A Galilean Moon Braking System: Fuel Conservation for a Chemical Spacecraft Entering Jovian Orbit

Jonathan Logan
Department of Physics, The College of Wooster, Wooster, Ohio 44691, USA
(Dated: May 3, 2021)

Using computational and numerical methods, I show that entering Jovian orbit via a gravity assist with the Galilean moons is inherently more deterministic than solely relying on deep space Δv maneuvers. By creating a simulation in Mathematica that plots a position vector as a function of time constrained by a set of acceleration and position vectors, one can find all possible orbits for a certain period of simulation time t. Additionally, if the mass rate of change is programmed into the acceleration vectors, one can plot trajectories for fuel conservation. While the simulation was greatly limited by the computational capabilities available, a physics engine was successfully programmed and all four Galilean moons could be considered. Fuel conservation is achieved when a spacecraft first gravity brakes or orbits around a Galilean moon before injecting into a stable Jovian orbit. Adding just a single moon like Io to the flight path corresponded to a 20 % increase in fuel conservation.

I. INTRODUCTION AND HISTORICAL PRECEDENCE

Jupiter, the fifth planet from the Sun, has been of great interest to astronomers and the destination of two NASA missions. In 1995, *Galileo* became the first spacecraft to enter Jovian orbit [1]. Then, in 2016 *Juno* also entered a stable orbit around Jupiter. In the 15th century, Galileo Galilei discovered the four largest moons of the gas giant: Io, Europa, Ganymede, and Callisto. *Galileo* and *Juno* both relied on deep space Δv maneuvers (DSM). Entering the orbit of a massive planet such as Jupiter via a DSM is stochastic and the final orbit is very sensitive to initial conditions [3].

The Galilean moons provide a natural gravitational braking system. Using the moons for a gravitational slingshot or going into orbit around one or more would greatly decrease the sensitivity of the final Jovian orbit to initial conditions [3]. According to flight data, *Galileo* performed a slingshot maneuver around Io before entering Jovian orbit, but *Juno* directly entered orbit via a DSM [1]. Future missions to Jupiter will be more costly. Designing missions that fly by or orbit the Galilean moons would maximize time spent by probes collecting data.

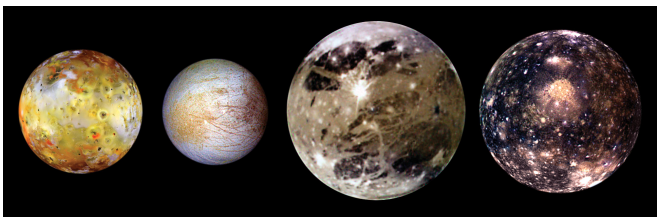

FIG. 1: The Galilean moons in order of distance from Jupiter: Io, Europa, Ganymede, Callisto. Relative sizes are too scale, however, spacing is not to scale. *Note:* Figure courtesy of *Science News*.

II. THEORY

The orbit of the Galilean moons can be assumed to be approximately circular. Their orbits follow Kepler's Laws of planetary motion where the period of an orbit is

$$T = \sqrt{\frac{4\pi^2}{GM}R^3}, \tag{1}$$

where G is the gravitational constant, M is the mass of the body being orbited (Jupiter) and R is the radial distance separating the satellite and the body being orbited [2]. All orbiting bodies obey the Law of Conservation of Angular Momentum and, therefore, have an angular velocity

$$\omega = \frac{2\pi}{T}. \tag{2}$$

Equation 1 can be rearranged to solve for exact radial distance of an orbit for a particular moon such that

$$R = \left(\frac{GM}{\omega^2}\right)^{\frac{1}{3}}. \tag{3}$$

Newton's Second Law states that $\vec{F} = m\vec{a}$. This equation will be the foundation for setting up an initial value problem wherein the force of Jupiter and each of the moons are considered acting on the mass of the spacecraft [2]. In order to plot and classify the orbit of a spacecraft, it is necessary to set up an initial radial vector dependent on time:

$$\vec{r}(t) = x(t)\hat{x} + y(t)\hat{y}. \tag{4}$$

In the computer simulation, this position vector is located by solving the differential equation that results from considering Newton's Second Law. The force of Jupiter on the spacecraft can be classified by considering the Universal Law of Gravitation such that

$$\vec{F_J} = -\frac{GM_J m_s}{\sqrt{\vec{r}(t)^2}}\hat{r}(t), \tag{5}$$

where F_J is the force on the spacecraft due to Jupiter, M_J is the mass of Jupiter, m_s is the mass of the spacecraft, and the radial vectors are normalized [5]. The mass of Jupiter M_J is assumed to be 1.

The force of a moon such as Io on the spacecraft is

$$\vec{F}_{Io} = -\frac{GM_{Io}m_s}{\sqrt{\vec{r}(t) - \vec{r}_{Io}(t)^2}}\hat{r}(t) - \hat{r}_{Io}(t), \tag{6}$$

where the vector $\vec{r}_{Io}(t)$ is a parametrically defined, approximately circular orbit such that

$$\vec{r}_{Io}(t) = 0.22(\cos(\omega_{Io}t) + \sin(\omega_{Io}t)). \tag{7}$$

The ratio 0.22 is the ratio of the distance of Io from Jupiter if the farthest moon, Callisto, is taken to be of radial distance $r = 1$ from Jupiter.

Given these equations and calling the force of the thrust vector F_T, one can establish a differential equation using $\vec{F} = m\vec{a}$:

$$(m_s)(\vec{r}''(t)) = \vec{F}_J + \vec{F}_{Io} + \vec{F}_{Eu} + \vec{F}_G + \vec{F}_C + \vec{F}_T, \tag{8}$$

where F_{Eu}, F_G, F_C are the forces of the moons Europa, Ganymede, and Callisto respectively on the spacecraft [5]. However, in order to conserve fuel, it is necessary to consider the m_s term as changing with time. A significant percentage of a spacecraft's total mass is fuel. Burning chemical fuel causes the mass to change by some incremental difference dm [4].

However, if the mass of the spacecraft is changing according to dm/dt, then the fixed mass m_s does not describe the situation. Dividing both sides by the fixed mass will allow for consideration of acceleration only [4]. That is, instead of considering both a changing force and a changing mass over time, one may only consider how acceleration vectors change over time. The final differential equation is

$$\vec{r}''(t) = \vec{a}_J + \vec{a}_{Io} + \vec{a}_{Eu} + \vec{a}_G + \vec{a}_C + \vec{a}_T, \tag{9}$$

where, for example, the acceleration vector due to any moon in general is

$$\vec{a}_{moon} = \frac{-GM_J}{\sqrt{\vec{r}(t)^2 - \vec{r}_{moon}(t)^2}}\hat{r}(t) - \hat{r}_{moon}(t). \tag{10}$$

III. MATHEMATICA SIMULATION

The purpose for using computational and numerical methods for determining the behavior of a spacecraft around the Jupiter-Galilean moons system is to exactly solve the differential equation for all possible approaches and points in space [6]. Additionally, the problem being considered is a restricted 5-body problem, which is very time consuming to solve analytically. The computer program can be thought of as an algorithm. The algorithm consists of creating two sets of vectors while the differential equation is solved for all possible trajectories, given initial conditions, that respect both sets of vectors. The differential equation is a component of an initial value problem (IVP). The other component of the IVP is a set of initial conditions.

A. Algorithm: Position Vectors

A spacecraft must avoid crashing into the moons and Jupiter. Prior to introducing a spacecraft, the positions of the moons and Jupiter were defined. Jupiter was considered to be a fixed point of mass $M_J = 1$, centered at the origin. Hence, the inertial reference frame is Jovicentric.

