Optimizing the Trajectory of a Rocket Launch to Orbit

Student: Ben Busschaert

Number of Pages: 22

Introduction

Rationale

Rocket trajectories can be described through sets of differential equations that may not be solvable. To 'solve' such equations, numerical solutions are generally utilised, which are a trial-and-error approach to the problem that produce an approximate solution for a specific instance of the problem. Rocket trajectories are far from the only type of problems for which analytical solutions do not exist (or have not yet been discovered). Many real-life problems which involve differential equations or partial differential equations, namely in physics, must be solved numerically. These problems involve modelling heat transfer, fluid motion or even just the swing of a pendulum. Modelling the trajectory of a rocket was a particularly interesting problem for me as I have been engaged in staying up to date with current space travel, often watching SpaceX or NASA rocket launches live. In some instances of the rocket launches, the trajectory the rocket follows can clearly be seen through the bright trail of fumes the rocket leaves behind. Seeing this many times over has made me wonder how the engineers of the rocket calculated and optimised such a trajectory.

Figure 1 --A long exposure of a SpaceX Falcon 9 rocket launch clearly showing the rocket's trajectory.

Aim

The aim of this investigation will be to model and optimize the trajectory of a single-staged rocket to orbit. To achieve this, I will construct a set of functions that can be used to describe the state of the rocket at a given instance in time. A set of constraints will then be formulated to depict the state of the rocket at the beginning and the end of the trajectory. From these constraints and functions, I will then attempt to optimize the trajectory of the rocket to minimize the fuel cost. This type of optimization problem is known as an optimal control problem and solutions to such problems typically cannot be found through analytical means.[1] I will therefore try solving this problem by formulating it as a nonlinear programming problem (NLP) for which numerical solutions can be approximated through computational means using of an NLP solver.

Creating a Model

In an orbit, the position of a rocket in relation to the body it is orbiting is always changing as the rocket is constantly moving. For real rockets, there are three dimensions of space to consider, however, I will only be considering two dimensions for the sake of simplicity. By measuring position in two dimensions, some information, such as the inclination of orbits, is lost. This makes this type of analysis perhaps unfit for the precise space travel of current days, however, it could still be a powerful tool when designing rockets, mainly to estimate how much fuel the rocket will need in order to achieve orbit. This is because when designing, precise position measurements are not needed and would just increase complexity and make the analysis dependent on values that may not yet be known, impeding on the process of designing.

I will also not be taking into account aerodynamic drag, again, for the sake of simplicity, so this model is more applicable for rocket launches from celestial bodies that do not have an atmosphere, such as a rocket launch from the Moon into lunar orbit. To create and assess the viability of the model, I will be using data from the ascent stage of the Apollo Program Lunar Module (LM), which is the part of the Apollo rockets that would take off from the Moon to go into lunar orbit. For specific orbital and positional values, data from the Apollo 11 mission will be used. The minimum fuel required calculated by this investigation will then be compared to the

[1] Kelly, Matthew. "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation." *SIAM Review*, vol. 59, no. 4, 2017, pp. 849–904., doi:10.1137/16m1062569.

real fuel used during the Apollo 11 LM ascent to orbit. Specifications of the LM ascent stage are depicted in the table below:

Engine thrust ($\lvert \underline{F}_T \rvert$):	15,346 newtons
Specific impulse (I_{sp}):	309.7 seconds
LM ascent stage wet mass (m_0):	4,821 kilograms
LM ascent stage dry mass (m_1):	2,445 kilograms

Table 1

State Variables

State variables are functions that can be used to describe the dynamics, or motion, of a system at a given point in time. To get into orbit, the main state variable to consider is the position of the rocket, $\underline{P}$. It is then imperative to be able to describe both the first and second derivative of the position, $\underline{\dot{P}}$ and $\underline{\ddot{P}}$, which depict the rocket's velocity and acceleration respectively. Since this model will be in two dimensions, these functions are all vectors with an x and y component. The position of the rocket will be measured as a comparison to the centre of the Moon, which will be at the origin:

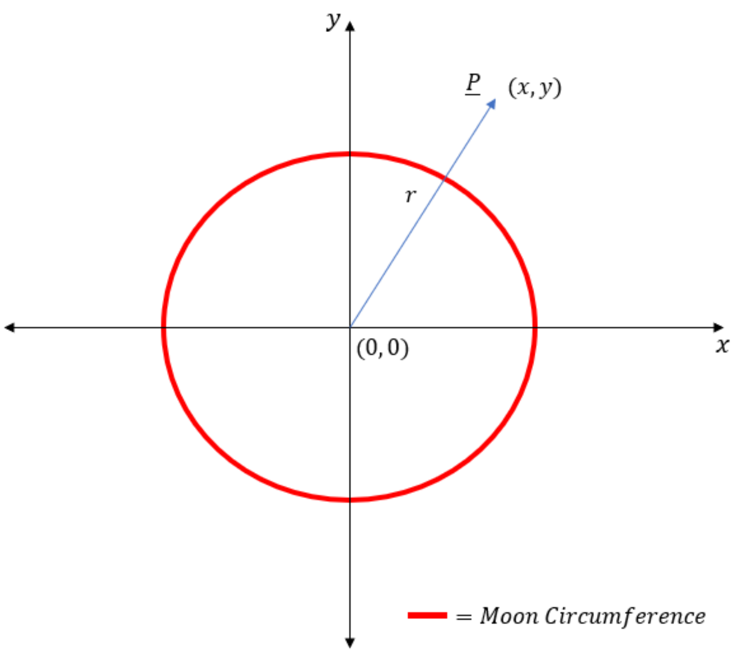


Figure 2 --A diagram showing how the position of the rocket will be measured

The distance to the centre of the moon is simply equal to the length of the rocket's position vector:

$$r = \lvert \underline{P} \rvert$$

$$r = \sqrt{x^2 + y^2} \quad (1)$$

Where r is the rocket's distance to the origin and x, y are the coordinates of the rocket. The rocket will be regarded as a point, as taking into account its size or shape would not add any particular useful information about the rocket's trajectory.

