Aerodynamic effects of integrated lifting surfaces on very low Earth orbit small satellites

William J. Bickett

AERODYNAMIC EFFECTS OF INTEGRATED LIFTING SURFACES ON VERY LOW EARTH ORBIT SMALL SATELLITES

by

WILLIAM J. BICKETT

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in
The Department of Mechanical and Aerospace Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2019

William Bickett 10/23/2019

William J. Bickett (date)

THESIS APPROVAL FORM

Submitted by William J. Bickett in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Systems Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Systems Engineering.

[illegible] *Dr. Kunning G. Xu*	10/23/19 (Date)	Committee Chair
[illegible] *Dr. Jason Cassibry*	10/28/19 (Date)	
[illegible] *Dr. James A Parsons*	23 Oct 2019 (Date)	
[illegible] *Dr. Keith Hollingsworth*	10/29/2019 (Date)	Department Chair
[illegible] *Dr. Shankar Mahalingam*	10/30/19 (Date)	College Dean
[illegible] *Dr. David Berkowitz*	11/1/19 (Date)	Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree	Masters of Science in Engineering	College/Dept.	Engineering/Mechanical and Aerospace Engineering

Name of Candidate William J. Bickett

Title Aerodynamic Effects of Integrated Lifting Surfaces on Very Low Earth Orbit Small Satellites

Small satellites are becoming increasingly popular due to their low cost and ease of launch. One limitation that they possess, however, is their inability to house the large imaging equipment needed for detailed observation of the Earths surface from high altitudes. Operating these satellites at very low orbits in order to decrease the observation range is a possible solution to this issue. Satellites orbiting at altitudes under 150 km are subject to the effects of atmospheric drag which causes the satellite to de-orbit quickly; however, if there is enough atmosphere to cause significant drag, then there is also enough to generate significant lift which could be harnessed as a maneuvering tool for adjustments to orbital parameters such as altitude and inclination. This thesis presents the modeling of the aerodynamic effects of integrated lifting surfaces on 6U sized satellite orbits ranging from 100 km to 150km. The results show that while the lifting surfaces create additional drag, they are also generate sufficient lift force to reduce the effect of orbital altitude decay and the overall thrust impulse which would be needed to sustain the satellites orbit.

Abstract Approval: Committee Chair [illegible]

Dr. Kunning G. Xu

Department Chair [illegible]

Dr. Keith Hollingsworth

Graduate Dean [illegible]

Dr. David Berkowitz

ACKNOWLEDGMENTS

I would foremost like to express my sincere appreciation to the members of my committee. Each of you has played an important role in my journey to this point in my academic career.

I am also extremely grateful for all of my friends and family. You have always been encouraging and kept me focused on my goals.

Lastly, the completion of this thesis would have not been possible without the love and support of my wife, Kelsey. Thank you for everything.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

SYMBOL	DEFINITION
3DOF	three-degrees-of-freedom
α	angle-of-attack of the satellite
λ	geodetic latitude
μ	standard gravitational parameter for Earth
ν	true anomaly
ω	argument of periapsis
Ω	longitude of the ascending node
ϕ	geodetic longitude
ρ	atmospheric density
ρ_0	atmospheric density at sea-level
a	semi-major axis
A	area
$A_{\text{satellite}}$	cross-sectional area of the satellite
A_{wing}	reference area for the wing(s)
C_{D_0}	drag coefficient for the satellite body
C_{D_α}	coefficient of drag due as a function of angle-of-attack

C_{L_α}	coefficient of lift due as a function of angle-of-attack
D	drag force
DCM	direction-cosine matrix
dt	iteration time-step for a simulation
e	orbital eccentricity
$\vec{e}$	eccentricity vector
ECEF	Earth-Centered-Earth-Fixed coordinate system
ECI	Earth-Centered-Intertial coordinate system
F_G	force due to gravity
F_{G_x}	X component of force due to gravity
F_{G_y}	Y component of force due to gravity
F_{G_z}	Z component of force due to gravity
F_{J2}	perturbation force due to the Earth's oblatness
F_{J2_x}	X component of perturbation force due to the Earth's oblatness
F_{J2_y}	Y component of perturbation force due to the Earth's oblatness
F_{J2_z}	Z component of perturbation force due to the Earth's oblatness
$\vec{h}$	angular momentum vector
H	atmospheric scale height

i	orbital inclination
J_2	parameter for the gravitational force perturbation due to the oblatness of the earth
	Boltzmann constant
Kn	Knudsen number
L	lift force
$m_{\text{satellite}}$	mass of the satellite
$\vec{n}$	normal vector
NED	North-East-Down coordinate system
p	semi-latus rectum
r	position
$\vec{r}$	position vector
r_e	Earth's equatorial radius
R	gas constant
RK4	4th-order Runge-Kutta algorithm
s	molecular speed ratio: $V/\sqrt{2\,R\,T}$
T	transformation matrix
v	velocity
$\vec{v}$	velocity vector

Somewhere, something incredible is waiting to be known.

—Carl Sagan

CHAPTER 1

INTRODUCTION

Anticipating problems and figuring out how to solve them is actually the opposite of worrying: its productive.

—Chris Hadfield

1.1 A Brief History of Small Satellites

The turn of the 21$^{\text{st}}$ century saw the advent of a new kind of satellite which would usher in a new era of space exploration and experimentation. These new satellites were small, so small that they were dubbed "mircosatellites" and because of their size, these miniaturized satellites were much cheaper and easier to launch than traditionally-sized satellites. First developed and launched by Stanford University, this new class of small satellites soon saw a multi-organizational movement towards standardization with the development of the CubeSat[1] design program [4]. Initially defined as a 10 cm cube with a mass not exceeding 1 kg [5], the goal of this program was to develop a universal standard for micro and smaller class satellites in order to increase the ease of development and streamline the production of these low cost space missions for the public and private sector alike. Cubesats have since seen wide-spread

[1]See cubesat.org for more info. *Last accessed on 10/21/2019.*

adoption [6]. The success of microsatellites and their even smaller counterparts, nano and picosatellites, has been undeniable with over one thousand CubeSats launched to date [1]. The usage of these satellites has grown from purely academic research to commercial use [1, 7]. The total number of CubeSats and nanosatellites launched to date and by organization type can be seen in Figure 1.1 and Figure 1.2.

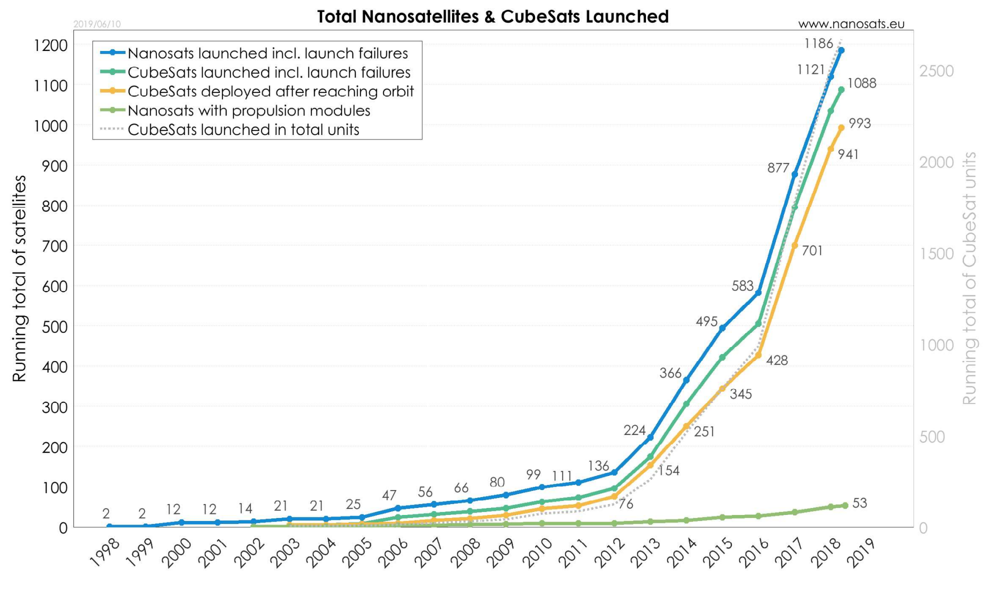

Figure 1.1: Total Number of Launches [1]

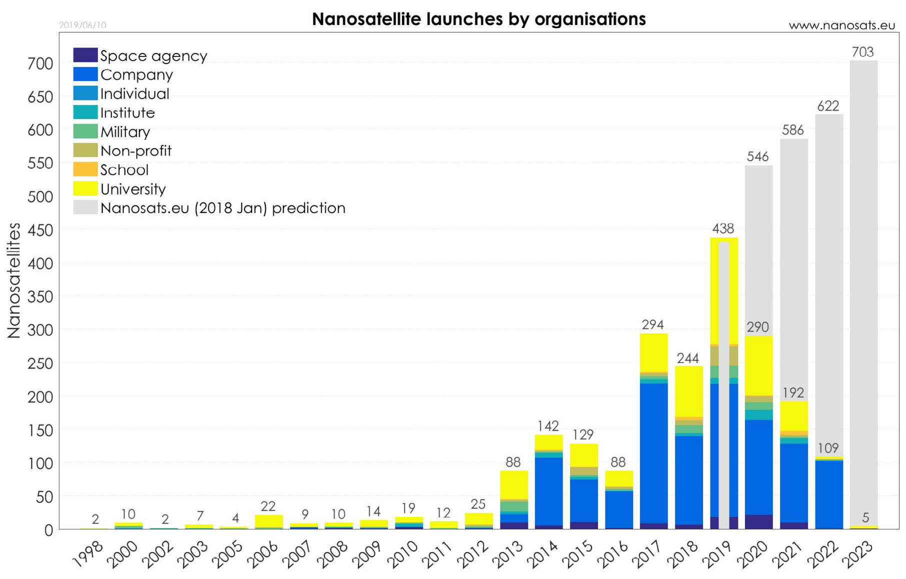

Figure 1.2: Launches by Organization Type [1]

1.2 Advantages and Disadvantages of Small Satellites

The low development and launch cost of small satellites is undoubtedly their largest advantage over traditionally-sized satellites which can reach hundreds of millions of dollars in total financial cost. Estimated development costs for a CubeSat range from \$50,000 to \$200,000 depending on size and complexity and current CubeSat launch costs range from \$50,000 to \$200,000 with an expected drop to between \$10,000 and \$85,000 by the year 2020 [8]. This low cost advantage is driven by their small size, which in turn is their largest disadvantage. Small satellites, with payload capacities ranging from 1 kg and > 72 in^3 (1U CubeSate) to 24 kg and 864 in^3 (12U CubeSat) [5], are incapable of supporting many of the payloads, components and subsystems which larger satellites can. Small satellites are unable to carry the large imaging systems used for highly-detailed earth surface observation [9] and imaging, full-scale attitude-control-systems [10], or sizeable thrust engines and the propellant or power-systems required to fuel them. For example, the Earth-imaging satellite Landsat 8 houses an imaging system capable of a 30 m per pixel resolution while orbiting at a distance of roughly 700 km but has a dry mass of 1512 kg and cost $855 million dollars [11]. Compare this to the M-Cubed satellite project being developed by the University of Michigan which only has a budget of $100,000 and hopes to achieve medium resolution (200 m per pixel) optical imaging of the Earth's surface at an altitude of 650 km [12].

1.3 Current Orbit Maintenance Solutions for LEO Satellites

The need for large and expensive imaging equipment for surface observation could be reduced by minimizing the effective observation range by decreasing the satellite's orbital altitude to the extreme lower limits of Low-Earth-Orbit where the altitude ranges span from 2000 km to as low as 100 km [13]. This solution presents additional problems though, chiefly, the presence of significant atmospheric density to impart drag on the satellite as it orbits. The increased drag force causes the satellite to rapidly lose velocity and altitude. Therefor the majority of satellites placed in Low-Earth-Orbit are fitted with various systems to maintain the orbit altitude over the course of the satellite's mission duration.

An example which illustrates the station-keeping requirements of a very low Earth orbit satellite mission is the Gravity Field and Steady-State Ocean Circulation Explorer or GOCE satellite which was developed and launched by the European Space

Agency in March of 2009 [14]. The GOCE orbited for 4 years and 8 months at an altitude of approximately 250 km before depleting its 40 kg of Xenon propulsion fuel for the QinetiQ T5 ion thruster used to counteract the atmospheric drag forces [14–16]. Using an average ISP of 2000 s [16] and a launch mass of 1100 kg [14], the yearly delta-v requirement for this mission comes out to be 177 m/s. It is important to note that while the GOCE is roughly 400 times larger than a 6U CubeSat [5,14], this still gives a reasonable estimate of the delta-v requirements for smaller satellites as well since aerodynamic drag scales linearly with spacecraft size, more specifically the presented drag area.

1.3.1 Using Thrust to Maintain Altitude

The most common method of orbit maintenance is to use a thrust force to increase the orbital velocity of the satellite and subsequently the altitude, such as done on the GOCE mission. The process of using controlled thrust forces to keep a satellite on its assigned orbit is known as orbital station-keeping [17]. Traditionally, this process is achieved using either liquid fuel chemical propulsion or cold-gas chemical propulsion. These methods can achieve high magnitudes of thrust and perform well on larger satellites where their is enough payload volume to house the propellant required for operation. Small satellites, while requiring smaller magnitudes of thrust due to their size, are still hampered by their reduced payload capacity and these systems are not usually ideal for use [9]. Recent technological developments in the field of electric propulsion have seen the development of miniature propulsion systems, such as ion, hall effect, electrospray, and pulsed plasma thrusters, which would be ideal for the use in small satellites [18–21]. These devices have a much lower impact on payload volume, but at the cost of very small thrust magnitudes which may be unable to overcome the aerodynamic drag forces present at extreme low orbits as well as adding to the satellite's total electrical power draw.

1.3.2 Tethers

Tethers have been used in space since as early as 1966 when the Gemini XI mission tested the concept of artificial gravity by spinning two spacecraft connected via a tether [22]. Now used for applications such as stabilization and attitude control, momentum exchange, maintaining relative positions for constellations of satellites, and also as a propulsive device via interaction with the Earth's magnetic field [23], tethers are seeing widespread research throughout the space industry. Tethers have yet to be successfully used in conjunction with small satellites [24–28], but are of a continued interest with a promising planned mission by the U.S. Naval Research Laboratory being design to test a propulsion concept that would be able to alter the altitude of the satellite system by several kilometers a day [29]. This technology, while exciting, still remains to be proven as feasible for small satellite usage.

1.3.3 Lift as a Solution for Low Earth Orbit Maintenance

The existence of drag effects at Low-Earth-Orbit altitudes also indicates the possibility for significant lift generation. Therefore, one possible solution to some of the problems that the micro and smaller classes of satellites face lies in the usage of integrated lifting-surfaces to act as a propellant-less orbital maneuvering tool. Small satellites could potentially gain the ability to passively alter their orbital parameters, such as inclination, improve their endurance, and reduce any amount of thrust needed for orbital control by leveraging the aerodynamic forces imparted upon the satellite at low orbit altitudes. The successful integration and operation of aerodynamic lifting and control surfaces onto small satellites could give way to an entire new breed of satellites which would be part low-orbit satellite and and part high-altitude aircraft.

CHAPTER 2

METHODOLOGY AND DESIGN OF EXPERIMENT

Research is what I'm doing
when I don't know what I'm doing.

—Wernher von Braun

2.1 Assumptions, Constraints, and Parameters

A high-fidelity physics-based simulation was developed in order to study the effects of integrated lifting surfaces on a small satellite and determine their feasibility. This chapter will cover the algorithms and equations implemented to govern the simulation used to conduct this study. The assumptions, characteristic parameters, and constraints chosen will also be detailed with explanations for their selection. The code for this simulation can be found in the Appendix.

Various assumption and constraints were placed on the simulation in this study in order to reduce its cost in both time and computation memory usage. The scope of the simulation was limited to three-degrees-of-freedom (3DOF), and the satellite was assumed to be a rigid-body. A more detailed description of the assumptions, constraints, and various parameters which drove the simulation can be found in the following subsections.

2.1.1 Earth Gravitation Model

This simulation was designed around the assumption of a two-body problem given the low altitude orbits being modeled. The only perturbations modeled for this study were the oblatness of the earth, the J_2 effect, and atmospheric lift and drag.

2.1.2 Satellite and Lifting Surface(s) Model

The characteristic parameters chosen to model the satellite and its integrated lifting surfaces for this study are in Table 2.1. The cross-sectional area for the satellite, $A_{Satellite}$, was chosen as the small side of the rectangular profile described in [5] for a 6U sized satellite, ($12 \times 24 \times 36\,\text{cm}$) and the mass, $m_{satellite}$ (12 kg) as the standard maximum allowed for the size. A representative rendering of a 6U CubeSat is shown in Figure 2.1 which comes from [2]. The coefficient of drag for the satellite body, C_{D0}, was picked as a rough estimate for an angled cube [30]. It should be noted that this simple estimation with realistic bounds was enough for the purposes of this study, and that it's effect is independent to the effects produced by the lifting surfaces. A lower value would have yielded less total drag and a higher value of more total drag. It should also be noted that it is assumed that the satellite has a fixed attitude which is always as commanded. The satellite would require an advanced control system to achieve this in reality but modeling this was outside the scope of this study.

