

**DTU Compute**  
Department of Applied Mathematics and Computer Science

# Low-Energy Transfer Orbits

## a Theoretical and Numerical Study

Gandalf Saxe (s113093)

Kongens Lyngby 2015



**DTU Compute**  
**Department of Applied Mathematics and Computer Science**  
**Technical University of Denmark**

Matematiktorvet  
Building 303B  
2800 Kongens Lyngby, Denmark  
Phone +45 4525 3031  
[compute@compute.dtu.dk](mailto:compute@compute.dtu.dk)  
[www.compute.dtu.dk](http://www.compute.dtu.dk)

# Abstract

---

When navigating a spacecraft between planets, moons and orbits, low-energy transfer orbits allows us to make the trade-off of a longer flight time in return for lower propellant energy requirements, suitable for unmanned space missions. Based on the restricted 3-body model and using 2nd order adaptive symplectic integrators implemented parallelized in Python, we brute-force search in many directions- and velocities from low Earth orbit for transfer orbits to the Moon. The lowest-energy orbit we found to the moon has  $\Delta v = 3795 \text{ km/s}$  and takes 194 days, which is even better than typical values in the literature. Our model- and implementation is validated by simulating the Apollo mission's 3-day Hohmann transfer-orbit to the Moon, found to agree within 2.5%.



# Preface

---

This bachelor's project was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a bachelor's degree in Physics and Nanotechnology.

Kongens Lyngby, June 11, 2015

Gandalf Saxe

Gandalf Saxe (s113093)



# Acknowledgements

---

People I would like to thank:

- My advisors Poul G. Hjorth and Hans Henrik Brandenburg Sørensen for excellent guidance throughout the project. This project would not have been possible without either of them.
- Laursen for the XeLaTeX thesis project template for DTU Compute [Lau].
- Kjell Magne Fauske for PGF/TikZ example of how to color equations [Fau].
- Good friend Tom Parton for feedback and suggestions.

Software acknowledgements:

1. Python - for implementing numerical algorithms to solve equations of motion.
2. Numbapro - a commerical, free-for-academics Python compiler that sped up my simulations 10-100 times.
3. Inkscape - for drawing figures in vector graphics.
4. Mathematica - for miscellaneous calculations.





# Contents

---

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Transfer Orbits</b>	<b>3</b>
2.1 Hohmann Transfer-Orbit . . . . .	5
2.2 Bi-elliptic Transfer-Orbit . . . . .	6
2.3 Hohmann-Approximated Transfer-Orbit to the Moon . . . . .	8
2.4 Low-Energy Transfer Orbits - Slow But Cost-Effective . . . . .	12
<b>3 Analytical Mechanics</b>	<b>17</b>
3.1 Newtonian Mechanics . . . . .	17
3.2 Lagrangian Mechanics . . . . .	17
3.3 Hamiltonian Mechanics . . . . .	18
3.4 Using Hamiltonian Mechanics . . . . .	19
3.5 The Restricted 3-body Problem . . . . .	21
<b>4 Numerical Analysis</b>	<b>27</b>
4.1 Explicit Euler algorithm . . . . .	28
4.2 Implicit Euler algorithm . . . . .	28
4.3 Symplectic Euler algorithm . . . . .	28
4.4 First Test Runs with First-Order Euler Methods . . . . .	29
4.5 Symplectic Störmer-Verlet . . . . .	29
4.6 Adaptive Symplectic Störmer-Verlet . . . . .	31
<b>5 Simulations of Transfer Orbits</b>	<b>35</b>
5.1 Setup and Circular Orbits . . . . .	35
5.2 Hohmann Transfer Orbits . . . . .	36
5.3 Low-Energy Transfer Orbits (LETO) . . . . .	38
5.4 Results Summary and Discussion . . . . .	44

<b>6</b>	<b>Conclusion</b>	<b>47</b>
<b>A</b>	<b>More details: Hohmann Transfer-Orbit to the Moon</b>	<b>49</b>
<b>B</b>	<b>Detailed Derivation: Coordinate Velocity Transform</b>	<b>55</b>
<b>C</b>	<b>Non-Dimensionalization of Hamiltons Equations</b>	<b>57</b>
<b>D</b>	<b>Miscellaneous</b>	<b>61</b>
D.1	Reduced 3-body Problem Symplectic Euler $x_{i+1}, y_{i+1}$ Mathematica) . . .	61
D.2	Implicit Euler algorithm . . . . .	61
D.3	Medium and Short Hohmann orbits . . . . .	61
<b>E</b>	<b>Python Code for Restricted 3-Body Problem</b>	<b>63</b>
E.1	Main Script . . . . .	63
E.2	Constants . . . . .	70
E.3	Plotter and Trajectory Finders . . . . .	71
E.4	Search Helper Functions . . . . .	76
E.5	Symplectic Adaptive Verlet and Symplectic Euler Integrators . . . . .	80
<b>F</b>	<b>Mathematica Data</b>	<b>85</b>
	<b>Bibliography</b>	<b>89</b>

# CHAPTER 1

## Introduction

---

*“The Moon is the first milestone on the road to the stars.”*

---

— Arthur C. Clarke

During the height of the cold war in December 1968 the NASA Apollo 8 spacecraft made history in being the first manned spacecraft to leave low Earth orbit (LEO) and also the first to orbit the moon. Less than a year later on July 20th 1969 the Apollo 11 spacecraft orbited and landed people on the moon, and was followed up by five more successful moon landing missions the next 3.5 years, Apollo(s) 12, 14, 15-17) [Wikc]. It was a remarkable feat of science and engineering, but manned missions into deep space (beyond LEO) have not been attempted since. Former US President G.W. Bush laid out a plan for NASA in 2004 to conduct manned missions to the moon within 16 years; however this was subsequently cancelled by current US President Barack Obama in 2010 [CNN][BBC]. Nevertheless, NASA recently tested its Orion spacecraft, designed to carry a crew of up to 4 people to asteroids or Mars, and the Space Launch System (SLS), the first rocket boosters designed for human exploration in deep space since the powerful Saturn V rockets of the 1960-70's [Ram15].

With the rise of private enterprise in the space industry, not least SpaceX, the future prospects of space exploration and colonization is promising. SpaceX was “founded with the goal of reducing space transportation costs and enabling the colonization of Mars” [Wike]. SpaceX even more ambitious about the scale of Mars colonization than any national- or intergovernmental organisations. SpaceX CEO Elon Musk wants to put 1 million people on Mars within a century. This will only be possible by making space travel two orders of magnitudes cheaper, driving down the price to tens of dollars per pound of weight. Elon Musk's opinion: the key to achieving this is reusable rockets. Musk in an interview: “‘Excluding organic growth, if you could take 100 people at a time, you would need 10,000 trips to get to a million people,’. ‘But you would also need a lot of cargo to support those people. In fact, your cargo to person ratio is going to be quite high. It would probably be 10 cargo trips for every human trip, so more like 100,000 trips. And we're talking 100,000 trips of a giant spaceship.’”[And].

Today it costs anywhere between \$10,000-\$20,000 per kilogram to launch the payload into LEO [Coo], although SpaceX is working these years on lowering the cost down to just below \$1000 per kilogram. However going into deep space, requires much more energy, and therefore much more fuel and money. The transportation cost for going to Moon orbit is on the order of \$100,000 per kilogram and to land on the moon is on the order of \$1,000,000 per kilogram [Ast][Oka15].

Whenever the destination is beyond LEO, any spacecraft is usually placed in a so-called parking orbit, which is usually a roughly circular orbit around the earth. From this position the propulsion system is fired to initiate a *transfer-orbit* towards the desired location (and often several more thrusts are required along the way to correct or change orbits). Finding the optimal transfer-orbit is therefore of great interest. In general, transfer orbits will always be a tradeoff between the time travelled and the fuel expended. For unmanned missions and supply shipments for a spacestation, base or colony, we want to use low-fuel routes at expense of flight time because they are cheaper. For the remainder of this project we will focus on low-energy transfer orbits to the moon, however all the principles and qualitative conclusions will apply equally well in general and thus for going to Mars [TB14].

We now explained what transfer orbits are and why they are interesting. Let's take a look at the roadmap:

**Chapter 2 transfer orbits** First we will introduce the term delta- $v$  and look at the two most basic types of transfer orbits Hohmann and bi-elliptic. We will show that they are practically identical in delta- $v$  and flight time for realistic Earth-Moon transfer orbits. We will then model a Hohmann transfer-orbit to the moon and derive delta- $v$  prediction for it. Finally we will discuss Lagrange points and their connection to the concept of low-energy transfer orbits.

**Chapter 3 Analytical Mechanics** Here we will give a quick overview of the three major formulations of classical mechanics: Newtonian, Lagrangian and Hamiltonian. We will see how Hamiltonian mechanics is the natural language for orbital mechanics, and how to use it in practice and finally derive the equations of motion for our model problem: the restricted 3-body problem.

**Chapter 4 Numerical Analysis** Here we will go through some numerical techniques for solving non-linear equations of motion and what the various limitations of these methods are. We will see the first-order explicit Euler and symplectic Euler and the second-order symplectic Störmer–Verlet method. An adaptive version of the Verlet method that takes a variable step-size will be explained, implemented and compared to the non-adaptive Verlet.

**Chapter 5 Simulations of Transfer Orbits** Here we will simulate transfer orbits from the Earth to the moon. First we will attempt to find a Hohmann transfer-orbit and compare the results with the results from Chapter 2 and to the Apollo mission. Finally we will attempt to find low-energy transfer orbits that has a lower delta- $v$  than Hohmann by brute force, i.e. shooting in many different directions from a circular Earth orbit with many different velocities.

All basic physical constants and characteristics such as the gravitational constant  $G$ , and planet properties are extracted from Wolfram Mathematica [Wol] and listed in appendix F.

# CHAPTER 2

## Transfer Orbits

---

*“It’s a strange, eerie sensation to fly a lunar landing trajectory—not difficult, but somewhat complex and unforgiving.”*

---

— Neil Armstrong

Transfer orbits is any trajectory taking a spacecraft between two different closed orbits, either around the same celestial body or two different bodies. In general, transfer orbits will always be a tradeoff between flight time (the total time required for the transfer orbit) and fuel consumption for a specific craft - or a tradeoff between flight time vs.  $\Delta v$  for any craft.

In spacecraft flight dynamics, the change in velocity<sup>1</sup> ( $\Delta v$  or delta- $v$ ) is fundamental property of a transfer orbit. The energy- and fuel consumption on the hand hand, does depend on the spacecraft. Energy consumption is a practical concern, but not a key part of the underlying physical problem. This is due to the fact that a spacecraft of any mass need the same change in velocity to perform identical orbital maneuvers. The basic reason for this is the cancellation of the gravitational mass in Newton’s law of universal gravitation and the inertial mass Newton’s 2nd law of motion [KH02], which are the same by the equivalence principles. Hence

$$\mathbf{F} = G \frac{Mm}{r^2} = m\mathbf{a} \quad (2.1)$$

$$\Leftrightarrow \mathbf{a} = \frac{GM}{r^2} \hat{\mathbf{r}}, \quad (2.2)$$

where  $G$  is the gravitational constant,  $M$  is the larger mass,  $m$  is the smaller mass,  $r$  is the separation distance between the two masses, where  $\mathbf{F}$  is the gravitational force vector from  $M$  on  $m$  pointing to  $M$  along the line connecting their center of masses with unit vector  $\hat{\mathbf{r}}$ , and  $\mathbf{a}$  is the acceleration vector of  $m$ . Thus for any spacecraft of mass  $m$ , the equation of motion (and hence its trajectory) will be the same independent of the mass  $m$ , given same initial conditions and that  $m \ll M$  so  $m$  does not affect the motion of  $M$ . We can easily generalize to an arbitrary number of large masses ( $M, m \dots$ ). This is all assuming point masses and that the small mass  $m$  is sufficiently small as to not affecting the motion of the big masses ( $M, m \dots$ ) significantly. This is the case for spacecrafts of

---

<sup>1</sup>Velocity and speed are often used interchangeably in orbital mechanics, but fortunately this rarely leads to confusion in practice.

the vicinity of the Earth, whose motion is dominated by the gravitational fields of the Sun, Earth and Moon.

So in planning- and optimizing orbital trajectories,  $\Delta v$  is the quantity of interest, a quantity that we may want to optimize along with flight time. Conversely, knowing the mass and fuel capacity of a spacecraft beforehand, determines the so-called  $\Delta v$  budget that the spacecraft have at its disposal for various maneuvers.

Before we delve into the different kinds of transfer orbits, we need to introduce some terminology. We imagine a large mass  $M$  placed at focus point  $F$  and a body of small mass  $m$  orbiting it, figure 2.1 [MD99]. Kepler's first law states that in a 2-body system with a large mass  $M$  and a small negligible mass  $m$ , the orbit of the small body is an ellipse with the large body at one of the foci [KH02]

$F, F' = \mathbf{Focus}$  Ellipses have two foci.

$\mathbf{r} = \mathbf{Position\ vector\ of\ smaller\ orbiting\ body}$

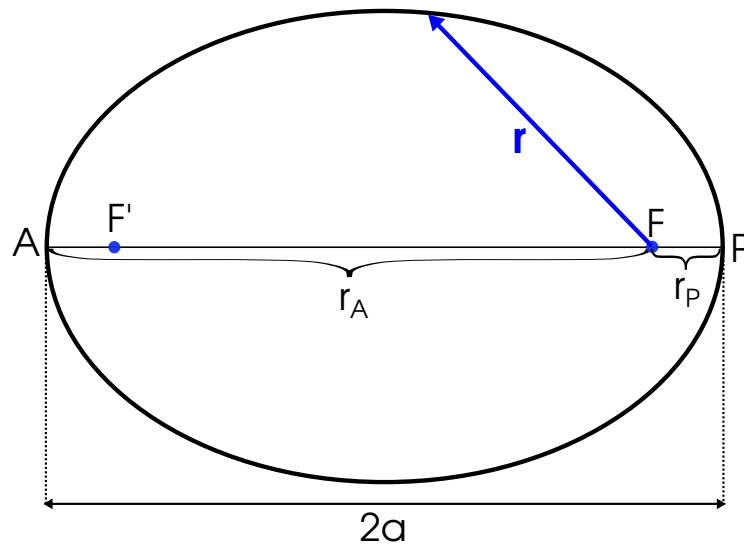
$A = \mathbf{Apoapsis}$  The point on the orbit of greatest distance to the central body<sup>2</sup>.

$P = \mathbf{Periapsis}$  The point on the orbit of least distance to the central body<sup>3</sup>.

$r_A = \mathbf{Distance\ from\ central\ body\ to\ apoapsis}$

$r_P = \mathbf{Distance\ from\ central\ body\ to\ periapsis}$

$a = \mathbf{Semi-major\ axis}$  Half the greatest diameter of the ellipse.



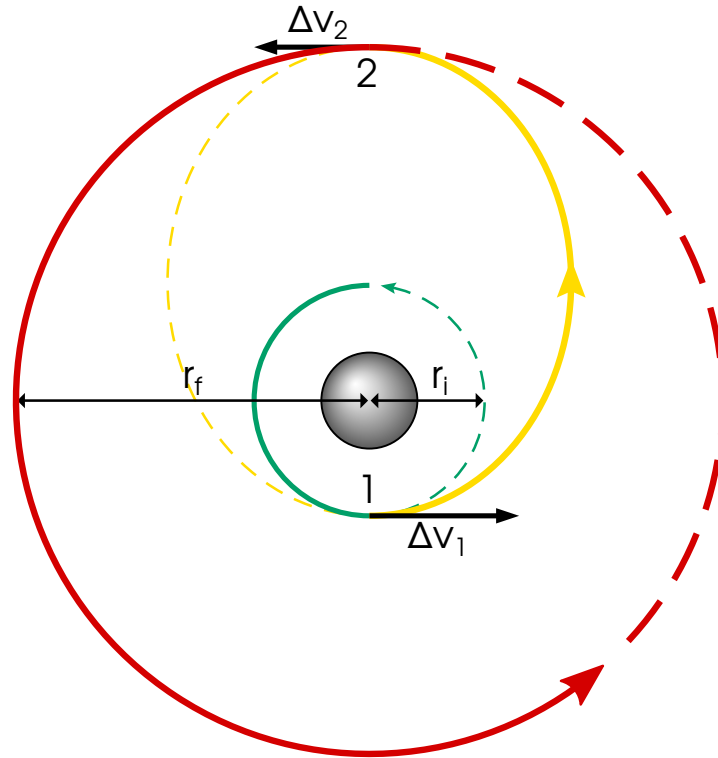
**Figure 2.1:** The elliptical orbit. See definitions just above figure. Source: [Use] (modified).

<sup>2</sup>Also known as the apocentre. If Sun/Earth is central body, also known as aphelion/apogee.

<sup>3</sup>Also known as the pericentre. If Sun/Earth is central body, also known as perihelion/perigee.

## 2.1 Hohmann Transfer-Orbit

The Hohmann transfer orbit is an elliptical orbit used to transfer between two circular orbits in the same plane. This is done by doing two engine impulses known as *burns* along the trajectory, see figure 2.2.



**Figure 2.2:** The Hohmann transfer orbit. We start in circular orbit around central body at distance  $r_i$ , 1.) Burn applied prograde (same direction as velocity vector) to change velocity by  $\Delta v_1$ , now in elliptical orbit around central body, 2.) Burn applied prograde (opposite velocity vector) at apoapsis to change velocity by  $\Delta v_2$ , now in circular orbit at distance  $r_f$ . Source: [Bar] (modified).

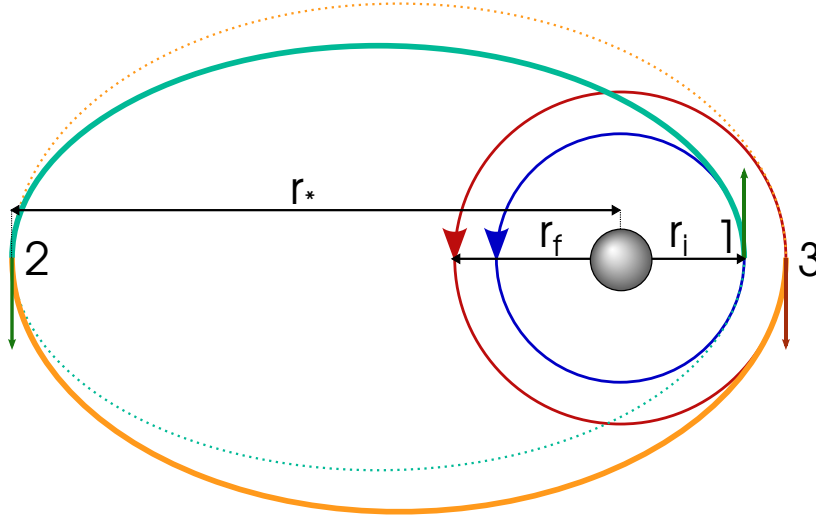
All the Apollo missions launched the spacecraft from Earth into a so-called parking orbit around Earth [Whe]<sup>4</sup>, then a Hohmann-like transfer orbit to the moon, where they achieved circular lunar orbit 99 km above the moon surface. We will assume an initial parking orbit of 160 km above the earth's surface and a capture orbit 100 km above the moon surface.

It has been proven that the Hohmann transfer-orbit is the most  $\Delta v$ -efficient transfer-orbit from circle-to-circle (or ellipse-to-ellipse) in a 1-body system when  $r_f/r_i$  is less than  $\approx 11.94$  [Pru92] [Pee] [BJ00].

<sup>4</sup>An intermediate low circular orbit that a spacecraft is placed into temporarily after launch to ensure that all systems are nominal and calculate the precise transfer orbit to the destination

## 2.2 Bi-elliptic Transfer-Orbit

The bi-elliptic transfer is two elliptical orbits used to transfer between two circular orbits. Thrust is applied at three points instead of Hohmann's two, see figure 2.3. The basic idea is that the first prograde burn will bring the spacecraft close to escape velocity, then at the apoapsis another prograde burn will increase the periapsis distance (see fig 2.1) to the desired distance. Finally at periapsis a retrograde (opposite velocity vector) burn will make the orbit circular.



**Figure 2.3:** The bi-elliptic transfer-orbit. We start in a circular orbit around the central body at distance  $r_i$ , 1.) Big burn applied prograde to change velocity close to escape velocity, now in arbitrarily big elliptical orbit with  $r_*$  as apoapsis distance, 2.) Small burn applied prograde at apoapsis to increase periapsis distance from  $r_i$  to  $r_f$ , 3.) Small burn applied retrograde at periapsis to decrease apoapsis distance from  $r_*$  to  $r_f$ , now in circular orbit at distance  $r_f$ . Source: [Buc] (modified).

It has been proven that the bi-elliptic transfer-orbit is the most  $\Delta v$ -efficient transfer-orbit from circle-to-circle (or ellipse-to-ellipse) in a 1-body system when  $r_f/r_i$  is greater than  $\approx 11.94$  [Pru92] [Pee]. We will not consider the bi-elliptic to the Moon. The reason for this is outlined below.

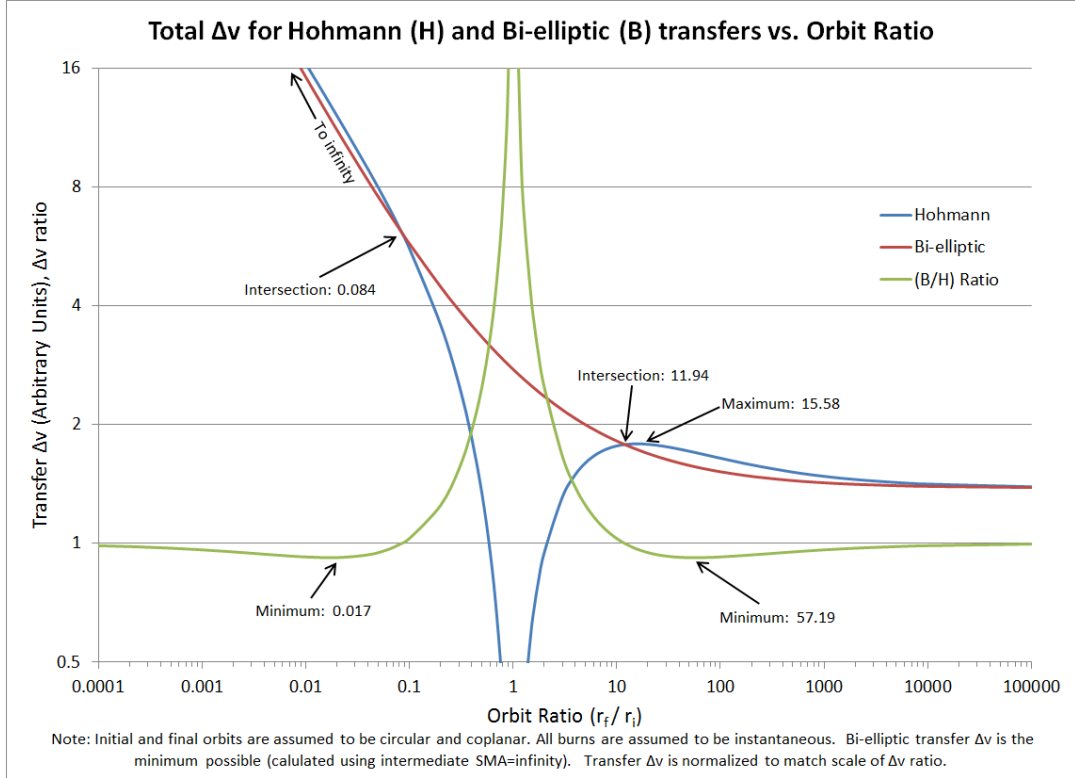
In our case  $r_i = 160$  km (Earth parking orbit) and  $r_f = 3.85 \times 10^5$  km so

$$r_f/r_i = \frac{3.85 \times 10^5 \text{ km}}{6367 \text{ km} + 160 \text{ km}} \approx 59.0 \quad (2.3)$$

The closer the velocity is to escape velocity right after the first burn at point in fig 2.3, the greater the apoapsis distance  $r_*$  will be, going to infinity at escape velocity. As  $r_*$  increases, the flight time tends to infinity and the bi-elliptic orbit gets more efficient. Note that the  $\Delta v$  of the Hohmann- and the bi-elliptic transfers converges to the same



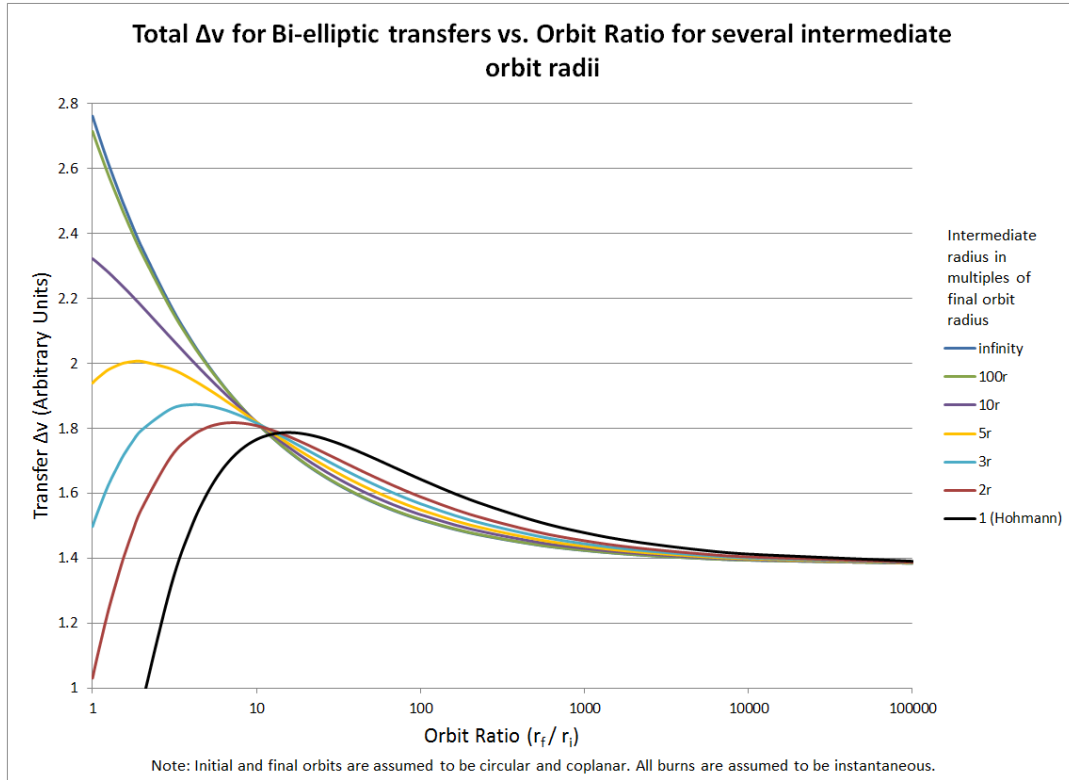
value in the limit as  $r_f/r_i \rightarrow \infty$  see figure 2.4. Note also the optimality of the bi-elliptic transfer at  $r_f/r_i = 57.19$  which is very close to our 59.0. Figure 2.5 shows efficiencies for



**Figure 2.4:** A plot of delta- $v$  as a function of the orbit ratio  $r_f/r_i$  for Hohmann and bi-elliptic transfer orbits. Here  $r_i$  is the initial orbit distance and  $r_f$  is the final orbit distance. We see that they intersect at  $r_f/r_i \approx 11.94$ , where bi-elliptical becomes more efficient than Hohmann, but they converge as  $r_f/r_i \rightarrow \infty$ . Note the local minimum of the ratio  $\Delta v_{\text{Hohmann}}/\Delta v_{\text{bi-elliptic}}$  at 57.19, very close to our 59.0 (caveat: note that  $r_*$  is set to  $\infty$ ). Source:[Cop13].

various intermediate orbital radii  $r_*$ . To take advantage of the bi-elliptic transfer, we must make  $r_*$  as large as possible. This is not achievable in practice due to both very long flight times and perturbations from other bodies. As an example, going from LEO at 6700 km above the earth to 1.3 times the distance to the moon, requires 99.0% of the Hohmann  $\Delta v$  [Wika]. So it quickly becomes a big tradeoff in flight time to save just a little  $\Delta v$ . While bi-elliptic transfers are considered more complicated and risky for bigger transfers and so are rarely used for that [Tay09, p. 78], its often used when performing small,  $0 < r_f/r_i < 1$  maneuvers<sup>5</sup>. We will not concern ourselves with the

<sup>5</sup>Soyuz capsules heading for ISS [ESA14], launched from Earth into parking orbit use bi-elliptic transfers to dock with ISS. The mid-burn gives opportunity to correct the trajectory. And safety is of great concern for a \$100 billion [Min] space station.