The Control Function

A rocket moves thanks to the force its engine creates. In this analysis, it will be assumed that the force produced by the engine is constant. This is accurate to the real LM ascent stage as its engine produced a fixed amount of thrust[2]. To go into orbit, the LM ascent stage needed to increase its height off the lunar surface and also gain enough tangential velocity to remain in an orbit at the desired height.

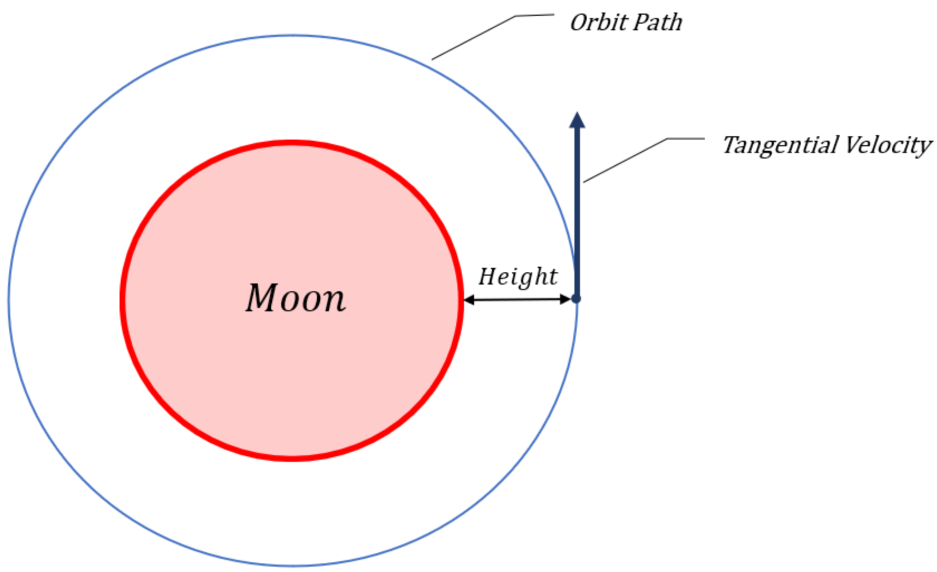


Figure 3 --To get into orbit, a rocket needs to not only increase its height off from the surface of the moon but must also gain enough velocity perpendicular to its position vector to remain in orbit.

This meant that the LM had to continuously change the angle of the force it produced, starting off by producing a purely vertical force to gain height, to at the very end produce a force perpendicular to its position vector, so as to only increase its tangential velocity. Since the main engine of the LM was fixed, it did this by continuously changing the angle of the entire rocket throughout its trajectory to orbit.

[2] Hooper, J.C. "Performance Analysis of the Ascent Propulsion System of the Apollo Spacecraft." NASA Lyndon B. Johnson Space Center, 1 Dec. 1973. Accessed Nov. 16 2022.

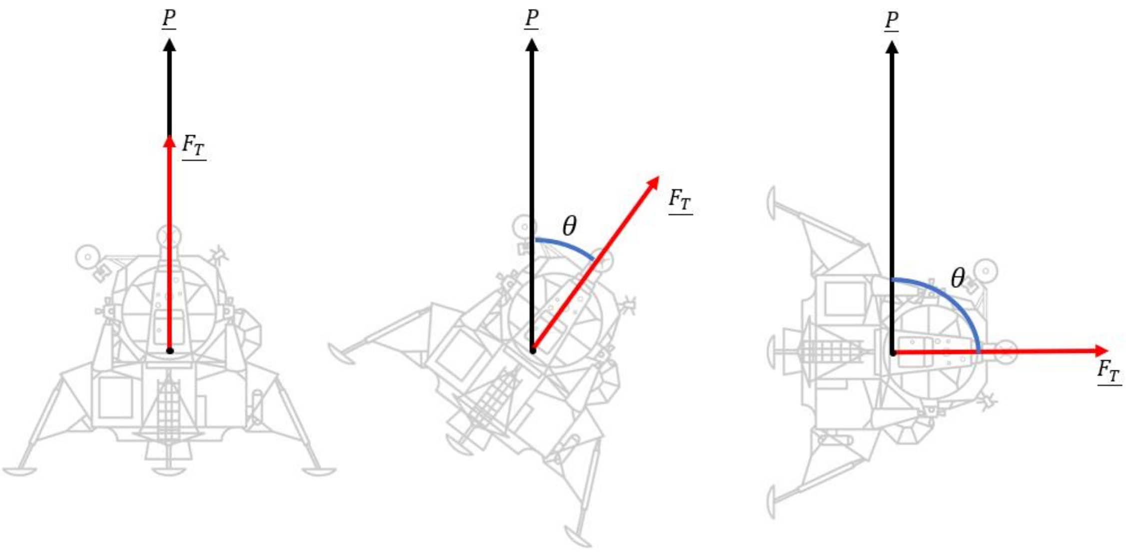


Figure 4 --How the angle between the LM's thrust vector and position vector changes as the LM ascends to orbit (left to right).

The function that will be controlled and optimized to minimize fuel cost will hence be the angle of the rocket, θ. This angle will be measured between the position vector of the rocket, $\underline{P}$, and the thrust vector, $\underline{F}_T$ (see *Figure 4*).