An infinitely thin, flat-plate wing was modeled as the primary lifting surface shape as it is the basic element of lifting surfaces and to reduce the overall complexity of the model. The area chosen for total lifting surface area was selected by computing the length-wise side panel area of the satellite and multiplying by three, suggesting a fold-out design. This parameter represents a semi-realistic value so that noticeable and appropriate lift magnitudes could be expected. A more detailed description of the lifting surfaces model is discussed in Section 2.2.3.

Table 2.1: Satellite Characteristic Parameters

Parameter	Value	Units
$A_{satellite}$	0.2880	m^2
A_{wing}	0.0648	m^2
$m_{satellite}$	12	kg
C_{D0}	0.8	

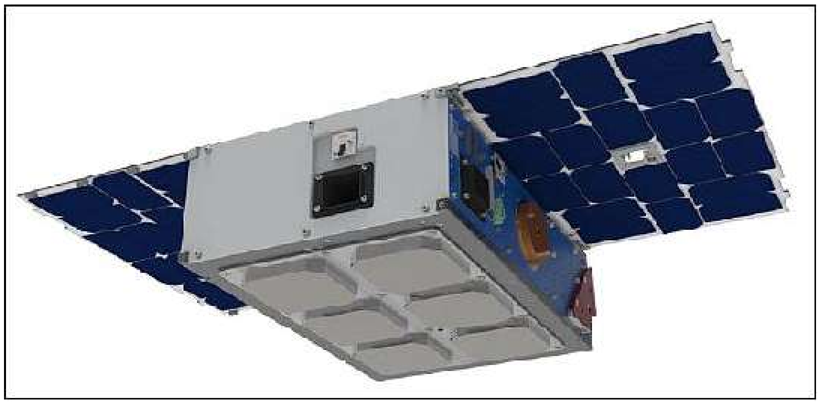

Figure 2.1: 6U CubeSat Render [2]

2.1.3 Atmospheric Density Model

The density distribution used to model the terrestrial atmosphere for this simulation is defined by [31] as Equation 2.1 which gives density as a continuous function of altitude. A plot of this function can be seen in Figure 2.2.

$$\rho(r) = \rho_0 \exp\left[-\frac{(r - r_e)}{H}\right] \tag{2.1}$$

Altitude is represented by r and the Earth's radius by $r_e = 6.4x10^6$m. $H = 8.5 \times 10^3 m$ is the atmospheric scale height and $\rho_0 \approx 1.3\ \frac{kg}{m^3}$. This model holds well for altitudes below 150km [31].

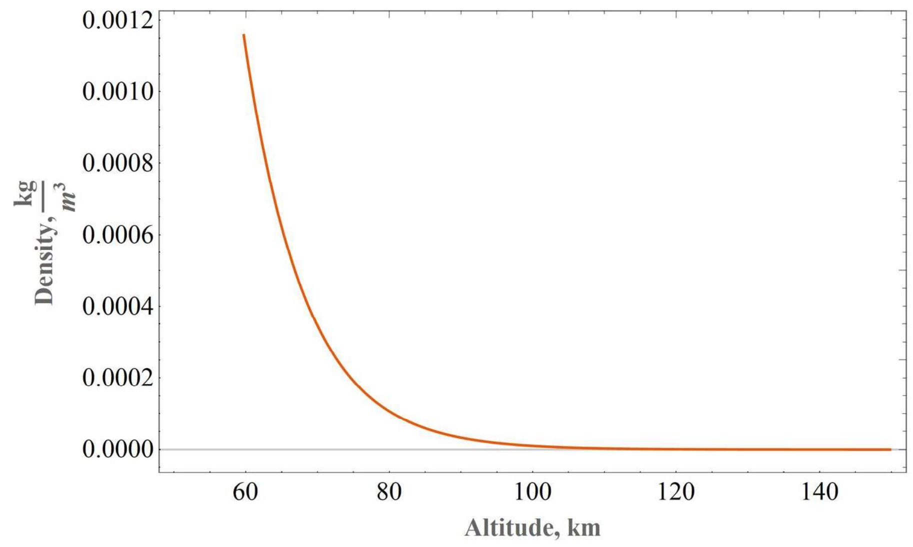

Figure 2.2: Atmospheric Density as a Function of Altitude

2.2 Forces and Perturbations

Any physics-based simulation will be ultimately governed by mathematical equations which represent the forces effecting the system. This section contains a detailed description of the forces and their perturbations which are modeled in this simulation.

2.2.1 Two-Body Gravitation Equations

The standard gravitational equation for satellites are defined in Ref. [13] as Equation 2.2. This can be expanded to form the necessary state-vector equations for integration as shown in Equation 2.2 through Equation 2.5.

$$F_G = -\frac{\mu}{r^2}\left(\frac{\vec{r}}{r}\right) \tag{2.2}$$

$$F_{G_x} = -\frac{\mu\, x}{\left(\sqrt{x^2+y^2+z^2}\right)^3} \tag{2.3}$$

$$F_{G_y} = -\frac{\mu\, z}{\left(\sqrt{x^2+y^2+z^2}\right)^3} \tag{2.4}$$

$$F_{G_z} = -\frac{\mu\, z}{\left(\sqrt{x^2+y^2+z^2}\right)^3} \tag{2.5}$$

The standard gravitational parameter of the Earth is represented by μ and has a value of $398600.436\frac{km^3}{s^2}$

2.2.2 J_2 Effect

The J_2 perturbation is an effect imparted on Earth orbiting bodies due to the oblateness of the Earth about its equator [32]. This effect is minimized in this study however by having the satellite orbit in a nearly-equatorial orbit. These equations are defined in Ref. [32] as Equation 2.6 through Equation 2.8.

$$F_{J2_x} = -\frac{3J_2\mu\, r_e^2\, x\left(x^2+y^2-4z^2\right)}{2\left(x^2+y^2+z^2\right)^{7/2}} \tag{2.6}$$

$$F_{J2_y} = -\frac{3J_2\mu\, r_e^2\, y\left(x^2+y^2-4z^2\right)}{2\left(x^2+y^2+z^2\right)^{7/2}} \tag{2.7}$$

$$F_{J2_z} = -\frac{3J_2\mu\, r_e^2\, z\left(-3\left(x^2+y^2\right)+2z^2\right)}{2\left(x^2+y^2+z^2\right)^{7/2}} \tag{2.8}$$

The J_2 parameter is defined as $1.082635666551098 \times 10^{-2}$ and $r_e = 6378.137km$, which is the Earth's radius.

2.2.3 Lift and Drag

The standard continuum model used for most cases in the field of aerodynamics begins to break down at altitudes around 75 km [33]. This is because the atmospheric density is so low, and the mean free path between the air molecules is so large that the flow ceases to act like a body of fluid and needs to instead be modeled using the kinetic theory of gases [34] which treats the air molecules as a distribution of particles rather than a continuous fluid body. The Knudsen number is a quantity which is used to determine the rarefaction of a flow and is expressed in Equation 2.9 where λ is the mean free path and L is some reference length [34]. The flow regimes which correspond to different Knudsen numbers are shown in Table 2.2 [35].

$$\text{Kn} = \frac{\lambda}{L} \tag{2.9}$$

$Kn < 0.01$	Continuum flow
$0.01 < Kn < 10$	Slip flow
$Kn > 10$	Free-molecular flow

Table 2.2: Knudsen Number Flow Regimes

The Knudsen number can be calculated by finding the mean free path from Equation 2.10, where k_B is the Boltzmann constant, T is the gas temperature,d^2_{avg} is the molecular diameter, and p is the total pressure [36].

$$\lambda = \frac{k_B\,T}{\sqrt{2}\,\pi\,d^2_{\text{avg}}\,p} \tag{2.10}$$

Figure 2.3 shows the relationship between the Knudsen number and altitude for the given constraints which correspond to the reference length for a Cubesat and the average diameter of molecular nitrogen which is generally used for computing mean free path in the atmosphere [36].

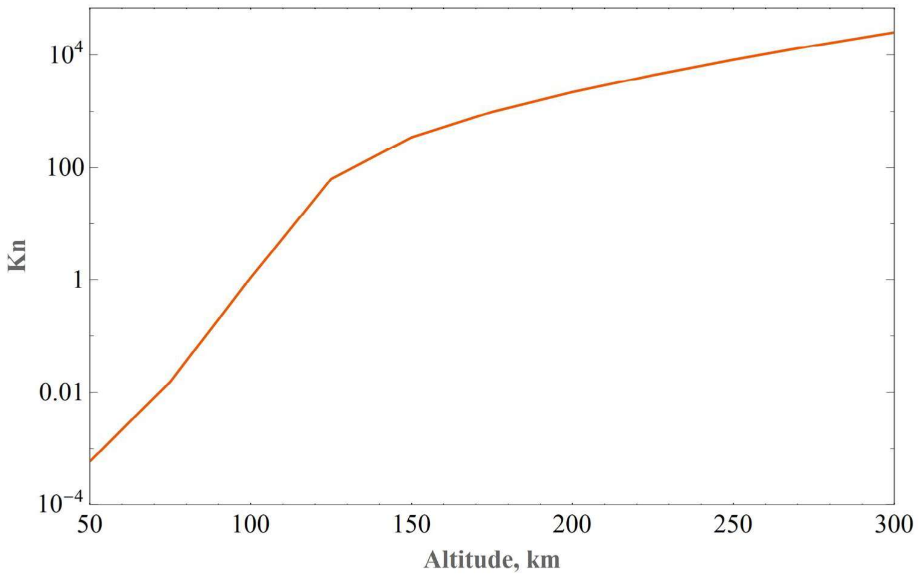

Figure 2.3: Knudsen Number vs. Altitude for $L = 10$ cm and $d_{\text{avg}} = 4.11 \times 10^{-10}$ m

Equations for the coefficients of lift and drag can be empirically derived for two different cases, diffuse reflection and specular reflection. Diffuse reflection assumes that the air molecules achieve complete thermal equilibrium with the incident surface, are absorbed, and then evaporated from the surface at some scattering angle. [37]. Specular assumes that no energy is exchanged in the interaction between the air molecule and the incident surface aside from the a change in the velocity vector angle in which the angle of reflection is the same as the angle of incidence [37]. This angle of incidence is expressed in Equation 2.12 and Equation 2.14 as the angle of attack.

The equations used to model lift and drag can be seen in Equations 2.11 and 2.12. C_{L_α} and C_{D_α} come from the assumption of an infinitely thin flat plate [38] in a completely rarefied flow regime for specular reflection which are consistent with the work done in Reference [33]. A drag polar curve for these equations can be seen in Figure 2.4. The assumption of an infinitely thin flat plate results in no lift or additional drag being generated when $\alpha = 0$, therefore at this condition, the satellite with integrated lifting surfaces behaves the same as an identical one without lifting surfaces. The drag force always acts in opposition to the satellite's velocity vector in any given frame [38] so it can be readily be expressed in Earth-Centered-Earth-Fixed, or ECEF, Cartesian coordinates. The lift force, however, always acts in the vertical axis of the local tangent plane [38]. The lift force vector is first defined in the North-East-Down, or NED, frame and then transformed into the ECEF frame so that it can be incorporated into the total force equation and then integrated to obtain the state-vectors. This transformation will be covered in more detail in Section 2.3.

$$L = \frac{1}{2} C_L \rho(r)\, v^2\, A \tag{2.11}$$

$$C_{L_\alpha} = \frac{4}{\sqrt{\pi}\, s} \cos\alpha \, \sin\alpha \, e^{-(s\,\sin\alpha)^2} + \cos\alpha \left(4\,\sin^2\alpha + \frac{2}{s^2}\right) \mathrm{erf}(s\,\sin\alpha) \tag{2.12}$$

$$D = \frac{1}{2} C_{D_\alpha} \rho(r)\, v^2\, A \tag{2.13}$$

$$C_{D_\alpha} = \left(\frac{4}{\sqrt{\pi}\, s} \sin^2\alpha\right) e^{-(s\,\sin\alpha)^2} + 2\sin\alpha \left[2(\sin^2\alpha + \frac{1}{2\,s^2})\right] \mathrm{erf}(s\,\sin\alpha) \tag{2.14}$$

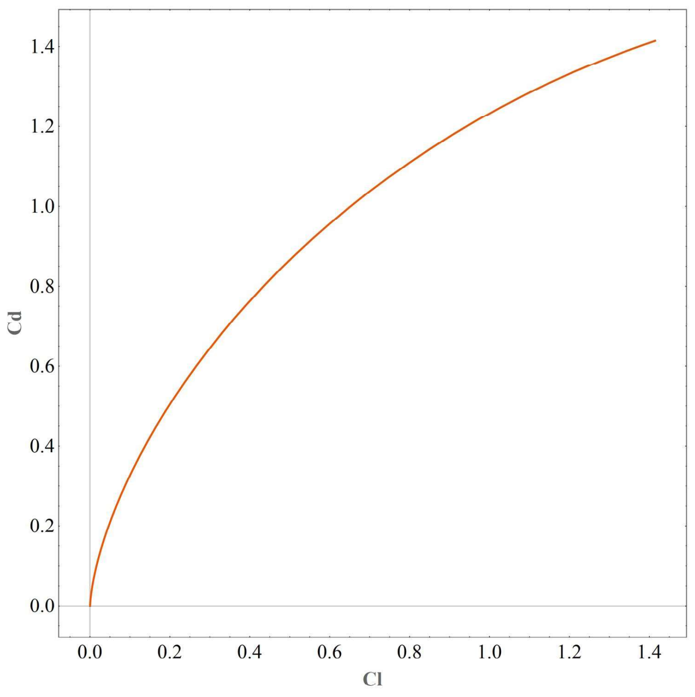

Figure 2.4: Rarefied Flow Flat Plate Drag Polar for the Case of Specular Reflection

2.2.4 Thrust Matching

Equations to simulate a thrust force were introduced to this simulation for part of the study in order to determine the thrust requirements for cancelling the effect of drag on the satellite to maintain orbit. This was achieved by simply inverting the drag force equations being modeled in Equation 2.13, such that just enough thrust was produced to counteract drag. Since an actual engine was not being modeled, using real thrust equations was not necessary.

2.3 Coordinate Frame Transformations

This section outlines the equations used for the various coordinate system transformations used throughout the simulation. Coordinate systems used include NED, ECEF, and perifocal which is expressed using the Keplerian orbital elements.