**Figure 2.5:**  $\Delta v$  for various intermediate orbit apoapsis radii  $r^*$  as function of the ratio  $r_f/r_i$ , where  $r_i$  is the initial orbit distance and  $r_f$  is the final orbit distance. Once gain we see the intersect at  $r_f/r_i = 11.94$ , where bi-elliptical becomes more efficient than Hohmann, the two of them converge as  $r_f/r_i \rightarrow \infty$ . Source: [Cop13].

bi-elliptic transfer in going to the moon, but consider it practically identical to the Hohmann transfer-orbit.

## 2.3 Hohmann-Approximated Transfer-Orbit to the Moon

We will now apply the Hohmann transfer orbit, a model for a 1-body system, as an approximation to find a transfer orbit to the moon, with the Earth-Moon system being a 2-body system. We work on the assumptions:

1. **Sun is neglected.** The gravitational pull from the sun is neglected. It is not insignificant compared to the earth and the moon, but it's a first-order effect. As long as we do not get too far away from the Earth-Moon system, it should not be a problem.

**2. Moon orbit assumed circular and co-planar with the earth** We assume orbital eccentricity  $e = 0$  (perfect circle) and inclination angle (angle between orbital planes of the moon and Earth)  $\phi = 0$ . These assumptions are valid, since the true values are  $e = 0.0549$  and  $\phi = 5.16^\circ$  [Wol]. This assumption reduces the problem from 3 to 2 dimensions, greatly simplifies all equations and does not affect the qualitative conclusions.

**3. Spacecraft does not affect Earth or Moon** The mass of the spacecraft itself is so small that it does not affect the motions of the earth and the moon. With this assumption the 3-body problem is called *restricted*.

**4. Hohmann transfer can approximated a trans-lunar injection + lunar orbit insertion**

A trans-lunar injection (TLI) is a propulsive maneuver used to set a spacecraft on a trajectory that will cause it to arrive at the moon [NAS66]. We assume that a Hohmann transfer, modified to end with a velocity that matches the centripetal velocity required for circular motion around the moon, is an adequate model. We will also neglect the gravitational attraction from the moon on the spacecraft while it is in transit, up until the point when the spacecraft is in the circular orbit around the moon.

Our Hohmann transfer orbit to the moon follows the same steps as the standard Hohmann transfer described in chapter 2.1, except we end with a slightly different velocity to match a circular orbit centered around the moon, see figure 2.6.

First we need the *vis viva* equation [Pis05], which is just energy conservation applied to elliptical orbits. It serves to relate the position and velocity in an ellipse:

$$v^2 = \mu \left( \frac{2}{r} - \frac{1}{a} \right), \quad (2.4)$$

$v$  is the velocity,  $\mu = GM$  is the standard gravitational parameter,  $M$  is the central body mass,  $r$  is distance from Earth (at focus) to spacecraft and  $a$  is the semi-major axis, see figure 2.6.

Using figure 2.6 as reference, we will calculate the total  $\Delta v$  to perform a Hohmann transfer orbit using figure 2.6 as reference, from a 160 km circular orbit around the earth to a 100 km circular orbit around the moon. We have the quantities [Wol]

$$\mu_E = Gm_E = 6.67 \times 10^{-11} \frac{\text{m}^3}{\text{kg s}^2} \cdot 5.97 \times 10^{24} \text{ kg} = 3.99 \times 10^5 \text{ km}^3/\text{s}^2 \quad (2.5)$$

$$\mu_M = Gm_M = 6.67 \times 10^{-11} \frac{\text{m}^3}{\text{kg s}^2} \cdot 7.35 \times 10^{22} \text{ kg} = 4.90 \times 10^3 \text{ km}^3/\text{s}^2 \quad (2.6)$$

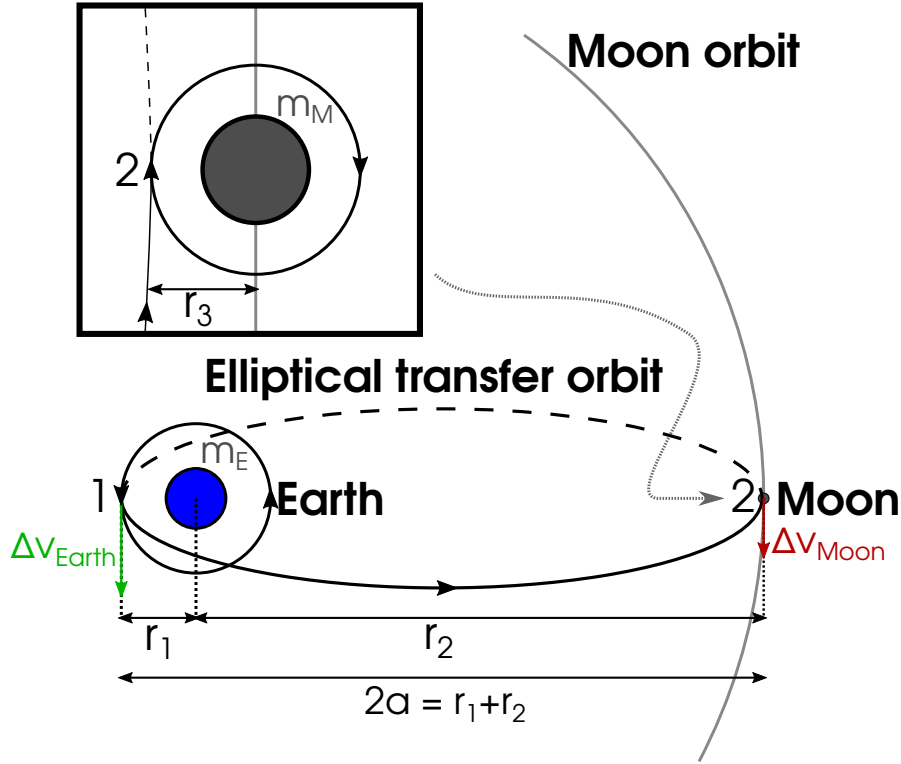
$$r_1 = \text{Earth radius} + 160 \text{ km} = 6367 \text{ km} + 160 \text{ km} = 6527 \text{ km} \quad (2.7)$$

$$r_2 = \text{average Earth-Moon distance} = 3.85 \times 10^5 \text{ km} \quad (2.8)$$

$$r_3 = \text{Moon radius} + 100 \text{ km} = 1738 \text{ km} + 100 \text{ km} = 1838 \text{ km}. \quad (2.9)$$

$$(2.10)$$

All the following  $\Delta v$  impulse calculations use the vis-viva equation (2.4) with various values for  $r$  and  $a$  and velocities are in the geocentric system unless otherwise specified.



**Figure 2.6:** A direct transfer orbit to the moon approximated by Hohmann transfer-orbit.  $m_E$  is the mass of the earth,  $m_M$  is the mass of the moon. We start in circular orbit around the earth at distance  $r_1$  with velocity  $v_{\text{earth}}$  1.) Burn applied prograde to change velocity by  $\Delta v_{\text{earth}}$ , now in elliptical orbit around the earth with  $r_2$  as apogee distance, 2.) Smaller burn applied retrograde at apogee to match a circular orbit around the moon at distance  $r_3$  to Moon and velocity  $v_{\text{moon}}$  in Moon-system.

### 2.3.1 Impulse at Earth

Velocity in circular Earth orbit at distance  $r_1$  ( $r = r_1$ ,  $a = r_1$ ):

$$v_{\text{earth}} = \sqrt{\mu_E \left( \frac{2}{r_1} - \frac{1}{r_1} \right)} = \sqrt{\frac{\mu_E}{r_1}} = 7.814 \text{ km/s} \quad (2.11)$$

Velocity in elliptical Earth orbit at perigee distance  $r_1$  ( $r = r_1$ ,  $a = (r_1 + r_2)/2$ ):

$$v_p = \sqrt{\mu_E \left( \frac{2}{r_1} - \frac{1}{\frac{r_1 + r_2}{2}} \right)} = \sqrt{\frac{\mu_E}{r_1} \frac{2r_2}{r_1 + r_2}} = 10.96 \text{ km/s} \quad (2.12)$$

So the necessary delta- $v$  at Earth:

$$\Delta v_{\text{earth}} = v_p - v_{\text{earth}} = 3.144 \text{ km/s} \quad (2.13)$$

### 2.3.2 Impulse at Moon orbit - ignoring the moon

Velocity in elliptical Earth orbit at apogee distance  $r_2$  ( $r = r_2$ ,  $a = (r_1 + r_2)/2$ ):

$$v_a = \sqrt{\mu_E \left( \frac{2}{r_2} - \frac{1}{\frac{r_1+r_2}{2}} \right)} = \sqrt{\frac{\mu_E}{r_2} \frac{2r_1}{r_1 + r_2}} = 0.1858 \text{ km/s} \quad (2.14)$$

Velocity in circular Earth orbit at distance  $r_2$  ( $r = r_2$ ,  $a = r_2$ ):

$$v_{\text{earth2}} = \sqrt{\frac{\mu_E}{r_2}} = 1.017 \text{ km/s} \quad (2.15)$$

So the necessary  $\Delta v$  at Moon orbit to enter circular Earth orbit (if the moon was not there) is an increase in velocity of

$$\Delta v_{\text{no-moon}} = v_{\text{earth2}} - v_a = 0.832 \text{ km/s} \quad (2.16)$$

### 2.3.3 Impulse at Moon orbit - including the moon

However we can't follow the moon stationary along its orbit, we need to go into a circular orbit around the moon, so we'll make a correction to our  $\Delta v_{\text{no-moon}}$ .

Velocity in circular orbit at distance 100 km above the moon surface *in Moon system*:

$$v_{\text{moon}} = \sqrt{\frac{\mu_M}{r_3}} = 1.633 \text{ km/s} \quad (2.17)$$

Now we can choose to add or subtract this value to  $\Delta v_{\text{no-moon}}$  since we can go into orbit both ways around the moon (clockwise and counterclockwise). We will subtract  $v_{\text{moon}}$  from  $\Delta v_{\text{no-moon}}$  since this minimizes the  $\Delta v$ .

So the necessary change in velocity needed to go from geocentric elliptical orbit to circular Moon orbit is a decrease in velocity:

$$\Delta v_{\text{moon}} = \Delta v_{\text{no-moon}} - v_{\text{moon}} = -0.8017 \text{ km/s} \quad (2.18)$$

So the total  $\Delta v$  for the whole trip is

$$\Delta v_{\text{total}} = |\Delta v_{\text{earth}}| + |\Delta v_{\text{moon}}| = 3.946 \text{ km/s}. \quad (2.19)$$

The flight time is approximately the time spend on the elliptical orbit, which is half the elliptical orbital period, so by Kepler's third law [MD99]:

$$t_{\text{Hohmann}} = \frac{1}{2} \sqrt{\frac{4\pi^2 \left( \frac{r_1+r_2}{2} \right)^3}{\mu_E}} = 5.0 \text{ days} \quad (2.20)$$

A more detailed analysis of a Hohmann-like LTI with identical initial and final orbits, turns out a very similar  $\Delta v_{\text{total}} = 3.959 \text{ km}$  [Swe91] [Juu08]. This seems to somewhat justify our fourth and perhaps seemingly most questionable assumption at the beginning of the section.

Coincidentally our  $\Delta v_{\text{total}}$  in 2.19 is actually pretty close to the moon-less scenario:

$$\Delta v_{\text{total-no-moon}} = |\Delta v_{\text{earth}}| + |\Delta v_{\text{no-moon}}| = 3.976 \text{ km/s} \quad (2.21)$$

This makes it relevant to pose the question: could we possibly minimize  $\Delta v_{\text{total}}$  by choosing a moon orbit altitude such that the necessary moon orbiting velocity in the moon system is equal to the incoming velocity on the ellipse at the intersection in the Earth system? In effect, meaning no velocity change necessary at point B in fig 2.6, i.e.  $\Delta v_{\text{moon}} = 0$ ? We set  $r_3 = \text{Moon radius} + x$  and solve  $\Delta v_B = 0$  for  $x$  and find  $x = 5350 \text{ km}$  and  $\Delta v_{\text{total-min}} = 3.144 \text{ km/s}$  (see A for calculations and  $\Delta v_{\text{total}}$  vs. Moon orbital distance plot). This is a valid orbital altitude since its well within the moon's Hill sphere of radius of roughly 62.000 km. The Hill sphere radius marks the point between Earth and Moon at which a spacecraft switches from orbiting one to orbiting the other. In more precise terms the Hill sphere is the approximate equipotential surface around a smaller body  $m$  orbiting a larger body  $M$  at distance  $r_{m,M}$ , indicating the boundary of the region where a smaller spacecraft  $m_s$  would orbit  $m$  rather than  $M$ . In other words its the distance from  $m$  where the forces from  $M$  and  $m$  add up to the centripetal force towards  $M$ . The radius of the Hill sphere is given by [MD99]

$$r_{\text{hill}} = r_{m,M} \sqrt[3]{\frac{m}{3M}} \quad (2.22)$$

The Hill sphere of the moon in our Earth-Moon system is (appendix A):

$$r_{\text{hill-moon}} = r_2 \sqrt[3]{\frac{m_M}{3m_E}} = 0.16r_2 = 62.000 \text{ km} \quad (2.23)$$

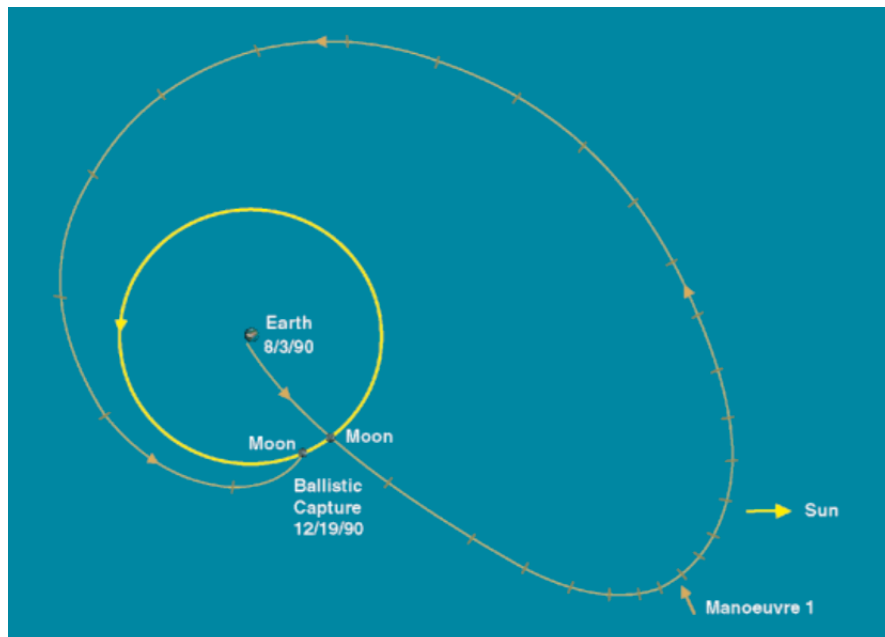
So the Hohmann transfer orbit is the most effective way of getting into a high lunar orbit. It is also the fastest way to the moon. The Apollo missions used 4.115 km/s and took 3.05 days. However ultimately we're not just interested in getting in *any* circular orbit around the moon, but in low lunar orbit (LLO) (lunar orbits less than 100 km altitude [NAS66]) at low  $\Delta v$  for the purpose of unmanned orbital missions- or landings.

Now that we fully understand the Hohmann-like TLI, its performance and limitations, we're ready to look at the low-energy transfer orbits.

## 2.4 Low-Energy Transfer Orbits - Slow But Cost-Effective

In 1990 the Japanese spacecraft "Hiten" was launched and failed part of its mission, ending in a high elliptical around the Earth, however with insufficient fuel to perform

the Hohmann transfer-orbit maneuver to reach the Moon. Edward Belbruno proposed an non-traditional trajectory that could get the spacecraft to the Moon anyway [Wil]. Originally invented by Belbruno in 1987 [See02], this marked the first practical use of a *low-energy transfer orbit* [NASc], that is an orbit that uses little fuel compared to traditional direct-route transfer-orbit such as Hohmann. Figure 2.7 shows the mission trajectory profile. It was a success and marked the beginning of an active research area in low-energy transfer orbits, also known as weak stability boundary trajectories or ballistic capture trajectories.



**Figure 2.7:** Hiten mission profile showing the Belbruno trajectory, which interacts with the moon twice in order to be captured ballistically by the moon. Tick marks are at 5 day intervals. which interacts with the moon twice in order to be captured ballistically by the moon. Source: [BJ00].

In the last section we found that Hohmann transfer orbits to the moon need to brake at the moon to shed 0.832 km/s. The idea of a low-energy transfer-orbit is to have the spacecraft captured ballistically by the moon, spending much less fuel on firing retrorockets when the spacecraft reaches the moon. We can expect savings in  $\Delta v$  up to 150 m/s  $\Delta v$  at the expense of the flight time going from the order of 5 days to 60-100 days [GG07]. A saving of 150 m/s is a small proportion of the total delta- $v$ , but since the  $\Delta v$  and fuel are being saved from a late point in the trip (braking at the moon), it takes a lot of fuel to transport the fuel itself to the moon and the effect is quite significant: upwards of 25% can be saved off the  $\Delta v$  after leaving LEO. As a result under some circumstances, double the payload mass that can be placed into LLO [BC00].

Another advantage is more flexible launch windows. A launch window is the time intervals during which the launch of a particular mission is possible from a specific

location. Launch windows vary in both size and frequency. During the Apollo missions, the monthly launch window consisted of a few days during a given month or lunar cycle. The daily window had a duration of a few hours during a given 24 hour period [NASb]. Since a low-energy transfer-orbit can approach its target from many more directions due to its chaotic nature, they typically provide more flexibility concerning launch time windows compared to the more traditional Hohmann transfers. It has been demonstrated that it is possible to achieve lunar orbit for a 20 minute launch window on a given day for about only 50 meters/second  $\Delta v$ . Furthermore, for an additional 50 meters/second a launch period of about 5 days per lunar cycle can be achieved [BC00].

The flexibility advantage is even more pronounced for Mars, where three advantages can be highlighted: the  $\Delta v$  performance, the much more flexible launch windows (Hohmann to Mars is just a few times a year), and finally its safer than Hohmann in the sense that the capture  $\Delta v$  is applied far from Mars and in a gradual manner. By comparison, the capture process for a Hohmann transfer needs to be done very quickly and be precisely timed or the spacecraft is lost. [TB14].

In any optimization scheme it is very useful to know the theoretical limit. For a trajectory that starts in a circular orbit 167 km above the Earth orbit and ends at a 100 km altitude circular polar moon orbit, [Swe91] found a lower bound of  $\Delta v$  at 3.721 km/s. Unfortunately, the associated trajectory does not exist since it requires, theoretically, an infinite time to approach  $L_1$  and to depart from it [TVB05].

Even in the simplest model, the planar circular restricted 3-body problem that we have chosen, the system is chaotic. Therefore the low-energy transfer orbits cannot be found analytically and we must apply some trial-and-error to find them. However instead of relying completely on brute force, randomly searching through all possible trajectories, we can be guided by theory some of the way by studying the energy contours of the system. To this end it is useful to introduce Lagrange points.

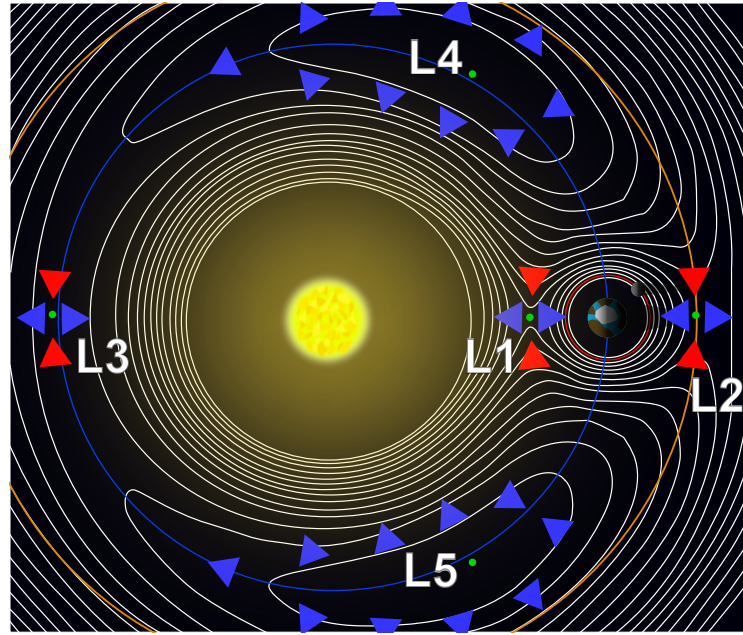
## 2.4.1 Lagrange Points and Low-Energy Transfer Orbits

For two large bodies of masses  $M$  and  $m$  there exists points in space where the sum of the gravitational forces from the two bodies equals the centripetal force required to orbit with them around the center of mass [MD99]. There exists five such points denoted by  $L_1 - L_5$ , see figure 2.8.

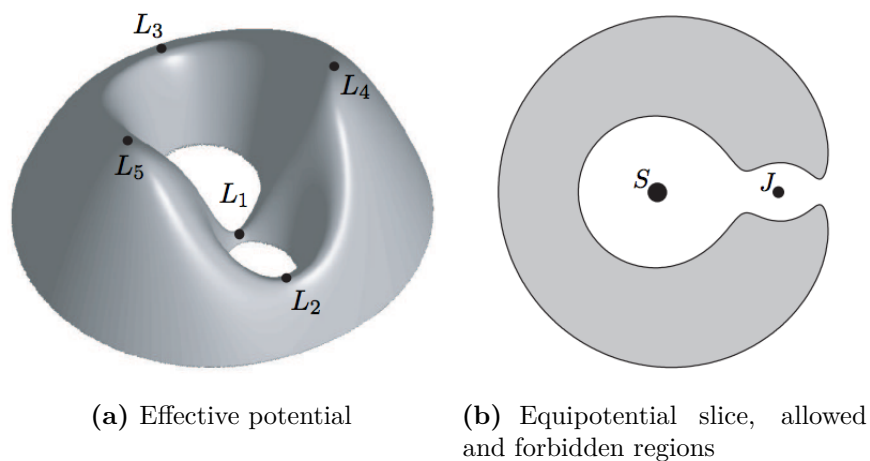
If  $M \gg m$ , then the distance to  $L_1$  and  $L_2$  from  $m$  are exactly equal to the Hill sphere (2.22).