Mass

The mass of a rocket changes as its engine burns fuel. For rocket engines, there are two general properties that describe the performance of an engine: thrust, $|\underline{F}_T|$, and specific impulse, I_{sp}. $|\underline{F}_T|$ is simply a measure of the magnitude of the force produced by the engine, and specific impulse is a measure of fuel efficiency. The mass flow rate out of the system, $\dot{m}$, is part of the equation for I_{sp}:

$$I_{sp} = \frac{|\underline{F}_T|}{\dot{m} g_0} \tag{2}$$

Where g_0 is the gravitational acceleration at sea level of $9.807\ m\,s^{-2}$. Rearranging for mass flow rate:

$$\dot{m} = \frac{|\underline{F}_T|}{I_{sp} g_0}$$

This is useful since $\dot{m}$ describes how the mass of a rocket changes with time. Since $|\underline{F}_T|$, I_{sp} and g_0 are all constants which are known (see *Table 1*), the mass flow rate is also constant, and its value can be determined:

$$\dot{m} = \frac{(15,346)}{(309.7) \times (9.807)}$$

$$\dot{m} = \frac{15,346}{3,037.2279}$$

$$\dot{m} \approx 5.053 \; kilograms \; per \; second$$

$\dot{m}$ is rounded to three decimal places as any greater precision to the value would be meaningless. This is because all the values used to calculate $\dot{m}$ are measured values which were measured with limited precision (of four or five significant figures). To remain consistent throughout the investigation, all values calculated will be rounded to three decimal places. $\dot{m}$ is the negative change in the mass of the rocket with respect to time, t (it is negative since it is the mass of fuel being expelled by the rocket):

$$\dot{m} = -\frac{dm}{dt}$$

$$\frac{dm}{dt} = -(5.053)$$

Integrating both sides:

$$m = -\int 5.053 \, dt$$

$$m = -[5.053t] + C$$

$$m = C - 5.053t$$

Where C is a constant. To find the value of C, we can inspect the state of the rocket at $t = 0$. The mass of the rocket $t = 0$ is simply the initial mass of the LM, m_0, (also referred to as the wet mass) which has a known value of $4,821 \; kilograms$ (see *Table 1*). Therefore:

$$4,821 = C - 5.053 \times (0)$$

$$C = 4,821 \; kilograms$$

Hence, the mass of the LM ascent stage throughout its trajectory to orbit can be described by the equation:

$$m = 4,821 - 5.053t \tag{3}$$

It is important to note that there are limits to the domain of the equation as the rocket does not have infinite fuel. The mass of the rocket when completely out of fuel (referred to as the dry mass, or m_1, in *Table 1*) is $2,445 \; kilograms$. This is the lower limit of the range for the mass of the LM. Substituting it into equation (3) and solving for time should give the upper limit of the domain of the equation:

$(2{,}445) = 4{,}821 - 5.053t$

$5.053t = 2{,}376$

$t \approx 470.216\ seconds$

This is the maximum amount of time the engine can be active. Since time can only be positive, the lower limit of the domain is 0. The domain of equation 2 is therefore:

$0 \leq t \leq 470.216$

It will be necessary to implement this time constraint when solving the model to ensure the solution is feasible.

Forces

There are two forces to consider for the LM ascent to orbit: gravitational force and thrust. Gravitational force acts towards the combined centre of mass of two bodies and its magnitude can be described by the equation:

$$\left|\underline{F}_g\right| = G\frac{Mm}{r^2} \tag{4}$$

Where $\underline{F}_g$ is the gravitational force vector, G is the gravitational constant[3] equal to 6.674×10^{-11}, M is the mass of the Moon[4] equal to around $7.346 \times 10^{22}\ kilograms$, m is the mass of the rocket and r is the distance between the rocket and the centre of the Moon, described in equation (1). The direction of the force is towards the combined centre of mass of both the Moon and the LM, but since the mass of the Moon is 19 orders of magnitude greater than that of the LM, it is reasonable to assume the force acts directly towards the centre of the Moon. Therefore, the unit vector of the rocket's position (which points directly away from the centre of the moon) gives the direction opposite of which gravitational force is acting. To calculate the gravitational force vector, the magnitude of the gravitational force can therefore be multiplied by the negative of the position unit vector:

$$\underline{F}_g = -\underline{\hat{P}}\,G\frac{Mm}{r^2} \tag{5}$$

[3] Mingsheng Zhan, Xincheng Xie. National Science Review, Volume 7, Issue 12, December 2020, Pages 1803–1817, https://doi.org/10.1093/nsr/nwaa165.
[4] Williams, David R. "Moon Fact Sheet." *NASA*, NASA, 20 Dec. 2021, nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html. Accessed Jan. 7 2022.

A unit vector of position is used rather than a normal vector so that the magnitude of the position does not affect the magnitude of the force. The second force to consider is that created by the engine:

$$|\underline{F_T}| = 15{,}346\ newtons \tag{6}$$

The magnitude of this force is constant, but its direction is dependent on the angle of the rocket. The significance of these forces will be explored in greater depth in the next sections, and a mathematical expression for the rocket's direction will be derived.