2.3.1 Cartesian State Vectors⟶ Keplerian Orbit Elements

The follow equations as described in [39] allow for the conversion of position and velocity vectors in the ECEF/ECI frame to the Keplerian elements used for describing the satellite and orbit at any given time. A diagram of the Kelperian elements can be seen in Figure 2.5 as shown in [3].

$$\vec{h} = \vec{r} \times \vec{v} \tag{2.15}$$

$$\vec{e} = \frac{\vec{v} \times \vec{h}}{\mu} \tag{2.16}$$

$$\vec{n} = (0,0,1)^T \times \vec{h} \tag{2.17}$$

$$\nu = \begin{cases} \arccos \frac{\vec{e}\cdot\vec{r}}{||\vec{e}||\,||\vec{r}||} & \vec{r}\cdot\vec{v} \geq 0 \\ 2\pi - \arccos \frac{\vec{e}\cdot\vec{r}}{||\vec{e}||\,||\vec{r}||} & else \end{cases} \tag{2.18}$$

$$i = \arccos \frac{h_z}{||\vec{h}||} \tag{2.19}$$

$$e = ||\vec{e}|| \tag{2.20}$$

$$\Omega = \begin{cases} \arccos \frac{\vec{n}\cdot\vec{e}}{||\vec{n}||\,||\vec{e}||} & e_z \geq 0 \\ 2\pi - \arccos \frac{\vec{n}\cdot\vec{e}}{||\vec{n}||\,||\vec{e}||} & e_z < 0 \end{cases} \tag{2.21}$$

$$\omega = \begin{cases} \arccos \frac{n_x}{||\vec{n}||} & n_y \geq 0 \\ 2\pi - \arccos \frac{n_x}{||\vec{n}||} & n_y < 0 \end{cases} \tag{2.22}$$

$$a = \left(\frac{2}{||\vec{r}||} - \frac{||\vec{v}||^2}{\mu} \right)^{-1} \tag{2.23}$$

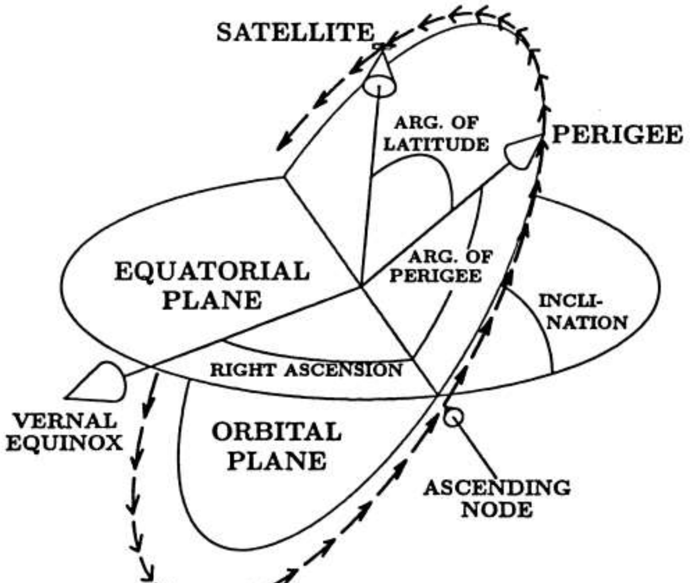

Figure 2.5: Keplerian Parameters for an Orbit [3]

2.3.2 Keplerian Orbit Elements$\longrightarrow$ Cartesian State Vectors

Equation 2.24 through Equation 2.33, given in [40] describe the steps for the inverse of Equation 2.15 through Equation 2.23. While the orbital elements are useful for more easily interpreting the state of the orbit and satellite's position in it they are not useful in the state-vector integration.

$$p = a\,(1 - e^2) \tag{2.24}$$

$$r = \frac{p}{1 + e\;\cos(\nu)} \tag{2.25}$$

$$\vec{r_p} = (r\;\cos(\nu), r\;\sin(\nu), 0) \tag{2.26}$$

$$\vec{v_p} = (-\sqrt{\frac{\mu}{p}}\;\sin(\nu), \sqrt{\frac{\mu}{p}}\,(e + \cos(\nu)), 0) \tag{2.27}$$

$$T1 = \begin{pmatrix} \cos(\Omega)\;\cos(\omega) - \sin(\Omega)\;\cos(i)\;\cos(\omega) \\ \sin(\Omega)\;\cos(\omega) + \cos(\Omega)\;\cos(i)\;\sin(\omega) \\ \sin(i)\;\sin(\omega) \end{pmatrix} \tag{2.28}$$

$$T2 = \begin{pmatrix} -\cos(\Omega)\;\sin(\omega) - \sin(\Omega)\;\cos(i)\,cos(\omega) \\ -\sin(\Omega)\;\sin(\omega) + \cos(\Omega)\;\cos(i)\;\cos(\omega) \\ \sin(i)\;\cos(\omega) \end{pmatrix} \tag{2.29}$$

$$T3 = \begin{pmatrix} \sin(\Omega)\;\sin(i) \\ -\cos(\Omega)\;\sin(i) \\ \cos(i) \end{pmatrix} \tag{2.30}$$

$$T = \begin{pmatrix} T1 & T2 & T3 \end{pmatrix} \tag{2.31}$$

$$\vec{r} = T \cdot \vec{r_p} \tag{2.32}$$

$$\vec{v} = T \cdot \vec{v_p} \tag{2.33}$$

2.3.3 ECEF$\longrightarrow$ Geodetic

The transformation from ECEF to Geodetic coordinates is necessary as part of the transformation of a vector described in the NED frame to the ECEF frame [3]. The geodetic latitude and longitude are used as the rotation angles in that transformation's direction-cosine matrix. The following set of equations come from [41] and outline a computationally efficient algorithm for deriving latitude and longitude from an ECEF position vector. Please note that the variable names used in this section only (Equation 2.34 through Equation 2.57), are unique to this algorithm and should not be confused with variables with similar names referenced in the rest of this paper.

$$w^2 = x^2 + y^2 \tag{2.34}$$

$$l = e^2/2 \tag{2.35}$$

$$m = (w/a)^2 \tag{2.36}$$

$$n = z^2(1 - e^2)/a^2 \tag{2.37}$$

$$p = (m + n - 4l^2)/6 \tag{2.38}$$

$$G = m\,n\,l^2 \tag{2.39}$$

$$H = 2p^3 + G \tag{2.40}$$

$$C = \frac{\sqrt[3]{H + G + 2\sqrt{H\,G}}}{\sqrt[3]{2}} \tag{2.41}$$

$$i = -(2l^2 + m + n)/2 \tag{2.42}$$

$$P = p^2 \tag{2.43}$$

$$\beta = i/3 - C - P/C \tag{2.44}$$

$$k = l^2(l^2 - m - n) \tag{2.45}$$

$$t = \sqrt{\sqrt{\beta^2 - k} - (\beta + i)/2} - \mathrm{sign}(m - n)\sqrt{|(\beta - i)/2|} \tag{2.46}$$

$$F = t^4 + 2i\,t^2 + 2l(m - n)t + k \tag{2.47}$$

$$dF/\mathrm{dt} = 4t^3 + 4i\,t + 2l(m - n) \tag{2.48}$$

$$\Delta t = -F/(dF/\mathrm{dt}) \tag{2.49}$$

$$u = t + \Delta t + l \tag{2.50}$$

$$v = t + \Delta t - l \tag{2.51}$$

$$w = \sqrt{w^2} \tag{2.52}$$

$$\phi = \arctan(z\,u, w\,v) \tag{2.53}$$

$$\Delta w = w(1 - 1/u) \tag{2.54}$$

$$\Delta z = z(1 - 1(1 - e^2)/v) \tag{2.55}$$

$$h = \text{sign}(u - 1)\sqrt{(\Delta w)^2 + (\Delta z)^2} \tag{2.56}$$

$$\lambda = \arctan(x, y) \tag{2.57}$$

2.3.4 NED$\longrightarrow$ ECEF

The transformation for a vector defined in the NED frame to the ECEF frame relies on a directional-cosine matrix which is defined using the geodetic latitude and longitude from an ECEF reference position vector. This matrix can be seen in Equation 2.58 and changes as the ECEF position vector does. Once the Directional-Cosine-Matrix, DCM is obtained all that is required for the transformation is the matrix multiplication show in Equation 2.59. The ECEF/ECI frame can be seen in Figure 2.6 as shown in [3].

$$\text{DCM}_{NED}^{ECEF} = \begin{pmatrix} -\cos(\lambda)\,\sin(\phi) & -\sin(\lambda) & -\cos(\lambda)\,\cos(\phi) \\ -\sin(\lambda)\,\sin(\phi) & \cos(\lambda) & -\sin(\lambda)\,\cos(\phi) \\ \cos(\phi) & 0 & -\sin(\phi) \end{pmatrix} \tag{2.58}$$

$$(u1, u2, u3)_{ECEF}^{T} = \text{DCM}_{NED}^{ECEF}\,(w1, w2, w3)_{NED}^{T} \tag{2.59}$$

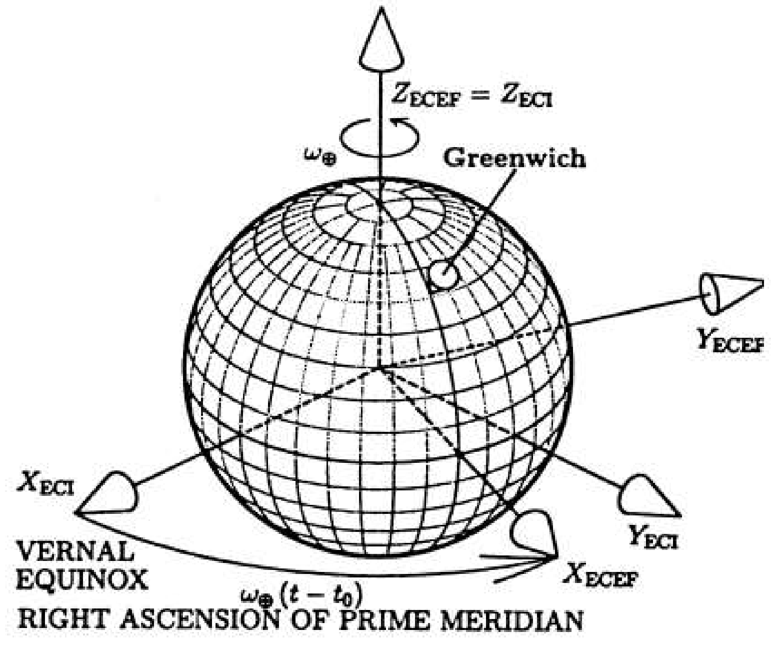

Figure 2.6: ECEF/ECI Coordinates [3]

2.4 4th-Order Runge-Kutta Integration

An RK4 integration method was used in this simulation to obtain the state-vectors from the total force-vector defined in Section 2.2. The general form of this method is shown in [42], and its adaptation for this simulation can be seen in Equation 2.60 through Equation 2.65 where dt is the timestep for the simulation. This method was chosen for its accuracy and low computational cost.

$$\begin{aligned} \mathrm{k1}_x &= \mathrm{dt}\, f_x(x, y, z, v_x, v_y, v_z) \\ \mathrm{k1}_y &= \mathrm{dt}\, f_y(x, y, z, v_x, v_y, v_z) \\ \mathrm{k1}_z &= \mathrm{dt}\, f_z(x, y, z, v_x, v_y, v_z) \end{aligned} \tag{2.60}$$

$$
\begin{aligned}
\mathrm{k2}_x &= \mathrm{dt}\, f_x(x + v_x\frac{\mathrm{dt}}{2}, y + v_y\frac{\mathrm{dt}}{2}, z + v_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \frac{\mathrm{k1}_x}{2}, v_y + \frac{\mathrm{k1}_y}{2}, v_z + \frac{\mathrm{k1}_z}{2}) \\
\mathrm{k2}_y &= \mathrm{dt}\, f_y(x + v_x\frac{\mathrm{dt}}{2}, y + v_y\frac{\mathrm{dt}}{2}, z + v_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \frac{\mathrm{k1}_x}{2}, v_y + \frac{\mathrm{k1}_y}{2}, v_z + \frac{\mathrm{k1}_z}{2}) \\
\mathrm{k2}_z &= \mathrm{dt}\, f_z(x + v_x\frac{\mathrm{dt}}{2}, y + v_y\frac{\mathrm{dt}}{2}, z + v_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \frac{\mathrm{k1}_x}{2}, v_y + \frac{\mathrm{k1}_y}{2}, v_z + \frac{\mathrm{k1}_z}{2})
\end{aligned}
\tag{2.61}
$$

$$
\begin{aligned}
\mathrm{k3}_x &= \mathrm{dt}\, f_x(x + v_x\frac{\mathrm{dt}}{2} + \mathrm{k1}_x\frac{\mathrm{dt}}{4}, y + v_y\frac{\mathrm{dt}}{2} + \mathrm{k1}_y\frac{\mathrm{dt}}{4}, z + v_z\frac{\mathrm{dt}}{2} + \mathrm{k1}_z\frac{\mathrm{dt}}{4}, ... \\
&\qquad ...v_x + \frac{\mathrm{k2}_x}{2}, v_y + \frac{\mathrm{k2}_y}{2}, v_z + \frac{\mathrm{k2}_z}{2}) \\
\mathrm{k3}_y &= \mathrm{dt}\, f_y(x + v_x\frac{\mathrm{dt}}{2} + \mathrm{k1}_x\frac{\mathrm{dt}}{4}, y + v_y\frac{\mathrm{dt}}{2} + \mathrm{k1}_y\frac{\mathrm{dt}}{4}, z + v_z\frac{\mathrm{dt}}{2} + \mathrm{k1}_z\frac{\mathrm{dt}}{4}, ... \\
&\qquad ...v_x + \frac{\mathrm{k2}_x}{2}, v_y + \frac{\mathrm{k2}_y}{2}, v_z + \frac{\mathrm{k2}_z}{2}) \\
\mathrm{k3}_z &= \mathrm{dt}\, f_z(x + v_x\frac{\mathrm{dt}}{2} + \mathrm{k1}_x\frac{\mathrm{dt}}{4}, y + v_y\frac{\mathrm{dt}}{2} + \mathrm{k1}_y\frac{\mathrm{dt}}{4}, z + v_z\frac{\mathrm{dt}}{2} + \mathrm{k1}_z\frac{\mathrm{dt}}{4}, ... \\
&\qquad ...v_x + \frac{\mathrm{k2}_x}{2}, v_y + \frac{\mathrm{k2}_y}{2}, v_z + \frac{\mathrm{k2}_z}{2})
\end{aligned}
\tag{2.62}
$$

$$
\begin{aligned}
\mathrm{k4}_x &= \mathrm{dt}\, f_x(x + v_x\,\mathrm{dt} + \mathrm{k2}_x\frac{\mathrm{dt}}{2}, y + v_y\,\mathrm{dt} + \mathrm{k2}_y\frac{\mathrm{dt}}{2}, z + v_z\,\mathrm{dt} + \mathrm{k2}_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \mathrm{k3}_x, v_y + \mathrm{k3}_y, v_z + \mathrm{k3}_z) \\
\mathrm{k4}_y &= \mathrm{dt}\, f_y(x + v_x\,\mathrm{dt} + \mathrm{k2}_x\frac{\mathrm{dt}}{2}, y + v_y\,\mathrm{dt} + \mathrm{k2}_y\frac{\mathrm{dt}}{2}, z + v_z\,\mathrm{dt} + \mathrm{k2}_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \mathrm{k3}_x, v_y + \mathrm{k3}_y, v_z + \mathrm{k3}_z) \\
\mathrm{k4}_z &= \mathrm{dt}\, f_z(x + v_x\,\mathrm{dt} + \mathrm{k2}_x\frac{\mathrm{dt}}{2}, y + v_y\,\mathrm{dt} + \mathrm{k2}_y\frac{\mathrm{dt}}{2}, z + v_z\,\mathrm{dt} + \mathrm{k2}_z\frac{\mathrm{dt}}{2}, ... \\
&\qquad ...v_x + \mathrm{k3}_x, v_y + \mathrm{k3}_y, v_z + \mathrm{k3}_z)
\end{aligned}
\tag{2.63}
$$

$$\begin{aligned}
x_{\text{new}} &= x + \text{dt}\, v_x + \text{dt}\frac{(\text{k}1_x + \text{k}2_x + \text{k}3_x)}{6} \\
y_{\text{new}} &= y + \text{dt}\, v_y + \text{dt}\frac{(\text{k}1_y + \text{k}2_y + \text{k}3_y)}{6} \\
z_{\text{new}} &= z + \text{dt}\, v_z + \text{dt}\frac{(\text{k}1_z + \text{k}2_z + \text{k}3_z)}{6}
\end{aligned} \tag{2.64}$$

$$\begin{aligned}
v_{x_{\text{new}}} &= v_x + \frac{\text{k}1_x + 2\,\text{k}2_x + 2\,\text{k}3_x + \text{k}4_x}{6} \\
v_{y_{\text{new}}} &= v_y + \frac{\text{k}1_y + 2\,\text{k}2_y + 2\,\text{k}3_y + \text{k}4_y}{6} \\
v_{z_{\text{new}}} &= v_z + \frac{\text{k}1_z + 2\,\text{k}2_z + 2\,\text{k}3_z + \text{k}4_z}{6}
\end{aligned} \tag{2.65}$$

2.5 Angle-of-Attack Control Loop

A simple control-loop as described in [43] was introduced to this simulation to modulate the angle-of-attack driving the lift and induced-drag forces. This was done to determine how a dynamic angle-of-attack would effect the simulation behavior. A gain value of $K = 0.1$ was used as it provided sufficient control to achieve the ideal angle-of-attack before the simulation ended but no optimization was performed as this was not the goal of this study. Equation 2.68 shows the equation used to recalculate the "commanded" angle-of-attack where v is the magnitude of the satellite's velocity state-vector. Limits of 45° and −15° where placed on the angle-of-attack to eliminate erratic behavior.

$$\text{Error}_{\text{altitude}} = r_{\text{desired}} - r_{\text{actual}} \tag{2.66}$$

$$\text{Error}_{v_z} = 0 - \frac{r_i - r_{i-1}}{\text{dt}} \tag{2.67}$$

$$\alpha_{\text{cmd}} = (\text{Error}_{\text{altitude}} + \text{Error}_{v_z})\frac{K}{v} \tag{2.68}$$

2.6 Algorithm Outline

This section provides a high level description of the overall simulation. Please reference the actual code in the Appendix for an explicit description.

2.6.1 Inputs and Outputs

Table 2.3 and Table 2.4 describe the various inputs[1] used to to drive the simulation and the outputs[2] from each run.

Table 2.3: Simulation Inputs

Input Parameter	Description	Units
dt	Iteration timestep size	s
t_{stop}	Simulation step limit	s
$\vec{oe}_{int}$	$(a, e, i, \Omega, \omega, \nu)$ at initial timestep	(km, N/A, deg, ...)
α_{int}	Initial angle-of-attack	deg

[1]For the non-control-loop variant of the simulation the initial α is fixed throughout the entirety of the simulation run.

[2]Each output described in the table is recorded for every timestep the simulation is run.