The Galilean moons' positions are a function of time. Kepler's Third Law gives the radial distance R of the moons from Jupiter, while angular velocity ω locates the moon on its orbit. Assuming that the orbits are circular, the radial distance and period parameters are programmed into the definition of a circle, which is defined using $\vec{r}(t) = R(\cos(\omega t) + \sin(\omega t)$. The radial distance R of each moon is in terms of Callisto's distance which is $R = 1$. The thought process is as follows:

Algorithm 1: Position Vectors Algorithm

Result: Moon and Spacecraft Vectors
`initialize;`
if $R_{Callisto} = 1$ **then**
  $R_{Io} = 0.22$;
  $R_{Europa} = 0.37$;
  $R_{Gany} = 0.57$;
end
;
→ Define Kepler's Third to solve for T using R's;
→ Define $\omega = 2\pi/T$ and solve for each moon;
FUNCTION: `Moon Positions (t);`
→ t replaces T so that there is more than 1 revolution (2π);
FUNCTION: `Spacecraft Position (t);`

In the algorithm, t replaces T. The period of an orbit was defined in terms of 2π. Therefore, the moons would only orbit once if we gave $\vec{r}(t) = R(\cos(\omega t) + \sin(\omega t))$ the period T. This allows the user to interact with the system over some defined simulation time t, instead of being limited to to just considering the time interval of one orbit for each moon. The result of this algorithm is four position vectors, one for each Galilean moon.

After defining and assigning the four moon position vectors, the spacecraft position vector is simply defined as a time-dependent radial vector with time-dependent x-

and y-components. The function is $\vec{r}(t) = x(t)\hat{x} + y(t)\hat{y}$. The definition is intentionally left completely general because the position $\vec{r}(t)$ is being solved for and plotted in the final simulation.

By solving an initial value problem for $\vec{r}(t)$, one could then plot the position vector in space. However, the position vector does not just respect the laws of physics in space unless there are objects that constrain its movement. Thus, Jupiter and the Galilean moons must be added. The physics is programmed separately from the positions.

The moons are symbolically represented as a circular orbit. While the exact positions of the moons can not be discerned in the simulation, the spacecraft is aware of their position in space and time because of the definition of the moon vectors. The simulation is also programmed to recognize when it is solving for points in space that would be inside a planet or moon. In this case, the simulation stops integrating and plotting solutions. The spacecraft is intelligently aware of whether or not it has crashed into a planet via the following algorithm:

Algorithm 2: Hit a Planet or Moon Algorithm

Result: Crashed!
`initialize;`
if $\vec{r}(t)^2 < R_J^2$ ***or*** $\vec{r}(t)^2 == \vec{r}_{moon}(t)$ **then**
$\vdash\!\rightarrow$ Stop Integration;
end

If the position of the spacecraft is less than that of the space occupied by the radius of Jupiter R_J, stop numerically integrating the flight path. Or, if the position of the spacecraft is equal to the position of a moon at the same simulation time t, stop numerically integrating the flight path. A crash is defined by these conditions and simulated by stopping the integration, which stops the plotting of a flight path. This algorithm is not implemented until later in the program when the flight path is being plotted after the differential equation is solved for $\vec{r}(t)$.

There now exists a full set of position vectors for the moons, Jupiter, and spacecraft that communicate with one another as the differential equation is solved for $\vec{r}(t)$.

B. Algorithm: Acceleration Vectors

The simulation plots an exact flight path in space determined by solving the differential equation in 9, for the spacecraft given time and fuel constraints. In order to address the problem of fuel conservation, an acceleration vector was created that acted in the direction opposite the acceleration vectors acting on the spacecraft. This vector simultaneously considered the mass rate of change dm/dt and whether or not the spacecraft was accelerating before, during, and after performing a Δv maneuver.

The three times - before, during, and after - refer to the times during which a thrusting event or Δv maneuver is being performed. Before the thrusting, the acceleration of spacecraft is $a_s = 0$. During thrusting, the spacecraft has an acceleration $a_s \neq 0$. After thrusting, $a_s = 0$.