Gravitational Acceleration

A force is an influence that can change the motion of a body. At a given instance in time, Newton's second law[5] can be used to describe any force through the following equation:

$$\underline{F} = m\underline{a} \tag{7}$$

Where $\underline{F}$ is a force experienced by a body, m the mass of the body and $\underline{a}$ the acceleration the body experiences. m is a scalar as it only has a magnitude, but $\underline{a}$ is a vector as it has both direction and magnitude. This is an important equation for this exploration as it is the key to relate how the forces experienced by the LM influence its motion. To describe the acceleration caused by gravity, equation (7) can be substituted into equation (5):

$$\underline{F_g} = -\underline{\hat{P}}\, G\frac{Mm}{r^2}$$

$$(m\underline{a_g}) = -\underline{\hat{P}}\, G\frac{Mm}{r^2}$$

$$\underline{a_g} = -\underline{\hat{P}}\, G\frac{M}{r^2}$$

Substituting equation (1) for r:

$$\underline{a_g} = -\underline{\hat{P}}\, GM\frac{1}{(\sqrt{x^2+y^2})^2}$$

$$\underline{a_g} = -\underline{\hat{P}}\, GM\frac{1}{x^2+y^2}$$

To further simplify, the unit vector of position, $\underline{\hat{P}}$, can be expressed by the following equation:

$$\underline{\hat{P}} = \frac{\underline{P}}{|\underline{P}|}$$

5 "Newton's second law of motion." *Physics for the IB Diploma*, by Mark Farrington and K. A. Tsokos, 6th ed., Cambridge University Press, 2014, pp. 67–75.

$$\underline{\hat{P}} = \frac{\underline{P}}{\sqrt{x^2 + y^2}}$$

Substituting this into the equation for gravitational acceleration:

$$\underline{a}_g = -\frac{\underline{P}}{\sqrt{x^2 + y^2}} GM \frac{1}{x^2 + y^2}$$

$$\underline{a}_g = -\underline{P}\, GM \frac{1}{(x^2 + y^2)^{\frac{3}{2}}}$$

And finally, substituting in the values for the gravitational constant, G, and the mass of the Moon, M:

$$\underline{a}_g = -\underline{P}\,(6.674 \times 10^{-11})(7.346 \times 10^{22}) \frac{1}{(x^2 + y^2)^{\frac{3}{2}}}$$

$$\underline{a}_g \approx -\underline{P}\, \frac{4.903 \times 10^{12}}{(x^2 + y^2)^{\frac{3}{2}}} \tag{8}$$

Where $\underline{a}_g$ is the acceleration vector experienced by the LM due to gravity. What this equation shows is that as the rocket gets further from the centre of the Moon, the force it experiences due to gravity tends to zero.

Direction of the Rocket

To get the vector of the acceleration produced by the engine, it is first necessary to be able to mathematically describe the direction the rocket is pointing towards. Let the unit vector of the rocket's position, $\underline{\hat{P}}$, be at an angle Ø from the x-axis and the engine's thrust unit vector, $\underline{\hat{F}_T}$, be at angle θ from the rocket's position unit vector:

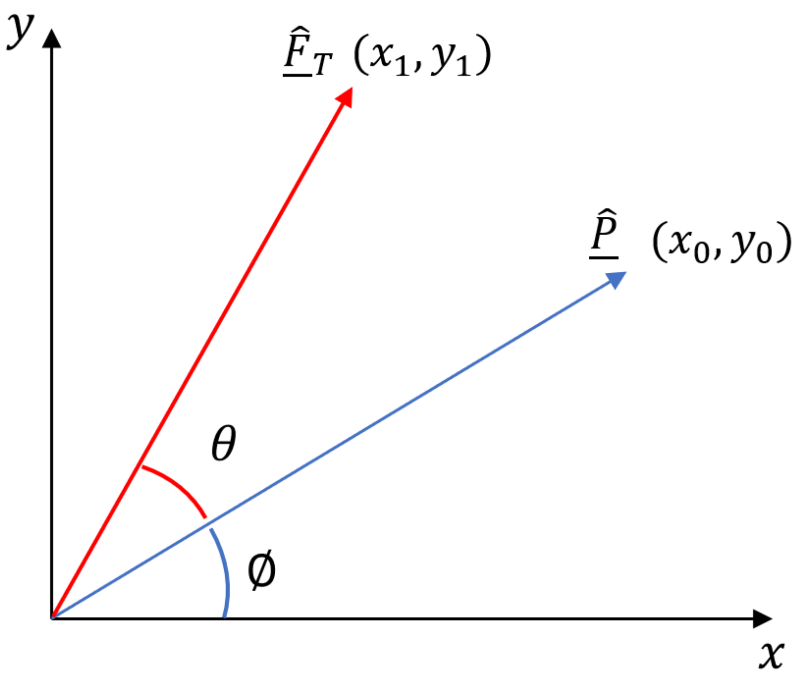


Figure 5 --Angle of the rocket's position unit vector and thrust unit vector.

The coordinates x_o and y_0 can be written as functions of the angle $\emptyset$:

$$x_o = |\underline{\hat{P}}|cos\emptyset$$

$$y_0 = |\underline{\hat{P}}|sin\emptyset$$

Since unit vectors have --by definition-- a magnitude of 1 unit:

$$x_o = cos\emptyset$$

$$y_0 = sin\emptyset$$

Similarly, for the coordinates of x_1 and y_1:

$$x_1 = \cos(\emptyset + \theta)$$

$$y_1 = \sin(\emptyset + \theta)$$

Using the sum of angles trigonometric identity $\cos(\emptyset + \theta) = cos\emptyset cos\theta - sin\emptyset sin\theta$, x_1 can be expressed as:

$$x_1 = cos\emptyset cos\theta - sin\emptyset sin\theta$$

Substituting in x_o for $cos\emptyset$ and y_0 for $sin\emptyset$:

$$x_1 = x_o cos\theta - y_0 sin\theta$$

Likewise, the sum of angles trigonometric identity $\sin(\emptyset + \theta) = sin\emptyset cos\theta + cos\emptyset sin\theta$ can be used to express y_1 as:

$$y_1 = sin\emptyset cos\theta + cos\emptyset sin\theta$$

And substituting in x_o for $cos\emptyset$ and y_0 for $sin\emptyset$:

$$y_1 = y_0 cos\theta + x_o sin\theta$$

Hence, the thrust unit vector, $\underline{\hat{F}}_T$, can be expressed in terms of the angle θ and x_o and y_0:

$$\underline{\hat{F}}_T = \begin{pmatrix} x_o cos\theta - y_0 sin\theta \\ y_0 cos\theta + x_o sin\theta \end{pmatrix}$$