Table 2.4: Simulation Outputs

Output Parameter	Description	Units
t	Simulation time	s
$\vec{r_{ECEF}}$	Satellite position-vector in ECEF	km
$\vec{v_{ECEF}}$	Satellite velocity-vector in ECEF	km/s
$\vec{F_{tot}}$	Total force-vector imparted on satellite	kN
$\vec{F_G}$	Gravity force-vector	kN
$\vec{F_L}$	Lift force-vector	kN
$\vec{F_{D_0}}$	Body-drag force-vector	kN
$\vec{F_{D_i}}$	Induced drag force-vector	kN
$\vec{F_{Thrust}}$	Thrust force-vector	kN
$\rho(r)$	Atmospheric density at current satellite altitude	kg/m^3
$\vec{oe}$	$(a, e, i, \Omega, \omega, \nu)$ at current time	(km, N/A, deg, ...)
α	angle-of-attack	deg

2.6.2 Steps

The following steps provide a verbal depiction of the algorithm describing the simulation.

1. Define the initial characteristic parameters from the simulation inputs. This step includes calculating parameters such as C_{D_α}, $\rho(r)$, and the position and velocity vectors from the initial inputs.

2. Calculate initial forces. Once the force equations are defined their values for the initial timestep must be calculated using the characteristic parameters defined in Step 1 so that they can be integrated.

3. Create output log and write initial parameters and forces to it. The output log is created and given a specified filename when the initial row is written

to it. This output log is then appended with a new row every timestep with parameters unique to that iteration.

4. Perform RK4 iteration on current state-vectors. The total-force vector is integrated using the RK4 method with the current position and velocity state-vectors which results in an updated state-vector for the next timestep.

5. Calculate updated parameters and forces. The parameters and forces defined in Steps 1 and 2 are recalculated using the updated position and velocity state-vectors from Step 4.

6. Write parameters and forces to log file for the current timestep.

7. Perform altitude and vertical speed error checks and compute commanded angle-of-attack at each timestep[3].

8. Recalculate C_{L_α} and C_{D_α} from commanded angle-of-attack[4].

9. Return to Step 4 until either desired runtime is reached or minimum altitude[5] is hit.

10. Close log file.

11. Exit Simulation.

[3]This step only applies to the control-loop variant of the simulation

[4]This step only applies to the control-loop variant of the simulation

[5]The minimum altitude condition for this simulation was determined to be 50 km.

2.7 Validation

The simulation code written for this study was validated against results for CubeSat lifetimes in [44]. The results of this validation can be seen in Table 2.5. It should be noted that in the baseline study conducted many perturbations such as solar flux, and higher J factors were modeled whose effects outweigh the effects felt from atmospheric drag at higher altitudes. These perturbations were not modeled in this simulation as the focus of the study is on orbits with altitudes of 150km and therefor this simulation is not well suited for accurately modeling the lifetimes of satellites at altitudes which there is little to no atmospheric density. This difference accounts for the slight increase in lifetime in which this study's simulation predicted for a 200 km orbit versus the predicted lifetime by that of the simulation used in [44]. The variation of altitude versus time for this validation can be seen in Figure 2.7. These results indicate that simulation code is on par with that used in similar publications and can be considered a valid representation of the system.

Table 2.5: Simulation Runs for a CubeSat in a 200 km Sun-Synchronous Orbit

Size	Simulation Lifetime	Baseline Lifetime [44]
6U	3.49 days	3 days
5U	4.01 days	4 days
4U	4.56 days	4 days
3U	3.50 days	3 days
2U	2.53 days	2 days
1U	2.76 days	2 days

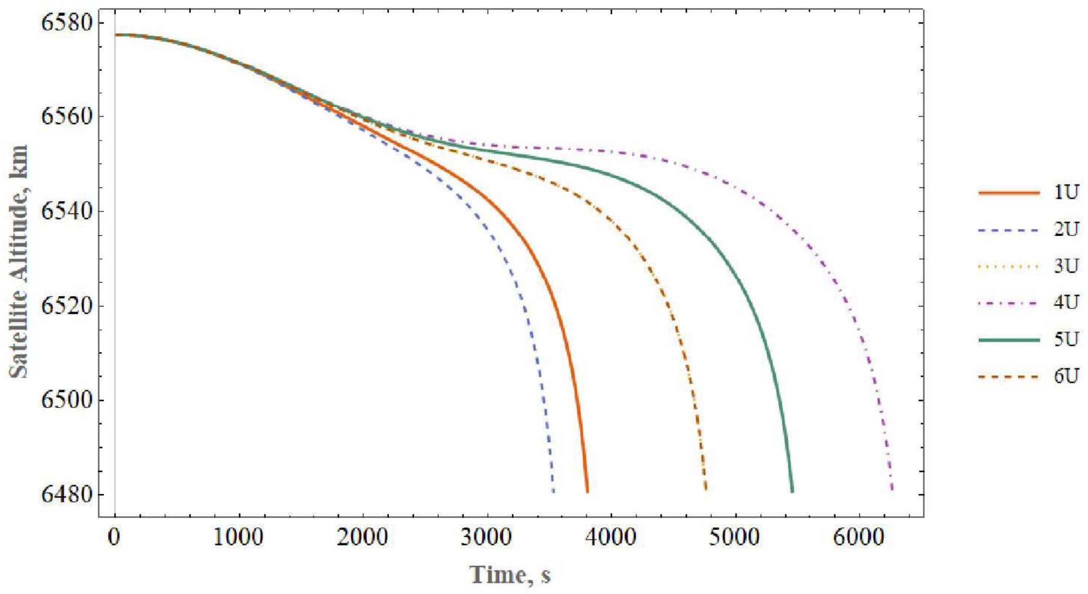

Figure 2.7: Satellite Altitude vs. Time for CubeSats in a 200 km Sun Synchronous Orbit

CHAPTER 3

SIMULATION ANALYSIS AND RESULTS

What I cannot create, I do not understand.

—Richard Feynman

3.1 Simulation Optimization

Initial testing of the simulation revealed that there was a large sensitivity to the timestep, dt, and large values resulted in unstable simulation behavior. This was attributed to the large impulses imparted on the system once it was within the significant drag regime. A series of convergence tests were then conducted in order to determine an adequate time-step for the study and to balance run time and memory costs. The results of the convergence test can be seen in Table 3.1. Simulation run time and output file size are seen to scale inversely with the time-step and the final orbit time converges around a value of 1603 seconds for the control case of a 6U Cubesat in an equatorial orbit at an altitude of 150km. It was determined from these results that a time-step value of 1 second would be adequate for the scope of this study as it achieved sufficient accuracy while minimizing computational costs.

Table 3.1: Time-Step Convergence Results

dt (s)	Orbit Time (s)	Run Time (s)	Output Size (KB)
0.01	1602.42	626.311	106845
0.1	1602.60	63.396	10686
0.5	1603.00	27.612	2101
1	1603.00	1.173	1048
5	1605.00	1.509	211
10	1610.00	0.909	106

3.2 Angle-of-Attack Variation

Variations on initial orbital parameters and applied angle-of-attack were implemented to generate respective behavioral profiles and document their effects on orbital endurance as well as other orbit characteristics in this study. A description of the analysis performed and the results can be found in this and the following sections.

A series of simulation runs was conducted with a varying angles-of-attack ranging from 0° to 40° in increments of 5° in order to study the effects of lift and drag on a low-orbiting satellite. The range of α was chosen so that significant variances in the effects on the orbital behavior imparted by the introduced aerodynamic lift and drag could be observed as these forces scale proportionally with α. It should be noted that boundary-layer separation effects do not occur in rarefied flows in which the simulation takes place and are not modeled in the kinetic gas theory [45]. The atmospheric densities and temperatures at 100 km present an extremely low Reynolds number environment in which thin plate airfoils have been shown to exhibit better performance than their traditional thick counterparts [46]. Table 3.2 shows the global operating parameters for the series of tests and Table 3.3 shows the operating parameters for each run and the length of time which the satellite's altitude remained above 50 km in the orbit.

Table 3.2: Global Parameters for α Runs

dt	t_{stop}	a_{int}	e_{int}	i_{int}	Ω_{int}	ω_{int}	ν_{int}
1 s	$2 \times \left(2\pi\sqrt{\frac{a_{int}^3}{\mu}}\right)$ s	150 km	0.0001	0.0001°	0°	0°	0°

Table 3.3: Local Parameters for α Runs

r_{int}	Run #	α	$t_{orbit}(s)$
150 km	1	0°	1603
	2	5°	1723
	3	10°	2132
	4	15°	2769
	5	20°	3321
	6	25°	3576
	7	30°	3549
	8	35°	3344
	9	40°	3056

3.2.1 Results

A relatively short simulation run-time of two orbital periods as determined by the initial conditions of the orbit was used for these runs because of the small timestep requirement. This amount of time was sufficient, however, for observing the desired behavior of the simulation. Orbits were initiated to be nearly circular and equatorial in order to more easily observe the effects that adding lift would have on the eccentricity and inclination. A value of 0.0001 was used to approximate zero for initial eccentricity and inclination in order to accommodate the limitations in the algorithms used for calculating Cartesian state-vectors from orbital elements.

The control case, Run 1, experienced an orbit time of approximately 27 minutes before its altitude degraded lower than 50 km. This short lifetime is in accordance with previous studies seen in [47]. The orbit lifetimes increase with α, culminating in an almost 125% difference in the case of $\alpha = 25°$, after which it begins to decrease again which is consistent with the theory described in [34]. While in no case is the satellite able to maintain its orbit, the increased orbit times indicate that by introducing lifting surfaces to the system, the orbital endurance improves despite the added drag. Figure 3.1 shows an X-Y plot of the orbits for each of the nine runs and shows none of the runs even complete a single orbit before deorbiting.

Figure 3.2 gives a more insight into the altitude degradation of the satellite's orbit for the various run conditions. The satellite's altitude degrades less as the angle-of-attack increases up to 25° with only a slightly diminished endurance for the case of 30° as compared to that for 25°. Note that for 40°, endurance is less than for 20°.

Figure 3.3 shows the semi-major axis vs time for the various angles-of-attack. The semi-major axis is a better indicator of the state of the entire orbit as opposed to altitude because it is independent of the satellite's position along the orbit. Note that the decline of a grows more gradual as angle-of-attack increases and the lift force becomes more prevalent until the critical value of 25° is reached and the effect begins to decline. Figure 3.4 shows the change in each orbit's eccentricity as time progresses. Note from Table 3.2 that each orbit starts as a circular orbit with an eccentricity of zero. This parameter increases as the orbit becomes more elliptical and begins changing into a parabolic trajectory towards the earth's surface. The drastic increases seen in eccentricity are primarily due to the drag effect of decreasing the satellite velocity which in turn dives the satellite down in altitude. Figure 3.4 is in agreement with Figure 3.2 and Figure 3.3, in that increases seen in eccentricity is lessened as the effects of lift increase with angle-of-attack. This can be primarily attributed to the positive radial force driving the satellite up and away from the denser lower altitudes where drag overcomes the satellite's velocity.