This requires three additional times to be established. Before thrusting is considered any simulation time t, which was previously established. For the three new times, there is the beginning of thrusting t_{start}, end of thrusting δt, and during thrusting $t_{start} + \delta t$.

The mass of the spacecraft is the sum of the mass of chemical propellant m_p and the actual hardware m_d. Accelerating causes the craft to lose mass according to dm/dt where $m_s = m_p + m_d$. The total mass m_s is decreasing by the rate dm/dt. This dm/dt occurs only during $t_{start} + \delta t$.

Simply, the force of thrusting F_T is causing an acceleration a_T over time $t_{start} + \delta t$ during which it is losing mass according to m_s - dm/dt. This is the thought process can be expressed as

$$\vec{a}_T = \frac{F_T}{(m_s - \frac{dm}{dt}) * (t_{start} + \delta t)}. \tag{11}$$

Then, the Galilean moons and Jupiter each have their own acceleration vectors as well. They can be generalized as

$$\vec{a}_J = \frac{-GM_J}{\sqrt{\vec{r}(t)^2}}\hat{r}(t), \tag{12}$$

and

$$\vec{a}_{moon} = \frac{-GM_J}{\sqrt{\vec{r}(t)^2 - \vec{r}_{moon}(t)^2}}\hat{r}(t) - \hat{r}_{moon}(t). \tag{13}$$

These three sets of acceleration vectors are the backbone of the simulation; they are the physics engine. While the equations are not as simple as those of the position vectors, they are much more simple to understand in the context of the program. The acceleration vectors are simply defined and then set equal to $\vec{r}''(t)$. They operate in the background, while the integration and plotting are the difficult tasks in the programming. Rendered as an algorithm,

Algorithm 3: Acceleration Vectors Algorithm

Result: Acceleration Vectors
`initialize;`
FUNCTION: `Spacecraft` $\vec{a_T}$ `(t);`
FUNCTION: `Jupiter` $\vec{a_J}$ `(t);`
FUNCTION: `Moon` $\vec{a_{moon}}$ `(t);`

The acceleration vectors have no responsibility until called again in the differential equation in the IVP. There are now two sets of vectors. The position vectors provide the visuals and the acceleration vectors provide the physics.

C. Algorithm: IVP and Simulation

In summary, there are two sets of vectors constantly communicating with one another as the differential equation is solved and trajectories are plotted. These two sets are the acceleration vectors and the position vectors, both of which depend only on time. The acceleration vectors contain the physics of $F = ma$, while the position vectors constrain the possible trajectories to space not occupied by a moon or Jupiter.

The acceleration vectors contain the relationship between thrust and mass rate of change dm/dt, which is necessary to consider fuel conservation. Then, the position vectors contain the information about where each of the moons and Jupiter is at any given time during the simulation. This information is communicated to the IVP through the differential equation

$$\vec{r}''(t) = \vec{a}_J + \vec{a}_{Io} + \vec{a}_{Eu} + \vec{a}_G + \vec{a}_C + \vec{a}_T, \tag{14}$$

with initial conditions

$$\vec{r}(t) = \vec{r}_0(t), \tag{15}$$

$$\vec{r}'(t) =< -v_0, 0 >, \tag{16}$$

so that it knows when to stop plotting solutions; otherwise the orbits would go straight through the moons and the gas giant. Equations 15, 16, and 17 together are the IVP.