Looking at figure 2.9, we see that the lowest energy orbits must go through the saddle point or “neck” with the Lagrange points  $L_1$  in the middle. However if we go straight to the Moon through the neck, we’ve seen that we arrive at the Moon with an excess velocity and must spend energy on breaking. Using a trajectory-search program to find trajectories through a region around  $L_1$ , we expect to find some Moon capture trajectories with lower delta- $v$  than Hohmann.





**Figure 2.8:** Contour plot of the effective potential due to gravity of a two-body system in a rotating frame of reference. The red and blue arrows indicate positive and negative potential gradients, respectively. Also shows all the Lagrange Points  $L_1 - L_5$  are indicated by green dots. They are points of equilibria, or stationary points where the all the gravitational forces and centrifugal force cancel out. Note that this picture depicts the potential and Lagrange points of Sun-Earth (the moon is only there for show) but it applies equally well the Earth-Moon or any other approximate 2-body system.



**Figure 2.9:** Visualizations of the potential. For some energy, 2.9(b) shows the allowed regions in white, forbidden in gray..



# CHAPTER 3

## Analytical Mechanics

---

*“If Nature has defined the mechanics problem of the thrown ball in so elegant a fashion, might She have defined other problems similarly. So it seems now. Indeed, at the present time it appears that we can describe all the fundamental forces in terms of a Lagrangian. The search for Nature’s One Equation, which rules all of the universe, has been largely a search for an adequate Lagrangian.”*

---

— Robert Adair, *The Great Design: Particles, Fields, and Creation*

### 3.1 Newtonian Mechanics

Since the early beginnings of classical mechanics, there have existed fundamentally two different approaches to the study of motion: vectorial (“classical”) and variational (“analytical”) treatment of mechanics. The first, a differential approach, the second, an integral approach.

The vectorial treatment was pioneered by Isaac Newton and is based on two fundamental vectors: momentum and force. Analysis of the forces acting on a body yield second-order differential equations of motion by Newton’s 2nd Law, equation (3.1). These differential equations can be solved to yield position and velocity at every point in time.

$$\text{Newton's 2nd Law} \quad \sum \mathbf{F} = \frac{d\mathbf{p}}{dt} \quad (3.1)$$

### 3.2 Lagrangian Mechanics

The first seeds of the variational approach was sown by G.W. Leibniz, a contemporary of Newton. Leibniz advocated another quantity “vis viva ” (living force), which differs from what we now call “kinetic energy” by a factor of 2. Thus Leibniz replaced Newton’s “momentum” with “kinetic energy” and “force” with “work of the force”. [Lan70, p. xxi]. The power of the variational approach is that these two scalar quantities can completely contain all the dynamics of any system. Around 1743 Euler discovered what later became known as the Euler-Lagrange equations (equations (3.3) [Lan70, p. 61]) using geometrical methods [Gol80, p. 67] [Sta]. In 1755 J.L. Lagrange re-derived Euler’s

results using just algebraic methods, a method that Euler dubbed “calculus of variations” [Gol80, p. xiv] [Sta]. Throughout the 18th century new developments analytical mechanics was made, culminating in Lagrange’s publication “Mécanique Analytique” in 1788 [Fra83, p. 197]. The central quantity in Lagrangian mechanics is called the Lagrangian [Hjo15, p. 3]

$$\textbf{Lagrangian} \quad L = T - V \quad (3.2)$$

, where  $T$  is the total kinetic energy of the body and  $V$  is the total potential energy of the system. The central equation is called the Euler-Lagrange Equation [Hjo15]

$$\textbf{Euler-Lagrange equation} \quad \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = 0 \quad , (k=1,2,\dots,n) \quad , \quad (3.3)$$

where  $q$  are the generalized coordinates (will be introduced in the next chapter) where the dot means the time derivative.

### 3.3 Hamiltonian Mechanics

Classical mechanics underwent one last important reformulation. In 1834 W.R. Hamilton reformulated classical mechanics based on Lagrange’s powerful formalism [NF02, p. 161]. Hamilton’s formalism is a systematic, automatic way to recast the 2nd order equations of Newton and Lagrange, and make it a first-order order system. It builds on the Lagrangian but relies on a new quantity called the generalized momentum

$$\textbf{Generalized momenta} \quad p_i(\mathbf{q}, \dot{\mathbf{q}}) = \frac{\partial L}{\partial \dot{q}_i}, \quad i = 1, \dots, n \quad (3.4)$$

$$\textbf{Hamiltonian} \quad H(\mathbf{q}, \mathbf{p}, t) = \sum_{i=1}^n p_i \dot{q}_i - L \quad (3.5)$$

$$\begin{aligned} \textbf{Hamilton's equation} \quad \dot{q}_i &= + \frac{\partial H}{\partial p_i}, \\ \dot{p}_i &= - \frac{\partial H}{\partial q_i} \end{aligned} \quad (3.6)$$

[Hjo15] Thus Lagrangian mechanics and Hamiltonian mechanics is energy-centric rather than force-centric. One of the great strengths of the variational approach is that the system can be more or less solved simply by appropriate choice of coordinates. This is due to the fact that appropriate coordinates automatically accounts for the constraints of movements (e.g. a bead on a string, a ball rolling on a surface or electrical charge moving on a surface) so we don’t have to account for these constraints by complicated

force analysis. These coordinates  $q_i$  we call *generalized coordinates*. They are “any quantitative attributes of the system (for example, strength of the magnetic field at a particular location; angle of a pulley; position of a particle in space; or degree of excitation of a particular eigenmode in a complex system) which are functions of the independent variable(s)” [Wikb]. The number of ways the system can move subject to the constraints is called the degrees of freedom. To fully describe the system, the number of generalized coordinates chosen for the system must equal the system’s degrees of freedom.

Thus instead of analyzing forces, including troublesome forces of constraints, we instead define and use coordinates that only describe states of the system that satisfies the constraints. Then, using the Euler-Lagrange’s equations (3.3) or Hamilton’s equations (3.6), we immediately obtain the equations of motion. Another advantage of the Hamiltonian method is that the underlying symmetries in the system are often more evident in first-order differential equations. On a theoretical level this is important due to Noether’s Theorem: “Each symmetry of a system leads to a physically conserved quantity” [Wei]. It’s also convenient on a practical level with regards to solving 1st order equations numerically. In other words, we prefer  $2n$  1st order partial differential equations (PDE) to  $n$  2nd order ordinary differential equations (ODE).

## 3.4 Using Hamiltonian Mechanics

We will now look at the procedure for using Hamiltonian mechanics in practice. In the next chapter “Numerical Methods” we will then solve the equations of motion, in part to demonstrate the validity of Hamilton’s equations of motion, and in part to analyze how various numerical techniques perform.

### Step 0 Lagrangian $L$

- a) Define general coordinates  $q_i$ ,  $i = 1, 2 \dots n$ , where  $n$  is the degrees of freedom for the system.
- b) Determine kinetic energy  $T(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2}mv^2$ .
- c) Determine potential energy  $V(\mathbf{q}, \dot{\mathbf{q}}, t)$ .
- d) Lagrangian:

$$L = T - V \quad (3.7)$$

### Step 1 Generalized momenta $p_i$

$$p_i(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{\partial L}{\partial \dot{q}_i} \quad (3.8)$$

, for all  $i = 1 \dots n$ .

## Step 2 Transform all the $\dot{q}_i$

Technically called a Legendre transform, this is the step that takes us from Lagrangian mechanics to Hamiltonian mechanics. We basically isolate the  $\dot{q}_i$  in the  $p_i$ -equations from step 1, and eliminate all  $\dot{q}_i$  in the equations in favor of  $p_i$ . Thus we go from independent variables  $(\mathbf{q}, \dot{\mathbf{q}})$  to  $(\mathbf{q}, \mathbf{p})$ .

$$\dot{q}_i = \dot{q}_i(\mathbf{q}, \mathbf{p}, t) \quad (3.9)$$

## Step 3 The Hamiltonian $H$

$$H(\mathbf{q}, \mathbf{p}, t) = \sum_{i=1}^n p_i \dot{q}_i - L \quad (3.10)$$

, for all  $n = 1 \dots n$ , where it is understood that all the  $q_i$  are substituted with expressions found in step 2.

## Step 4 Hamilton's Equations of Motion

$$\begin{aligned} \dot{q}_i &= + \frac{\partial H}{\partial p_i}, \\ \dot{p}_i &= - \frac{\partial H}{\partial q_i} \end{aligned} \quad (3.11)$$

, for  $i = 1 \dots n$ , which gives us  $2n$  1st order coupled PDEs of  $2n$  variables  $(q_1, q_2, \dots, q_n, p_1, p_2, \dots, p_n)$ .

In general  $T$  and  $V$  can be time-dependent (and therefore  $L$  and  $H$  can too). However in many applications, including the our model problem, they are not time-dependent.

An important property of the Hamiltonian is that if it is not explicitly time dependent then it is conserved  $dH/dt = 0$  along the  $(\mathbf{p}(t), \mathbf{q}(t))$  flow [Hjo15].

### 3.4.1 $H$ vs. $E$

An important characteristic of a closed physical system is it's energy  $E$ . By a clever choice of coordinate system, it is possible to have systems where Hamiltonian  $H$  is conserved, but the total mechanical energy  $E$  is not. In that sense in can be argued that the Hamiltonian is a more general concept than energy. It can be shown that  $H = E$  if and only if the following three conditions are met: [GPS02, p. 60-64] [Bra12] [HT10]

1. Equations of constraints,  $T$  and  $V$  have no explicit time-dependency.
2.  $V$  is independent of  $\dot{\mathbf{q}}$ .
3.  $T$  is a homogeneous quadratic in the  $\dot{q}$ 's, in particular if  $T(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \dot{\mathbf{q}}^\top M(\mathbf{q}) \dot{\mathbf{q}}$ , where  $M(\mathbf{q})$  is some symmetric and positive definite matrix ,

Meeting these conditions also implies that the generalized impulses  $p_i$  will be equal to well known conserved quantities such as linear momentum, angular momentum etc. We will later see that the equations for restricted 3-body system satisfy the conditions. Another way to see if  $T$  is a quadratic form is if it's a homogenous polynomial, i.e. all terms have same degree in a number of variables. For example  $P(x, y) = 4x^2 + 2xy + 3y^2$  [Wikd] is quadratic.

## 3.5 The Restricted 3-body Problem

### 3.5.1 Assumptions and Setup

We make the same first three assumptions as in chapter 2.3: The gravitational effects of the sun can be neglected, the moon's orbit around the Earth is assumed to be circular and spacecraft mass has negligible influence on the Earth and Moon. Unlike chapter 2.3, we will now include the additional gravitational influence of the moon, without which it would be impossible to find low-energy transfers to the Moon.

We define two cartesian coordinate systems, both with their origin in the Moon-Earth center of mass (CM). The  $(\mathcal{X}, \mathcal{Y})$  system is a stationary inertial frame, and the  $(x, y)$  system is co-rotating with the Earth and the Moon in their orbits, see figure 3.1.

We have two choices of coordinate systems to use for Lagrangian:

1. If we choose the inertial frame of reference,  $T$  is straight forward, but distances in the  $V$  becomes bothersome, albeit we can use the straightforward gravitational potential function.
2. If we choose the rotating  $(x, y)$  system, distances for  $V$  is straight forward but then we either have to use polar coordinates, use an effective potential function instead of the standard gravitational potential, or use cartesian coordinates but find a coordinate transformation  $X(x, y)$  and  $Y(x, y)$ .

I arbitrarily choose the latter option: use  $(x, y)$  coordinates and find  $X(x, y)$  and  $Y(x, y)$  transformation.

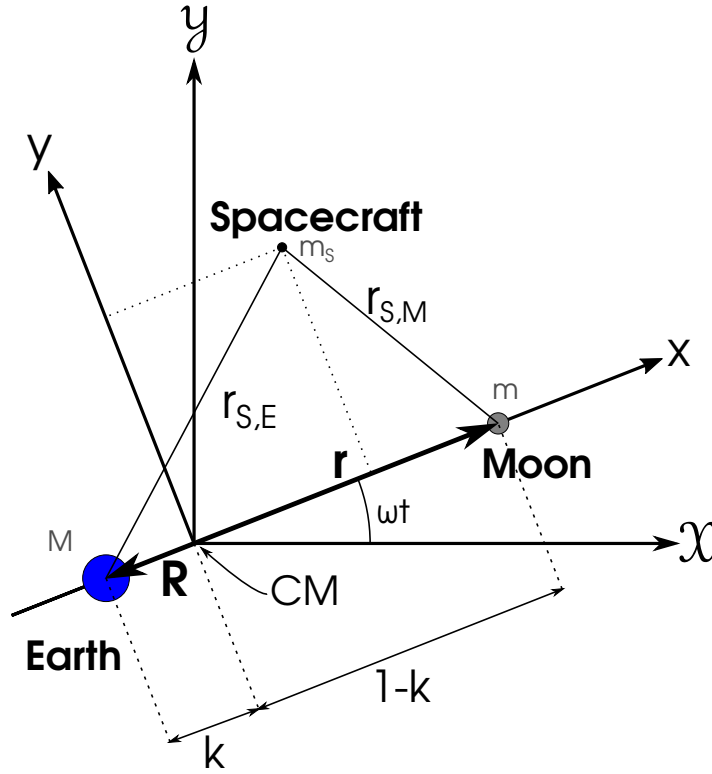
The distances between spacecraft and Earth/Moon are delightfully easy to work out:

$$r_{S,E} = \sqrt{(x + R)^2 + y^2} \quad (3.12)$$

$$r_{S,M} = \sqrt{(x - r)^2 + y^2} \quad (3.13)$$

In anticipation for calculating the kinetic energy of the system, we need the quantity

$$\dot{x}^2 + \dot{y}^2 \quad (3.14)$$



**Figure 3.1:** Restricted 3-body problem. Two coordinate systems both with center of mass as origin:  $(X, Y)$  is a stationary inertial frame,  $(x, y)$  is a co-rotating non-inertial frame that rotates with the moon at angular frequency  $\omega$ .  $M =$  mass of Earth,  $m =$  mass of Moon,  $\mathbf{R} =$  vector from CM to earth,  $\mathbf{r} =$  vector from CM to Moon,  $m_s =$  mass of spacecraft,  $r_{S,E} =$  distance from spacecraft to Earth,  $r_{S,M} =$  distance from spacecraft to Moon. In the dimensionless variables we introduce later, unit distance is  $R + r$ , which makes the dimensionless constant  $k = \frac{R}{R + r} =$  the CM-Earth distance and  $1 - k = \frac{r}{R + r} =$  the CM-Moon distance.

In matrix notation the coordinate transform between  $(X, Y)$  and  $(x, y)$  is obtained by the multiplication of the rotation matrix on the inertial frame coordinates

$$\begin{aligned} \begin{bmatrix} X \\ Y \end{bmatrix} &= \begin{bmatrix} \cos \omega t & -\sin \omega t \\ \sin \omega t & \cos \omega t \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ \Rightarrow \begin{cases} X = x \cos \omega t - y \sin \omega t \\ Y = x \sin \omega t + y \cos \omega t \end{cases} \end{aligned} \quad (3.15)$$

By using the pythagorean trigonometric identity<sup>1</sup> a number of times (see appendix B

---

<sup>1</sup> $\cos^2 \theta + \sin^2 \theta = 1$



for detailed derivation) we obtain

$$\dot{\mathcal{X}}^2 + \dot{\mathcal{Y}}^2 = (\dot{x} - \omega y)^2 + (\dot{y} + \omega x)^2 \quad (3.16)$$

Now we can really see another convenience of the  $(x, y)$  coordinate system: it eliminates explicit time-dependence of  $L$  that would have been there in  $V$  in the  $(\mathcal{X}, \mathcal{Y})$  system.

We are now ready to derive the equations of motion using the procedure from chapter 3.4.

### 3.5.2 Equations of Motion

The kinetic energy is

$$\begin{aligned} T &= \frac{1}{2} m_s (\dot{\mathcal{X}}^2 + \dot{\mathcal{Y}}^2) \\ &= \frac{1}{2} m_s [(\dot{x} - \omega y)^2 + (\dot{y} + \omega x)^2] \end{aligned} \quad (3.17)$$

The potential energy is

$$\begin{aligned} V &= -Gm_s \left( \frac{M}{r_{S,E}} + \frac{m}{r_{S,M}} \right) \\ &= -Gm_s \left( \frac{M}{\sqrt{(x+R)^2 + y^2}} + \frac{m}{\sqrt{(x-r)^2 + y^2}} \right) \end{aligned} \quad (3.18)$$

The Lagrangian is

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2} m_s [(\dot{x} - \omega y)^2 + (\dot{y} + \omega x)^2] \\ &\quad + Gm_s \left( \frac{M}{\sqrt{(x+R)^2 + y^2}} + \frac{m}{\sqrt{(x-r)^2 + y^2}} \right) \end{aligned} \quad (3.19)$$

The generalized momenta are

$$p_x = \frac{\partial L}{\partial \dot{x}} = m_s (\dot{x} - \omega y) \quad (3.20)$$

$$p_y = \frac{\partial L}{\partial \dot{y}} = m_s (\dot{y} + \omega x) \quad (3.21)$$

From which we get the  $\dot{x}(x, p_x)$  and  $\dot{y}(y, p_y)$

$$\dot{x} = \frac{p_x}{m_s} + \omega y \quad (3.22)$$

$$\dot{y} = \frac{p_y}{m_s} - \omega x \quad (3.23)$$

The Hamiltonian is

$$H = \sum_{i=1}^n p_i \dot{q}_i - L \quad (3.24)$$

$$= \frac{p_x^2 + p_y^2}{m_s} + p_x \omega y - p_y \omega x - \frac{1}{2} m_s \left[ \left( \frac{p_x}{m_s} \right)^2 + \left( \frac{p_y}{m_s} \right)^2 \right] \quad (3.25)$$

$$- G m_s \left( \frac{M}{\sqrt{(x+R)^2 + y^2}} + \frac{m}{\sqrt{(x-r)^2 + y^2}} \right) \quad (3.26)$$

$$= \frac{p_x^2 + p_y^2}{2m_s} + p_x \omega y - p_y \omega x - G m_s \left( \frac{M}{\sqrt{(x+R)^2 + y^2}} + \frac{m}{\sqrt{(x-r)^2 + y^2}} \right) \quad (3.27)$$

so Hamilton's equations are

$$\dot{x} = + \frac{\partial H}{\partial p_x} = \frac{p_x}{m_s} + \omega y \quad (3.28)$$

$$\dot{y} = + \frac{\partial H}{\partial p_y} = \frac{p_y}{m_s} - \omega x \quad (3.29)$$

$$\dot{p}_x = - \frac{\partial H}{\partial x} = \omega p_y - G m_s \left[ - \frac{M(x+R)}{((x+R)^2 + y^2)^{3/2}} + \frac{m(x-r)}{((x-r)^2 + y^2)^{3/2}} \right] \quad (3.30)$$

$$\dot{p}_y = - \frac{\partial H}{\partial y} = -\omega p_x - G m_s \left[ - \frac{M y}{((x+R)^2 + y^2)^{3/2}} - \frac{m y}{((x-r)^2 + y^2)^{3/2}} \right] \quad (3.31)$$

By appropriate choice of units we can render the equations dimensionless.

A characteristic distance must be the Earth-Moon distance so:

$$\text{One unit distance} = R + r. \quad (3.32)$$

A natural choice for the characteristic time is the moon's orbital period. Because the time dimension is present in the equations as  $\omega$  we use Kepler's third law to express: [MD99]

$$\text{One unit time} = \frac{1}{\omega} = \sqrt{\frac{(R+r)^3}{G(M+m)}} \quad (3.33)$$

The momentums  $p_x p_y$  of the spacecraft are going to be proportional with the spacecraft's mass  $m_s$ , so let's choose:

$$\text{One unit mass} = m_s \quad (3.34)$$

With these we can calculate:

$$\begin{aligned}
 \text{One unit momentum} &= \text{One unit velocity} = (\text{unit mass}) \frac{\text{unit distance}}{\text{unit time}} \\
 &= m_s(R+r) \sqrt{\frac{G(M+m)}{(R+r)^3}} \\
 &= m_s \sqrt{\frac{G(M+m)}{(R+r)}}
 \end{aligned} \tag{3.35}$$

We can now non-dimensionalize all the equations and get (see appendix C)

$$\boxed{\dot{X} = P_x + Y} \tag{3.36}$$

$$\boxed{\dot{Y} = P_Y - X} \tag{3.37}$$

$$\boxed{\dot{P}_x = P_y - \frac{(1-k)(X+k)}{((X+k)^2 + Y^2)^{3/2}} - \frac{k(X-1-k)}{((X-1-k)^2 + Y^2)^{3/2}}} \tag{3.38}$$

$$\boxed{\dot{P}_y = -P_x - \frac{(1-k)Y}{((X+k)^2 + Y^2)^{3/2}} - \frac{kY}{((X-1-k)^2 + Y^2)^{3/2}}}, \tag{3.39}$$

where  $T$ ,  $X$ ,  $Y$ ,  $P_x$  and  $P_y$  are our dimensionless variables, see appendix C for details.

For the rest of the time we will denote the non-dimensionalized units simply as  $x, y, p_x, p_y$  instead of  $X, Y, P_x, P_y$  for readability.

To reiterate, we have chosen units such that

$$\text{unit length} = 3.85 \times 10^5 \text{ km} \tag{3.40}$$

$$\text{unit time} = 4.3484 \text{ days} \tag{3.41}$$

$$\text{unit velocity} = 1.025 \text{ km/s} \tag{3.42}$$

Since all of the conditions in subsection 3.4.1 are fulfilled, we have that  $H = E$  and since  $H$  does not depend explicitly on time, the Hamiltonian, and thus the energy, is conserved.



# CHAPTER 4

## Numerical Analysis

---

*“A computer lets you make more mistakes faster than any invention in human history - with the possible exceptions of handguns and tequila.”*

— Mitch Ratliffe, Technology Review, April 1992

Some systems can be solved analytically in terms of time-dependent coordinates. Other systems are non-linear and the vast majority cannot be solved analytically. This is the case for the restricted 3-body problem, (3.36) to (3.39). To solve these equations of motion, we must use numerical methods.

The simplest numerical integration is the first order linear approximation known as the Euler method. There are three variants of the Euler method: explicit, implicit and symplectic. As we will see, the explicit Euler will typically increase the energy, implicit Euler will decrease the energy and the symplectic Euler will oscillate the energy slightly but on average conserve it to a high degree.

We will now solve the restricted 3-body problem equations of motion, (3.36) to (3.39). First we discretize the equations

$$\Delta x = +\frac{\partial H}{\partial p_x} \Delta t = (p_x + y) \Delta t \quad (4.1)$$

$$\Delta y = +\frac{\partial H}{\partial p_y} \Delta t = (p_y - x) \Delta t \quad (4.2)$$

$$\Delta p_x = -\frac{\partial H}{\partial x} \Delta t = \left( p_y - \frac{(1-k)(k+x)}{((k+x)^2 + y^2)^{3/2}} + \frac{k(1-k-x)}{((1-k-x)^2 + y^2)^{3/2}} \right) \Delta t \quad (4.3)$$

$$\Delta p_y = -\frac{\partial H}{\partial y} \Delta t = \left( -p_x - \frac{(1-k)y}{((k+x)^2 + y^2)^{3/2}} - \frac{ky}{((1-k-x)^2 + y^2)^{3/2}} \right) \Delta t \quad (4.4)$$

From now on we will use  $h$  instead of  $\Delta t$  for the time step size.

## 4.1 Explicit Euler algorithm

All new values  $(x, y, p_x, p_y)_{i+1}$  refer to known values from the previous timestep  $(x, y, p_x, p_y)_i$ .

$$x_{i+1} = (p_{x,i} + y_i)h + x_i \quad (4.5)$$

$$y_{i+1} = (p_{y,i} - x_i)h + y_i \quad (4.6)$$

$$p_{x,i+1} = \left( p_{y,i} - \frac{(1-k)(k+x_i)}{((k+x_i)^2 + y_i^2)^{3/2}} + \frac{k(1-k-x_i)}{((1-k-x_i)^2 + y_i^2)^{3/2}} \right) h + p_{x,i} \quad (4.7)$$

$$p_{y,i+1} = \left( -p_{x,i} - \frac{(1-k)y_i}{((k+x_i)^2 + y_i^2)^{3/2}} - \frac{ky_i}{((1-k-x_i)^2 + y_i^2)^{3/2}} \right) h + p_{y,i} \quad (4.8)$$

## 4.2 Implicit Euler algorithm

Same as explicit Euler, except that all variables in the right-hand side refers to variables in the current timestep,  $i + 1$ . This algorithm would involve finding four roots simultaneously and numerically. Since we know the Symplectic to be more correct, we skip the implicit Euler for the restricted 3-body problem (algorithm included in appendix D.2).