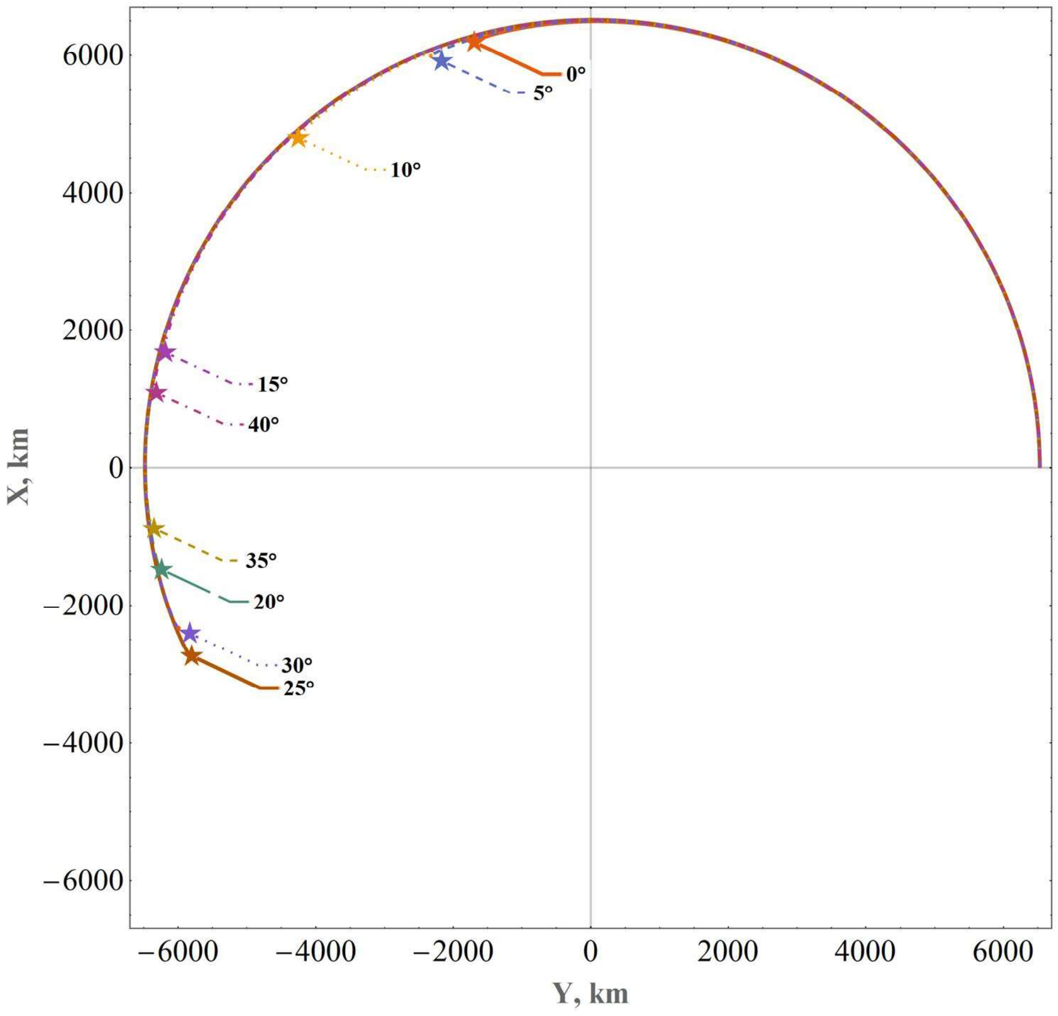

Figure 3.1: Satellite X & Y Position for Varying Angle-of-Attack

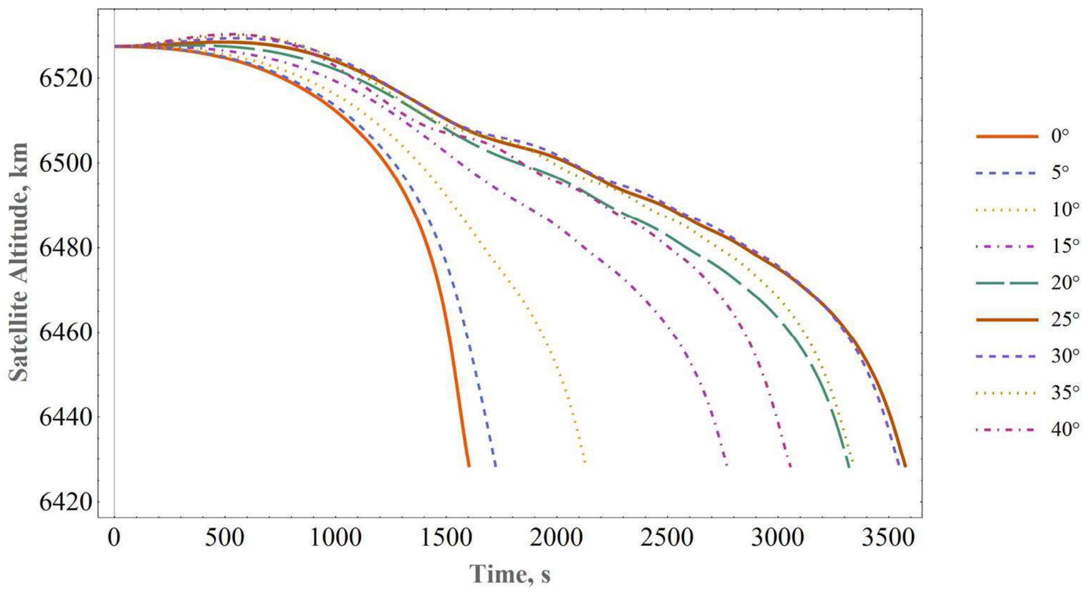

Figure 3.2: Satellite Altitude vs. Time for varying Angle-of-Attack

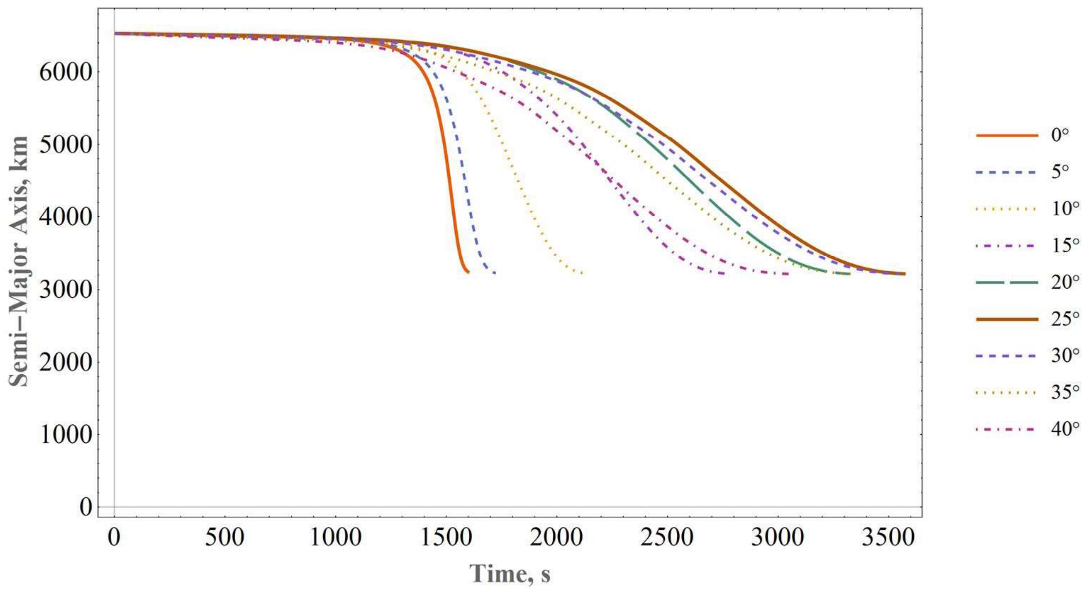

Figure 3.3: Semi-Major Axis vs. Time for varying Angle-of-Attack

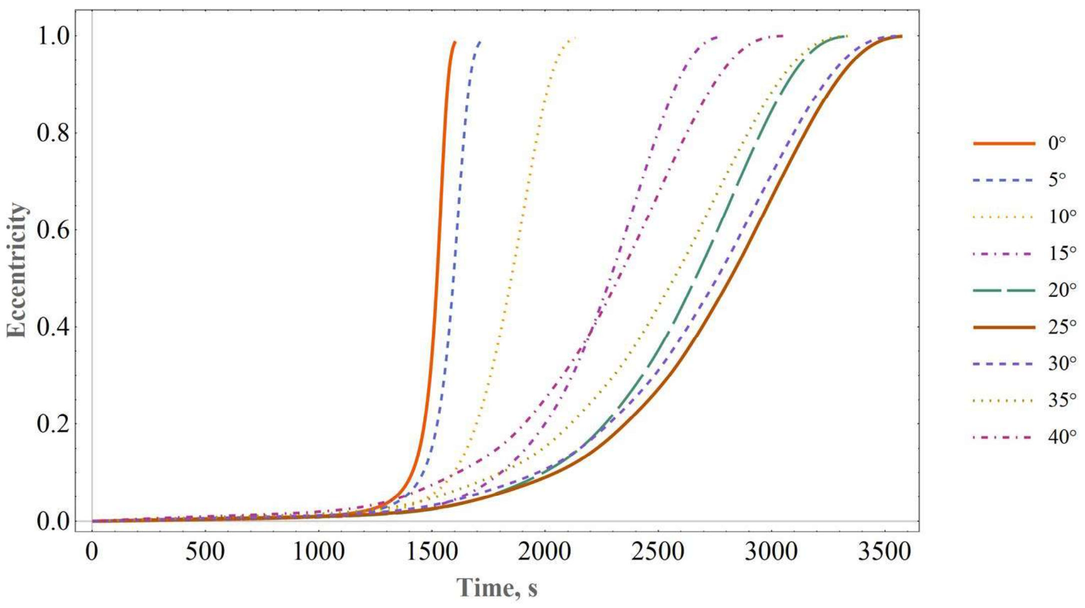

Figure 3.4: Eccentricity vs. Time for varying Angle-of-Attack

3.3 Controlled Angle-of-Attack

A rudimentary pitch control loop was incorporated for altitude hold, once is was determined that adding the capability to generate lift to the satellite produced a significant positive effect. This was to determine if modulating the angle-of-attack would have any effect and to create an isolated data set for more detailed analysis of the behavior and forces as opposed to the previous simulation runs where α was fixed for the duration of the orbit.

Table 3.4: Global Parameters for α-Controlled Runs

dt	t_{stop}	a_{int}	e_{int}	i_{int}	Ω_{int}	ω_{int}	ν_{int}
1 s	$2 \times \left(2\pi\sqrt{\frac{a_{int}^3}{\mu}}\right)$ s	150 km	0.0001	0.0001°	0°	0°	0°

Table 3.5: Local Parameters for α-Controlled Runs

r_{int}	Run #	Lift (Y/N)	$t_{orbit}(s)$
150 km	1	N	1603
	2	Y	3555

3.3.1 Results

The control loop was effective in smoothing out the initial bouncing and allowed a more clear analysis of the effects driving the system. The initial conditions and constraints are listed in Table 3.4 and are identical to those in Table 3.2. Results are shown in Table 3.5. There is an increase of about 122% in orbit time which is a reduction from the maximum increase seen in the fixed angle-of-attack study. This value is more realistic since the system begins in a steady-state with no initial lift vector versus in the fixed case where a large initial impulse is imparted on the system by the lift force.

Ultimately the control loop resulted in a total saturation of the upper angle-of-attack limit of 25°. This limit was determined from the previous results where angles-of-attack greater than 25° were seen to have negative effects on the satellite's orbital endurance. Adding lift increased the life-time of the orbit but still resulted in the satellite de-orbiting indicating that the lift effects were not sufficient to maintain altitude. The drastic losses in orbital velocity indicate large amounts of energy being bled from the system due to drag and resulting in the observed orbital decay. Figure 3.5 shows the angle-of-attack commanded by the control loop throughout the simulation run and should be referenced when observing the other figures for this study in order to understand the variations in the lift force present on the system at any given time. Note that this value becomes saturated at its limit of 25° almost instantly. The extremely limited amounts of lift able to be generated demanded that the maximum amount of lift be provided at all times.

Figure 3.6 shows some oscillatory behavior, and can be attributed to changes in the eccentricity and velocity of the satellite's orbit as well as the slight reactive buoyancy forces experienced by the satellite falling into a more dense atmosphere where there is significantly more lift and drag. Orbital velocity as a function of orbit time is shown in Figure 3.7. The sharp decrease for no lift versus the gradual decline with lift is indicative of the behavior as orbital velocity is the driving state variable responsible for the orbital conditions.

The plot of semi-major axis for the orbit versus time in Figure 3.8 is in agreement with Figure 3.3 and shows a sharp decrease in a for the case of no lift in contrast to the gradual decline seen in the case with lift.

Eccentricity is seen to sharply increase as the orbital velocity decreases in no lift in Figure 3.9 and gradually increase with lift. This is still in agreement with the fixed angle-of-attack simulation runs. A breakdown of the forces acting on the simulation and their individual components can be seen in Figure 3.10 and Figure 3.12 for the case with lift. Gravity remains smooth and oscillatory as expected until the time of de-orbiting. The total drag force increases in amplitude as the orbit time progresses. This is due to the increased atmospheric density as the altitude decreases. The lift force follows the same trend for similar reasons. The oscillations in the lift force are not due to the angle-of-attack variation as the controller is maxed out at that time but instead is due to the satellite "bobbing" against the increasing atmospheric density as its altitude decreases and the changes in its eccentricity and velocity. It is important to note that in all of the force plots negligible forces in the Z-axis indicate there is little to no influence on the inclination of the orbit, which agrees with the notation that all the major forces being modeled are acting in the orbital plane. The exception is the J2 perturbation which has an extremely small magnitude and does not have time to manifest any significant changes due to the short duration of the simulation runs.

The results in this section indicate that adding lifting surfaces to a satellite and modulating the angle-of-attack in order to maintain altitude can reduce orbital decay due to the significant drag forces at low altitude orbits without destabilizing the orbit.