$\begin{pmatrix} x_o \\ y_0 \end{pmatrix}$ is the unit vector of the position of the rocket so to get $\underline{\hat{F}}_T$ in terms of x and y, x_o and y_0 must be multiplied by the magnitude of the position vector $\underline{P}$:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_o \\ y_0 \end{pmatrix} |\underline{P}|$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_o \\ y_0 \end{pmatrix} \sqrt{x^2 + y^2}$$

Therefore $\begin{pmatrix} x_o \\ y_0 \end{pmatrix}$ can be expressed as:

$$\begin{pmatrix} x_o \\ y_0 \end{pmatrix} = \frac{1}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix}$$

Substituting this in for the unit vector of thrust:

$$\underline{\hat{F}}_T = \begin{pmatrix} \frac{x}{\sqrt{x^2+y^2}}cos\theta - \frac{y}{\sqrt{x^2+y^2}}sin\theta \\ \frac{y}{\sqrt{x^2+y^2}}cos\theta + \frac{x}{\sqrt{x^2+y^2}}sin\theta \end{pmatrix}$$

Which simplifies to:

$$\underline{\hat{F}}_T = \frac{1}{\sqrt{x^2+y^2}}\begin{pmatrix} xcos\theta - ysin\theta \\ ycos\theta + xsin\theta \end{pmatrix} \tag{9}$$

This expression is useful since it gives the direction which the rocket is pointing solely based off the position of the rocket, which is a state variable, and the angle θ, which is the control function (the function that will be optimized).

Acceleration Produced by the Engine

To find the magnitude of the acceleration produced by the engine, Newton's second law (equation (7)) can again be used:

$$\underline{F}_T = m\underline{a}_T$$

$$|\underline{F}_T| = m|\underline{a}_T|$$

$$|\underline{a}_T| = \frac{|\underline{F}_T|}{m}$$

Substituting in the magnitude of the thrust force, $|\underline{F}_T|$ (from equation (6)):

$$|\underline{a}_T| = \frac{(15{,}346)}{m}$$

m is the function for the mass of the rocket, which is given by equation *(3)*. Substituting equation *(3)* in for m:

$$|\underline{a}_T| = \frac{15{,}346}{(4{,}821 - 5.053t)}$$

Where $\underline{a}_T$ is the acceleration vector due to thrust. Practically, this equation shows that as the rocket burns fuel, the magnitude of the acceleration experienced by the LM due to its engine increases.

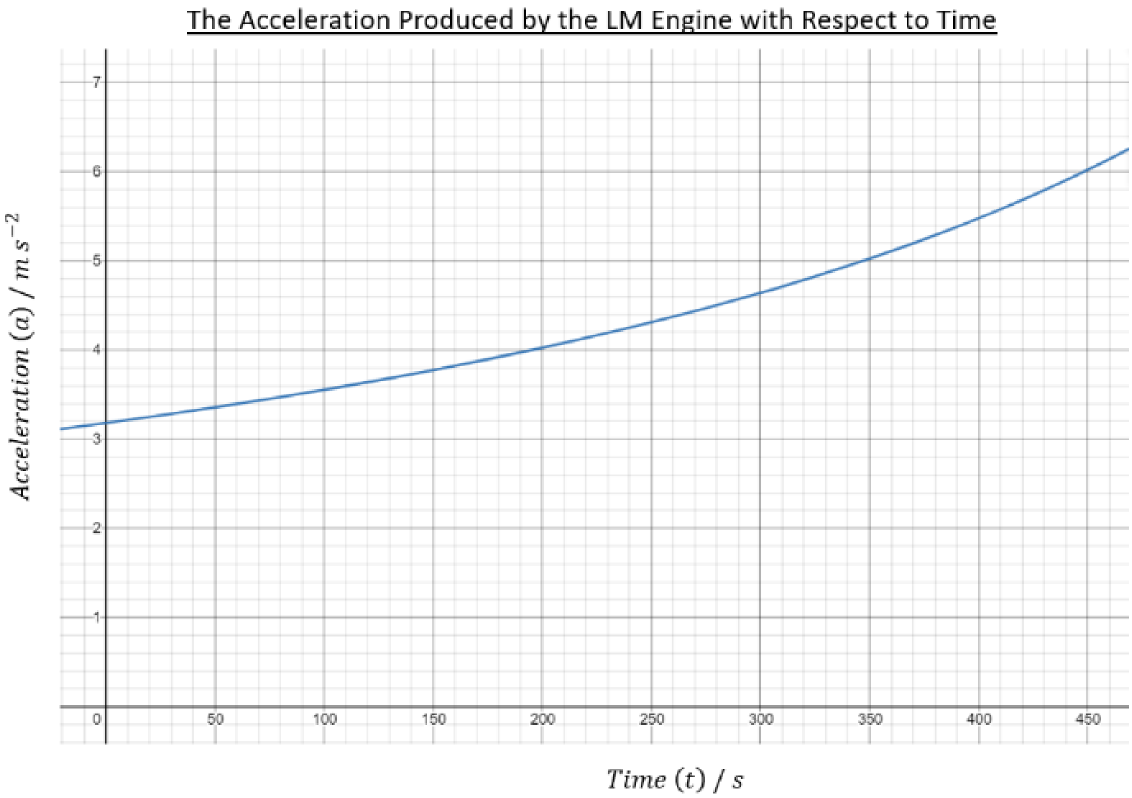


Figure 6

The vector $\underline{a_T}$ is equal to the magnitude of the vector multiplied by a unit vector in its same direction, such as the unit vector of the thrust, $\underline{\hat{F}_T}$:

$$\underline{a_T} = \left|\underline{a_T}\right| \underline{\hat{F}_T}$$

$$\underline{a_T} = \left(\frac{15{,}346}{4{,}821 - 5.053t}\right)\left(\frac{1}{\sqrt{x^2+y^2}}\right)\begin{pmatrix} x\cos\theta - y\sin\theta \\ y\cos\theta + x\sin\theta \end{pmatrix}$$

$$\underline{a_T} = \left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)\begin{pmatrix} x\cos\theta - y\sin\theta \\ y\cos\theta + x\sin\theta \end{pmatrix} \tag{10}$$

Sum of Accelerations

To get the resultant acceleration experienced by the LM at an instance in time, the gravitational acceleration vector and the thrust acceleration vector can be added:

$$\underline{\ddot{P}} = \underline{a_T} + \underline{a_g}$$

$$\underline{\ddot{P}} = \left(\left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)\begin{pmatrix} x\cos\theta - y\sin\theta \\ y\cos\theta + x\sin\theta \end{pmatrix}\right) - \left(\underline{P}\,\frac{4.903\times 10^{12}}{(x^2+y^2)^{\frac{3}{2}}}\right)$$

$$\underline{\ddot{P}} = \begin{pmatrix} \left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)(x\cos\theta - y\sin\theta) - x\left(\frac{4.903 \times 10^{12}}{(x^2+y^2)^{\frac{3}{2}}}\right) \\ \left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)(y\cos\theta + x\sin\theta) - y\left(\frac{4.903 \times 10^{12}}{(x^2+y^2)^{\frac{3}{2}}}\right) \end{pmatrix} \quad (11)$$

$\underline{\ddot{P}}$ is used to represent the resultant acceleration as it makes it clear the variable represents the second derivative of position. This equation is part of the *system dynamics* of the problem as it describes how the rocket's trajectory changes with time. Instead of fully writing equation (11) out as the function of acceleration, $\underline{\ddot{P}}$ will be referenced as:

$\underline{\ddot{P}} = f(x, y, t, \theta)$

Where:

$$f(x, y, t, \theta) = \begin{pmatrix} \left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)(x\cos\theta - y\sin\theta) - x\left(\frac{4.903 \times 10^{12}}{(x^2+y^2)^{\frac{3}{2}}}\right) \\ \left(\frac{15{,}346}{(4{,}821 - 5.053t)\sqrt{x^2+y^2}}\right)(y\cos\theta + x\sin\theta) - y\left(\frac{4.903 \times 10^{12}}{(x^2+y^2)^{\frac{3}{2}}}\right) \end{pmatrix}$$

This is used to represent all the variables $\underline{\ddot{P}}$ is dependent on and will be useful when defining the boundary conditions.

Initial Boundary Conditions

Boundary conditions are necessary to find a numerical solution to the trajectory as they determine the starting and ending state of the system, which will be used by the computer solver to set constraints on the solutions produced, so as to only produce relevant solutions. The initial conditions will be formed as close to the Apollo 11 LM take-off conditions as possible. The Moon is not a perfect sphere, but its radius at its equator and at its poles is known. The Apollo 11 LM landed $0.6875°\ north$ of the equator[6], which is almost on the equator (out of a maximum $north$ coordinate of $90°$, $0.6875°$ represents less than a 1% offset from the equator). Therefore, the radius of the equator will be used as the Moon's radius for the take-off of the rocket, which has a value of $1.7381 \times 10^6\ meters$:[7]

[6] Jones, Eric M, and Ken Glover. "Landing Site Coordinates." *NASA*, NASA, www.hq.nasa.gov/alsj/alsjcoords.html. Accessed Jan. 7 2022.
[7] Williams, David R. "Moon Fact Sheet." *NASA*, NASA, 20 Dec. 2021, nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html. Accessed Jan. 7 2022.

$r_0 = 1.7381 \times 10^6$

The Apollo 11 LM launched into an almost perfectly equatorial orbit[8], meaning it had almost zero degrees of inclination compared to the plane of the Moon's equator. Therefore, it can reasonably be assumed that the final orbit of the rocket is in the same plane as the equator of the Moon. Hence, the two-dimensional plane of the Moon represented in this investigation will essentially be a 'slice' of the Moon at its equator so that the entire trajectory of the rocket can be represented in the plane. Since the equator is assumed to have a constant radius, the exact launch coordinates for the rocket, as long as it touches the circumference of the moon, are arbitrary. The initial position vector was then picked to be:

$$\underline{P}_0 = \begin{pmatrix} 0 \\ 1.7381 \times 10^6 \end{pmatrix}$$

For the velocity of the LM, it starts its trajectory off as a stationary point on the surface of the Moon, so at $t = 0$, it has a velocity of 0:

$$\underline{\dot{P}}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The LM starts off by launching vertically (i.e., pointing directly away from the centre of the Moon), so the thrust vector and position vector initially have the same direction. Therefore, the angle between the vectors also starts off by being 0:

$$\theta_0 = 0$$

At the start of the launch, both the force of gravity and that of the engine are acting on the rocket, so the initial acceleration can be described by:

$$\underline{\ddot{P}}_0 = f(x_0, y_0, t_0, \theta_0)$$

Where x_0 and y_0 are the x and y components of $\underline{\dot{P}}_0$, not to be confused with the x_0 and y_0 used earlier on in the exploration to describe the components of the unit vector of position.