Each of the two vector algorithms and the IVP are contained within one *Manipulate* function that also contains the *ParametricPlot* function that plots the trajectories or solutions. This allows for user interaction. From a high level, the entire program runs as follows:

Algorithm 4: Full Algorithm

Result: Acceleration Vectors
`initialize`;
if $R_{Callisto} = 1$ **then**
  $R_{Io} = 0.22$;
  $R_{Europa} = 0.37$;
  $R_{Gany} = 0.57$;
end
;
→ Define Kepler's Third to solve for T using R's;
→ Define $\omega = 2\pi/T$ and solve for each moon;
FUNCTION: `Moon Positions (t);`
→ t replaces T;
FUNCTION: `Spacecraft Position (t);`
;
FUNCTION: `Spacecraft` $\vec{a_T}$ `(t);`
FUNCTION: `Jupiter` $\vec{a_J}$ `(t);`
FUNCTION: `Moon` $\vec{a_{moon}}$ `(t);`
;
→ Define IVP;
FUNCTION: `Manipulate` (IVP);
→ Solve IVP for $\vec{r}(t)$;
if $\vec{r}(t)^2 < R_J^2$ ***or*** $\vec{r}(t)^2 == \vec{r}_{moon}(t)$ **then**
  ↦ Stop Integration;
else
  ↦ Plot position using `ParametricPlot`
end

There is, of course, other functionality built in that allows the user to control various initial parameters and the coloration of the flight path according to time and duration of thrusting events. In summary, the program that runs the simulation can be thought of as two sets of vectors that are brought together by the solving of an initial value problem.

IV. LIMITATIONS

Simulations are, at best, approximations of the real world based on a set of rules or the laws of physics. The algorithm used to code the Mathematica program is no exception. I ran into a range of problems typically associated with processing power of the computer running the simulation or values that were too large or too small for the integration

A. Small Masses

The primary limitation of the program is in the *small-mass limit*. For masses much, much smaller than Jupiter or its moons, the computers on which I ran the simulation simply could not process the numbers. The fault often occurred in the *NDSolve* function,

which solves the differential equation. The mass of Jupiter is realistically 28 orders of magnitude larger than most spacecraft and 6 to 7 orders of magnitude larger than any of the four moons. The simulation ran for the Jupiter-moons system, but broke for the Jupiter-moons-spacecraft system in the small-mass limit. Thus, the spacecraft in most of my simulation runs is, at best, only 6 orders of magnitude less than M_J. If M_J is taken to be 1, then $m_s = m_p + m_d \leq 0.1$.

B. Times

Another obstacle associated with this approach is the problem of times. There are three levels of time: period time T, simulation time t, and thrust times t_{start} and $t_{start} + \delta t$. Simulation time t was substituted in for T, however, t could never be surpassed by t_{start} and $t_{start} + \delta t$. Additionally, there was a global variable t_{stop} that stopped the integration at simulation time $t = t_{stop}$ according to Algorithm 2. If t_{stop} is not defined as a global variable, then t, which is a global variable, will not stop integrating when the spacecraft crashes.

Following from the previous obstacle, the simulation will break if the mass rate of change dm/dt exceeds a certain point that can only be found once the other parameters are changed. This breaks the simulation because the time during which thrusting occurs ($t_{start} + \delta t$) is long enough that dm/dt takes the mass negative, which is impossible. However, this is encouraging, because it is also realistic; one just has to be intelligent in their use of the parameters.

V. RESULTS & ANALYSIS

A. Fuel Conservation

The moons of Jupiter provide natural gravitational brakes if the spacecraft approaches the system in the correct way and performs a well-timed thrust or Δv maneuver. The rocket equation states that for any increase in acceleration $\vec{r}''(t)$, there must be a decrease in the time rate of change of mass dm/dt. The primary goal of this simulation is to show that fuel can be conserved by using the moons of Jupiter to maneuver into a stable orbit [6]. To determine the viability of this, one can compare the total mass of the propellant m_p left over after injecting directly into a Jovian orbit versus the mass of the propellant left over after first orbiting an increasing number of Galilean moons.

The rate of fuel consumption is varied so that the mass of the propellant determines the final, stable orbit of the spacecraft. The hardware mass m_d is 0.1 and the propellant mass is $m_p = 0.1$, so the initial total mass is $m_s = 0.2$. For direct insertion into Jupiter's orbit, it is found that the total mass of propellant after a maneuver is $m_p = 0.058$ 2. This is determined by subtracting the propellant mass by the rate of change over simulation time: $m_s = m_p$ - (dm/dt).