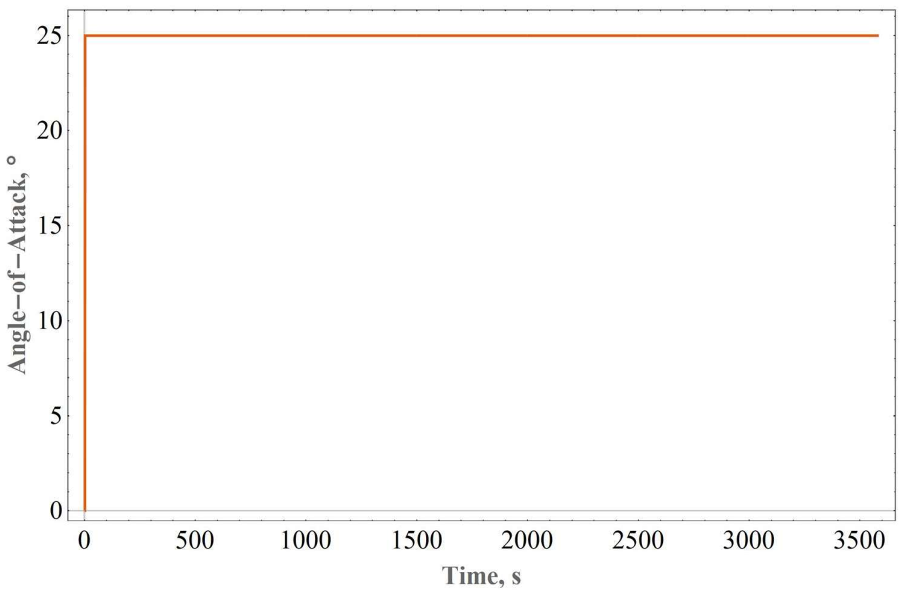

Figure 3.5: Controlled Angle of Attack vs. Time

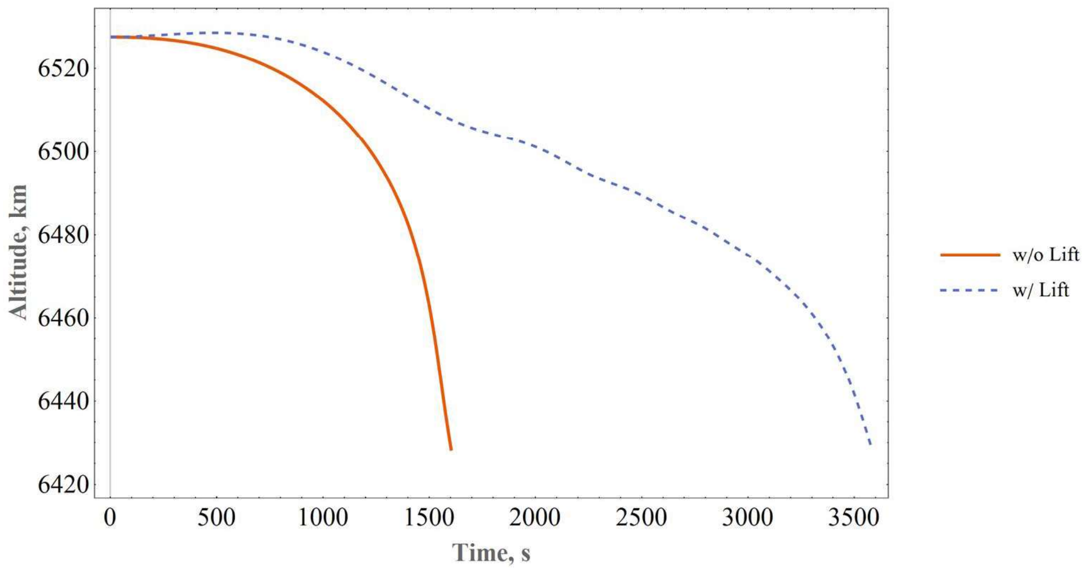

Figure 3.6: Altitude vs. Time for a Controlled Angle-of-Attack

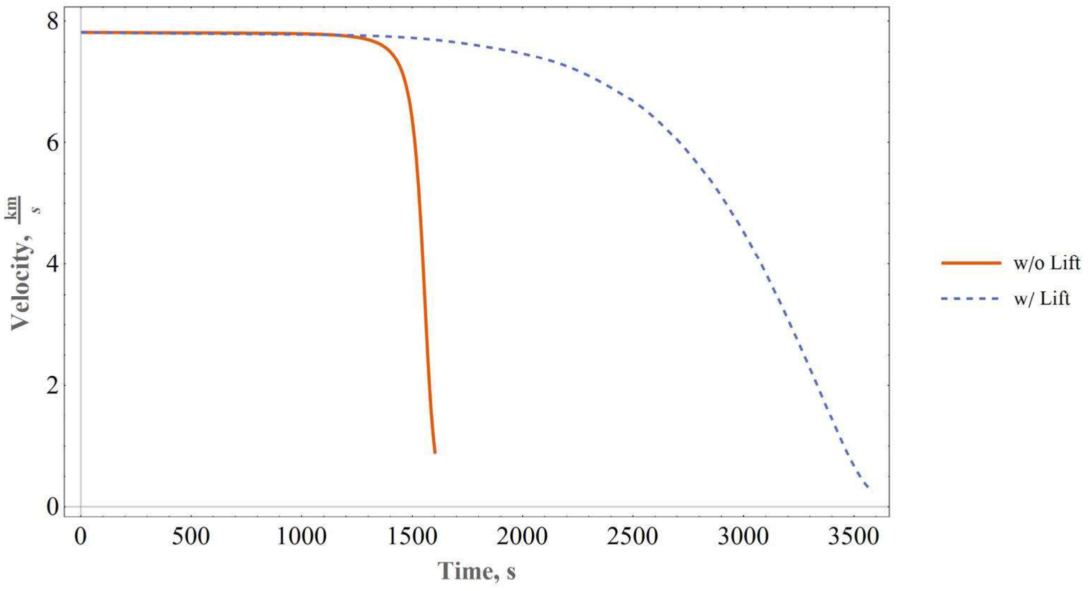

Figure 3.7: Orbital Velocity vs. Time for a Controlled Angle-of-Attack

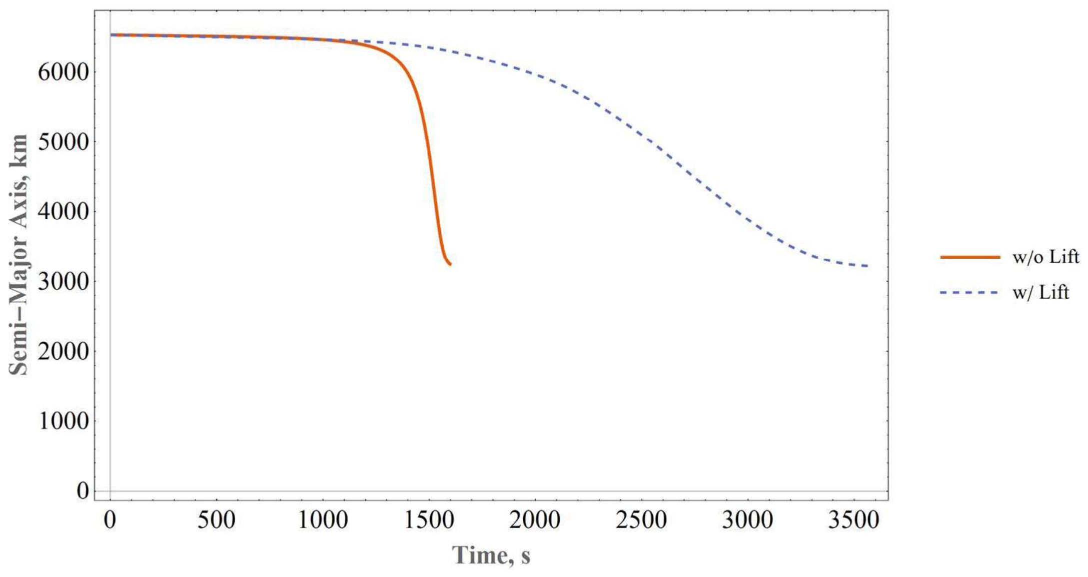

Figure 3.8: Semi-Major Axis vs. Time for a Controlled Angle-of-Attack

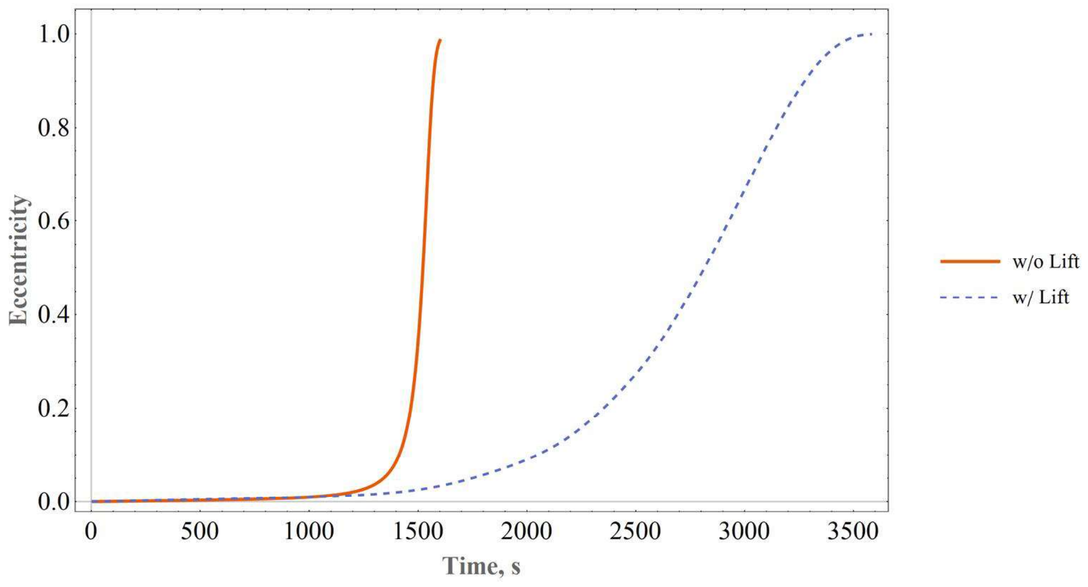

Figure 3.9: Eccentricity vs. Time for a Controlled Angle-of-Attack

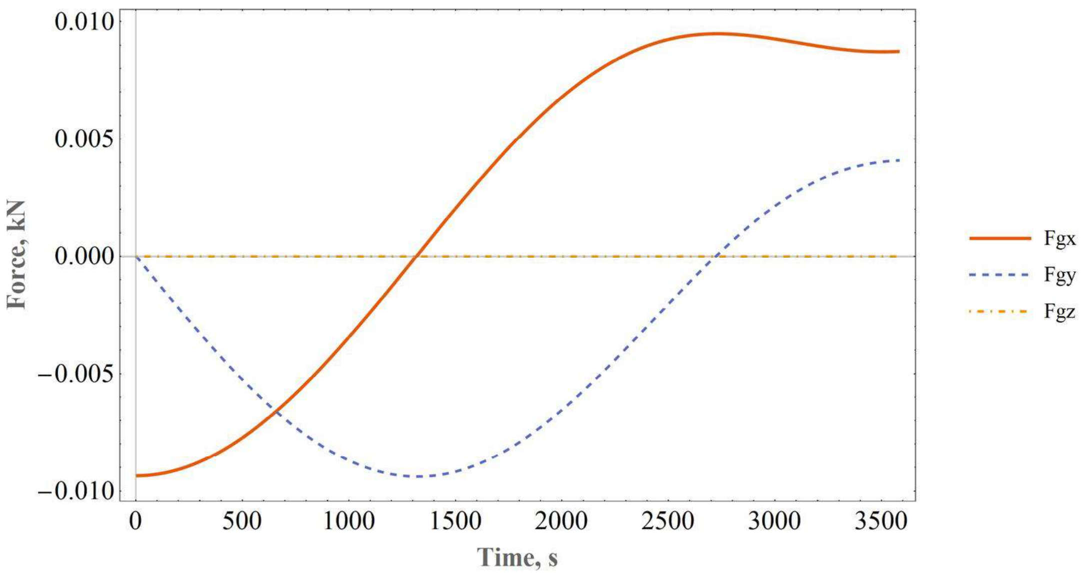

Figure 3.10: Gravity Force vs. Time for a Controlled Angle-of-Attack

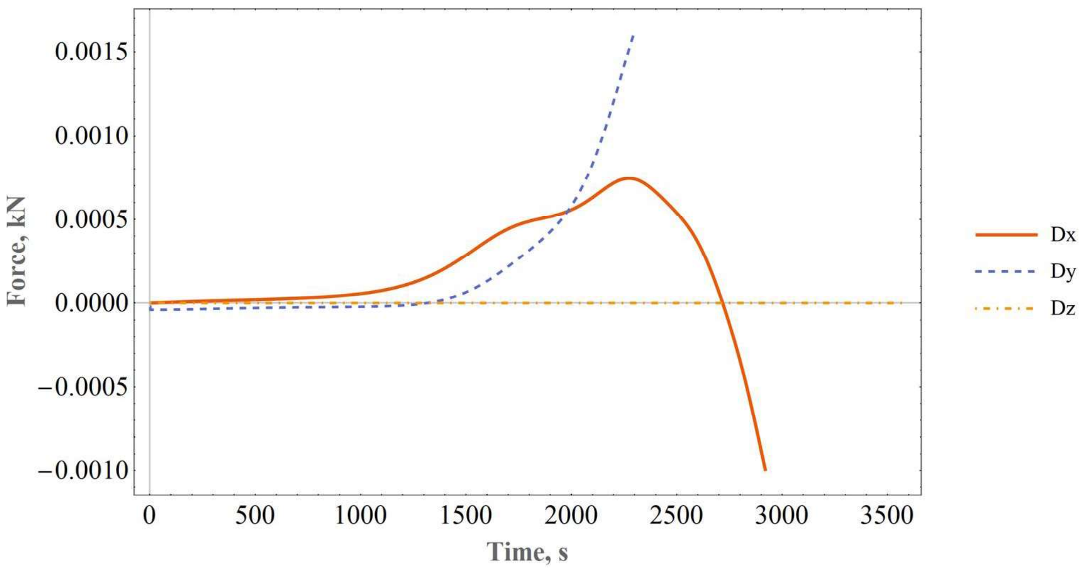

Figure 3.11: Total Drag Force vs. Time for a Controlled Angle-of-Attack

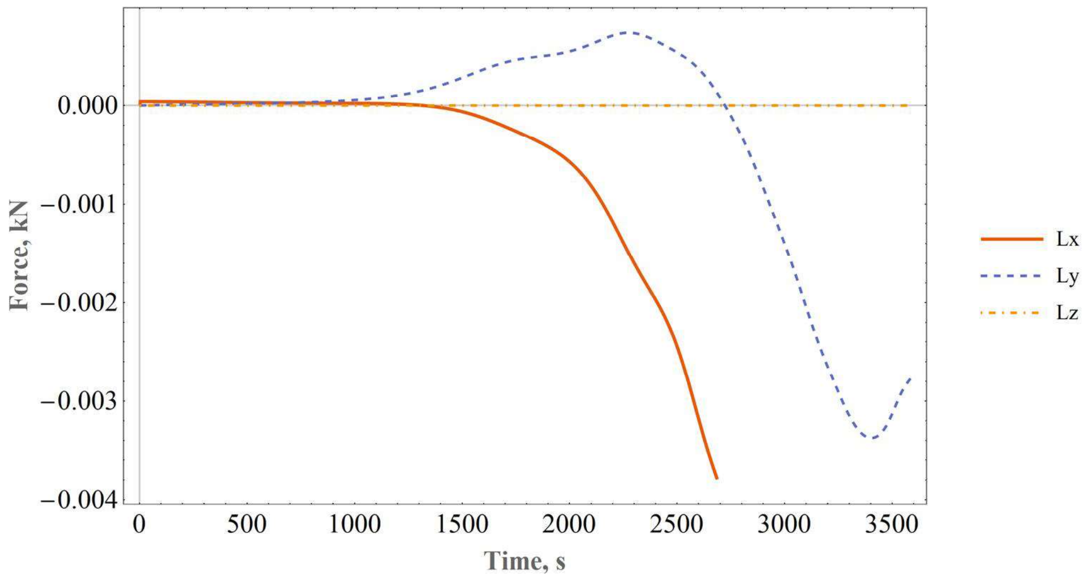

Figure 3.12: Lift Force vs. Time for a Controlled Angle-of-Attack

3.4 Thrust Matching

Drag cancelling thrust equations were modeled into the simulation as an extension of the previous runs to determine the total amount of thrust necessary to maintain the satellite's orbit, with lifting surfaces included and without. The simulation initial parameters and constraints were the same as in Table 3.4 and the results are in Table 3.6.

Table 3.6: Parameters for Simulation α-Controlled with Thrust Matching Runs

r_{int}	Run #	Lift (Y/N)	Total Thrust Impulse $\text{kN} \cdot \text{s}$
150 km	1	N	7.85×10^{-13}
	2	Y	1.44×10^{-14}

3.4.1 Results

The thrust magnitude as a function of time for both without lift and with lift are shown in Figure 3.13. This should be interpreted as the total thrust magnitude required to cancel the effects of drag on the satellite as it orbits at an altitude of 150 km. Note that the magnitudes without lift are much greater than those with lift. This data was discretely integrated to find the total thrust impulse imparted over the 2 orbits modeled in these runs as seen in Table 3.6. The amount of impulse required without lift is 55 times that for the case with lift. This indicates that for the same engine and satellite, roughly 1.834% of the fuel required for drag cancelling would be needed with integrated lifting surfaces versus without.

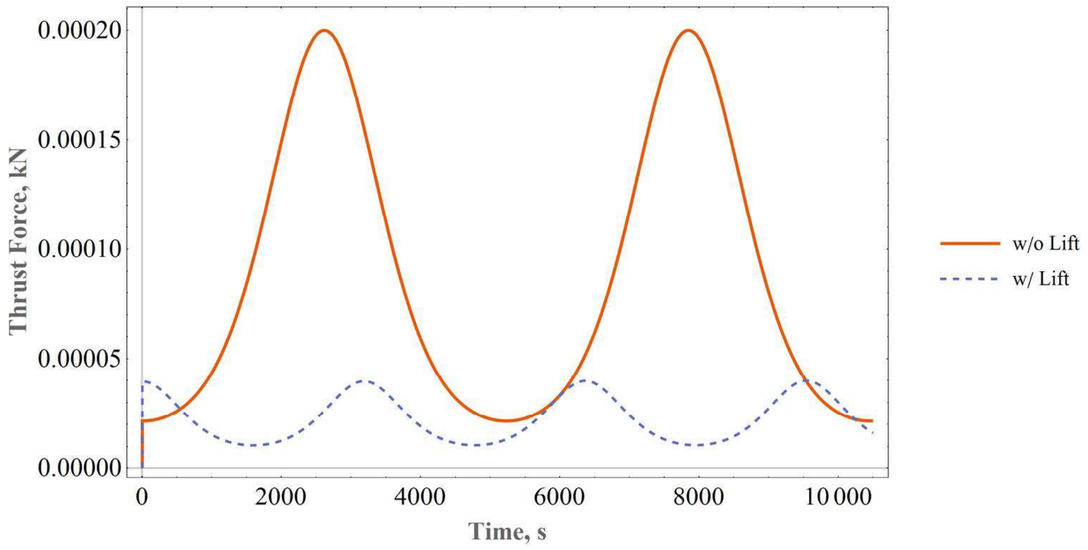

Figure 3.13: Thrust Force vs. Time for a Controlled Angle-of-Attack

CHAPTER 4

CONCLUSIONS

Finishing races is important, but racing is more important.

—Dale Earnhardt

4.1 Overview

The current limitations of small satellites make them ill-suited for missions which require large payloads. Many of these payloads could be drastically reduced in size if the mission altitude was reduced to Very Low Earth Orbit. It is difficult to maintain an orbit in this region of space due to significant drag from the Earth's atmosphere. These effects could be mitigated by incorporating lifting surfaces to the satellite in order to harness the aerodynamic forces which previously only hindered the satellite. This study's purpose was to characterize the effects of integrating lifting surfaces onto a small satellite in Very Low Earth Orbit and determine whether they would be beneficial to improving orbital endurance and could assist in orbit maintenance. Answering this question is an important step in extending the operating range of small satellites and expanding their role in the theater of space.

A physics-based simulation which modeled the various forces that govern the motion of a satellite while in orbit was designed for the purposes of this study. The major perturbations in the regime of interest which were modeled included the J2 effect and atmospheric drag. The satellite model was based off a 6U Cubesat with infinitely thin flat plate lifting surfaces. The control case for this study was conducted used the same satellite model without lifting surfaces. All runs were conducted using initial conditions for 150 km equatorial circular orbits. Suites of simulation runs were first conducted while varying a fixed angle-of-attack from 0° to 40° in order to profile the effects on orbital endurance. Increases in orbital endurance were seen to diminish after an angle-of-attack value of 25°. Improvements of 125% in orbit life time were seen when adding lifting surfaces to the satellite as opposed to the control case without lifting surfaces. An altitude-hold control loop was then used to dynamically vary the angle-of-attack so that maximum amount of lift would be generated and a more in-depth analysis into the individual forces effecting the system could be conducted. The control loop was seen to be saturated for the duration of the run at the maximum value of 25° which produced similar results as the previous case of a fixed angle-of-attack at 25°. Small oscillatory effects seen in the fixed angle-of-attack runs were absent in this case due to the lack of an initial lift vector on the system. Finally, an analysis of total thrust required to cancel the drag forces on the satellite was conducted for both the control case and with integrated lifting surfaces. It was observed that the total thrust required was reduced by a factor of 55 when lifting surfaces were added to the satellite. The reduced total thrust requirement can be extrapolated to indicate a propellant mass requirement of only 2% that of an identical satellite without integrated lifting surfaces. A reduction in propellant mass required for a given mission which would allow for either a larger payload, or much longer mission times for the same amount of fuel.

The results from this study show that adding lifting surfaces are capable of significantly reducing the orbit decay due to atmospheric drag at low orbital altitudes as well as the amount of thrust needed to cancel drag in order to maintain altitude. This indicates that there are potential benefits of integrated lifting surfaces for satellite altitude maintenance and fuel reduction at Very Low Earth Orbit.

4.2 Research Contribution

Low-Earth-Orbit CubeSats have been of interest to the scientific community and space industry since their inception. Many studies have investigated aerodynamic perturbations in Low-Earth-Orbit and the effects on a satellite's orbit [33, 44, 48, 49]. These studies, however, are only concerned with the perturbations caused by an aerodynamic lift force and not how to harness it. The concept for a class of Very-Low-Earth-Orbit satellites whose primary mission focus would be Earth observation is explored in [50], but relies on the idea of novel air-breathing-drag-compensating propulsive systems for altitude maintenance. This study provides detailed insight into how the incorporation of lifting surfaces to Very-Low-Earth-Orbit satellites could be used as a low-cost method for orbit maintenance to make such missions much more feasible.

4.3 Future Work

Several assumptions were made in order to reduce the complexity of the simulation, thus future work is required to better understand and possibly optimize the effect of lifting surfaces in Very-Low-Earth-Orbit. More detailed studies should be done on optimizing the airfoil used for the integrated lifting surfaces in order to reduce induced drag as well as how to make the satellite body more aerodynamic to reduce body drag. This airfoil optimization study should incorporate modern numerical methods to directly simulate the interactions between the air molecules in the rarefied flows and the lifting surfaces. A study performed by NASA using the direct simulation Monte Carlo method has shown that for an infinitely flat plate, the lift-to-drag ratio increases by up to 90% are possible compared to the free molecular lift and drag coefficient equations used in this study [51]. A reduction of total drag would greatly improve the impact that the lift force is able to make on the satellite-orbit system. A study on the design of an attitude-control system in order to achieve a commanded angle of attack would also be necessary to confirm the feasibility of this study. This could be done by incorporating additional control surfaces to the lifting surfaces, by attitude control systems seen in traditional satellites, or some combination. Different methods of engine modeling would also yield further insight. Constant thrust was modeled in this study but periodic engine use is also a valid approach. Different methods of engine operation may impact the propellant use differently and should be compared in order to determine the optimal solution. A more detailed study into the atmospheric density conditions and how they vary with solar activity as well as solar flux should also be conducted to determine a higher fidelity profile of orbit perturbations at VLEO altitudes. Finally a study of thermal conditions due to the drag forces at such low orbital altitudes and if current materials are capable of withstanding them should be conducted as well.

If these remaining questions can be answered then the potential exists for a new class of low-cost satellites which could navigate the extreme reaches of the earth's atmosphere like aircraft allowing for dynamic orbit alterations and extended mission times.