## 4.3 Symplectic Euler algorithm

New values of  $(x, y)_{i+1}$  refer to unknown values from the same timestep  $(x, y, p_x, p_y)_{i+1}$  but all new values of  $(p_x, p_y)_{i+1}$  refer only to known values from the previous timestep  $(x, y, p_x, p_y)_i$ .

$$x_{i+1} = (p_{x,i+1} + y_{i+1})h + x_i \quad (4.9)$$

$$y_{i+1} = (p_{y,i+1} - x_{i+1})h + y_i \quad (4.10)$$

$$p_{x,i+1} = \left( p_{y,i} - \frac{(1-k)(k+x_i)}{((k+x_i)^2 + y_i^2)^{3/2}} + \frac{k(1-k-x_i)}{((1-k-x_i)^2 + y_i^2)^{3/2}} \right) h + p_{x,i} \quad (4.11)$$

$$p_{y,i+1} = \left( -p_{x,i} - \frac{(1-k)y_i}{((k+x_i)^2 + y_i^2)^{3/2}} - \frac{ky_i}{((1-k-x_i)^2 + y_i^2)^{3/2}} \right) h + p_{y,i} \quad (4.12)$$

As opposed to implicit Euler, this can easily be solved by solving (4.9) and (4.10) for  $x_{i+1}$  and  $y_{i+1}$

$$x_{i+1} = \frac{x_i + h(hp_{y,i+1} + p_{x,i+1} + y_i)}{1 + h^2} \quad (4.13)$$

$$y_{i+1} = \frac{y_i - h(hp_{x,i+1} - p_{y,i} + x_i)}{1 + h^2}, \quad (4.14)$$

and then we can run the algorithm in the order (4.11), (4.12), (4.9) and (4.10).

## 4.4 First Test Runs with First-Order Euler Methods

The explicit and symplectic Euler algorithm was implemented in Python. The first two things we want to investigate are:

1. Does the trajectories look like we expect them to?
2. Is the Hamiltonian conserved?

For symplectic integrators the trajectories look like we expect them to (see 4.2, but not for the non-symplectic explicit Euler (see 4.1). The Hamiltonian is not conserved for either first order (Euler) Methods, but  $H$  for the symplectic Euler was conserved on average over an orbit in a closed orbit. The simulations in figure 4.1 4.2 was run with the following parameters:

```

1 # Duration and stepsize
2 DURATION = 1.5*(2*np.pi)
3 H = 0.0001
4
5 # Initial Conditions
6 X0=0.5
7 Y0=0.2
8 PX0=0
9 PY0=0.5

```

**Listing 4.1:** Initial conditions, step size and duration for initial test run.

Having  $H$  conserved on average over an orbit is fine for a closed orbit, but for chaotic orbits around both Earth and Moon, it's not nearly good enough. We want a higher order method to conserve  $H$  better, to better gauge the numerical errors, and to allow an adaptive numerical method.

Making the algorithm select a stepsize adaptively decreases the runtime of a given simulation (though not necessarily the correctness of the solution). This is a big advantage because it allows us to search for many more orbits, which statistically will give us better orbits.

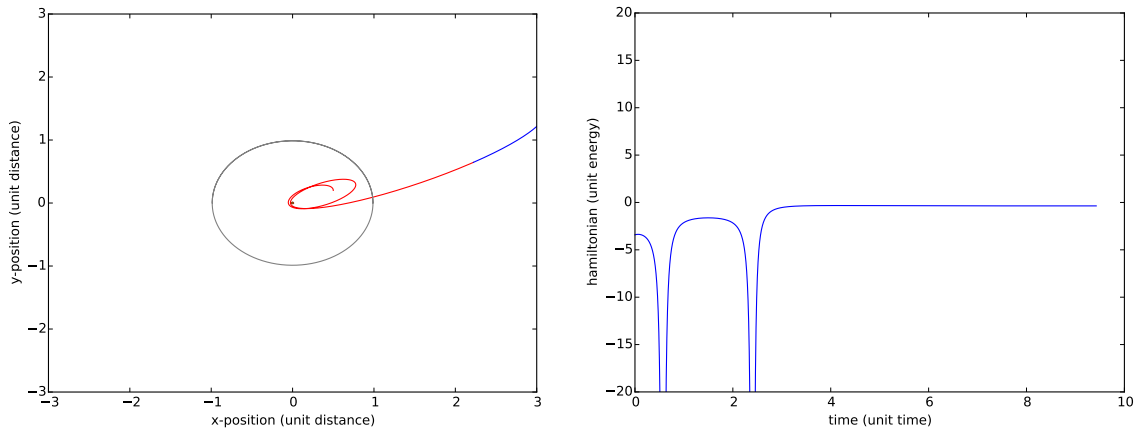
## 4.5 Symplectic Störmer-Verlet

We will now describe the second-order Störmer-Verlet method [Hoc08]:

$$q_{i+1/2} = q_i + \frac{h}{2} H_p(q_{i+1/2}, p_i) \quad (4.15)$$

$$p_{i+1} = p_i + \frac{h}{2} (H_q(q_{i+1/2}, p_i) + H_q(q_{i+1/2}, p_{i+1})) \quad (4.16)$$

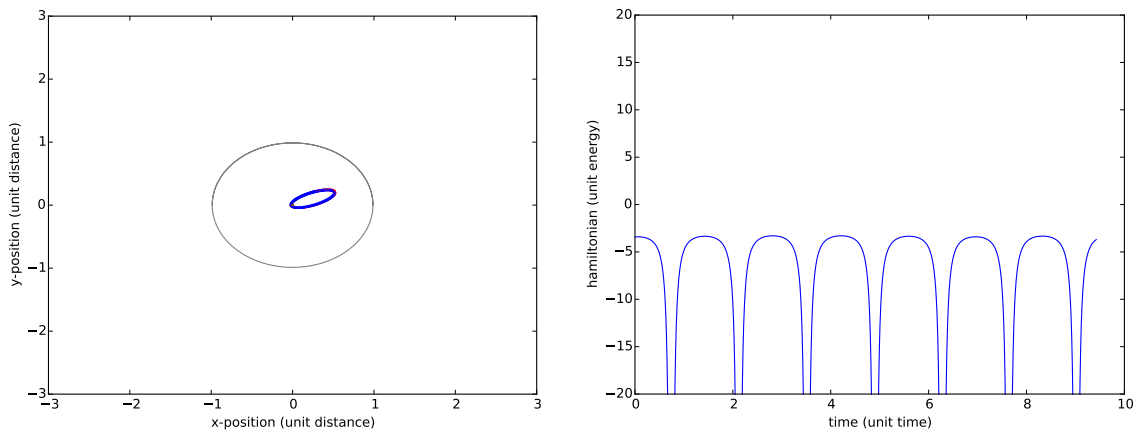
$$q_{i+1} = q_{i+1/2} + \frac{h}{2} H_p(q_{i+1/2}, p_{i+1/2}) \quad (4.17)$$



(a) Explicit Euler  $(\mathcal{X}, \mathcal{Y})$  trajectory. Earth in origin, moon orbit in grey, first half of spacecraft trajectory is red, last half is blue.

(b) Explicit Euler  $H(t)$ .

**Figure 4.1:** Explicit Euler trajectory and Hamiltonian. Ideally the  $H(t)$  should be constant, but instead we see temporary dips for every Earth pass, and worse, the spacecraft picks up energy after every pass of Earth. Finally the spacecraft escapes the Earth with no apparent energy input. This is clearly wrong and not a valid trajectory.



(a) Symplectic Euler  $(\mathcal{X}, \mathcal{Y})$  trajectory. Earth in origin, moon orbit in grey, first half of spacecraft trajectory is red, last half is blue.

(b) Symplectic Euler  $H(t)$ .

**Figure 4.2:** Symplectic Euler trajectory and Hamiltonian. The first half trajectory is colored red, the last half is blue. First we note the approximate elliptical orbit as expected. The  $H(t)$  shows dips as the spacecraft passes the Earth, but on average over an orbit,  $H(t)$  is approximately constant.



where  $H_q$  and  $H_p$  denotes the partial derivatives of  $H$ . Implemented for our equations we get

$$x_{i+1/2} = (p_{x,i} + y_{i+1/2}) \frac{h}{2} + x_i \quad (4.18)$$

$$y_{i+1/2} = (p_{y,i} - x_{i+1/2}) \frac{h}{2} + y_i \quad (4.19)$$

$$p_{x,i+1} = \left[ -H_{x,i} + (p_{x,i+1} + y_{i+1/2}) \right] \frac{\Delta t}{2} + p_{x,i} \quad (4.20)$$

$$p_{y,i+1} = \left[ -H_{y,i} + (p_{y,i+1} - x_{i+1/2}) \right] \frac{\Delta t}{2} + p_{y,i} \quad (4.21)$$

$$x_{i_1} = (p_{x,i+1} + y_{i+1/2}) \frac{h}{2} \quad (4.22)$$

$$y_{i_1} = (p_{y,i+1} - x_{i+1/2}) \frac{h}{2}, \quad (4.23)$$

where

$$-H_{x,i} = \left( p_{y,i+1} - \frac{(1-k)(k+x_{i+1})}{((k+x_{i+1})^2 + y_{i+1}^2)^{3/2}} + \frac{k(1-k-x_{i+1})}{((1-k-x_{i+1})^2 + y_{i+1}^2)^{3/2}} \right) \quad (4.24)$$

$$-H_{y,i} = \left( -p_{x,i+1} - \frac{(1-k)y_{i+1}}{((k+x_{i+1})^2 + y_{i+1}^2)^{3/2}} - \frac{ky_{i+1}}{((1-k-x_{i+1})^2 + y_{i+1}^2)^{3/2}} \right) \quad (4.25)$$

## 4.6 Adaptive Symplectic Störmer-Verlet

Let  $z$  denote a vector of the position variables of the variables  $(x, y)$ .

$$\text{Euler step result:} \quad z_1 = z + O(h) \quad (4.26)$$

$$\text{Verlet step result:} \quad z_2 = z + O(h^2), \quad (4.27)$$

$$(4.28)$$

where  $O(h)$  denotes an error term of order  $h$ . Then we take the difference

$$\|z_1 - z_2\| = O(h) - O(h^2) \quad (4.29)$$

$$\approx O(h), \quad (4.30)$$

since  $O(h) \gg O(h^2)$ . Thus we approximate the error difference between the Euler and Verlet method as the actual error we make at stepsize  $h$ . The idea is to make both an Euler and a Verlet step for every time-step to assess the error and adjust the stepsize accordingly as implemented in listing 4.2

```

1 if err < tol or h <= hmin:
2
3     # Accept step
4     x = x2
5     y = y2
6     px = px2
7     py = py2
8
9     # Forward time by step
10    t = t+h
11    h = max(hmin, h*max(0.1, 0.8*sqrt(tol/err)))
12
13 else:
14     # No accept, reduce h to half
15     h = max(hmin, 0.5*h)

```

**Listing 4.2:** Adaptive method implemented in python. We accept the Verlet step only if the error is no more than a given tolerance  $\text{tol}$ . Subsequently we estimate a new  $h$  that will yield an error of  $\text{tol}$  on the next step and use 0.8 of this value to avoid frequent rejects. If the step is rejected we reduce the step size by half.

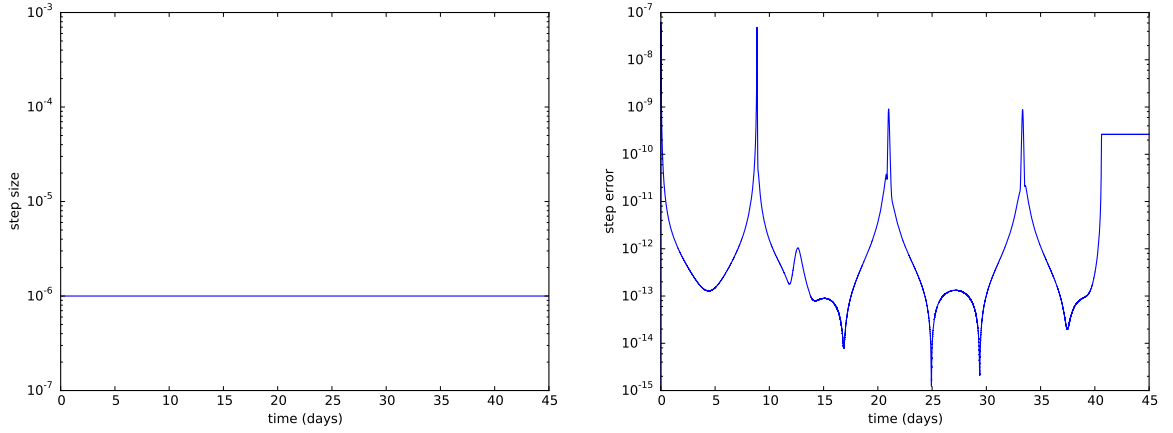
For every single step the step size is changed either up or down, depending on the errors and tolerance. As a result we always stay near the same error in every step, only taking as small steps as necessary in each iteration.

The non-adaptive algorithm is fixed in step size but varies in error per step. The adaptive algorithm varies in step size in an attempt to fix the error per step.

For all simulations we have set  $10^{-9}$  as the maximum tolerated error per step in the adaptive algorithm. For the non-adaptive algorithm we select a fixed stepsize,  $10^{-6}$ , to ensure that it's reasonably low most of the time. For comparison a simulation was run with same initial conditions using the adaptive and non-adaptive algorithm, see figures 4.3 - 4.4. Position plot 4.5 shown for completeness.

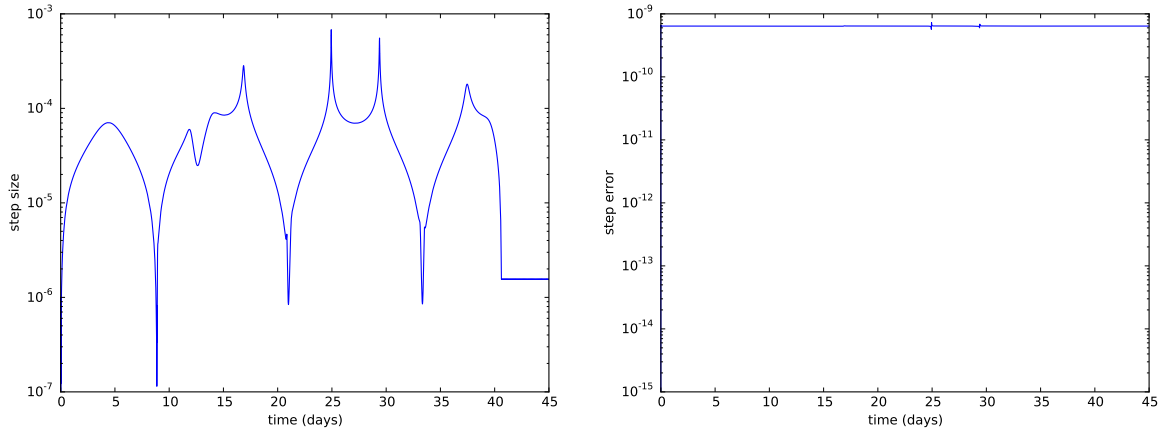
Note on the stepsize figure 4.4(a) that the adaptive ends up with constant stepsize at the end at 40 days once captured by the Moon. Thus when we are in circular orbit around a single body, we don't gain much by using adaptive methods. However as the spacecraft are in free space far away from both Earth/Moon, we gain a lot by taking longer steps (raising the stepsize). However as we make close passes to Earth or Moon, then the step size dips sharply to maintain constant error. We see this around day 9, 21 and 33 in 4.4(a). For most simulations, the adaptive method ran 10-100 times faster.

The code for the full Python implementation is in appendix E.



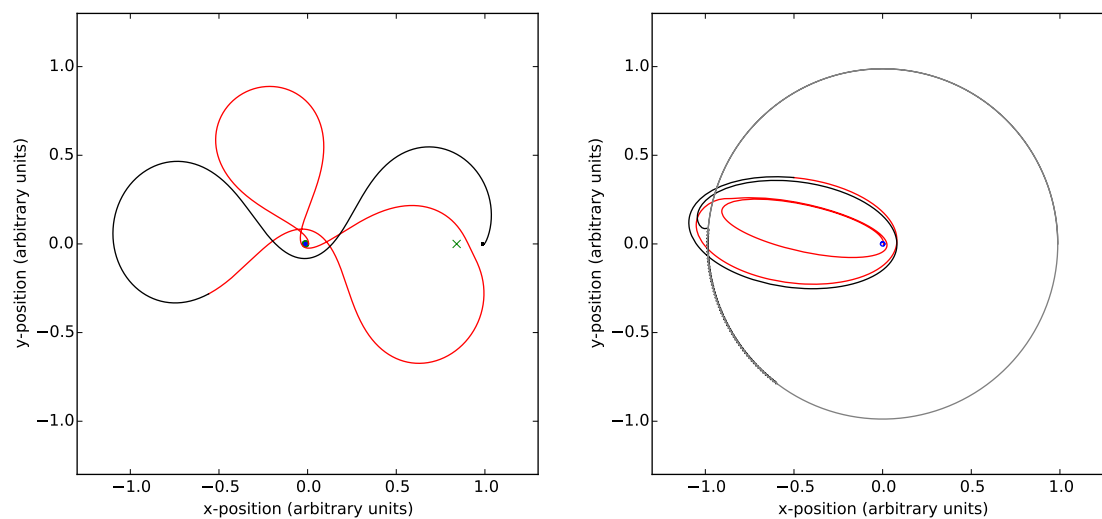
(a) Non-adaptive symplectic Verlet step size. (b) Non-adaptive symplectic Verlet error per step.

**Figure 4.3:** Non-adaptive method step size and error per step. Step size is fixed and as a result the error varies. All the work spend on calculating the many steps with lower error than  $10^{-9}$  in 4.3(b) are wasted since the error spikes to somewhere  $10^{-8}$  and  $10^{-7}$  during a close fly-by, which might invalidate all decimals after the 7th decimal on the non-adaptive method. The solutions are practically identical, but the adaptive is much faster.



(a) Adaptive symplectic Verlet step size. (b) Adaptive symplectic Verlet error per step.

**Figure 4.4:** The adaptive method constantly varies the step-size to ensure a constant error around  $10^{-9}$ . As a result it takes longer steps when there is less change in the variabels, in the space far from the bodies, and shorter steps when passing close by the Earth or Moon.



(a) Non-adaptive symplectic Verlet  $(x, y)$  trajectory. (b) Non-adaptive symplectic Verlet  $(X, Y)$  trajectory.

**Figure 4.5:** The trajectories of same initial conditions for the adaptive and non-adaptive method was visually indistinguishable. Included for context to figures 4.3 - 4.4.

# CHAPTER 5

## Simulations of Transfer Orbits

---

*“If you don’t put your breaks on when you reach the moon with a spaceship with people in it, you just keep going. And you’re dead. So the Hohmann transfer is risky, it’s a little bit dangerous and it uses a lot of fuel because you have to slow down (...). I think Apollo used a couple hundred pounds of fuel to slow down. It’s a million dollars a pound to bring anything to the Moon, including fuel. So just to slow down was a quarter of a billion dollars. I’m thinking, can you go into orbit without using your engines? Then you save all that money.”*

---

— Edward Belbruno, *Painting the Way to the Moon* (2015)

### 5.1 Setup and Circular Orbits

Our search algorithm is basically a brute force search. All orbits are found by search method that start from circular Earth Orbit 160 km altitude to circular Moon Orbit 100 km altitude.

A search algorithm in our program then initiates many trajectory calculations with many slightly different initial conditions in the following manner:

1. Many different initial positions in Earth parking orbit denoted by  $\theta$ , the angle from the positive  $x$  axis, see figure 3.1 for reference.
2. Many  $\Delta v_{\text{earth}}$  around some value.
3. Many small variations of  $\Delta\phi$ , the angle between impulse  $\Delta\mathbf{v}$  and start velocity vector  $\mathbf{v}_0$ .

Each of the trajectories are iteratively calculated using the adaptive Verlet algorithm described in chapter 4 with the additional conditions:

1. If we hit the earth, the trajectory is discarded.
2. If we hit the moon at altitude  $(100 \pm 10)$  km, the necessary  $\Delta v_{\text{moon}}$  to enter circular orbit is calculated and applied.

Throughout a simulation run, only the trajectory with the lowest  $\Delta v_{\text{moon}}$  is saved. This effectively filters out all the trajectories hitting the Moon head-on or high velocities; the best orbit will enter the Moon's orbit on an angle approximately tangent to a 100 km circular orbit.

## 5.2 Hohmann Transfer Orbits

If the moon was stationary in an inertial system, we would simply shoot from the opposite side of the Earth, at  $\theta = \pi$ . But because we must shoot not for where the moon is, but for where the moon will be. We also know that the flight time is approximately 5 days. We search for Hohmann orbits with simulation times of 6 days as follows:

1. Angular position from Earth varied in 100 positions uniformly spaced around  $\theta = -3\pi/4$  in range  $\pm \pi/4$
2. Earth burn  $\Delta v_{\text{earth}}$  varied in 200 velocities uniformly around 3.11 km/s (found by trial-and-error, starting from 2.13) in range  $\pm 0.1 * \text{unit\_velocity} = 0.102 \text{ km/s}$
3. One refinement-run on the best trajectory as initial guess, where all the ranges are set to 1/10 their previous value.

Listing 5.1 show the lowest energy Hohmann trans-lunar injection found among  $100 \times 200 = 20,000$  trajectories.

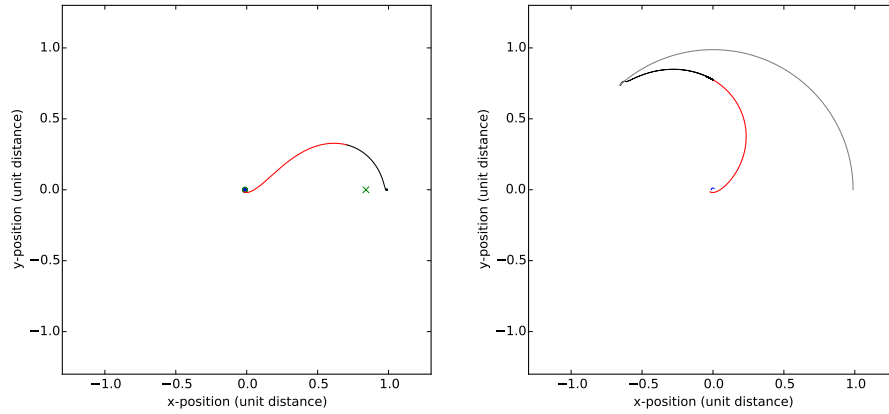
```

1 # -----
2 duration = 5/unit_time
3 pos      = -2.086814820119193
4 ang      = -0.000122173047640
5 burn     = 3.111181716545691/unit_vel
6 x0       = -0.020532317163607
7 y0       = -0.014769797663479
8 px0      = 9.302400979050308
9 py0      = -5.289712560652044
10 # -----
11 # dV(earth-escape) = 3.111182 km/s
12 # dV(moon-capture) = 0.800682 km/s
13 # dV(total)        = 3.911863 km/s
14 # Flight-time      = 4.300078 days
15 # -----
16 # Runtime = 0.51s
17 # Final position: 0.992505 -0.001507
18 # Final impulse: 0.503920 2.476928
19 # Final H: -2.730124
20 # Total runtime = 0.67s
21 # -----

```

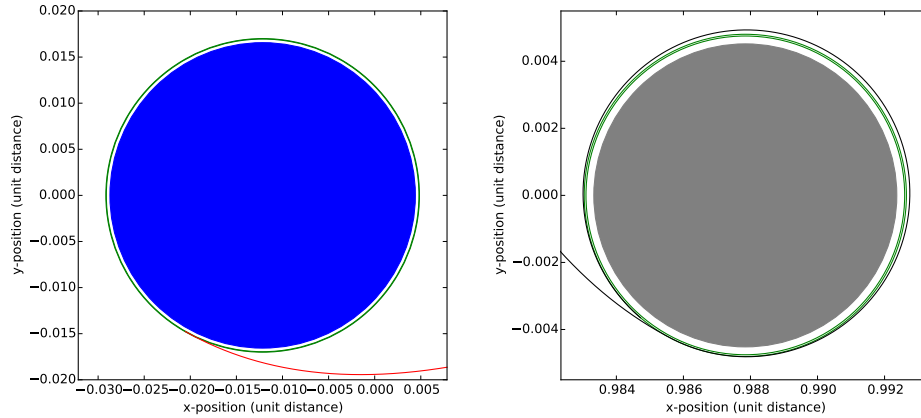
**Listing 5.1:** Best Hohmann orbit. `pos` = angular difference with start angle (here  $\theta = -3\pi/4$ ), `ang` = angle to velocity vector in Earth parking orbit, `burn` =  $\Delta v_{\text{earth}}$ , `(x0,y0,px0,py0)` are the initial conditions..

See figure 5.1 - 5.4 for trajectory position-, entry/exit- and Hamiltonian plots.



(a) Short LETO in co-rotating  $(x, y)$ , (b) Hohmann in  $(X, Y)$ , inertial system Earth and Moon stationary. stationary on CM.