Direct Jupiter Insertion

$m_p = 0.058 \approx 40$ % of propellant

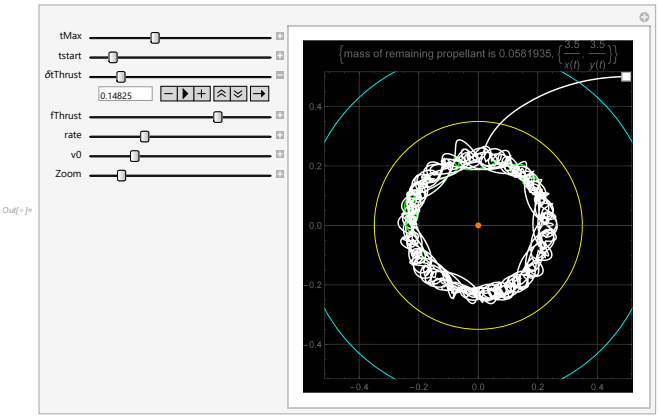

FIG. 2: Spacecraft directly entering Jovian orbit from beyond Europa (blue). The final orbit is slightly chaotic, but this is due to the effects of Io and the craft did not crash over a long period of time.

Jupiter Insertion via Io

$m_p = 0.078 \approx 20$ % of propellant

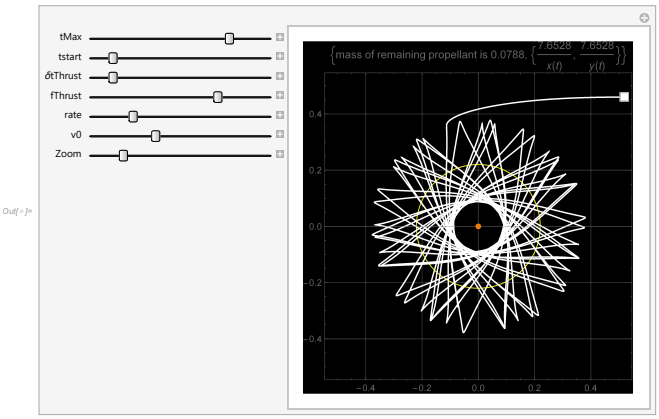

FIG. 3: Insertion into Jovian orbit via Io from beyond Europa (not shown to minimize chaotic effects). The spacecraft orbits Io before entering a stable, circular orbit around Jupiter.

Using Io for a gravity assist in a restricted 3-body problem does save fuel. Orbiting Io conserves 20 % more fuel and even allows for a higher initial velocity v_0 3. Figure 3 demonstrates this phenomenon well. The spacecraft starts from beyond the orbit of Europa and enters a stable orbit around Europa. The result of this particular trajectory is a final, nearly circu-

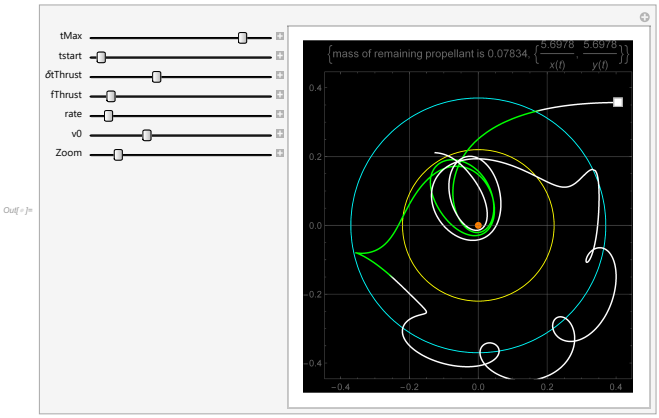

FIG. 4: Orbiting Europa before going into a highly elliptical, stable orbit around Jupiter. The green piece of the flight path is the time during which the spacecraft is thrusting to get into orbit around Jupiter.

lar orbit around Jupiter having conserved 20% more fuel.