APPENDICES

APPENDIX A

CODE

A.1 Simulation Code

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 19 11:06:11 2019 in Python 3.5

@author: wbickett
"""

import numpy as np
import csv
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# Constants

mu = 398600.436
```

```
re = 6378.137  # km

J2 = 0.1082635666551098e-2

Cd0 = 0.8

A = 0.0288e-6  # km^2

Aw = (0.0648e-6)*1  # km^2

m = 1  # kg




# Subroutines



# Elements to Vector





def elem2vec(elem):

# define elements

a = elem[0]

e = elem[1]

i = elem[2]*np.pi/180

o = elem[3]*np.pi/180

w = elem[4]*np.pi/180

nu = elem[5]*np.pi/180

# calculations

p = a*(1-e**2)
```

```
r = p/(1+e*np.cos(nu))

rp = np.array([r*np.cos(nu), r*np.sin(nu), 0])

vp = np.array([-np.sqrt(mu/p)*np.sin(nu),

  np.sqrt(mu/p)*(e+np.cos(nu)), 0])

T = np.array([[np.cos(o)*np.cos(w)-np.sin(o)*np.cos(i)*np.sin(w),

  -np.cos(o)*np.sin(w)-np.sin(o)*np.cos(i)*np.cos(w),

  np.sin(o)*np.sin(i)],

  [np.sin(o)*np.cos(w)+np.cos(o)*np.cos(i)*np.sin(w),

  -np.sin(o)*np.sin(w)+np.cos(o)*np.cos(i)*np.cos(w),

  -np.cos(o)*np.sin(i)],

  [np.sin(i)*np.sin(w), np.sin(i)*np.cos(w), np.cos(i)]])

R = np.dot(T, rp)

V = np.dot(T, vp)

vec = [R, V]

return vec



# Vector to Elements



def vec2elem(R, V):

r = np.linalg.norm(R)

v = np.linalg.norm(V)
```

```
E = ((v**2)/mu-(1/r))*R-(1/mu)*np.dot(R, V)*V

e = np.linalg.norm(E)

H = np.cross(R, V)

h = np.linalg.norm(H)

p = (h**2)/mu

a = p/(1-(e**2))

i = np.arccos(H[2]/h)

Nv = np.cross(np.array([0, 0, 1]), H)

nv = np.linalg.norm(Nv)

if Nv[0] == 0 or nv == 0:

o = 0.0

else:

o = np.arccos(Nv[0]/nv)

if Nv[1] >= 0:

o = o

else:

o = 2*np.pi-o

if np.dot(Nv, E) == 0 or nv*e == 0:

w = 0.0

else:

w = np.arccos(np.dot(Nv, E)/(nv*e))

if E[2] >= 0:

w = w
```

```
else:

w = 2*np.pi-w

if np.dot(E, R) == 0 or (e*r) == 0:

nu = 0.0

else:

nu = np.arccos(np.dot(E, R)/(e*r))

if np.dot(R, V) >= 0:

nu = nu

else:

nu = 2*np.pi-nu

elem = [a, e, i*180/np.pi, o*180/np.pi, w*180/np.pi, nu*180/np.pi]

return elem


# ECEF 2 LLA Transformation



def ecef2lla(evec):

a = 6378137e-3

e = 0.08181919084

b = a*np.sqrt(1-e**2)

w2 = evec[0]**2+evec[1]**2

l = e**2/2

m = w2/a**2
```

```
n = ((1-e**2)*evec[2]/b**2)**2

p = (m+n-4*l**2)/6

G = m*n*l**2

H = 2*p**3+G

C = ((H+G+2*np.sqrt(H*G))**(1/3))/(2**(1/3))

i = -(2*l**2+m+n)/2

P = p**2

beta = i/3-C-P/C

k = (l**2)*(l**2-m-n)

t = np.sqrt(np.sqrt(beta**2-k)-(beta+i)/2)...

    ...-np.sign(m-n)*np.sqrt(np.abs((beta-i)/2))

F = t**4+2*i*t**2+2*l*(m-n*t+k)

dFdt = 4*t**3+4*i*t+2*l*(m-n)

dt = -F/dFdt

u = t+dt+l

v = t+dt-l

w = np.sqrt(w2)

latitude = np.arctan2(evec[2]*u, w*v)

dw = w*(1-1/u)

dz = evec[2]*(1-(1-e**2)/v)

h = np.sign(u-1)*np.sqrt(dw**2+dz**2)

longitude = np.arctan2(evec[1], evec[0])

return latitude, longitude, h
```

```
# Body frame to ECEF frame Transformation

def body2ecef(roll, pitch, yaw, i, bvec, evec):
roll = roll*np.pi/180
pitch = pitch*np.pi/180
i = i*np.pi/180
lat, lon, h = ecef2lla(evec)
#    yaw =  np.arcsin(np.cos(i)/np.cos(lat))+yaw
body2nedDCM = np.array([
    [np.cos(yaw)*np.cos(pitch),...
    ...-np.sin(yaw)*np.cos(roll)+np.cos(yaw)*np.sin(pitch)*np.sin(roll),...
    ...np.sin(yaw)*np.sin(roll)+np.cos(yaw)*np.sin(pitch)*np.cos(roll)],
[np.sin(yaw)*np.cos(pitch),...
...np.cos(yaw)*np.cos(roll)+np.sin(yaw)*np.sin(pitch)*np.sin(roll),...
...-np.cos(yaw)*np.sin(roll)+np.sin(yaw)*np.sin(pitch)*np.cos(roll)],
[-np.sin(pitch),...
...np.cos(pitch)*np.sin(roll),...
...np.cos(pitch)*np.cos(roll)]])
#    body2enuDCM = np.array([
        [np.sin(yaw)*np.cos(pitch),...
        ...np.cos(roll)*np.cos(yaw)+np.sin(roll)*np.sin(yaw)*np.sin(pitch),...
        ...-np.sin(roll)*np.cos(yaw)+np.cos(roll)*np.sin(yaw)*np.sin(pitch)],
#       [np.cos(yaw)*np.cos(pitch),...
        ...-np.cos(roll)*np.sin(yaw)+np.sin(roll)*np.cos(yaw)*np.sin(pitch),...
        ...np.sin(roll)*np.sin(yaw)+np.cos(roll)*np.cos(yaw)*np.sin(pitch)],
#       [np.sin(pitch),...
        ...-np.sin(roll)*np.cos(pitch),-np.cos(roll)*np.cos(pitch)]])
ned2ecefDCM = np.array([
        [-np.cos(lon)*np.sin(lat),...
        ...-np.sin(lon),-np.cos(lon)*np.cos(lat)],
        [-np.sin(lon)*np.sin(lat),...
    ...np.cos(lon),-np.sin(lon)*np.cos(lat)],
    [np.cos(lat),0,-np.sin(lat)]])
#    enu2ecefDCM = np.array([
        [-np.sin(lon),...
        ...-np.cos(lon)*np.sin(lat),...
        ...np.cos(lon)*np.cos(lat)],
#       [np.cos(lon),...
        ...-np.sin(lon)*np.sin(lat),...
        ...np.sin(lon)*np.cos(lat)],
#       [0,np.cos(lat),np.sin(lat)]])
vecNED = np.matmul(body2nedDCM,bvec)
#    vecENU = np.matmul(body2enuDCM,bvec)
vecECEF1 = np.matmul(ned2ecefDCM,vecNED)
```

```
#    vecECEF2 = np.matmul(enu2ecefDCM,vecENU)

return vecECEF1 #, vecECEF2


# Body frame to Ned frame Transformation



def body2ned(roll, pitch, yaw, bvec):

roll = roll*np.pi/180

pitch = pitch*np.pi/180

yaw = yaw*np.pi/180

body2nedDCM = np.array([

    [np.cos(yaw)*np.cos(pitch),...

    ...-np.sin(yaw)*np.cos(roll)+np.cos(yaw)*np.sin(pitch)*np.sin(roll),...

    ...np.sin(yaw)*np.sin(roll)+np.cos(yaw)*np.sin(pitch)*np.cos(roll)],

    [np.sin(yaw)*np.cos(pitch),...

    ...np.cos(yaw)*np.cos(roll)+np.sin(yaw)*np.sin(pitch)*np.sin(roll),...

    ...-np.cos( yaw)*np.sin(roll)+np.sin(yaw)*np.sin(pitch)*np.cos(roll)],

[-np.sin(pitch),....

...np.cos(pitch)*np.sin(roll),...

...np.cos(pitch)*np.cos(roll)]])

vecNED = np.matmul(body2nedDCM, bvec)

return vecNED
```

```
# Ned Frame to ECEF frame Transformation



def ned2ecef(nvec, evec):

lat, lon, h = ecef2lla(evec)

ned2ecefDCM = np.array([

    [-np.cos(lon)*np.sin(lat),...-np.sin(lon),-np.cos(lon)*np.cos(lat)],

[-np.sin(lon)*np.sin(lat),np.cos(lon),-np.sin(lon)*np.cos(lat)],

[np.cos(lat),0,-np.sin(lat)]])

vecECEF = np.matmul(ned2ecefDCM, nvec)

return vecECEF


# Saturation



def saturate(value,min,max):

if value > max:

value = max

if value < min:

value = min

else:

value = value

return value
```

```
# Main Code

def solver(step, stop, elem, attitude, filename):

# Define Orbital Elements

a = elem[0]

e = elem[1]

i = elem[2]

o = elem[3]

w = elem[4]

nu = elem[5]

# Define Euler Angles

roll = attitude[0]

pitch = attitude[1]

yaw = attitude[2]

# Calculate Vectors

Vec = elem2vec(elem)

rVec = Vec[0]

vVec = Vec[1]

# Define Constants

#    dt = step
```

```
timeCount = 0

# Define State Vector Elements

X = rVec[0]

Y = rVec[1]

Z = rVec[2]

Vx = vVec[0]

Vy = vVec[1]

Vz = vVec[2]

# Define Iteration State Vector Elements

Xj = X

Yj = Y

Zj = Z

Vxj = Vx

Vyj = Vy

Vzj = Vz

Rj = np.array([X, Y, Z])

Vj = np.array([Vx, Vy, Vz])

rj = np.linalg.norm(Rj)

vj = np.linalg.norm(Vj)

rhoj = (1.3e9)*np.exp((-(rj-re)*(10**3))/(8.5e3))

# Calculate CL and CD

    sr = (vj*10**3)/np.sqrt(2*R*Temp)

    Cl = (
        4/(np.sqrt(np.pi)*sr)*np.cos(pitch*np.pi/180)...
        ...*np.sin(pitch*np.pi/180)*np.exp(-(sr*np.sin(pitch*np.pi/180)**2))
        + np.cos(pitch*np.pi/180)*(4*np.sin(pitch*np.pi/180)**2...
        ...+(2/(sr**2)))*math.erf(sr*np.sin(pitch*np.pi/180))
            )
    Cdw = (
       ((4/(np.sqrt(np.pi)*sr))*np.sin(pitch*np.pi/180)**2)...
       ...*np.exp(-(sr*np.sin(pitch*np.pi/180))**2)
       + 2*np.sin(pitch*np.pi/180)*(2*(np.sin(pitch*np.pi/180)**2...
       ...+(1/(2*sr**2))))*math.erf(sr*np.sin(pitch*np.pi/180))
            )
# Define Equations of Motion


def fx(rho, lift, thrust, x, y, z, Vx, Vy, Vz):
force = (-(mu*x)/((x**2+y**2+z**2)**(3/2))
 -(3*J2*mu*re**2*x*(x**2+y**2-4*z**2))/(2*(x**2+y**2+z**2)**(7/2))
 -(A*Cd0*Vx*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 -(Aw*Cdw*Vx*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 +lift+thrust) # *np.sqrt(Vx**2+Vy**2+Vz**2)**2)
return force


def fy(rho, lift, thrust, x, y, z, Vx, Vy, Vz):
force = (-(mu*y)/((x**2+y**2+z**2)**(3/2))
 -(3*J2*mu*re**2*y*(x**2+y**2-4*z**2))/(2*(x**2+y**2+z**2)**(7/2))
 -(A*Cd0*Vy*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 -(Aw*Cdw*Vy*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 +lift+thrust) # *np.sqrt(Vx**2+Vy**2+Vz**2)**2)
return force


def fz(rho, lift, thrust, x, y, z, Vx, Vy, Vz):
force = (-(mu*z)/((x**2+y**2+z**2)**(3/2))
 +(3*J2*mu*re**2*z*(-3*(x**2+y**2)+2*z**2))/(2*(x**2+y**2+z**2)**(7/2))
 -(A*Cd0*Vz*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 -(Aw*Cdw*Vz*np.sqrt(Vx**2+Vy**2+Vz**2)*rho)/(2*m)
 +lift+thrust) # *np.sqrt(Vx**2+Vy**2+Vz**2)**2)
return force
#    def fx(rho, x, y, z, vx, vy, vz):
#        force = (-(mu*x)/((x**2+y**2+z**2)**(3/2)))
#        return force
#
#    def fy(rho, x, y, z, vx, vy, vz):
#        force = (-(mu*y)/((x**2+y**2+z**2)**(3/2)))
#        return force
#
#    def fz(rho, x, y, z, vx, vy, vz):
#        force = (-(mu*z)/((x**2+y**2+z**2)**(3/2)))
#        return force

# Calculate Initial Lift Force

#    lift_body = np.array([0, 0, -(Cl*Aw*rhoj*(vj**2))/(2*m)])

#    lift_ned = body2ned(roll, 0, 0, lift_body)

lift_ned = np.array([0, 0, -(Cl*Aw*rhoj*(vj**2))/(2*m)])

#    lift_ecef = body2ecef(roll, pitch, yaw, i, lift_body, Rj)

lift_ecef = ned2ecef(lift_ned, Rj)

# Initial Thrust is Zero

thrust = np.array([0, 0, 0])

# Calculate Force Components for recording

Fgrav = [-(mu*Xj)/((Xj**2+Yj**2+Zj**2)**(3/2)),

 -(mu*Yj)/((Xj**2+Yj**2+Zj**2)**(3/2)),

 -(mu*Zj)/((Xj**2+Yj**2+Zj**2)**(3/2))]

Fj2 = [

-(3*J2*mu*re**2*Xj*(Xj**2+Yj**2-4*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2)),

-(3*J2*mu*re**2*Yj*(Xj**2+Yj**2-4*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2)),

+(3*J2*mu*re**2*Zj*(-3*(Zj**2+Yj**2)+2*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2))]

FD0 = [-(A*Cd0*Vxj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),

   -(A*Cd0*Vyj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),

   -(A*Cd0*Vzj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m)]

FDw = [-(Aw*Cdw*Vxj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),

   -(Aw*Cdw*Vyj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),

   -(Aw*Cdw*Vzj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m)]
```

```
# Calculate Initial Forces

fxj = fx(rhoj, lift_ecef[0], thrust[0], Xj, Yj, Zj, Vxj, Vyj, Vzj)

fyj = fy(rhoj, lift_ecef[1], thrust[1], Xj, Yj, Zj, Vxj, Vyj, Vzj)

fzj = fz(rhoj, lift_ecef[2], thrust[2], Xj, Yj, Zj, Vxj, Vyj, Vzj)

# Create Log File

with open(filename, 'w', newline='') as csvFile:

writer = csv.writer(csvFile)

writer.writerows([['Time', 'X', 'Y', 'Z', 'Vx', 'Vy', 'Vz',

   'Fx', 'Fy', 'Fz', 'Lx', 'Ly', 'Lz', 'rho', 'rj',

   'a', 'e', 'i', 'o', 'w', 'nu', 'pitch',

   'Fgrav', 'FgravX', 'FgravY', 'FgravZ',

   'Fj2', 'Fj2X', 'Fj2Y', 'Fj2Z',

   'FD0', 'FD0X', 'FD0Y', 'FD0Z',

   'FDw', 'FDwX', 'FDwY', 'FDwZ', 'Thrust'],

  [str(timeCount), str(Xj), str(Yj), str(Zj),

   str(Vxj), str(Vyj), str(Vzj),

   str(fxj), str(fyj), str(fzj),

   str(lift_ecef[0]), str(lift_ecef[1]), str(lift_ecef[2]),

   str(rhoj), str(rj), str(a), str(e),

   str(i),str(o), str(w),str(nu), str(pitch),

   str(np.linalg.norm(Fgrav)), str(Fgrav[0]), str(Fgrav[1]), str(Fgrav[2]),

   str(np.linalg.norm(Fj2)), str(Fj2[0]), str(Fj2[1]), str(Fj2[2]),

   str(np.linalg.norm(FD0)), str(FD0[0]), str(FD0[1]), str(FD0[2]),
   str(np.linalg.norm(FDw)), str(FDw[0]), str(FDw[1]), str(FDw[2]),
   str(np.linalg.norm(thrust))]])
csvFile.close()
# RK4-4 Integrator
while timeCount < stop and rj > (re+50):
#        if rj < (re+120):
#            dt = step
#        else:
#            dt = (step*10)
dt = step
# Step 1
k1x = dt*fx(rhoj, lift_ecef[0], thrust[0], Xj, Yj, Zj, Vxj, Vyj, Vzj)
k1y = dt*fy(rhoj, lift_ecef[1], thrust[1], Xj, Yj, Zj, Vxj, Vyj, Vzj)
k1z = dt*fz(rhoj, lift_ecef[2], thrust[2], Xj, Yj, Zj, Vxj, Vyj, Vzj)
# Step 2
k2x = dt*fx(rhoj, lift_ecef[0], thrust[0],...
...Xj+Vxj*(dt/2), Yj+Vyj*(dt/2), Zj+Vzj*(dt/2),...
...Vxj+(k1x/2), Vyj+(k1y/2), Vzj+(k1z)/2)
k2y = dt*fy(rhoj, lift_ecef[1], thrust[1],...
...Xj+Vxj*(dt/2), Yj+Vyj*(dt/2), Zj+Vzj*(dt/2),...
...Vxj+(k1x/2), Vyj+(k1y/2), Vzj+(k1z)/2)
k2z = dt*fz(rhoj, lift_ecef[2], thrust[2],...
Xj+Vxj*(dt/2), Yj+Vyj*(dt/2), Zj+Vzj*(dt/2),...
```

```
Vxj+(k1x/2), Vyj+(k1y/2), Vzj+(k1z)/2)

# Step 3

k3x = dt*fx(rhoj, lift_ecef[0], thrust[0],...