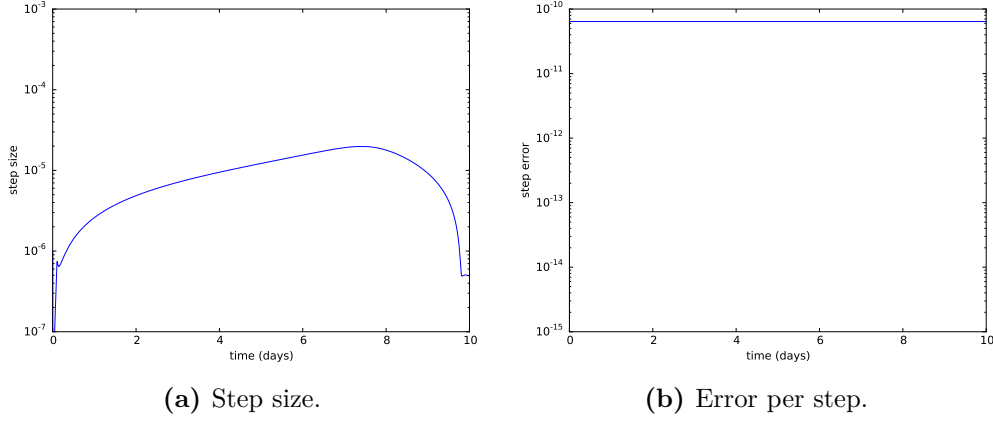
**Figure 5.1:** Position plots of simulated Hohmann transfer-orbit. Earth in origin, moon orbit in grey, first half of trajectory is red, last half is black, green cross is first Lagrange point  $L_1$ .



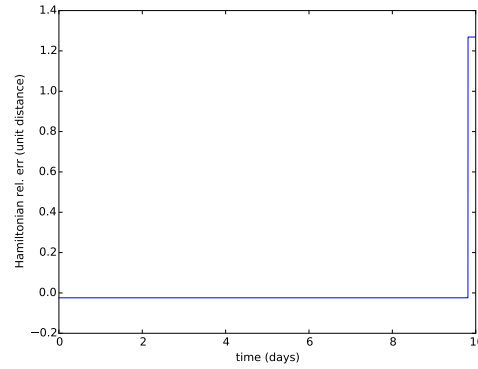
(a) Exit from circular Earth parking orbit, 100 km altitude (b) Entry to circular Moon orbit, 100 km altitude

**Figure 5.2:** Exit- and entry orbits of simulated Hohmann. The green band around the moon is the altitude range of  $\pm 10$  km around 100 km that triggers a capture, meaning a  $\Delta v_{\text{moon}}$  is calculated and applied if a discrete time step lands inside this band.

Note from listing 5.1 that  $\Delta v_{\text{total}} = 3.911$  km is in within reasonable agreement with the prediction from the simple Hohmann model in chapter 2 of  $\Delta v_{\text{total}} = 3.946$  km. Now we just need to see if we can find a low-energy transfer-orbit significantly cheaper than this.



**Figure 5.3:** Step size and error per step of simulated Hohmann. As expected step size is longer in weaker gravitational field and vice versa, and constant in circular orbit. Error is maintained at  $10^{-9}$ , as ensured by the adaptive method.



**Figure 5.4:** Hamiltonian  $H$  is conserved along the trajectory until  $\Delta v_{\text{moon}}$  is applied, as expected.

## 5.3 Low-Energy Transfer Orbits (LETO)

We searched for LETOs running simulations up to 200 days as follows:

1. Angular position from Earth varied in 55 positions uniformly spaced around  $\theta = -3\pi/4$  in range  $\pm \pi/4$
2. Earth burn  $\Delta v_{\text{earth}}$  varied in 55 velocities uniformly around 3.11 km/s (found by trial-and-error) in range  $\pm 0.1 * \text{unit\_velocity} = 0.102 \text{ km/s}$
3. Angle  $\phi$  to velocity vector in circular orbit varied in 55 angles uniformly spaced around  $\phi = 0$  in range  $\pm \pi/100$



4. 7 refinement-runs around the iteratively best trajectory, where all the ranges are set to 1/10 their previous value.

We will look at two LETOs:

**Short LETO** Flight time of 40.6 days and  $\Delta v_{\text{total}} = 3.896 \text{ km/s}$ , found in simulation run with maximum duration of 41 days.

**Long LETO** Flight time of 194.3 days and  $\Delta v_{\text{total}} = 3.795 \text{ km/s}$ , found in a simulation run of 195 days.

Both was found with the same search parameters, except simulation duration among  $55 \times 55 \times 55 \times 8 = 1,331,000$  trajectories as described above.

### 5.3.1 Short LETO

Listing 5.2 show the lowest  $\Delta v_{\text{total}}$  LETO trans-lunar injection

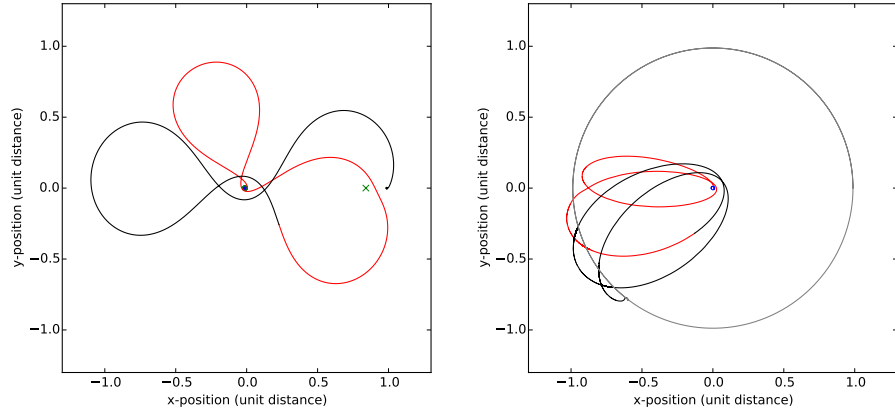
```

1 # -----
2 duration = 41/unit_time
3 pos      = -0.138042744751570
4 ang      = -0.144259374836607
5 burn     = 3.127288444444444/unit_vel
6 x0       = 0.004665728429046
7 y0       = -0.002336647636098
8 px0      = 1.904735175752430
9 py0      = 10.504985512873279
10 # -----
11 # dV(earth-escape) = 3.127288 km/s
12 # dV(moon-capture) = 0.768534 km/s
13 # dV(total)        = 3.895822 km/s
14 # Flight-time      = 40.617871 days
15 # -----
16 # Runtime = 0.54s
17 # Final position: 0.990701 0.003888
18 # Final impulse: 1.277242 0.047338
19 # Final H: -2.730117
20 # Total runtime = 0.71s
21 # -----

```

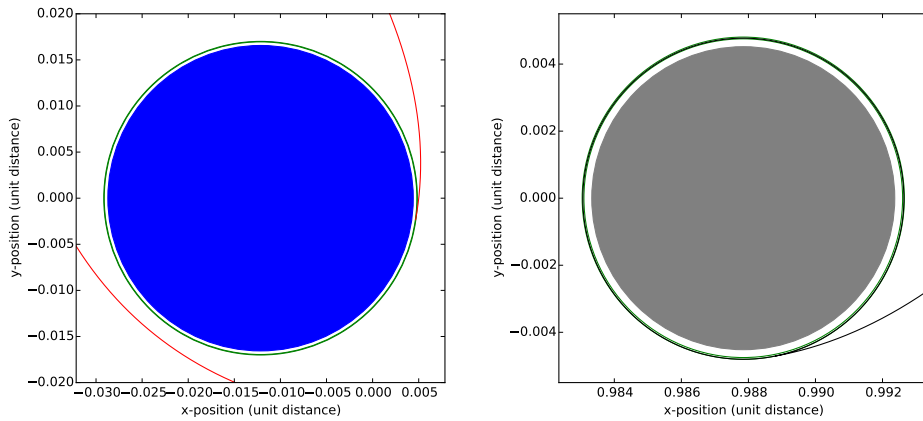
**Listing 5.2:** Short LETO. `pos` = angular difference with start angle (here  $\theta = -3\pi/4$ ), `ang` = angle to velocity vector in Earth parking orbit, `burn` =  $\Delta v_{\text{earth}}$ , `(x0,y0,px0,py0)` are the initial conditions..

See figure 5.5 - 5.8 for trajectory position-, entry/exit- and Hamiltonian plots.



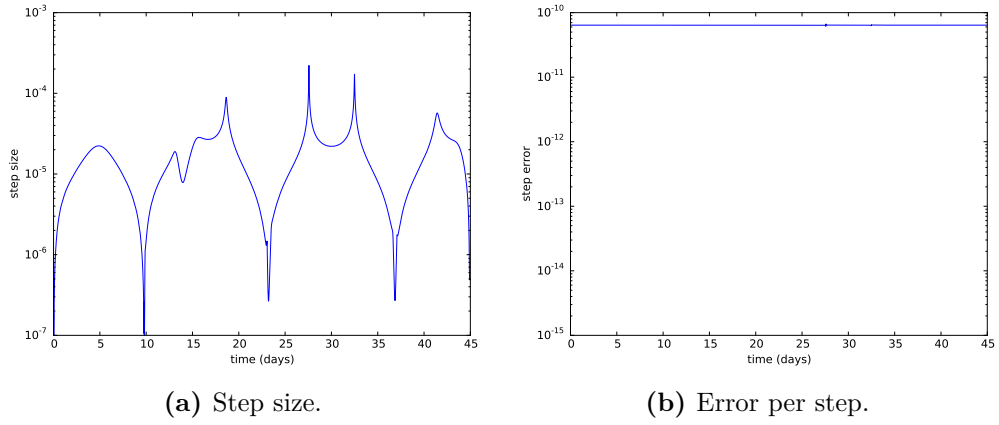
(a) Short LETO in co-rotating  $(x, y)$ , Earth and Moon stationary. (b) Short LETO in  $(X, Y)$ , inertial system stationary on CM.

**Figure 5.5:** Position plots of short LETO. Earth in origin, moon orbit in grey, first half of trajectory is red, last half is black, green cross is first Lagrangepoint  $L_1$ .

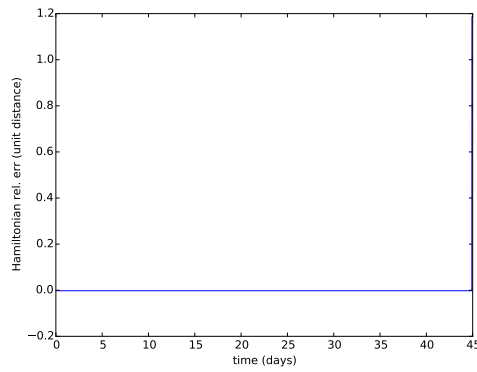


(a) Exit from circular Earth parking orbit, 100 km altitude (b) Entry to circular Moon orbit, 100 km altitude

**Figure 5.6:** Exit- and entry orbits of short LETO. The green band around the moon is the altitude range of  $\pm 10$  km around 100 km that triggers a capture.



**Figure 5.7:** Step size and error per step of simulated short LETO.



**Figure 5.8:** Hamiltonian  $H$  is conserved along the trajectory until  $\Delta v_{\text{moon}}$  is applied, as expected.

### 5.3.2 Long LETO

Listing 5.3 show the lowest  $\Delta v_{\text{total}}$  LETO trans-lunar injection

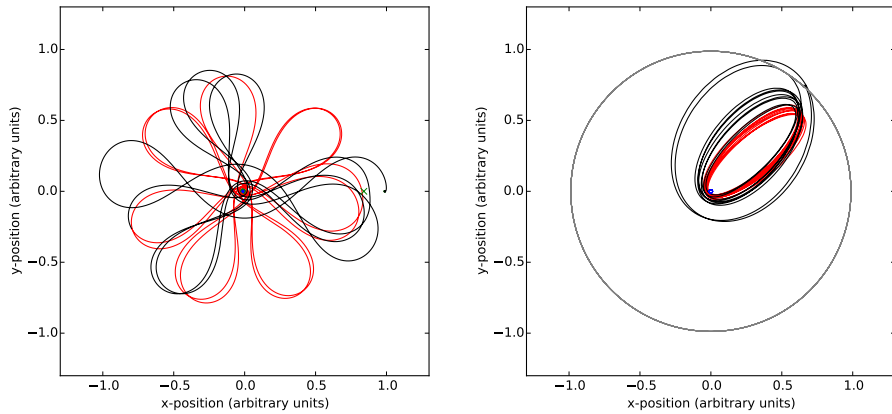
```

1 # -----
2 duration = 195/unit_time
3 pos      = 3.794182930145708
4 ang      = 0.023901745288554
5 burn     = 3.090702702702703/unit_vel
6 x0       = -0.025645129237870
7 y0       = -0.010311570301966
8 px0      = 6.539303578815582
9 py0      = -8.449205705334165
10 # -----
11 # dV(earth-escape) = 3.090703 km/s
12 # dV(moon-capture) = 0.704113 km/s
13 # dV(total)        = 3.794815 km/s
14 # Flight-time      = 194.275487 days
15 # -----
16 # Runtime = 0.98s
17 # Final position: 0.992404 0.001553
18 # Final impulse: 0.509302 -0.516927
19 # Final H: -2.730123
20 # Total runtime = 1.16s
21 # -----

```

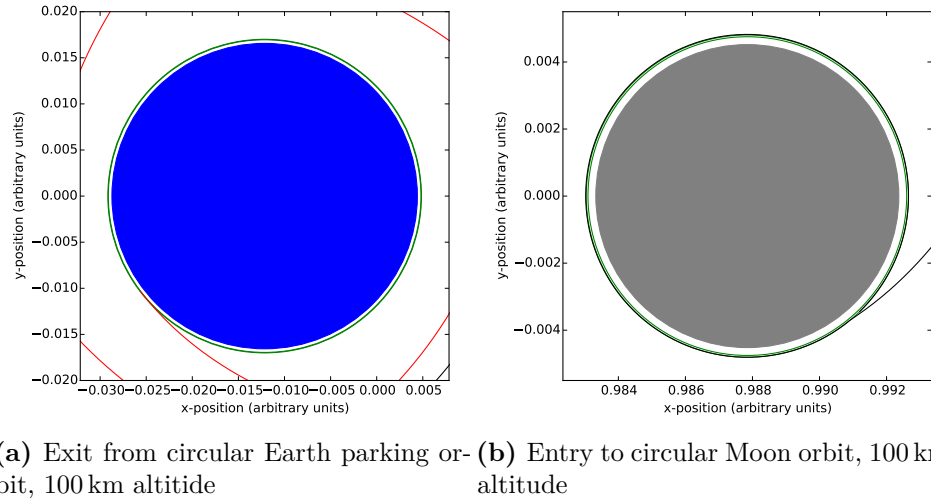
**Listing 5.3:** Long LETO. pos = angular difference with start angle (here  $\theta = -3\pi/4$ ), ang = angle to velocity vector in Earth parking orbit, burn =  $\Delta v_{\text{earth}}$ , (x0,y0,px0,py0) are the initial conditions..

See figure 5.9 - 5.12 for trajectory position-, entry/exit- and Hamiltonian plots.

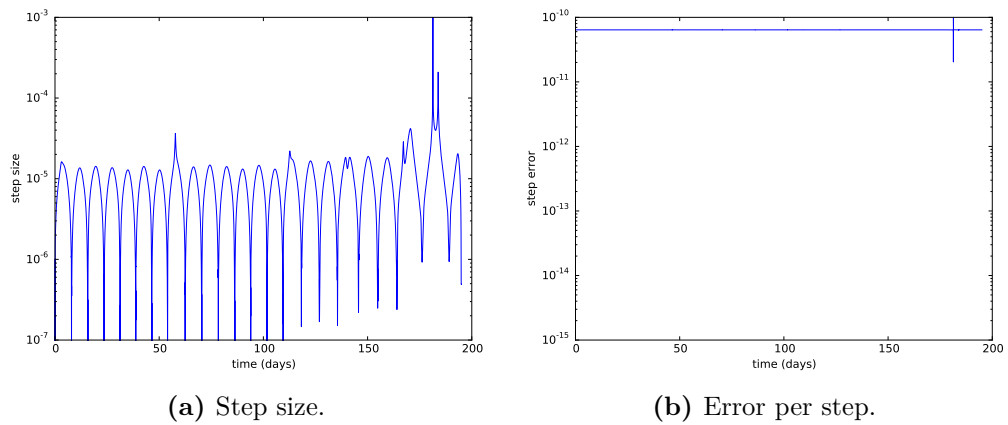


(a) Long LETO in co-rotating  $(x, y)$ , Earth and Moon stationary. (b) Long LETO in  $(X, Y)$ , inertial system stationary on CM.

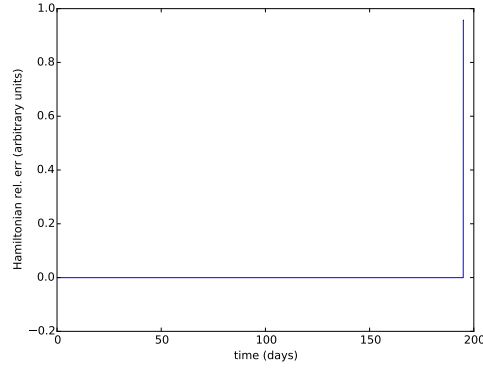
**Figure 5.9:** Position plots of long LETO. Earth in origin, moon orbit in grey, first half of trajectory is red, last half is black, green cross is first Lagrange point  $L_1$ .



**Figure 5.10:** Exit- and entry orbits of long LETO. The green band around the moon is the altitude range of  $\pm 10$  km around 100 km that triggers a capture.



**Figure 5.11:** Step size and error per step of simulated long LETO.



**Figure 5.12:** Hamiltonian  $H$  is conserved along the trajectory until  $\Delta v_{\text{moon}}$  is applied, as expected.

## 5.4 Results Summary and Discussion

Table 5.1 summarizes the  $\Delta v$  we found for the Hohmann, and the low-energy transfer orbits, short LETO and long LETO. In addition we've added a few more Hohmann (appendix D.3) to showcase the *Deltav* vs. flight time dynamics and to compare with the Apollo missions. In early simulations, the brute force approach was good enough to find a good Hohmann transfer, but not good enough for low-energy. After some code optimization, parallelization and running sufficiently many simulations on multi-core processors yielded good results even for low-energy transfers in the end.

- The brute force method worked surprisingly well; we were able to find low-energy transfer orbits of lower  $\Delta v$  than any of the two cited from literature.
- Results for the 3-day Hohmann are with 2.5% of the Apollo  $\Delta v$  and flight time, which adds to the credibility of our problem model and implementation.
- It would possibly be advantageous to go up to a symplectic 4.-5.th order Runge-Kutta method, but was not implemented due to time-constraints. However we do see that the error can be kept small around  $10^{-9}$  within a reasonable runtime, which can somewhat justify staying in 2nd order methods. In other words, we would not necessarily find better results, just a bit lower runtime since we could take fewer steps with a 4th order method.
- In further investigations it would be interesting to include:
  1. Patch together orbits in Lagrange points. This could be done by patching together brute forced trajectories from Earth to  $L_1$  (integrated forwards) and  $L_1$  to moon (integrated backwards).
  2. To from 2 to 3 dimensions.

3. Include the gravitational force from the Sun. This allows us to exploit the dynamics of the Sun-Earth-Moon system. For example deliberately approach the moon from the far side ( $L_2$ ) [Koo+01].
4. Exploit the dynamics of the Moon resonances as described in [TVB05].

Trajectory	Flight time	$\Delta v_{\text{total}}$ (km/s)	$\Delta v_{\text{earth}}$ (km/s)	$\Delta v_{\text{moon}}$ (km/s)
Minimum	N/A	3.721	3.099	0.622
Long LETO	194 days	3.795	3.091	0.704
Belbruno-Miller	3 months	3.838	3.187	0.651
Topputo	8 months	3.895	3.265	0.630
Short LETO	41 days	3.896	3.127	0.769
Hohmann - long (sim)	4.3 days	3.912	3.111	0.801
Hohmann - (model)	5.0 days	3.946	3.144	0.802
Hohmann - medium (sim)	3.00 days	4.015	3.136	0.880
Apollo (Hohman)	3.05 days	4.115	3.048	1.067
Hohmann - short (sim)	1.00 days	6.823	3.809	3.014

**Table 5.1:** Summerized flight time and  $\Delta v$  for various trajectories, sorted by  $\Delta v_{\text{total}}$ . Belbruno and Topputo from [Juu08] are for orbits going from 167 km Earth parking to 100 km circular moon orbit. Apollo mission data form [NAS66] [NASa]..





# CHAPTER 6

## Conclusion

---

To make it clear, low-energy transfer orbits are not fit for manned missions. Those types of missions require optimization for flight time due to resource usage of maintaining the lives of the astronauts.

We were able to find a low-energy transfer orbit to the moon with a flight time of 194 days and a  $\Delta v$  as low as 3795 km/s, which is better than any of the values we compared with from the literature but still higher than the theoretical minimum of  $\Delta v = 3721$  km/s, as it should be. We ensured that these results were numerically solid by implementing an adaptive Störmer–Verlet method that maintained a step-error of  $10^{-9}$ . Furthermore the code was implemented in Python optimized for parallelization, enabling feasible brute force search for several kinds of orbits. We also found another with a more moderate flight time of 41 days and  $\Delta v$  of respectable 3896 km/s, which is very close to value found by Tuppoto [TVB05], but that had a flight time of 8 months. Furthermore a Hohmann transfer to the Moon was modelled and gave a prediction of  $\Delta v = 9.946$  km/s with a flight time of 5.0 days. Our simulation found a Hohmann orbit of  $\Delta v = 9.912$  km/s and flight time 4.3 days. We also found the 3-day Hohmann to be within 2.5% agreement with the real world 3-day trip to the moon realized by the Apollo missions, which validates our model and code implementation.