[8] Williams, David R. "Moon Fact Sheet." *NASA*, NASA, 20 Dec. 2021, nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html. Accessed Jan. 7 2022.

Final Boundary Conditions

The Apollo 11 LM did not initially launch into a perfectly circular orbit. Instead, it launched into an elliptical orbit, and then fired another engine to circularize the orbit. This adds a lot of complexity to the problem, so only the first part of the orbit will be considered. This first part of the orbit will be approximated as circular as this greatly increase the simplicity of the problem. The average height of the Apollo 11 LM ascent stage's initial lunar orbit was around $5.311 \times 10^4\ meters$[9] above the surface of the moon. To find the orbital radius, this height can be added to the radius of the moon:

$$r_F = 5.311 \times 10^4 + 1.7381 \times 10^6$$

$$r_F \approx 1.7912 \times 10^6$$

Substituting equation (1) for r_F:

$$\sqrt{x_F^2 + y_F^2} = 1.7912 \times 10^6 \tag{12}$$

Where x_F and y_F are the final components of the LM's position vector and r_F is the desired radius of the final orbit. Note how there are no direct bounds on x_F and y_F. This is because the trajectory should be able to end on any point on the final orbit, so equation (12) ensures that only points on the desired orbit satisfy this condition.

For an object to remain in orbit, it must follow a curved path around the body it is orbiting. The velocity, since it is the derivative of position, will always be tangential to the path of the orbit and therefore always changing. From Newton's second law, it is known that if an object experiences a change of velocity, a force must be acting on it. In the case of an object in orbit, this force is the gravitational force. However, a force that makes an object follow a curved path can also be represented as a centripetal force. The magnitude of this force can be described by the equation:

$$|\underline{F_c}| = \frac{m|\underline{v_t}|^2}{r_c} \tag{13}$$

Where $\underline{F_c}$ is the centripetal force, $\underline{v_t}$ the tangential velocity and r_c the radius of curvature. Radius of curvature is the radius of a curved path at a given point on the path. For a circular orbit, the radius of curvature is simply

[9] Loff, Sarah. "Apollo 11 Mission Overview." *NASA*, NASA, 17 Apr. 2015, www.nasa.gov/mission_pages/apollo/missions/apollo11.html. Accessed Mar. 3 2022.

equal to the radius of the orbit, r. To remain in a circular orbit at that radius, the magnitude of the centripetal force must be equal to the magnitude of the gravitational force[10] (from equation (4)):

$$|\underline{F}_g| = |\underline{F}_c|$$

$$G\frac{Mm}{r^2} = \frac{m|\underline{v}_t|^2}{r}$$

Solving for $|\underline{v}_t|$:

$$|\underline{v}_t|^2 = G\frac{M}{r}$$

$$|\underline{v}_t| = \sqrt{\frac{GM}{r}}$$

Substituting the values for G and M (given on page 7):

$$|\underline{v}_t| = \sqrt{\frac{(6.674 \times 10^{-11})(7.346 \times 10^{22})}{r}}$$

$$|\underline{v}_t| \approx \sqrt{\frac{4.903 \times 10^{12}}{r}}$$

The Tangential Velocity Required to Orbit the Moon at a Given Radius

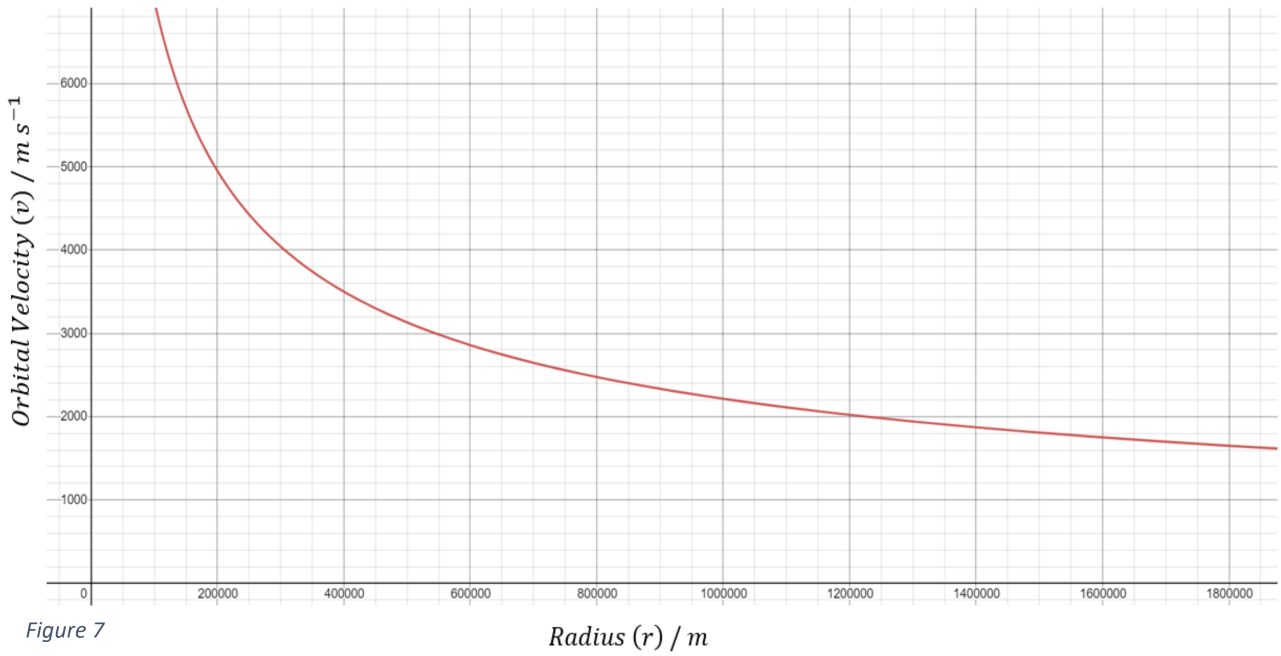


Figure 7

[10] "Circular Motion." *Physics for the IB Diploma*, by Mark Farrington and K. A. Tsokos, 6th ed., Cambridge University Press, 2014, pp. 262–263.