...Xj+Vxj*(dt/2)+k1x*(dt/4), Yj+Vyj*(dt/2)+k1y*(dt/4),...

...Zj+Vzj*(dt/2)+k1z*(dt/4),...

...Vxj+(k2x/2), Vyj+(k2y/2), Vzj+(k2z)/2)

k3y = dt*fy(rhoj, lift_ecef[1], thrust[1],...

...Xj+Vxj*(dt/2)+k1x*(dt/4), Yj+Vyj*(dt/2)+k1y*(dt/4),...

...Zj+Vzj*(dt/2)+k1z*(dt/4),...

...Vxj+(k2x/2), Vyj+(k2y/2), Vzj+(k2z)/2)

k3z = dt*fz(rhoj, lift_ecef[2], thrust[2],...

...Xj+Vxj*(dt/2)+k1x*(dt/4), Yj+Vyj*(dt/2)+k1y*(dt/4),...

...Zj+Vzj*(dt/2)+k1z*(dt/4),...

...Vxj+(k2x/2), Vyj+(k2y/2), Vzj+(k2z)/2)

# Step 4

k4x = dt*fx(rhoj, lift_ecef[0], thrust[0],...

...Xj+Vxj*dt+k2x*(dt/2), Yj+Vyj*dt+k2y*(dt/2), Zj+Vzj*dt+k2z*(dt/2),...

...Vxj+k3x, Vyj+k3y, Vzj+k3z)

k4y = dt*fy(rhoj, lift_ecef[1], thrust[1],...

...Xj+Vxj*dt+k2x*(dt/2), Yj+Vyj*dt+k2y*(dt/2), Zj+Vzj*dt+k2z*(dt/2),...

...Vxj+k3x, Vyj+k3y, Vzj+k3z)

k4z = dt*fz(rhoj, lift_ecef[2], thrust[2],...

...Xj+Vxj*dt+k2x*(dt/2), Yj+Vyj*dt+k2y*(dt/2), Zj+Vzj*dt+k2z*(dt/2),...
```

```
...Vxj+k3x, Vyj+k3y, Vzj+k3z)

# Update State Variables

Xj = Xj+dt*Vxj+dt*((k1x+k2x+k3x)/6)

Yj = Yj+dt*Vyj+dt*((k1y+k2y+k3y)/6)

Zj = Zj+dt*Vzj+dt*((k1z+k2z+k3z)/6)

Vxj = Vxj+((k1x+2*k2x+2*k3x+k4x)/6)

Vyj = Vyj+((k1y+2*k2y+2*k3y+k4y)/6)

Vzj = Vzj+((k1z+2*k2z+2*k3z+k4z)/6)

# Calculate New State Parameters

Rj = np.array([Xj, Yj, Zj])

Vj = np.array([Vxj, Vyj, Vzj])

timeCount = timeCount+dt

rjOld = rj

rj = np.linalg.norm(Rj)

vj = np.linalg.norm(Vj)

elemj = vec2elem(Rj, Vj)

rhoj = (1.3e9)*np.exp((-(rj-re)*(10**3))/(8.5e3))

# Calculate Forces

# Lift

#         lift_body = np.array([0, 0, -(Cl*Aw*rhoj*10**3)/2])

#         lift_ecef = body2ecef(roll, pitch, yaw, elemj[2], lift_body, Rj)

lift_ned = np.array([0, 0, -(Cl*Aw*rhoj*(vj**2))/(2*m)])

#         lift_body = np.array([0, 0, -(Cl*Aw*rhoj*(vj**2))/(2*m)])
#        lift_ned = body2ned(roll, 0, 0, lift_body)
lift_ecef = ned2ecef(lift_ned, Rj)
# Calculate Force Components for recording
Fgrav = [-(mu*Xj)/((Xj**2+Yj**2+Zj**2)**(3/2)),
 -(mu*Yj)/((Xj**2+Yj**2+Zj**2)**(3/2)),
 -(mu*Zj)/((Xj**2+Yj**2+Zj**2)**(3/2))]
Fj2 = [
-(3*J2*mu*re**2*Xj*(Xj**2+Yj**2-4*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2)),
-(3*J2*mu*re**2*Yj*(Xj**2+Yj**2-4*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2)),
+(3*J2*mu*re**2*Zj*(-3*(Zj**2+Yj**2)+2*Zj**2))/(2*(Xj**2+Yj**2+Zj**2)**(7/2))]
FD0 = [-(A*Cd0*Vxj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),
   -(A*Cd0*Vyj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),
   -(A*Cd0*Vzj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m)]
FDw = [-(Aw*Cdw*Vxj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),
   -(Aw*Cdw*Vyj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m),
   -(Aw*Cdw*Vzj*np.sqrt(Vxj**2+Vyj**2+Vzj**2)*rhoj)/(2*m)]
# Calculate Thrust
thrust = -1*np.array([FD0[0]+FDw[0], FD0[1]+FDw[1], FD0[2]+FDw[2]])
# Total Force
fxj = fx(rhoj, lift_ecef[0], thrust[0], Xj, Yj, Zj, Vxj, Vyj, Vzj)
fyj = fy(rhoj, lift_ecef[1], thrust[1], Xj, Yj, Zj, Vxj, Vyj, Vzj)
fzj = fz(rhoj, lift_ecef[2], thrust[2], Xj, Yj, Zj, Vxj, Vyj, Vzj)
# Write State Info to csv
```

```
with open(filename, 'a', newline='') as csvFile:

writer = csv.writer(csvFile)

writer.writerow([str(timeCount), str(Xj), str(Yj), str(Zj),

 str(Vxj), str(Vyj), str(Vzj),

 str(fxj), str(fyj), str(fzj),

 str(lift_ecef[0]), str(lift_ecef[1]), str(lift_ecef[2]),

 str(rhoj), str(rj), str(elemj[0]), str(elemj[1]),

 str(elemj[2]), str(elemj[3]),

 str(elemj[4]), str(elemj[5]), str(pitch),

 str(np.linalg.norm(Fgrav)), str(Fgrav[0]), str(Fgrav[1]), str(Fgrav[2]),

 str(np.linalg.norm(Fj2)), str(Fj2[0]), str(Fj2[1]), str(Fj2[2]),

 str(np.linalg.norm(FD0)), str(FD0[0]), str(FD0[1]), str(FD0[2]),

 str(np.linalg.norm(FDw)), str(FDw[0]), str(FDw[1]), str(FDw[2]),

 str(np.linalg.norm(thrust))])

csvFile.close()

# Pitch Control Loop

gain = 0.1

altErr = a-rj

vertSpeedErr = 0-(rj-rjOld)/dt

pitch = saturate((altErr+vertSpeedErr)*gain/vj,...

...-15*np.pi/180, 45*np.pi/180)*180/np.pi

# Calculate CL and CD

    sr = (vj*10**3)/np.sqrt(2*R*Temp)
```

```
Cl = (

    4/(np.sqrt(np.pi)*sr)*np.cos(pitch*np.pi/180)...

    ...*np.sin(pitch*np.pi/180)*np.exp(-(sr*np.sin(pitch*np.pi/180)**2))

    + np.cos(pitch*np.pi/180)*(4*np.sin(pitch*np.pi/180)**2...

    ...+(2/(sr**2)))*math.erf(sr*np.sin(pitch*np.pi/180))

        )

Cdw = (

   ((4/(np.sqrt(np.pi)*sr))*np.sin(pitch*np.pi/180)**2)...

   ...*np.exp(-(sr*np.sin(pitch*np.pi/180))**2)

   + 2*np.sin(pitch*np.pi/180)*(2*(np.sin(pitch*np.pi/180)**2...

   ...+(1/(2*sr**2))))*math.erf(sr*np.sin(pitch*np.pi/180))

        )
```

REFERENCES

[1] Erik Kulu. Nanosats database. https://www.nanosats.eu/. Accessed: 2019-10-22.

[2] Thomas Yunck, Conrad Lautenbacher, Mike McGrath, Peter Withnell, Penina Axelrad, Danielle Nuding, and John Kreisher. Cicero: Community initiative for continuing earth radio occultation. 6th FORMOSAT-3/COSMIC data users workshop. Boulder, CO, USA, 2012.

[3] Mohinder S Grewal, Lawrence R Weill, and Angus P Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[4] Hank Heidt, Jordi Puig-Suari, Augustus Moore, Shinichi Nakasuka, and Robert Twiggs. Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation. 2000.

[5] Ryan Hevner, Walter Holemans, Jordi Puig-Suari, and Robert Twiggs. An advanced standard for cubesats. 2011.

[6] Alexander Chin, Roland Coelho, Ryan Nugent, Riki Munakata, and Jordi Puig-Suari. Cubesat: the pico-satellite standard for research and education. In *AIAA Space 2008 Conference & Exposition*, page 7734, 2008.

[7] Isaac Nason, Jordi Puig-Suari, and Robert Twiggs. Development of a family of picosatellite deployers based on the cubesat standard. In *Proceedings, IEEE Aerospace Conference*, volume 1, pages 1–1. IEEE, 2002.

[8] Anders Kose Nervold, Joshua Berk, Jeremy Straub, and David Whalen. A pathway to small satellite market growth. *Advances in Aerospace Science and Technology*, 1(01):14, 2016.

[9] Daniel Selva and David Krejci. A survey and assessment of the capabilities of cubesats for earth observation. *Acta Astronautica*, 74:50–68, 2012.

[10] Junquan Li, Mark Post, Thomas Wright, and Regina Lee. Design of attitude control systems for cubesat-class nanosatellite. *Journal of Control Science and Engineering*, 2013:4, 2013.

[11] NASA. Landsat 8. https://landsat.gsfc.nasa.gov/landsat-8/. Accessed: 2019-10-22.

[12] Kiril A Dontchev, Kartik Ghorakavi, Cameron E Haag, Thomas M Liu, and Rafael Ramos. M-cubed: University of michigan multipurpose minisatellite with optical imager payload. In *Proceedings of the AIAA Space 2010 Conference & Exhibition*. Citeseer, 2010.

[13] Roger R Bate, Donald D Mueller, and Jerry E White. *Fundamentals of astrodynamics*. Courier Corporation, 1971.

[14] ESA. Goce facts and figures. http://www.esa.int/Our$_{A}ctivities/Observing_{t}he_{E}arth/GOCE$ $2019 - 10 - 22$.

[15] Jaime Perez Luna, Rhodri A Lewis, and Mark Hutchins. Qinetiq high power electric propulsion system and architectural options for future applications. 2017.

[16] CH Edwards, NC Wallace, C Tato, and P Van Put. The t5 ion propulsion assembly for drag compensation on goce. In *Second International GOCE User Workshop" GOCE, The Geoid and Oceanography*, 2004.

[17] Chia-Chun George Chao. *Applied orbit perturbation and maintenance*. American Institute of Aeronautics and Astronautics, Inc., 2005.

[18] Samudra E Haque, Michael Keidar, and Taeyoung Lee. Low-thrust orbital maneuver analysis for cubesat spacecraft with a micro-cathode arc thruster subsystem. In *33rd International Electric Propulsion Conference*, 2013.

[19] BA Arkhipov, NM Vertakov, BE Didenko, and AG Niatin. Control strategies for orbit maintenance of leo small satellites with feep. In *Spacecraft Propulsion*, volume 465, page 733, 2000.

[20] Andrea Garulli, Antonio Giannitrapani, Mirko Leomanni, and Fabrizio Scortecci. Autonomous low-earth-orbit station-keeping with electric propulsion. *Journal of Guidance, Control, and Dynamics*, 34(6):1683–1693, 2011.

[21] Akshay Tummala and Atri Dutta. An overview of cube-satellite propulsion technologies and trends. *Aerospace*, 4(4):58, 2017.

[22] Yi Chen, Rui Huang, Xianlin Ren, Liping He, and Ye He. History of the tether concept and tether missions: a review. *ISRN astronomy and astrophysics*, 2013, 2013.

[23] Mario L Cosmo and Enrico C Lorenzini. Tethers in space handbook. 1997.

[24] Robert Hoyt and Jeffrey Slostad. The multi-application survivable tether (mast) experiment. In *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, page 5219, 2003.

[25] Robert Hoyt, Tyrel Newton, Ian Barnes, Jack Shepherd, S Scott Frank, Jeff Slostad, Belgacem Jaroux, Robert Twiggs, et al. Early results of the multi-application survivable tether (mast) space tether experiment. 2007.

[26] Masahiro Nohmi. Initial experimental result of pico-satellite kukai on orbit. In *2009 International Conference on Mechatronics and Automation*, pages 2946–2951. IEEE, 2009.

[27] Masahiro Nohmi, Takeshi Yamamoto, Akira Andatsu, Yohei Takagi, Yusuke Nishikawa, Takashi Kaneko, and Daisuke Kunitom. Kagawa satellite stars in shikoku. *TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES, SPACE TECHNOLOGY JAPAN*, 7(ists26):Tu_7–Tu_12, 2009.

[28] Jouni Envall, Pekka Janhunen, Petri Toivanen, Mihkel Pajusalu, Erik Ilbis, Jaanus Kalde, Matis Averin, Henri Kuuste, Kaspars Laizans, Viljo Allik, et al. E-sail test payload of estcube-1 nanosatellite. *arXiv preprint arXiv:1404.6961*, 2014.

[29] S Coffey, B Kelm, A Hoskins, J Carroll, and E Levin. Tethered electrodynamic propulsion cubesat experiment (tepce). In *Air Force Orbital Resources Ionosphere Conference, Dayton, Ohio*, pages 12–14, 2010.

[30] A Haider and O Levenspiel. Drag coefficient and terminal velocity of spherical and nonspherical particles. *Powder technology*, 58(1):63–70, 1989.

[31] Richard Fitzpatrick. Effect of atmospheric drag on artificial satellite orbits. http://farside.ph.utexas.edu/teaching/celestial/Celestialhtml/node94.html. Accessed: 2019-10-22.

[32] Jeffery Scott Ginn. Spacecraft formation flight: Analysis of the perturbed j2-modified hill-clohessy-wiltshire equation. Master's thesis, The University of Texas at Arlington, 2007.

[33] Carlos L Pulido. Aerodynamic lift and drag effects on the orbital lifetime low earth orbit (leo) satellites. *University of Colorado Boulder*, 2007.

[34] Samuel A Schaaf, Lawrence Talbot, and GN Patterson. Handbook of supersonic aerodynamics, section 16, mechanics of rarefied gases, 1960.

[35] George Karniadakis, Ali Beskok, and Narayan Aluru. *Microflows and nanoflows: fundamentals and simulation*, volume 29. Springer Science & Business Media, 2006.

[36] Graeme A Bird and JM Brady. *Molecular gas dynamics and the direct simulation of gas flows*, volume 5. Clarendon press Oxford, 1994.

[37] Jose F Padilla and Iain D Boyd. Assessment of gas-surface interaction models for computation of rarefied hypersonic flow. *Journal of Thermophysics and Heat Transfer*, 23(1):96–105, 2009.

[38] John David Anderson Jr. *Fundamentals of aerodynamics.* Tata McGraw-Hill Education, 2010.

[39] M Eng and René Schwarz. Cartesian state vectors- keplerian orbit elements. 2017.

[40] M Eng and René Schwarz. Keplerian orbit elements- cartesian state vectors. 2017.

[41] Karl Osen. Accurate conversion of earth-fixed earth-centered coordinates to geodetic coordinates. 2017.

[42] Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, page 1259, 1981.

[43] Jan Roskam. *Airplane flight dynamics and automatic flight controls*. DARcorporation, 1998.

[44] Li Qiao, Chris Rizos, and Andrew G Dempster. Analysis and comparison of cubesat lifetime. In *Proceedings of the 12th Australian Space Conference*, pages 249–260. Citeseer, 2013.

[45] Leonard B Loeb. *The kinetic theory of gases*. Courier Corporation, 2004.

[46] Justin Winslow, Hikaru Otsuka, Bharath Govindarajan, and Inderjit Chopra. Basic understanding of airfoil characteristics at low reynolds numbers (10 4–10 5). *Journal of Aircraft*, 55(3):1050–1061, 2017.

[47] Daniel Oltrogge and Kyle Leveque. An evaluation of cubesat orbital decay. 2011.

[48] TP Brito, CC Celestino, and RV Moraes. Study of the decay time of a cubesat type satellite considering perturbations due to the earth's oblateness and atmospheric drag. In *Journal of Physics: Conference Series*, volume 641, page 012026. IOP Publishing, 2015.

[49] Sahadeo Ramjatan, Norman Fitz-Coy, and Alvin Yew. Magnus effect on a spinning satellite in low earth orbit. In *AIAA/AAS Astrodynamics Specialist Conference*, page 5257, 2016.

[50] Andrew Bacon and Ben Olivier. Skimsats: bringing down the cost of earth observation. In *Proceedings of the 12th Reinventing Space Conference*, pages 1–7. Springer, 2017.

[51] Virendra K Dogra, James N Moss, and Joseph M Price. Rarefied flow past a flat plate at incidence. 1988.