Clohessy-Wiltshire (HCW) Equations of Motion (EOMs). In this study, the implementation of a Jetson TX2i as an onboard flight computer is discussed and accomplished. Each hardware component used in the process of adding embedded computing as well as the software and paths of communication are all discussed in detail. Tests are conducted to assess the reliability of CASpR and the embedded computer.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj Costantinos Zagaris
U	U	U	UU	113	**19b. TELEPHONE NUMBER** *(Include area code)* 937-255-3636 ext. 4774

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18