## APPENDIX **A**

### More details: Hohmann Transfer-Orbit to the Moon

---

```
In[12]:= Quiet@Remove["`*"]
```

```
In[13]:= SetOptions[SelectedNotebook[],  
  PrintingStyleEnvironment -> "Printout", ShowSyntaxStyles -> True]
```

---

## Setup

```
In[14]:= (*Pariapsis: Point of least distance in elliptical orbit  
  Apoapsis: Point of greatest distance in elliptical orbit*)
```

```
In[15]:=  $\mu_E = \text{UnitConvert}[G * \text{PlanetData}[\text{Earth (planet)}, "Mass"], "km^3/s^2"]$ 
```

```
Out[15]=  $3.986 \times 10^5 \text{ km}^3/\text{s}^2$ 
```

```
In[16]:=  $\mu_M = \text{UnitConvert}[G * \text{PlanetaryMoonData}[\text{Moon (planetary moon)}, "Mass"], "km^3/s^2"]$ 
```

```
Out[16]=  $4.90 \times 10^3 \text{ km}^3/\text{s}^2$ 
```

```
In[17]:= (*Distance of Satellite's circular orbit around Earth in Earth system*)
```

```
  r1 = PlanetData[Earth (planet), "Radius"] + 160 km
```

```
Out[17]= 6527.4447 km
```

```
In[18]:= (*Distance of Satellite's elliptical  
  orbit around Earth at apoapsis in Earth system*)
```

```
  r2 = PlanetaryMoonData[Moon (planetary moon), "AverageDistanceFromEarth"]
```

```
Out[18]=  $3.850 \times 10^5 \text{ km}$ 
```

```
In[19]:= (*Distance of Satellite's circular orbit around Moon in Moon system*)
```

```
In[20]:= r3 = PlanetaryMoonData[Moon (planetary moon), "Radius"] + 100 km
```

```
Out[20]= 1837.5 km
```

---

## Hohmann transfer orbit to the Moon

### Impulse A

```
In[21]:= (*Speed in circular orbit, distance r1, at point A in Earth system*)
```

$$v_{\text{Earth}} = \sqrt{\frac{\mu_E}{r_1}}$$

```
Out[21]= 7.814 km/s
```

```
In[22]:= (*Speed in elliptical orbit, distance r1, at point A in Earth system*)
```

$$v_p = \sqrt{\frac{\mu_E}{r_1} \frac{2 r_2}{r_1 + r_2}}$$

```
Out[22]= 10.96 km/s
```

In[23]:= (\*Necessary  $\Delta v$  in point A, a speed-up of 3.14 km/s\*)

$$\Delta v_{\text{Earth}} = \sqrt{\frac{\mu_E}{r_1}} \left( \sqrt{\frac{2 r_2}{r_1 + r_2}} - 1 \right) // \text{N}(*v_p - v_{\text{Earth}}*)$$

Out[23]= 3.14423 km/s

## Impulse B - neglecting circular motion around moon

In[24]:= (\*Speed in elliptical orbit, distance  $r_2$ , at point B in Earth system\*)

$$v_a = \sqrt{\frac{\mu_E}{r_2} \frac{2 r_1}{r_1 + r_2}}$$

Out[24]= 0.1858 km/s

In[25]:= (\*Speed in circular orbit, distance  $r_2$ , at point B in Earth system  
(i.e. speed of the Moon in Earth system)\*)

$$v_{\text{NoMoon}} = \sqrt{\frac{\mu_E}{r_2}}$$

Out[25]= 1.017 km/s

In[26]:=

In[27]:= (\*Change of speed needed to "follow along with the moon",  
a speed-up of 0.831 km/s\*)

$$\Delta v_{\text{NoMoon}} = \sqrt{\frac{\mu_E}{r_2}} \left( 1 - \sqrt{\frac{2 r_1}{r_1 + r_2}} \right) // \text{N}(*v_3 - v_{2B}*)$$

Out[27]= 0.831692 km/s

## Moon circular orbit correction

In[28]:= (\*However we can't follow the moon stationary along it's orbit  
since the Moon attracts the satellite and it would crash into the  
surface. We need to go into a circular orbit around the moon\*)

In[29]:= (\*Speed  $v_3$  in circular orbit,  
distance 100 km above the Moon surface, in Moon system\*)

$$v_{\text{Moon}} = \sqrt{\left( \frac{\mu_M}{r_3} \right)} // \text{N}$$

Out[29]= 1.63342 km/s

In[30]:= (\*We can choose to add or subtract this value to  $\Delta v_{\text{BNoMoon}}$  since  
we can go around in orbit both ways. Subtracting  $v_3$  from  $\Delta v_{\text{BNoMoon}}$   
gives numerically the smallest  $\Delta v$  so that's what we'll do\*)

In[31]:= (\*Change in speed needed to go from geocentric elliptical  
orbit to circular Moon orbit, a speed-down of 0.802 km/s\*)

$$\Delta v_{\text{Moon}} = \Delta v_{\text{NoMoon}} - v_{\text{Moon}}$$

Out[31]= -0.801723 km/s

## Total

```
In[32]:= (*Regarding fuel/energy cost,
we're interested in in how much we need to change the speed,
irregardless of directio. Total Δv for
the whole trip: impulse at A plus impulse at B*)
ΔvTotal = Abs[ΔvEarth] + Abs[ΔvMoon] // N
Out[32]= 3.94595 km/s
```

```
In[33]:= (*Result if we had neglected the circular motion
around the moon at the end. Close, but not identical*)
ΔvTotalNoMoon = Abs[ΔvMoon] + Abs[ΔvNoMoon]
Out[33]= 1.63342 km/s
```

## Flight time

```
In[34]:= tHohmann = UnitConvert[ $\frac{1}{2} \sqrt{\frac{4 \pi^2 * \left(\frac{r1+r2}{2}\right)^3}{\mu E}}$ , "Days"]
Out[34]= 4.99 days
```

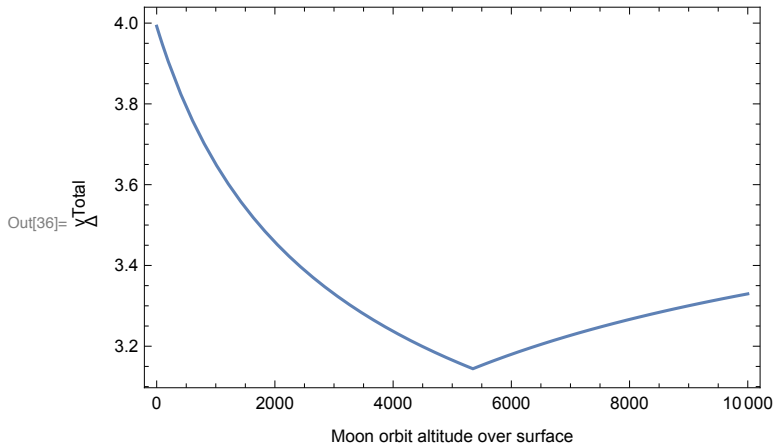
---

## Trying to minimize ΔvTotal by choosing appropriate moon orbit altitude

```
In[35]:= RMoon = PlanetaryMoonData[Moon (planetary moon), "Radius"]
Out[35]= 1737.5 km
```

## Plot

```
In[36]:= Plot[Abs[ $\sqrt{\frac{\mu E}{r1}} \left( \sqrt{\frac{2 r2}{r1 + r2}} - 1 \right)$ ] + Abs[ $\sqrt{\frac{\mu E}{r2}} \left( 1 - \sqrt{\frac{2 r1}{r1 + r2}} \right)$ ] -  $\sqrt{\left( \frac{\mu M}{R_{\text{Moon}} + \text{altitude}} \right)}$ ] /.  
altitude -> Quantity[x, "Kilometers"]], {x, 0, 10^4}, Frame -> True,  
FrameLabel -> {"Moon orbit altitude over surface", " $\Delta v_{\text{Total}}$ "}
```



## Moon orbit altitude that gives minimum $\Delta v_{\text{Total}}$

```
In[37]:= Solve[ $\sqrt{\frac{\mu E}{r2}} \left( 1 - \sqrt{\frac{2 r1}{r1 + r2}} \right)$  -  $\sqrt{\left( \frac{\mu M}{R_{\text{Moon}} + \text{altitude}} \right)}$  == 0, altitude] // N
```

```
Out[37]= {{altitude -> 5350.04 km}}
```

## Minimum $\Delta v_{\text{Total}}$

```
In[38]:= ΔvTotalMin = Abs[ $\sqrt{\frac{\mu E}{r1}} \left( \sqrt{\frac{2 r2}{r1 + r2}} - 1 \right)$ ] + Abs[ $\sqrt{\frac{\mu E}{r2}} \left( 1 - \sqrt{\frac{2 r1}{r1 + r2}} \right)$ ] -  $\sqrt{\left( \frac{\mu M}{R_{\text{Moon}} + x} \right)}$  /.  
x -> 5350.0393315174060602869`2.9098829878759 km] // N
```



```
Out[38]= 3.14423 km/s
```

```
In[39]:= (*Yes but not practical*)
```

## Moon hill sphere

```
In[40]:= rHillMoon = N@ (PlanetaryMoonData[Moon (planetary moon), "Mass"] /  
(3 * PlanetData[Earth (planet), "Mass"])))^(1/3)
```

```
Out[40]= 0.160053
```

```
In[41]:= rHillMoon *  PlanetaryMoonData[  Moon (planetary moon) , "AverageOrbitDistance" ]
```

```
Out[41]= 61 618.7 km
```



# APPENDIX B

## Detailed Derivation: Coordinate Velocity Transform

---

Recall  $(\mathcal{X}, \mathcal{Y})$  is the inertial frame coordinates and  $(x, y)$  is the rotating frame coordinates  
3.1. The goal is to go from  $\mathcal{X}(x, y)$  and  $\mathcal{Y}(x, y)$  to  $(\dot{\mathcal{X}}^2 + \dot{\mathcal{Y}}^2)(x, y)$ . This is used in (3.17) to obtain the kinetic energy  $T(x, y)$  From equation (3.15) we had

$$\begin{cases} \mathcal{X} = x \cos \omega t - y \sin \omega t \\ \mathcal{Y} = x \sin \omega t + y \cos \omega t \end{cases}$$

Taking the time derivative we get

$$\begin{cases} \dot{\mathcal{X}} = \dot{x} \cos \omega t - \dot{y} \sin \omega t - \omega x \sin \omega t - \omega y \cos \omega t \\ \dot{\mathcal{Y}} = \dot{x} \sin \omega t + \dot{y} \cos \omega t + \omega x \cos \omega t - \omega y \sin \omega t \end{cases} \quad (\text{B.1})$$

We'll take the squares, but first, to get a good overview of the terms, we note that  $\dot{\mathcal{X}}^2$  matches

$$\begin{aligned} (a - b - c - d)^2 \\ = a^2 + b^2 + c^2 + d^2 \\ - 2ab - 2ac - 2ad \\ + 2bc + 2bd + 2cd \end{aligned} \quad (\text{B.2})$$

Now we take the squares, align the terms as in (B.2) and note what terms add and cancel in the sum  $\dot{\mathcal{X}}^2 + \dot{\mathcal{Y}}^2$ :

$$\begin{aligned} \dot{\mathcal{X}}^2 &= (\dot{x} \cos \omega t - \dot{y} \sin \omega t - \omega x \sin \omega t - \omega y \cos \omega t)(\dot{x} \cos \omega t - \dot{y} \sin \omega t - \omega x \sin \omega t - \omega y \cos \omega t) \\ &= \dot{x}^2 \cos^2 t + \dot{y}^2 \sin^2 t + \omega^2 x^2 \sin^2 t + \omega^2 y^2 \cos^2 t \\ &\quad - 2\dot{x}\dot{y} \cos \omega t \sin \omega t - 2\omega x \dot{x} \cos \omega t \sin \omega t - 2\omega \dot{x} y \cos^2 t \\ &\quad + 2\omega x \dot{y} \sin^2 t + 2\omega y \dot{y} \cos \omega t \sin \omega t + 2\omega^2 x y \cos \omega t \sin \omega t \end{aligned} \quad (\text{B.3})$$

Likewise  $\dot{y}^2$  matches

$$\begin{aligned}
 & (a + b + c - d)^2 \\
 &= a^2 + b^2 + c^2 + d^2 \\
 &+ 2ab + 2ac - 2ad \\
 &+ 2bc - 2bd - 2cd
 \end{aligned} \tag{B.4}$$

so we have

$$\begin{aligned}
 \dot{y}^2 &= (\dot{x} \sin \omega t + \dot{y} \cos \omega t + \omega x \cos \omega t - \omega y \sin \omega t)(\dot{x} \sin \omega t + \dot{y} \cos \omega t + \omega x \cos \omega t - \omega y \sin \omega t) \\
 &= \dot{x}^2 \sin^2 t + \dot{y}^2 \cos^2 t + \omega^2 x^2 \cos^2 t + \omega^2 y^2 \sin^2 t \\
 &+ 2\dot{x}\dot{y} \cos \omega t \sin \omega t + 2\omega x \dot{x} \cos \omega t \sin \omega t - 2\omega x y \sin^2 t \\
 &+ 2\omega x y \cos^2 t - 2\omega y \dot{y} \cos \omega t \sin \omega t - 2\omega^2 x y \cos \omega t \sin \omega t
 \end{aligned} \tag{B.5}$$

- Terms that add in  $\mathcal{X}^2 + \mathcal{Y}^2$
- Terms that cancel in  $\mathcal{X}^2 + \mathcal{Y}^2$

(B.6)

Cancelling terms and repeated use of  $\cos^2 t + \sin^2 t = 1$  finally gives us

$$\begin{aligned}
 \dot{\mathcal{X}}^2 + \dot{\mathcal{Y}}^2 &= \dot{x}^2 + \dot{y}^2 + \omega^2 x^2 + \omega^2 y^2 - 2\omega x y + 2\omega x y \\
 &= (\dot{x} - \omega y)^2 + (\dot{y} + \omega x)^2
 \end{aligned} \tag{B.7}$$

# APPENDIX C

## Non-Dimensionalization of Hamiltons Equations

---

We set:

$$t = k_t T \quad (\text{C.1})$$

$$x = k_x X \quad (\text{C.2})$$

$$y = k_y Y \quad (\text{C.3})$$

$$p_x = k_{px} P_x \quad (\text{C.4})$$

$$p_y = k_{py} P_Y, \quad (\text{C.5})$$

with

$$k_t = \frac{1}{\omega} = \sqrt{\frac{(R+r)^3}{G(M+m)}} \quad (\text{C.6})$$

$$k_x = k_y = R + r \quad (\text{C.7})$$

$$p_x = p_y = m_s \sqrt{\frac{G(M+m)}{(R+r)}}. \quad (\text{C.8})$$

So we get for  $\dot{X}$

$$\frac{k_x}{k_t} \frac{dX}{dT} = \frac{k_{px} P_x}{m_s} + \omega k_y Y \quad (\text{C.9})$$

$$\Leftrightarrow \frac{dX}{dT} = \frac{k_t k_{px}}{k_x} \frac{P_x}{m_s} + \omega k_t \frac{k_y}{k_x} Y \quad (\text{C.10})$$

$$\Leftrightarrow \dot{X} = P_x + Y. \quad (\text{C.11})$$

And for  $\dot{Y}$  by symmetry

$$\dot{Y} = P_Y - X, \quad (\text{C.12})$$

For  $\dot{P}_x$  we get

$$\frac{dP_x}{dT} = \frac{k_{py}}{k_{px}} k_t \omega P_y - \frac{k_t}{k_{px}} G m_s \left[ \frac{M(k_x X + R)}{(k_x X + R)^2 + k_y^2 Y^2}^{3/2} + \frac{m(k_x X - r)}{((k_x X - r)^2 + k_y^2 Y^2)}^{3/2} \right] \quad (C.13)$$

$$= P_y - \left( \sqrt{\frac{(R+r)^3}{G(M+m)}} \right) \left( \frac{1}{m_s} \sqrt{\frac{(R+r)}{G(M+m)}} \right) G m_s. \quad (C.14)$$

$$\left[ \frac{M[(R+r)X + R]}{[(R+r)X + R]^2 + (R+r)^2 Y^2}^{3/2} + \frac{m[(R+r)X - r]}{[(R+r)X - r]^2 + (R+r)^2 Y^2}^{3/2} \right] \\ = P_y - \frac{(R+r)^2}{M+m}. \quad (C.15)$$

$$\left[ \frac{M(R+r) \left( X + \frac{R}{R+r} \right)}{(R+r)^3 \left[ \left( X + \frac{R}{R+r} \right)^2 + Y^2 \right]^{3/2}} + \frac{m(R+r) \left( X - \frac{r}{R+r} \right)}{(R+r)^3 \left[ \left( X - \frac{r}{R+r} \right)^2 + Y^2 \right]^{3/2}} \right] \\ = P_y - \left[ \frac{\frac{M}{M+m} \left( X + \frac{R}{R+r} \right)}{\left( \left( X + \frac{R}{R+r} \right)^2 + Y^2 \right)^{3/2}} + \frac{\frac{m}{M+m} \left( X - \frac{r}{R+r} \right)}{\left( \left( X - \frac{r}{R+r} \right)^2 + Y^2 \right)^{3/2}} \right], \quad (C.16)$$

where  $\frac{M}{M+m}$  and  $\frac{R}{R+r}$  are dimensionless parameters. There is a simple relation between the two given by the equation for the center of mass:

$$M\mathbf{R} + m\mathbf{r} = \mathbf{0} \quad (C.17)$$

$$\Rightarrow \frac{m}{M+m} = \frac{R}{R+r} = k \quad (C.18)$$

$$\Leftrightarrow \frac{M}{M+m} = \frac{r}{R+r} = 1 - k \quad (C.19)$$

With that in mind we finally get

$$\dot{P}_x = P_y - \frac{(1-k)(X+k)}{((X+k)^2 + Y^2)^{3/2}} - \frac{k(X-1-k)}{((X-1-k)^2 + Y^2)^{3/2}} \quad (\text{C.20})$$

by symmetry for  $P_y$  we get

$$\dot{P}_y = -P_x - \frac{(1-k)Y}{((X+k)^2 + Y^2)^{3/2}} - \frac{kY}{((X-1-k)^2 + Y^2)^{3/2}} \quad (\text{C.21})$$



# APPENDIX D

## Miscellaneous

### D.1 Reduced 3-body Problem Symplectic Euler $x_{i+1}, y_{i+1}$ (Mathematica)

```
In[89]:= eqn1 = x1 == (px1 + y1) h + x0;
```

```
In[90]:= eqn2 = y1 == (py1 - x1) h + y0;
```

```
In[91]:= Solve[{eqn1, eqn2}, {x1, y1}] // FullSimplify
```

```
Out[91]= {{x1 -> (x0 + h (px1 + h py1 + y0)) / (1 + h^2), y1 -> (-h (h px1 - py1 + x0) + y0) / (1 + h^2)}}
```

### D.2 Implicit Euler algorithm

All new values  $(x, y, p_x, p_y)_{i+1}$  refer to the same unknown values in the same timestep  $(x, y, p_x, p_y)_{i+1}$ .

$$x_{i+1} = (p_{x,i+1} + y_{i+1})h + x_{i+1} \quad (\text{D.1})$$

$$y_{i+1} = (p_{y,i+1} - x_{i+1})h + y_{i+1} \quad (\text{D.2})$$

$$p_{x,i+1} = \left( p_{y,i+1} - \frac{(1-k)(k+x_{i+1})}{((k+x_{i+1})^2 + y_{i+1}^2)^{3/2}} + \frac{k(1-k-x_{i+1})}{((1-k-x_{i+1})^2 + y_{i+1}^2)^{3/2}} \right) h + p_{x,i} \quad (\text{D.3})$$

$$p_{y,i+1} = \left( -p_{x,i+1} - \frac{(1-k)y_{i+1}}{((k+x_{i+1})^2 + y_{i+1}^2)^{3/2}} - \frac{ky_{i+1}}{((1-k-x_{i+1})^2 + y_{i+1}^2)^{3/2}} \right) h + p_{y,i} \quad (\text{D.4})$$

### D.3 Medium and Short Hohmann orbits

```
1 # -----
2 duration = 3/unit_time
3 pos      = -2.272183066647597
4 ang      = -0.075821466029764
```

```
5 burn      = 3.135519748743719/unit_vel
6 x0        = -0.023110975767437
7 y0        = -0.012972499765730
8 px0       = 8.032228991913522
9 py0       = -7.100537706154897
10 # -----
11 # dV(earth-escape) = 3.135520 km/s
12 # dV(moon-capture) = 0.879826 km/s
13 # dV(total)        = 4.015346 km/s
14 # Flight-time      = 2.999939 days
15 # -----
```

**Listing D.1:** Medium duration Hohmann.

```
1 # -----
2 duration = 2/unit_time
3 pos      = -2.277784119105456
4 ang      = 0.046759232345463
5 burn     = 3.809267777777778/unit_vel
6 x0       = -0.023183463163465
7 y0       = -0.012910923775798
8 px0      = 8.760501647975921
9 py0      = -7.267405934327472
10 # -----
11 # dV(earth-escape) = 3.809268 km/s
12 # dV(moon-capture) = 3.014142 km/s
13 # dV(total)        = 6.823410 km/s
14 # Flight-time      = 1.000007 days
15 # -----
```

**Listing D.2:** Fast duration Hohmann.



# APPENDIX E

## Python Code for Restricted 3-Body Problem

---

### E.1 Main Script

Calling all the other functions, initiating searches, and storing interesting orbits that has been found (filename: testing.py).

```
1 import os
2 import time
3 from math import pi, cos, sin
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from const import *
7 import reduced3body as r3b
8
9 try:
10     threads = int(os.environ["OMP_NUM_THREADS"])
11 except KeyError:
12     threads = 1
13
14 runtime = time.time()
15
16 # Precalculated initial Conditions
17 #demo = 'earth_orbit'
18 #demo = 'lunar_orbit'
19 #demo = 'hohmann'
20 #demo = '3_day_hohmann'
21 #demo = '1_day_hohmann'
22 #demo = 'reverse_hohmann'
23 demo = 'low_energy_short'
24 #demo = 'low_energy_long'
25 #demo = 'earth_to_L1'
26
27 # or Search for trajectories
28 #demo = 'search_hohmann'
29 #demo = 'search_low_energy'
30 #demo = 'search_low_energy_parts8'
31 #demo = 'search_refine'
32
33 n = 1000000
34
```

```

35 # Set coordinates
36 if demo == 'earth_orbit':
37     duration = (2.0*pi*leo_orbit/leo_orbit_vel)/(unit_time*day)
38     r = leo_orbit/unit_len
39     v = 0.99732*leo_orbit_vel/unit_vel
40     theta = 0
41     x = r*cos(theta)
42     y = r*sin(theta)
43     vx = -v*y/r
44     vy = v*x/r
45     pos = 0
46     ang = 0
47     burn = 0
48     x0 = earth_pos_x+x
49     y0 = y
50     px0 = vx-y0
51     py0 = vy+x0
52 elif demo == 'lunar_orbit':
53     duration = (2.0*pi*lunar_orbit/lunar_orbit_vel)/(unit_time*day)
54     r = lunar_orbit/unit_len
55     v = 0.99732*lunar_orbit_vel/unit_vel
56     theta = 0
57     x = r*cos(theta)
58     y = r*sin(theta)
59     vx = -v*y/r
60     vy = v*x/r
61     pos = 0
62     ang = 0
63     burn = 0
64     x0 = moon_pos_x+x
65     y0 = y
66     px0 = vx-y0
67     py0 = vy+x0
68 elif demo == 'hohmann':
69     #demo = 'search_refine'
70 # -----
71     duration = 5/unit_time
72     pos = -2.086814820119193
73     ang = -0.000122173047640
74     burn = 3.111181716545691/unit_vel
75     x0 = -0.020532317163607
76     y0 = -0.014769797663479
77     px0 = 9.302400979050308
78     py0 = -5.289712560652044
79 # -----
80 # dV(earth-escape) = 3.111182 km/s
81 # dV(moon-capture) = 0.800682 km/s
82 # dV(total) = 3.911863 km/s
83 # Flight-time = 4.300078 days
84 # -----
85 elif demo == 'reverse_hohmann':
86 # -----

```

```

87     duration = 4/unit_time
88     pos      = -2.282942228154665
89     ang      = 0.000000000000000
90     burn     = -3.149483130653266/unit_vel
91     x0       = -0.023249912090507
92     y0       = -0.012853859046429
93     px0      = -8.098481905534163
94     py0      = 6.978997254692934
95 # -----
96 # dV(earth-escape) = 3.149483 km/s
97 # dV(moon-capture) = 0.968488 km/s
98 # dV(total)        = 4.117971 km/s
99 # Flight-time      = 3.875497 days
100 # -----
101 elif demo == 'low_energy_long':
102 # -----
103     duration = 195/unit_time
104     pos      = 3.794182930145708
105     ang      = 0.023901745288554
106     burn     = 3.090702702702703/unit_vel
107     x0       = -0.025645129237870
108     y0       = -0.010311570301966
109     px0      = 6.539303578815582
110     py0      = -8.449205705334165
111 # -----
112 # dV(earth-escape) = 3.090703 km/s
113 # dV(moon-capture) = 0.704114 km/s
114 # dV(total)        = 3.794816 km/s
115 # Flight-time      = 194.275480 days
116 # -----
117 # -----
118     #demo = 'search_refine'
119     # duration = 195/unit_time
120     # pos      = 3.794182930145708
121     # ang      = 0.023901745288554
122     # burn     = 3.090702702702703/unit_vel
123     # x0       = -0.025645129237870
124     # y0       = -0.010311570301966
125     # px0      = 6.539303578815583
126     # py0      = -8.449205705334164
127 # -----
128 # dV(earth-escape) = 3.090703 km/s
129 # dV(moon-capture) = 0.704114 km/s
130 # dV(total)        = 3.794817 km/s
131 # Flight-time      = 194.275480 days
132 # -----
133 elif demo == 'low_energy_short':
134     #demo = 'search_refine'
135 # -----
136     duration = 41/unit_time
137     pos      = -0.138042744751570
138     ang      = -0.144259374836607

```

```

139     burn      = 3.127288444444444/unit_vel
140     x0        = 0.004665728429046
141     y0        = -0.002336647636098
142     px0       = 1.904735175752430
143     py0       = 10.504985512873279
144 # -----
145 # dV(earth-escape) = 3.127288 km/s
146 # dV(moon-capture) = 0.768534 km/s
147 # dV(total)        = 3.895822 km/s
148 # Flight-time      = 40.617871 days
149 # -----
150 elif demo == '3_day_hohmann':
151     #demo = 'search_refine'
152 # -----
153     duration = 3/unit_time
154     pos      = -2.272183066647597
155     ang      = -0.075821466029764
156     burn     = 3.135519748743719/unit_vel
157     x0       = -0.023110975767437
158     y0       = -0.012972499765730
159     px0      = 8.032228991913522
160     py0      = -7.100537706154897
161 # -----
162 # dV(earth-escape) = 3.135520 km/s
163 # dV(moon-capture) = 0.879826 km/s
164 # dV(total)        = 4.015346 km/s
165 # Flight-time      = 2.999939 days
166 # -----
167 elif demo == '1_day_hohmann':
168     #demo = 'search_refine'
169     duration = 1/unit_time
170     pos      = -2.277654673852600
171     ang      = 0.047996554429844
172     burn     = 3.810000000000000/unit_vel
173     x0       = -0.023181791813268
174     y0       = -0.012912351430812
175     px0      = 8.764829132987316
176     py0      = -7.263069305305378
177 # -----
178 # dV(earth-escape) = 3.810000 km/s
179 # dV(moon-capture) = 3.319455 km/s
180 # dV(total)        = 7.129455 km/s
181 # Flight-time      = 0.997234 days
182 # -----
183 elif demo == 'earth_to_L1':
184     demo = 'search_refine'
185 # -----
186     duration = 191/unit_time
187     pos      = 2.843432239707429
188     ang      = 0.000000000000000
189     burn     = 3.091851851851852/unit_vel
190     x0       = -0.028385246222264