One can expand their starting position to encompass both the orbits of Io and Europa, so as to conserve even more fuel and enter a more stable Jovian orbit. This is a restricted 4-body problem and the orbits take hours to find by locator. Considering or even trying to find orbits above the 4-body situation is not a timely proposition unless one has access to phenomenal computing power. Entering Jovian orbit via Europa is indeed possible, however, it requires the full use of boosters over an extended period of time; this is shown by the green trajectory. During the green phase, the thrusters are firing over a very long period of time. This allows the spacecraft to enter Jupiter's orbit and conserve fuel. Orbiting Europa or Europa and Io requires the spacecraft to only use 20% of the propellant or $m_p = 0.078$. Additionally, the spacecraft is traveling much slower, as evidenced by the fact that the $tMax$ slider is the same in both simulations, but the orbits do not overlap nearly as much.

Notice that the orbits become much more difficult to predict and sensitive to initial conditions as more moons are considered. Less fuel is required to enter Jovian orbit via the Galilean moons based on the results of this simulation. More fuel is conserved by taking advantage of the gravitational fields of the Galilean moons. This conclusion relies heavily on the exact initial conditions.

VI. CONCLUSION

Traveling to deep space, whether it be *Juno* or a colony ship of the future, is inherently unpredictable, and orbital injection around a planet as large as Jupiter is stochastic in nature. The goal of astrodynamics and space flight planning is to make deep space exploration a deterministic endeavor in which the final orbit of a spacecraft is predictable. Moons provide a natural tool with which spacecraft can gravity assist or gravity brake around. Not only does this make the process more deterministic, but it also conserves fuel if the thrusting is timed right.

The Mathematica simulation outlined in the previous sections applies these principles to the Galilean moons of Jupiter. The lack of computing power and a lack of understanding of the underpinnings of Mathematica led to considerable limitations, especially where realistic masses are concerned. However, fuel conservation was simulated for direct insertion into Jovian orbit via deep space (no moons), Io, and Europa. The percent of propellant burned decreased from roughly 42 % to 22 % as more moons were introduced. The Galilean moons, when the spacecraft has the exact initial conditions and thrusts at the optimal time, allow the craft to conserve fuel.

VII. ACKNOWLEDGMENT

I would like to thank Professor Lindner for being available to answer all of my questions and for investing so much time into my interest. Thank you to Professor Lehman for her nudges in the right direction and for keeping me on track. Katie Shideler and Andrew Kunkel also played a major role in motivating me to think through challenges along the way.

[1] D'Amario, A. Louis, Bright, E. Larry, Wolf, A. Aron, "Galileo Trajectory Design," Space Science Reviews, Vol. 60., **16, 23, 78**, (1992).
[2] Guido, Colasurdo, "Astrodynamics," Politencio di Torino, Space Exploration and Development (SEEDS), 2nd Ed., Ver. 2.0.1. **1-10, 51-62, 91-98** (2006-07). LEP 5.3.04 -01.
[3] "Basics of Space Flight," National Aeronautics and Space Administration, 2020, https://solarsystem.nasa.gov/basics/chapter13-1/. LEP 5.3.01.
[4] Peraire, J., Widnall, S., "Variable-Mass Sytems: the Rocket Equation," Ver. 2.0 **4-6**, (Fall 2008).
[5] Curtis, Howard, "Orbital Mechanics for Engineering Students," Elsevier Butterworth-Heinemann, Elsevier Aerospace Engineering Series. **107-134, 149-187, 347-391** (2005).

[6] Ross, S.D., Koon, W.S., Lo, M.W., Marsden, J.E., "Design of a multi-moon orbiter," Advances in the Astronautical Sciences: Spaceflight Mechanics, No. 114. **1-15**, (2003).