```

```

191     y0      = 0.004988337832881
192     px0     = -3.136296304910217
193     py0     = -10.217405925499762
194 # -----
195 # dV(earth-escape) = 3.091852 km/s
196 # dV(at L1)       = 0.676226 km/s
197 # dV(total)       = 3.768078 km/s
198 # Flight-time     = 190.001881 days
199 # -----
200
201 ##### FUNCTION CALLS #####
202
203 if demo == 'search_hohmann':
204     tlist,xlist,ylist,pxlist,pylist,errlist,hlist = r3b.hohmann(threads,n)
205 elif demo == 'search_low_energy':
206     tlist,xlist,ylist,pxlist,pylist,errlist,hlist = r3b.low_energy(threads,n
207 )
208 elif demo == 'search_low_energy_parts8':
209     tlist,xlist,ylist,pxlist,pylist,errlist,hlist = r3b.low_energy_parts8(
210         threads,n)
211 elif demo == 'search_refine':
212     tlist,xlist,ylist,pxlist,pylist,errlist,hlist = r3b.refine(threads,n,
213         duration,pos,ang,burn,x0,y0,px0,py0)
214 else:
215     tlist,xlist,ylist,pxlist,pylist,errlist,hlist = r3b.trajectory(n,
216         duration,pos,ang,burn,x0,y0,px0,py0)
217 Hlist = pxlist**2/2 + pylist**2/2 + ylist*pxlist - xlist*pylist - (1-mu)/np.
218     sqrt(np.power(mu+xlist,2)+np.power(ylist,2)) - mu/np.sqrt(np.power(1-mu-
219     xlist,2)+np.power(ylist,2))
220 print("# Final position: %f %f" %(xlist[n-1],ylist[n-1]))
221 print("# Final impulse: %f %f" % (pxlist[n-1],pylist[n-1]))
222 print("# Final H: %f" % (Hlist[n-1]))
223 runtime = time.time()-runtime
224 print("# Total runtime = %3.2fs" % (runtime))
225 print("#
226
227     ")
228 print("# --- Done with FUNCTION CALLS")
229 #exit()
230
231 ##### PLOTS: POSITION #####
232
233 xlist1 = xlist[:n/2]
234 ylist1 = ylist[:n/2]
235 xlist2 = xlist[n/2:]
236 ylist2 = ylist[n/2:]
237
238 Xlist1 = xlist[:n/2]*np.cos(tlist[:n/2]) - ylist[:n/2]*np.sin(tlist[:n/2])
239 Ylist1 = xlist[:n/2]*np.sin(tlist[:n/2]) + ylist[:n/2]*np.cos(tlist[:n/2])
240 Xlist2 = xlist[n/2:]*np.cos(tlist[n/2:]) - ylist[n/2:]*np.sin(tlist[n/2:])
241 Ylist2 = xlist[n/2:]*np.sin(tlist[n/2:]) + ylist[n/2:]*np.cos(tlist[n/2:])
242

```

```

235 Xlist_earth = earth_pos_x*np.cos(tlist)
236 Ylist_earth = -earth_pos_x*np.sin(tlist)
237
238 Xlist_moon = moon_pos_x*np.cos(tlist)
239 Ylist_moon = moon_pos_x*np.sin(tlist)
240
241 # Rel. err
242 plt.figure()
243 plt.plot(tlist*unit_time, errlist)
244 plt.xlabel("time (days)")
245 plt.ylabel("step error")
246 plt.yscale('log')
247
248 # Step sizes
249 plt.figure()
250 plt.plot(tlist*unit_time, hlist)
251 plt.xlabel("time (days)")
252 plt.ylabel("step size")
253 plt.yscale('log')
254
255 # Total energy
256 havg = np.sum(Hlist)/n
257 hrelerr = (Hlist-havg)/havg
258 plt.figure()
259 plt.plot(tlist*unit_time, hrelerr)
260 plt.xlabel("time (days)")
261 plt.ylabel("Hamiltonian rel. err (arbitrary units)")
262
263 # Zoom earth
264 xlim = 0.02
265 ylim = 0.02
266 xmin = earth_pos_x-xlim
267 xmax = earth_pos_x+xlim
268 ymin = -ylim
269 ymax = ylim
270 plt.figure()
271 earth=plt.Circle((earth_pos_x,0),earth_radius/unit_len,color='blue')
272 earthorbit1=plt.Circle((earth_pos_x,0),(leo_orbit-orbit_range)/unit_len,
273     color='g',fill=False)
273 earthorbit2=plt.Circle((earth_pos_x,0),(leo_orbit+orbit_range)/unit_len,
274     color='g',fill=False)
274 plt.gcf().gca().add_artist(earth)
275 plt.gcf().gca().add_artist(earthorbit1)
276 plt.gcf().gca().add_artist(earthorbit2)
277 plt.plot(xlist1,ylist1,'r-')
278 plt.plot(xlist2,ylist2,'k-')
279 plt.xlim(xmin,xmax)
280 plt.ylim(ymin,ymax)
281 plt.gca().set_aspect('equal', adjustable='box')
282 plt.xlabel("x-position (arbitrary units)")
283 plt.ylabel("y-position (arbitrary units)")
284

```

```

285 # Zoom moon
286 xlim = 0.0055
287 ylim = 0.0055
288 xmin = moon_pos_x-xlim
289 xmax = moon_pos_x+xlim
290 ymin = -ylim
291 ymax = ylim
292 plt.figure()
293 moon=plt.Circle((moon_pos_x,0),moon_radius/unit_len,color='grey')
294 moonorbit1=plt.Circle((moon_pos_x,0),(lunar_orbit-orbit_range)/unit_len,
    color='g',fill=False)
295 moonorbit2=plt.Circle((moon_pos_x,0),(lunar_orbit+orbit_range)/unit_len,
    color='g',fill=False)
296 plt.gcf().gca().add_artist(moon)
297 plt.gcf().gca().add_artist(moonorbit1)
298 plt.gcf().gca().add_artist(moonorbit2)
299 plt.plot(xlist1,ylist1,'r-')
300 plt.plot(xlist2,ylist2,'k-')
301 plt.xlim(xmin,xmax)
302 plt.ylim(ymin,ymax)
303 plt.gca().set_aspect('equal', adjustable='box')
304 plt.xlabel("x-position (arbitrary units)")
305 plt.ylabel("y-position (arbitrary units)")
306
307 # View center of mass
308 xlim = 1.3
309 ylim = 1.3
310 xmin = -xlim
311 xmax = xlim
312 ymin = -ylim
313 ymax = ylim
314
315 # Position plot (X,Y)
316 plt.figure()
317 plt.plot(Xlist1,Ylist1,'r')
318 plt.plot(Xlist2,Ylist2,'k')
319 plt.plot(Xlist_earth, Ylist_earth, 'blue')
320 plt.plot(Xlist_moon, Ylist_moon, 'grey')
321 plt.xlim(xmin,xmax)
322 plt.ylim(ymin,ymax)
323 plt.gca().set_aspect('equal', adjustable='box')
324 plt.xlabel("x-position (arbitrary units)")
325 plt.ylabel("y-position (arbitrary units)")
326
327 # Position plot (x,y)
328 plt.figure()
329 plt.plot(xlist1,ylist1,'r-')
330 plt.plot(xlist2,ylist2,'k-')
331 earth=plt.Circle((earth_pos_x,0),earth_radius/unit_len,color='blue')
332 earthorbit1=plt.Circle((earth_pos_x,0),(leo_orbit-orbit_range)/unit_len,
    color='g',fill=False)
333 earthorbit2=plt.Circle((earth_pos_x,0),(leo_orbit+orbit_range)/unit_len,

```

```

        color='g',fill=False)
334 moon=plt.Circle((moon_pos_x,0),moon_radius/unit_len,color='grey')
335 moonorbit1=plt.Circle((moon_pos_x,0),(lunar_orbit-orbit_range)/unit_len,
        color='g',fill=False)
336 moonorbit2=plt.Circle((moon_pos_x,0),(lunar_orbit+orbit_range)/unit_len,
        color='g',fill=False)
337 plt.gcf().gca().add_artist(earth)
338 plt.gcf().gca().add_artist(earthorbit1)
339 plt.gcf().gca().add_artist(earthorbit2)
340 plt.gcf().gca().add_artist(moon)
341 plt.gcf().gca().add_artist(moonorbit1)
342 plt.gcf().gca().add_artist(moonorbit2)
343 plt.plot(L1_pos_x,0,'gx')
344 plt.xlim(xmin,xmax)
345 plt.ylim(ymin,ymax)
346 plt.gca().set_aspect('equal', adjustable='box')
347 plt.xlabel("x-position (arbitrary units)")
348 plt.ylabel("y-position (arbitrary units)")
349 #plt.savefig('fig/r3b/r3b_y(x)_euler_symplectic.pdf',bbox_inches='tight')
350 plt.show()
351 plt.close()
352 print("# --- Done with PLOTS")

```

## E.2 Constants

Constants defined for the rest of the program to use (filename: const.py)

```

1 from math import pi,sqrt,pow
2
3 # Table constants
4 earth_moon_distance = 384400.0 # km
5 moon_orbit_duration = 27.322 # days
6 earth_radius = 6367.4447 # km
7 earth_mass = 5.9721986e24
8 moon_mass = 7.34767309e22
9 G = 6.67384e-11 # m^3 kg^-1 s^-2
10 moon_radius = 1737.1 # km
11 leo_orbit = 160.0+earth_radius # km
12 lunar_orbit = 100.0+moon_radius # km
13 orbit_range = 10.0 # km
14 day = 24.0*3600.0 # s
15
16 # Units
17 unit_len = earth_moon_distance # km
18 unit_time = moon_orbit_duration/(2.0*pi) # days
19 unit_vel = unit_len/(unit_time*day) # km/s
20
21 # Gravitational constants

```



```

22 mu = moon_mass/(earth_mass+moon_mass) # 1
23
24 # Calculation constants
25 leo_orbit_vel = sqrt(G*earth_mass/(leo_orbit*1000.0))/1000.0 # km/s
26 lunar_orbit_vel = sqrt(G*moon_mass/(lunar_orbit*1000.0))/1000.0 # km/s
27 moon_pos_x = 1-mu
28 earth_pos_x = -mu
29 L1_pos_x = (1-pow(mu/3.0,1.0/3.0))

```

## E.3 Plotter and Trajectory Finders

trajectory plot trajectories, hohmann and low\_energy finds transfer orbits (filename: reduced3body.py - most of it).

```

1 import time
2 from math import pi,sqrt
3 import numpy as np
4 from const import *
5 from numbapro import *
6 from search import search_mt, search, print_search_results
7 from symplectic import symplectic
8
9 # **BRUGER IKKE pos, ang, burn til noget, kun til print
10 def trajectory(n,duration,pos,ang,burn,x0,y0,px0,py0):
11     """Integrate trajectory for the reduced 3-body problem.
12
13     Args:
14         n (int): Positions stored.
15         duration (float): Time duration of simulation.
16         x0 (float): Initial x-coordinate
17         y0 (float): Initial y-coordinate
18         px0 (float): Initial generalized x-momentum
19         py0 (float): Initial generalized y-momentum
20
21     Returns:
22         Tuple of time-, x-, y-, px- and py lists.
23
24     """
25     print("# Running trajectory.")
26
27     # Initialize arrays
28     tlist = np.linspace(0,duration,n)
29     xlist = np.zeros(n)
30     ylist = np.zeros(n)
31     pxlist = np.zeros(n)
32     pylist = np.zeros(n)
33     errlist = np.zeros(n)
34     hlist = np.zeros(n)

```

```

35     info = np.zeros(2)
36
37     # Find orbits
38     runtime = time.time()
39     status = symplectic(n,duration,x0,y0,px0,py0,xlist,ylist,pxlist,pylist,
40                        errlist,hlist,info)
41     runtime = time.time()-runtime
42
43     # Display result
44     print_search_results(status,pos,ang,burn,x0,y0,px0,py0,info[0],info[1])
45     print("# Runtime = %3.2fs" % (runtime))
46     return tlist,xlist,ylist,pxlist,pylist,errlist,hlist
47
48 def hohmann(threads,n):
49     """Finding Hohmann trajectory for the reduced 3-body problem.
50
51     Args:
52         n (int): Positions stored.
53
54     Returns:
55         Tuple of time-, x-, y-, px- and py lists.
56
57     """
58
59     print("# Running Hohmann.")
60
61     # Hohmann trajectory < 6 days
62     duration = 6/unit_time
63     best_total_dv = 1e9
64     positions = 100
65     angles = 1
66     burns = 200
67     pos = -3*pi/4
68     ang = 0
69     burn = 3.11/unit_vel # Forward Hohmann
70     #burn_low = -3.14/unit_vel # Reverse Hohmann
71
72     # Super fast Hohmann trajectory < 1 days
73     #duration = 3/unit_time
74     #best_total_dv = 1e9
75     #positions = 10
76     #angles = 10
77     #burns = 200
78     #pos = -3*pi/4
79     #ang = 0
80     #burn = 3.7/unit_vel # Forward Hohmann
81
82     pos_range = pi/4
83     ang_range = pi/8
84     burn_range = 0.1/unit_vel
85
86     # Start search

```

```

86     searches = 0
87     max_searches = 5
88     while searches < max_searches:
89         runtime = time.time()
90         searches += 1
91         print("##### Search %i #####" % (searches))
92         print("# pos          = %f" % (pos))
93         print("# ang          = %f" % (ang))
94         print("# burn         = %f" % (burn))
95         pos_low = pos-pos_range
96         pos_high = pos+pos_range
97         ang_low = ang-ang_range
98         ang_high = ang+ang_range
99         burn_low = burn-burn_range
100        burn_high = burn+burn_range
101        stat,pos,ang,burn,x0,y0,px0,py0,dv,toa = search_mt(threads,1,
            duration,positions,angles,burns,pos_low,pos_high,ang_low,ang_high
            ,burn_low,burn_high)
102
103        if stat < 0:
104            total_dv = abs(burn)+dv
105            if best_total_dv > total_dv:
106                best_total_dv = total_dv
107                best_stat = stat
108                best_pos = pos
109                best_ang = ang
110                best_burn = burn
111                best_x0 = x0
112                best_y0 = y0
113                best_px0 = px0
114                best_py0 = py0
115                best_dv = dv
116                best_toa = toa
117            else:
118                break
119
120        pos_range *= 0.1
121        ang_range *= 0.1
122        burn_range *= 0.1
123
124        runtime = time.time()-runtime
125        print("# Search runtime    = %3.2fs" % (runtime))
126
127    # Print best result
128    print("##### Best #####")
129    print("# Best dV(total)    = %f km/s" % (best_total_dv*unit_vel))
130    print_search_results(best_stat,best_pos,best_ang,best_burn,best_x0,
        best_y0,best_px0,best_py0,best_dv,best_toa)
131
132    # Initialize arrays
133    tlist = np.linspace(0,duration,n)
134    xlist = np.zeros(n)

```

```

135     ylist = np.zeros(n)
136     pxlist = np.zeros(n)
137     pylist = np.zeros(n)
138     errlist = np.zeros(n)
139     hlist = np.zeros(n)
140     info = np.zeros(2)
141
142     # Do trajectory
143     duration = 10/unit_time
144     status = symplectic(n,duration,x0,y0,px0,py0,xlist,ylist,pxlist,pylist,
145                        errlist,hlist,info)
146
147     return tlist,xlist,ylist,pxlist,pylist,errlist,hlist
148
149 def low_energy(threads,n):
150     """Finding low energy transfer trajectory for the reduced 3-body problem
151     .
152
153     Args:
154         n (int): Positions stored.
155
156     Returns:
157         Tuple of time-, x-, y-, px- and py lists.
158
159     """
160
161     print("# Running low_energy.")
162
163     # Low-energy trajectory < 200 days
164     duration = 200/unit_time
165     best_total_dv = 1e9
166     positions = 100
167     angles = 1
168     burns = 200
169     pos = -3*pi/4
170     ang = 0
171     burn = 3.12/unit_vel
172     pos_range = pi
173     ang_range = 0
174     burn_range = 0.01/unit_vel
175
176     # Start search
177     searches = 0
178     max_searches = 1
179     while searches < max_searches:
180         runtime = time.time()
181         searches += 1
182         print("##### Search %i #####" % (searches))
183         print("# pos          = %f" % (pos))
184         print("# ang          = %f" % (ang))
185         print("# burn          = %f" % (burn))
186         pos_low = pos-pos_range

```

```

185     pos_high = pos+pos_range
186     ang_low = ang-ang_range
187     ang_high = ang+ang_range
188     burn_low = burn-burn_range
189     burn_high = burn+burn_range
190     stat,pos,ang,burn,x0,y0,px0,py0,dv,toa = search_mt(threads,1,
        duration,positions,angles,burns,pos_low,pos_high,ang_low,ang_high
        ,burn_low,burn_high)
191
192     if stat < 0:
193         total_dv = abs(burn)+dv
194         if best_total_dv > total_dv:
195             best_total_dv = total_dv
196             best_stat = stat
197             best_pos = pos
198             best_ang = ang
199             best_burn = burn
200             best_x0 = x0
201             best_y0 = y0
202             best_px0 = px0
203             best_py0 = py0
204             best_dv = dv
205             best_toa = toa
206         else:
207             break
208
209     pos_range *= 0.1
210     ang_range *= 0.1
211     burn_range *= 0.1
212
213     runtime = time.time()-runtime
214     print("# Search runtime      = %3.2fs" % (runtime))
215
216     # Initialize arrays
217     tlist = np.linspace(0,duration,n)
218     xlist = np.zeros(n)
219     ylist = np.zeros(n)
220     pxlist = np.zeros(n)
221     pylist = np.zeros(n)
222     errlist = np.zeros(n)
223     hlist = np.zeros(n)
224     info = np.zeros(2)
225
226     # Do trajectory
227     duration = toa+(2.0*pi*lunar_orbit/lunar_orbit_vel)/(unit_time*day)
228     status = symplectic(n,duration,x0,y0,px0,py0,xlist,ylist,pxlist,pylist,
        errlist,hlist,info)
229     exit()
230     return tlist,xlist,ylist,pxlist,pylist,errlist,hlist

```

## E.4 Search Helper Functions

Search helper functions, also implements all the paralellization (filename: search.py).

```

1  """
2  Brute-force search Module for reduced 3-body solver
3  =====
4
5  """
6
7  from __future__ import print_function
8  import sys
9  import time
10 from math import ceil
11 import numpy as np
12 from multiprocessing import Pool
13 from const import *
14 from symplectic import symplectic
15
16 def print_search_results(stat, pos, ang, burn, x0, y0, px0, py0, dv, toa):
17     print("#
18         -----
19         ")
20     print("duration = %i/unit_time" % (max(1, int(ceil(toa*unit_time)))))
21     print("pos      = %.15lf" % (pos))
22     print("ang      = %.15lf" % (ang))
23     print("burn     = %.15lf/unit_vel" % (burn*unit_vel))
24     print("x0      = %.15lf" % (x0))
25     print("y0      = %.15lf" % (y0))
26     print("px0     = %.15lf" % (px0))
27     print("py0     = %.15lf" % (py0))
28     print("#
29         -----
30         ")
31     print("# dV(earth-escape) = %f km/s" % (abs(burn)*unit_vel))
32     if stat > 0 and stat < 100:
33         print("# Min moon distance= %f" % (stat))
34     elif stat < 0:
35         print("# dV(moon-capture) = %f km/s" % (dv*unit_vel))
36         print("# dV(total)      = %f km/s" % ((abs(burn)+dv)*unit_vel))
37         print("# Flight-time      = %f days" % (toa*unit_time))
38     else:
39         print("# Crashed on earth!")
40     print("#
41         -----
42         ")
43     sys.stdout.flush()
44
45 def search(thread, threads, n, duration, positions, angles, burns):
46     #print("Start thread=%i" % (thread))
47
48     # Initialize arrays

```

```

44     xlist = np.zeros(n)
45     ylist = np.zeros(n)
46     pxlist = np.zeros(n)
47     pylist = np.zeros(n)
48     errlist = np.zeros(n)
49     hlist = np.zeros(n)
50     info = np.zeros(2)
51
52     # Search for orbits
53     trials = len(positions)*len(angles)*len(burns)
54     ld1 = len(angles)*len(burns)
55     ld2 = len(burns)
56     trial = 0
57     hit_earth = 0
58     hit_moon = 0
59     best_status = 1e9
60     progress = -1
61     i = thread
62     while i < trials:
63
64         # One-to-one mapping of i -> (pos_i,ang_i,burn_i)
65         pos_i = i//ld1
66         ang_i = (i-pos_i*ld1)//ld2
67         burn_i = i-pos_i*ld1-ang_i*ld2
68         i += threads
69
70         # Find launch setup
71         pos = positions[pos_i]
72         ang = angles[ang_i]
73         burn = burns[burn_i]
74
75         # Calculate initial conditions
76         r = leo_orbit/unit_len
77         x0 = np.cos(pos)*r
78         y0 = np.sin(pos)*r
79         rx = -y0/r
80         ry = x0/r
81         vx = (leo_orbit_vel/unit_vel)*rx
82         vy = (leo_orbit_vel/unit_vel)*ry
83         x0 += earth_pos_x
84         bx = np.cos(ang)*rx-np.sin(ang)*ry
85         by = np.sin(ang)*rx+np.cos(ang)*ry
86         px0 = (burn/abs(burn))*vx+burn*bx-y0 # Sign of burn decides
            rotational direction of launch
87         py0 = (burn/abs(burn))*vy+burn*by+x0
88
89         # Call symplectic integration
90         # status > 0      : Closest distance to moon achieved
91         # status < 0      : Hit the moon using status=dV(moon)-10000 to get
            into orbit
92         # status == 100   : Collided with earth
93         #if thread == 1:

```

```

94     # print(n,duration,x0,y0,px0,py0)
95     status = symplectic(n,duration,x0,y0,px0,py0,xlist,ylist,pxlist,
96         pylist,errlist,hlist,info)
97     if status == 100:
98         hit_earth += 1
99     if status < 0:
100         hit_moon += 1
101     if status < best_status:
102         best_status = status
103         best_pos = pos
104         best_ang = ang
105         best_burn = burn
106         best_x0 = x0
107         best_y0 = y0
108         best_px0 = px0
109         best_py0 = py0
110         best_dv = info[0]
111         best_toa = info[1]
112
113     # Show progress
114     if thread == 0: # only thread 0
115         if (100*trial/(1+trials//threads))//10 > progress:
116             progress = (100*trial/(1+trials//threads))//10
117             print(progress*10,end="% ")
118             sys.stdout.flush()
119             trial += 1
120     #if thread == 13:
121     #    print("thread=%i status=%f best_status=%f trial=%i(%i) pos=%f
122     #    ang=%f burn=%f" % (thread,status,best_status,trial,trials,pos,
123     #    ang,burn))
124
125     #print("End thread=%i" % (thread))
126
127     return best_status,best_pos,best_ang,best_burn,best_x0,best_y0,best_px0,
128         best_py0,best_dv,best_toa,hit_earth,hit_moon
129
130 def search_worker(args):
131     return search(args[0], args[1], args[2], args[3], args[4], args[5], args
132         [6])
133
134 def search_mt(threads,n,duration,num_pos,num_ang,num_burn,pos_low,pos_high,
135     ang_low,ang_high,burn_low,burn_high):
136
137     # Time search
138     runtime = time.time()
139
140     # Set search space
141     positions = np.linspace(pos_low,pos_high,num_pos)
142     angles = np.linspace(ang_low,ang_high,num_ang)
143     burns = np.linspace(burn_low,burn_high,num_burn)
144     if num_pos == 1:
145         positions[0] = (pos_high+pos_low)/2.0

```



```

140     if num_ang == 1:
141         angles[0] = (ang_high+ang_low)/2.0
142     if num_burn == 1:
143         burns[0] = (burn_high+burn_low)/2.0
144     if num_burn == 2:
145         burns[0] = burn_low
146         burns[1] = burn_high
147     trials = num_pos*num_ang*num_burn
148     print(positions)
149     print(angles)
150     print(burns*unit_vel)
151
152     # Set threads
153     threads = min(threads, num_pos)
154
155     # Do multi-threaded search
156     print("#
157         -----
158         ")
159     print("# Threads:          %6i" % (threads))
160     print("# Trials:          %6i (" % (trials), end="")
161     if threads == 1:
162         # Single thread
163         best_status, best_pos, best_ang, best_burn, best_x0, best_y0, best_px0,
164             best_py0, best_dv, best_toa, hit_earth, hit_moon = search(0, num_pos, n,
165                 duration, positions, angles, burns)
166     else:
167         # Multi-threading
168         chunk = num_pos/threads
169         args = [[i, threads, n, duration, positions, angles, burns] for i in range
170             (threads)]
171         pool = Pool()
172         result = pool.map(search_worker, args)
173         pool.close()
174         pool.join()
175
176         # Reduce results from all threads
177         best_status = 1e9
178         hit_earth = 0
179         hit_moon = 0
180         for i in range(threads):
181             status = result[i][0]
182             if status < best_status:
183                 best_status = status
184                 best_pos = result[i][1]
185                 best_ang = result[i][2]
186                 best_burn = result[i][3]
187                 best_x0 = result[i][4]
188                 best_y0 = result[i][5]
189                 best_px0 = result[i][6]
190                 best_py0 = result[i][7]
191                 best_dv = result[i][8]

```

```

187         best_toa = result[i][9]
188         hit_earth += result[i][10]
189         hit_moon += result[i][11]
190
191     print("100%")
192     print("# No interception:  %6i (%i%%)" % (trials-hit_earth-hit_moon
193         ,100*(trials-hit_earth-hit_moon)/trials))
194     print("# Crashed on earth: %6i (%i%%)" % (hit_earth,100*hit_earth/trials
195         ))
196     print("# Hit moon:          %6i (%i%%)" % (hit_moon,100*hit_moon/trials))
197     runtime = time.time()-runtime
198     print("# Runtime:          %6.2fs" % (runtime))
199     if best_status < 100:
200         print_search_results(best_status,best_pos,best_ang,best_burn,best_x0
201             ,best_y0,best_px0,best_py0,best_dv,best_toa)
202         return best_status,best_pos,best_ang,best_burn,best_x0,best_y0,
203             best_px0,best_py0,best_dv,best_toa
204     else:
205         return best_status,0,0,0,0,0,0,0,0,0

```

## E.5 Symplectic Adaptive Verlet and Symplectic Euler Integrators

Implementing the symplectic algorithms from Chapter 4 (filename: symplectic.py).

```

1  """
2  Reduced 3-Body Problem Solver Module
3  =====
4  """
5
6  from math import pi,sqrt
7  import numpy as np
8  from numbapro import *
9  from const import *
10
11  @jit
12  def F(x,y):
13      mux = mu+x
14      mux2 = mux*mux
15      mumx = 1-mux
16      mumx2 = mumx*mumx
17      y2 = y*y
18      denum1 = 1.0/(((mux2+y2)*sqrt(mux2+y2))
19      denum2 = 1.0/(((mumx2+y2)*sqrt(mumx2+y2))
20      Fx = (mu-1.0)*mux*denum1+mu*mumx*denum2
21      Fy = (mu-1.0)*y*denum1-mu*y*denum2
22      return Fx,Fy

```

```

23
24 @jit
25 def explicit_euler_step(h,x,y,px,py):
26     Fx,Fy = F(x,y)
27     vx = px+y
28     vy = py-x
29     vpx = Fx+py
30     vpy = Fy-px
31     x += vx*h
32     y += vy*h
33     px += vpx*h
34     py += vpy*h
35     return x,y,px,py
36
37 @jit
38 def symplectic_euler_step(h,x,y,px,py):
39     # Step 1
40     vx = px+y
41     x = (x+(vx+py*h)*h)/(1.0+h*h)
42     vy = py-x
43     y += vy*h
44     # Step 2
45     Fx,Fy = F(x,y)
46     vpx = Fx+py
47     vpy = Fy-px
48     px += vpx*h
49     py += vpy*h
50     return x,y,px,py
51
52 @jit
53 def symplectic_verlet_step(h,x,y,px,py):
54     hh = 0.5*h
55     denum = 1.0/(1.0+hh*hh)
56     # Step 1
57     vx = px+y
58     x = (x+(vx+py*hh)*hh)*denum
59     vy = py-x
60     y += vy*hh
61     # Step 2
62     Fx,Fy = F(x,y)
63     vpx = Fx+py
64     vpy = Fy-px
65     px = (px+(2.0*vpx+(vpy+Fy)*hh)*hh)*denum
66     py += (vpy+Fy-px)*hh
67     # Step 3
68     vx = px+y
69     vy = py-x
70     x += vx*hh
71     y += vy*hh
72     return x,y,px,py
73
74 @jit #('void(int64, float64, float64, float64, float64, float64, float64,

```

```

float64[:,], float64[:,], float64[:,], float64[:,])')
75 def symplectic(n,duration,x0,y0,px0,py0,xlist,ylist,pxlist,pylist,errlist,
    hlist,info):
76
77     # Initialize initial conditions
78     h = 1e-6
79     hmin = 1e-10
80     # tol = 1e-9
81     tol = 10
82     maxsteps = duration
83     x = x0
84     y = y0
85     px = px0
86     py = py0
87     err = 1e-15
88     status = 1
89     target_dist = 1
90     target = 1; target_pos_x = moon_pos_x
91     #target = 2; target_pos_x = L1_pos_x
92     target_pos_y = 0
93
94     # Time reset
95     t = 0
96     for i in range(n):
97
98         # Store position
99         xlist[i] = x
100        ylist[i] = y
101        pxlist[i] = px
102        pylist[i] = py
103        errlist[i] = err
104        hlist[i] = h
105
106        # Integrate time period
107        dt = duration*(i+1)/n
108        count = 0
109        while t < dt:
110            # Safety on iterations
111            count += 1
112            if count > 100000000:
113                count = 0
114                hmin = 2*hmin
115
116            # Adaptive symplectic euler/midpoint
117            x1,y1,px1,py1 = symplectic_euler_step(h,x,y,px,py)
118            x2,y2,px2,py2 = symplectic_verlet_step(h,x,y,px,py)
119
120            # Relative local error of step
121            err = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)/(x2*x2+y2*y2))
122
123            # Accept the step only if the weighted error is no more than the
124            # tolerance tol. Estimate an h that will yield an error of tol

```

```

125         on
126         # the next step and use 0.8 of this value to avoid failures.
127         if err < tol or h <= hmin:
128
129             # Accept step
130             x = x2
131             y = y2
132             px = px2
133             py = py2
134
135             # Forward time by step
136             t = t+h
137             # h = max(hmin, h*max(0.1, 0.8*sqrt(tol/err)))
138
139         else:
140             # No accept, reduce h to half
141             h = max(hmin, 0.5*h)
142
143         # How close are we to the moon?
144         rx = x-target_pos_x
145         ry = y-target_pos_y
146         r = sqrt(rx*rx+ry*ry)
147         target_dist = min(target_dist,r)
148
149         # Check if we hit the target
150         if status == 1:
151             if target == 1:
152                 r_low = (lunar_orbit-orbit_range)/unit_len
153                 r_high = (lunar_orbit+orbit_range)/unit_len
154             else:
155                 r_low = 0
156                 r_high = orbit_range/unit_len
157
158             if r > r_low and r < r_high:
159
160                 # Current velocity
161                 vx = px+y
162                 vy = py-x
163
164                 if target == 1:
165
166                     # Project velocity onto radius vector and subtract
167                     # so velocity vector is along orbit
168                     vr = (vx*rx+vy*ry)/r
169                     vx = vx-vr*rx/r
170                     vy = vy-vr*ry/r
171
172                     # Now ajust velocity to lunar orbit velocity
173                     vt = sqrt(vx*vx+vy*vy)
174                     px = (lunar_orbit_vel/unit_vel)*vx/vt-y
175                     py = (lunar_orbit_vel/unit_vel)*vy/vt+x

```

```
176         # Total velocity change
177         dv = sqrt(vr*vr+(vt-lunar_orbit_vel/unit_vel)*(vt-
178                 lunar_orbit_vel/unit_vel))
179     else:
180         dv = sqrt(vx*vx+vy*vy)
181
182     # Store info
183     info[0] = dv
184     info[1] = t
185
186     # Finish?
187     status = -10000+dv
188     if n == 1:
189         return status
190
191     # Check if we hit the earth
192     r = (x-earth_pos_x)*(x-earth_pos_x)+y*y
193     r_high = earth_radius/unit_len
194     if r < r_high*r_high:
195         return 100 # Hit earth surface
196
197 if status >= 0:
198     status = target_dist
199
200 return status
```

# APPENDIX F

## Mathematica Data

---

```
In[42]:= Quiet@Remove["`*"]
```

```
In[43]:= SetOptions[SelectedNotebook[],  
  PrintingStyleEnvironment -> "Printout", ShowSyntaxStyles -> True]
```

---

## Date and version

Notebook run / all data acquired from Wolfram's servers at time:

```
In[44]:= Now[]
```

```
Out[44]=  Wed 10 Jun 2015 19:42:38 []
```

```
In[45]:= $Version
```

```
Out[45]= 10.0 for Mac OS X x86 (64-bit) (June 29, 2014)
```

---

## Raw data

### Universal

```
In[46]:= UnitConvert[Quantity[1, "GravitationalConstant"], "SIBase"]
```

```
Out[46]=  $6.67 \times 10^{-11} \text{ m}^3 / (\text{kg s}^2)$ 
```

### Earth

```
In[47]:= PlanetData[Earth (planet), "Mass"]
```

```
Out[47]=  $5.9721986 \times 10^{24} \text{ kg}$ 
```

```
In[48]:= PlanetData[Earth (planet), "Radius"]
```

```
Out[48]= 6367.4447 km
```

### Moon

```
In[49]:= PlanetaryMoonData[Moon (planetary moon), "Mass"]
```

```
Out[49]=  $7.3459 \times 10^{22} \text{ kg}$ 
```

```
In[50]:= PlanetaryMoonData[Moon (planetary moon), "Radius"]
```

```
Out[50]= 1737.5 km
```

```
In[51]:= PlanetaryMoonData[Moon (planetary moon), "Inclination"]
```

```
Out[51]=  $5.16^\circ$ 
```



## Earth-Moon

In[52]:= **PlanetaryMoonData** [ **Moon** (planetary moon) , "AverageDistanceFromEarth" ]

Out[52]=  $3.850 \times 10^5$  km

In[53]:= **PlanetaryMoonData** [ **Moon** (planetary moon) , "OrbitPeriod" ]

Out[53]= 27.322 days

## Derived Units

### Unit velocities

In[54]:= **unitTime** =  $\frac{1}{2 \text{ Pi}}$  **PlanetaryMoonData** [ **Moon** (planetary moon) , "OrbitPeriod" ]

Out[54]= 4.3484 days

In[55]:= **unitLength** = **PlanetaryMoonData** [ **Moon** (planetary moon) , "AverageDistanceFromEarth" ]

Out[55]=  $3.850 \times 10^5$  km

In[56]:= **unitVelocity** = **UnitConvert** [  $\frac{\text{unitLength}}{\text{unitTime}}$  , "km/s" ]

Out[56]= 1.025 km/s

## Earth sphere of influence

In[72]:= **=** earth sphere of influence / average moon orbit distance

**PlanetData** [ **Earth** (planet) , "SphereOfInfluenceRadius" ] /  
**PlanetaryMoonData** [ **Moon** (planetary moon) , "SemimajorAxis" ]

Out[72]= 2.41

## Parking Velocities

### Earth

In[57]:= **earthParkingVelocity** =

**UnitConvert** [  $\sqrt{\left( G \frac{\text{PlanetData} [ \text{Earth} (\text{planet}) , "Mass" ]}{\text{PlanetData} [ \text{Earth} (\text{planet}) , "Radius" ] + 160 \text{ km}} \right)}$  , "km/s" ]

Out[57]= 7.814 km/s

```
In[58]:= earthParkingVelocity / unitVelocity
```

```
Out[58]:= 7.63
```

## Moon

```
In[59]:= moonParkingVelocity = UnitConvert[  


$$\sqrt{\left( G \frac{\text{PlanetaryMoonData}[\text{Moon (planetary moon)}, \text{"Mass"}]}{\text{PlanetaryMoonData}[\text{Moon (planetary moon)}, \text{"Radius"}] + 100 \text{ km}} \right), \text{"km/s"}]$$

```

```
Out[59]:= 1.633 km/s
```

```
In[60]:= moonParkingVelocity / unitVelocity
```

```
Out[60]:= 1.594
```

## Miscellaneous

```
In[61]:=  $\mu = G * \text{PlanetData}[\text{Earth (planet)}, \text{"Mass"}]$ 
```

```
Out[61]:=  $5.9721986 \times 10^{24} \text{ kg } G$ 
```

```
In[62]:= 0.1 * unitVelocity
```

```
Out[62]:= 0.102471 km/s
```

---

## Apollo delta-v

```
In[69]:= (*Trans-lunar injection*)  
apolloEarth = UnitConvert[Quantity[10 000, "Feet per second"], "km/s"] // N
```

```
Out[69]:= 3.048 km/s
```

```
In[71]:= (*Lunar orbit insertion (PDF p. 27 in source*)  
apolloMoon = UnitConvert[Quantity[3500, "Feet per second"], "km/s"] // N
```

```
Out[71]:= 1.0668 km/s
```

```
In[67]:= UnitConvert[apolloEarth + apolloMoon, "Km/s"] // N
```

```
Out[67]:= 4.1148 km/s
```

# Bibliography

---

- [And] Ross Andersen. *The Elon Musk interview on Mars colonisation – Ross Andersen – Aeon*. URL: <http://aeon.co/magazine/technology/the-elon-musk-interview-on-mars/> (visited on June 4, 2015).
- [Ast] Astrobotic. *Lunar Delivery | Astrobotic*. URL: <https://www.astrobotic.com/lunar-delivery> (visited on June 4, 2015).
- [Bar] Hubert Bartkowiak. *File:Hohmann transfer orbit.svg - Wikimedia Commons*. URL: [http://commons.wikimedia.org/wiki/File:Hohmann\\_transfer\\_orbit.svg](http://commons.wikimedia.org/wiki/File:Hohmann_transfer_orbit.svg) (visited on June 6, 2015).
- [BBC] BBC. *BBC News - Obama cancels Moon return project*. URL: <http://news.bbc.co.uk/2/hi/science/nature/8489097.stm> (visited on June 4, 2015).
- [BC00] Edward a Belbruno and John P Carrico. “Calculation of Weak Stability Boundary Ballistic Lunar Transfer Trajectories”. In: *Astrodynamics specialist conference*. Denver, Colorado, 2000. DOI: doi:10.2514/6.2000-4142.
- [BJ00] R Biesbroek and G Janin. “Ways to the Moon?” In: *Earth* august (2000), pages 92–99.
- [Bra12] Jim Branson. *Recalling Lagrangian Mechanics*. 2012. URL: [http://hepweb.ucsd.edu/ph110b/110b\\_notes/node86.html](http://hepweb.ucsd.edu/ph110b/110b_notes/node86.html) (visited on June 8, 2015).
- [Buc] Andrew Buck. *File:Bi-elliptic transfer.svg - Wikimedia Commons*. URL: [http://commons.wikimedia.org/wiki/File:Bi-elliptic\\_transfer.svg](http://commons.wikimedia.org/wiki/File:Bi-elliptic_transfer.svg) (visited on June 6, 2015).
- [CNN] CNN. *CNN.com - Bush unveils vision for moon and beyond - Jan. 15, 2004*. URL: <http://edition.cnn.com/2004/TECH/space/01/14/bush.space/> (visited on June 4, 2015).
- [Coo] Jonathan Coopersmith. *Affordable Access to Space | Issues in Science and Technology*. URL: <http://issues.org/29-1/jonathan/> (visited on June 4, 2015).
- [Cop13] Copperheadtnp. *I was curious about the delta v requirements for different transfers. So I made this plot. At best, a bi-elliptic transfer will require 92% of the delta v of a Hohmann transfer. : KerbalSpaceProgram*. 2013. URL: [http://www.reddit.com/r/KerbalSpaceProgram/comments/1ajru7/i\\_was\\_curious\\_about\\_the\\_delta\\_v\\_requirements\\_for/](http://www.reddit.com/r/KerbalSpaceProgram/comments/1ajru7/i_was_curious_about_the_delta_v_requirements_for/) (visited on June 6, 2015).

- [ESA14] ESA. *Soyuz rendezvous and docking explained*. 2014. URL: [https://www.youtube.com/watch?v=M2\\_NeFbFcSw%5C&feature=youtu.be%5C&t=9m55s](https://www.youtube.com/watch?v=M2_NeFbFcSw%5C&feature=youtu.be%5C&t=9m55s).
- [Fau] Kjell Magne Fauske. *Global nodes / TikZ example*. URL: <http://www.texample.net/tikz/examples/global-nodes/> (visited on May 29, 2015).
- [Fra83] Craig Fraser. “J. L. Lagrange’s early contributions to the principles and methods of mechanics”. In: *Archive for History of Exact Sciences* 28.3 (1983), pages 197–241. ISSN: 00039519. DOI: 10.1007/BF00328268. URL: <http://link.springer.com/10.1007/BF00328268>.
- [GG07] F. García and G. Gómez. “A note on weak stability boundaries”. In: *Celestial Mechanics and Dynamical Astronomy* 97.2 (2007), pages 87–100. ISSN: 09232958. DOI: 10.1007/s10569-006-9053-6.
- [Gol80] Herman Heine Goldstine. *A history of the calculus of variations from the 17th through the 19th century*. First Edit. Springer-Verlag, 1980, page 410. ISBN: 3540905219. URL: <https://books.google.com/books?id=ymQPAQAAMAAJ%5C&pgis=1>.
- [GPS02] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics*. 3rd. Addison Wesley, 2002, page 638. ISBN: 0201657023. URL: [https://books.google.dk/books/about/Classical\\_Mechanics.html?id=tJCuQgAACAAJ%5C&pgis=1](https://books.google.dk/books/about/Classical_Mechanics.html?id=tJCuQgAACAAJ%5C&pgis=1).
- [Hjo15] P G Hjorth. *A few Concepts from Analytical Mechanics, DTU (unpublished)*. 2015.
- [Hoc08] Marlis Hochbruck. “Symplectic Integrators”. In: (2008). URL: <https://na.math.kit.edu/marlis/download/meetings/08Oberwolfach/symplectic1-print.pdf>.
- [HT10] Ernst Hairer and M Tu. *Lecture 1 : Hamiltonian systems*. 2010. URL: [http://www.unige.ch/~hairer/poly\\_geoint/week1.pdf](http://www.unige.ch/~hairer/poly_geoint/week1.pdf) (visited on June 8, 2015).
- [Juu08] Jeppe Søgaard Juul. *Low Energy Trajectories to the Moon*. Technical report. University of Copenhagen, 2008, page 36. URL: <http://jeppejuul.com/research/projects/bachelors-project-low-energy-trajectories-to-the-moon/>.
- [KH02] Jens Martin Knudsen and Poul G. Hjorth. *Elements of Newtonian Mechanics*. 3rd Editio. Springer-Verlag, 2002, page 451. ISBN: 3-540-67652-X.
- [Koo+01] W.S. Koon et al. “Low Energy Transfer to the Moon”. In: *Celestial Mechanics and Dynamical Astronomy* 81.1-2 (2001), pages 63–73. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.307.2362>.
- [Lan70] Cornelius Lanczos. *The Variational Principles of Mechanics*. Fourth Edi. Dover Publications, 1970, page 418. ISBN: 978-0486650678.

- [Lau] Laursen. *\_laursen / Laursen's XeLaTeX thesis template — Bitbucket*. URL: [https://bitbucket.org/\\_laursen/laursens-xelatex-thesis-template/overview](https://bitbucket.org/_laursen/laursens-xelatex-thesis-template/overview) (visited on May 29, 2015).
- [MD99] Carl D. Murray and Stanley F. Dermott. *Solar System Dynamics*. 1st. Cambridge University Press, 1999, page 592. ISBN: 0-521-57295-9.
- [Min] J.R. Minkel. *Is the International Space Station Worth \$100 Billion?* URL: <http://www.space.com/9435-international-space-station-worth-100-billion.html> (visited on June 9, 2015).
- [NASa] NASA. *Apollo 11 Mission Overview*. URL: [http://www.lpi.usra.edu/lunar/missions/apollo/apollo\\_11/overview/](http://www.lpi.usra.edu/lunar/missions/apollo/apollo_11/overview/) (visited on June 11, 2015).
- [NASb] NASA. *Launch Windows Essay*. URL: <http://history.nasa.gov/afj/launchwindow/lw1.html> (visited on June 8, 2015).
- [NASc] NASA. *NASA - NSSDC - Spacecraft - Details*. URL: <http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1990-007A> (visited on June 8, 2015).
- [NAS66] NASA. “Apollo Lunar Landing Mission Symposium”. In: 1966. URL: [http://www.hq.nasa.gov/alsj/LunarLandingMissionSymposium1966\\_1978075303.pdf](http://www.hq.nasa.gov/alsj/LunarLandingMissionSymposium1966_1978075303.pdf).
- [NF02] Michiyo Nakane and Craig G. Fraser. “The Early History of Hamilton-Jacobi Dynamics 1834?1837”. In: *Centaurus* 44.3-4 (December 2002), pages 161–227. ISSN: 0008-8994. DOI: 10.1111/j.1600-0498.2002.tb00613.x. URL: <http://doi.wiley.com/10.1111/j.1600-0498.2002.tb00613.x>.
- [Oka15] Jacob Akira Okada. *Painting the Way to the Moon*. 2015. URL: <http://www.paintingthewaytothemoon.com/%20http://www.imdb.com/title/tt4294246/>.
- [Pee] Matthew M. Peet. *Spacecraft and Aircraft Dynamics - Lecture 9: Bi-elliptics and Out-of-Plane Maneuvers*. URL: <http://mmae.iit.edu/~mpeet/Classes/MMAE441/Spacecraft/441Lecture21.pdf>.
- [Pis05] Vincent L. Pisacane. *Fundamentals of Space Systems*. 2nd. Oxford University Press, 2005, page 828. ISBN: 978-0-19-516205-9.
- [Pru92] John E. Prussing. “Simple proof of the global optimality of the Hohmann transfer”. en. In: *Journal of Guidance, Control, and Dynamics* 15.4 (July 1992), pages 1037–1038. ISSN: 0731-5090. DOI: 10.2514/3.20941. URL: <http://arc.aiaa.org/globalproxy.cvt.dk/doi/abs/10.2514/3.20941>.
- [Ram15] Sarah Ramsey. *NASA's Space Launch System Booster Passes Major Ground Test*. 2015. URL: <http://www.nasa.gov/press/2015/march/nasas-space-launch-system-booster-passes-major-ground-test>.
- [See02] Wolfgang Seefelder. *Lunar Transfer Orbits utilizing Solar Perturbations and Ballistic Capture*. 1st. Herbert Utz Verlag, 2002, page 171. ISBN: 3-8316-0155-0.

- [Sta] Stackexchange. *Stackexchange Mathematics - Who came up with the Euler-Lagrange equation first?* URL: <http://math.stackexchange.com/questions/177243/who-came-up-with-the-euler-lagrange-equation-first> (visited on June 2, 2015).
- [Swe91] Theodore H. Sweetser. "Estimate of the global minimum DV needed for earth-moon transfer". In: *Advances in the Astronautical Sciences* 75.pt 1 (1991), pages 111–120. ISSN: 00653438.
- [Tay09] Travis S. Taylor. *Introduction to Rocket Science and Engineering*. 1st. CRC Press, 2009, page 310. ISBN: 1420075292. URL: <https://books.google.com/books?id=dQHMBQAAQBAJ%5C&pgis=1>.
- [TB14] Francesco Topputo and E Belbruno. "Earth–Mars Transfers with Ballistic Capture". In: *arXiv preprint arXiv:1410.8856* (2014), pages 1–21. ISSN: 0923-2958. DOI: 10.1007/s10569-015-9605-8. arXiv: 1410.8856. URL: <http://arxiv.org/abs/1410.8856>.
- [TVB05] Francesco Topputo, Massimiliano Vasile, and Franco Bernelli-Zazzera. "Earth-to-moon low energy transfers targeting L1 hyperbolic transit orbits". In: *Annals of the New York Academy of Sciences*. June. 2005. DOI: 10.1196/annals.1370.025. URL: [https://www.researchgate.net/profile/Francesco\\_Topputo/publication/45417269\\_Earth-to-Moon\\_Low\\_Energy\\_Transfers\\_Targeting\\_L1\\_Hyperbolic\\_Transit\\_Orbits/links/0046351f230274d70a000000.pdf](https://www.researchgate.net/profile/Francesco_Topputo/publication/45417269_Earth-to-Moon_Low_Energy_Transfers_Targeting_L1_Hyperbolic_Transit_Orbits/links/0046351f230274d70a000000.pdf).
- [Use] Stanmar (Wikipedia User). *File:EllipseInPolarCoords2.svg - Wikimedia Commons*. URL: <http://commons.wikimedia.org/wiki/File:EllipseInPolarCoords2.svg> (visited on June 6, 2015).
- [Wei] Eric W. Weisstein. *Noether's Symmetry Theorem*. en. URL: <http://mathworld.wolfram.com/NoethersSymmetryTheorem.html>.
- [Whe] Robin Wheeler. *Apollo lunar landing launch window: The controlling factors and constraints*. URL: <http://history.nasa.gov/afj/launchwindow/lw1.html%5C#EPO> (visited on June 4, 2015).
- [Wika] Wikipedia. *Wikipedia - Bi-elliptic transfer*. URL: [http://en.wikipedia.org/wiki/Bi-elliptic\\_transfer%5C#Example](http://en.wikipedia.org/wiki/Bi-elliptic_transfer%5C#Example) (visited on June 6, 2015).
- [Wikb] Wikipedia. *Wikipedia: Lagrangian - Advantages over other methods*. URL: [http://en.wikipedia.org/wiki/Lagrangian%5C#Advantages\\_over\\_other\\_methods](http://en.wikipedia.org/wiki/Lagrangian%5C#Advantages_over_other_methods) (visited on June 1, 2015).
- [Wike] Wikipedia. *Wikipedia: List of missions to the Moon*. URL: [http://en.wikipedia.org/wiki/List\\_of\\_missions\\_to\\_the\\_Moon](http://en.wikipedia.org/wiki/List_of_missions_to_the_Moon) (visited on June 4, 2015).
- [Wikd] Wikipedia. *Wikipedia: Quadratic form*. URL: [http://en.wikipedia.org/wiki/Quadratic\\_form](http://en.wikipedia.org/wiki/Quadratic_form) (visited on June 2, 2015).
- [Wike] Wikipedia. *Wikipedia - SpaceX*. (Visited on June 4, 2015).

- [Wil] Matt Williams. *Making the Trip to Mars Cheaper and Easier: The Case for Ballistic Capture*. URL: <http://www.universetoday.com/117615/making-the-trip-to-mars-cheaper-and-easier-the-case-for-ballistic-capture/> (visited on June 8, 2015).
- [Wol] Inc. Wolfram Research. *Mathematica* (see appendix "Mathematica Data" for details). Champaign, Illinois.