The magnitude of the final velocity of the LM will therefore have to be equal to $|\underline{v}_t|$, in order to ensure the LM is going at the right speed to remain in a circular orbit:

$$|\dot{\underline{P}}_F| = |\underline{v}_t|$$

$$|\dot{\underline{P}}_F| = \sqrt{\frac{4.903 \times 10^{12}}{r_F}}$$

$$|\dot{\underline{P}}_F| = \sqrt{\frac{4.903 \times 10^{12}}{\sqrt{x_F^2 + y_F^2}}}$$

Using the standard magnitude of vectors formula:

$$\sqrt{\dot{x}_F^2 + \dot{y}_F^2} = \sqrt{\frac{4.903 \times 10^{12}}{\sqrt{x_F^2 + y_F^2}}}$$

$$\dot{x}_F^2 + \dot{y}_F^2 = \frac{4.903 \times 10^{12}}{\sqrt{x_F^2 + y_F^2}} \qquad (13)$$

Where $\dot{x}_F$ and $\dot{y}_F$ are the components of the final velocity vector $\dot{\underline{P}}_F$. Since the velocity is tangential to the orbital path, and the orbital path is circular, the velocity is by definition perpendicular to the radius of the orbit:

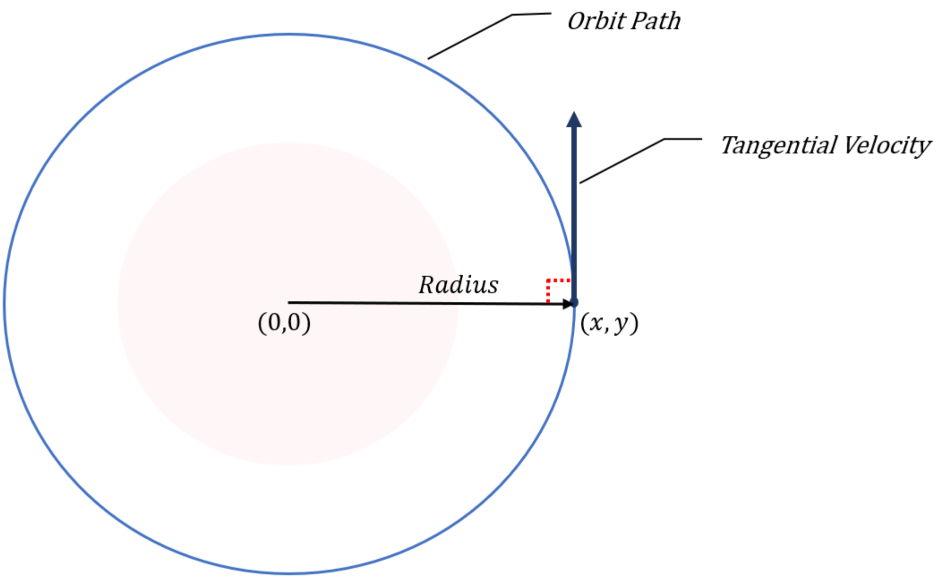


Figure 8 --The velocity of a spacecraft in a circular orbit is always perpendicular to the radius of that orbit.

The radius of the orbit is just the rocket's final position vector, $\underline{P}_F$. To ensure the final velocity is perpendicular to the orbital radius, we take the dot product of the rocket's final velocity vector and final position vector. The dot product of two perpendicular vectors should be equal to zero:

$\underline{P}_F \cdot \underline{\dot{P}}_F = 0$

$\begin{pmatrix} x_F \\ y_F \end{pmatrix} \cdot \begin{pmatrix} \dot{x}_F \\ \dot{y}_F \end{pmatrix} = 0$

$x_F \dot{x}_F + y_F \dot{y}_F = 0$

This constraint ensures that the LM's velocity vector is in the right direction. The constraint for the duration of the launch, t_F, will be the domain of t in equation (3) (given at the top of page 7) as this represents the domain of time for which the LM has fuel:

$0 \leq t_F \ \leq 470.216$

The final angle of the rocket, θ_F, will be left unbound as there is no particular value of θ_F needed for the rocket to successfully enter an orbit. The final acceleration can hence be described by:

$\underline{\ddot{P}}_F = f(x_F, y_F, t_F, \theta_F)$

The Objective Function

The objective is to simply minimize the fuel used during the rocket launch. The total fuel used during the trajectory is the integral of the mass flow rate, $\dot{m}$, over the entire duration of the launch:

$m_{fuel} = \ \int_0^{t_F} \dot{m} \, dt$

$m_{fuel} = \ \int_0^{t_F} 5.053 \, dt$

$m_{fuel} = \ [5.053t]_0^{t_F}$

$m_{fuel} = \ 5.053 t_F$

Therefore, the fuel used is directly proportional to the duration of the launch, t_F. Hence, fuel usage can be minimized by minimizing time. This can be expressed as: $\min\,(t_F)$

Solving the Model

Describing the Problem

To be able to find a solution to the rocket's trajectory, an already existing NLP solver named *APOPT* was used. To interface with this solver, Python, with the library *GEKKO*, was used. The problem had to then be described in Python and passed on to *APOPT* in order to be numerically solved. The problem was formulated as such:

$t \in [0, 470.216]$

$\theta \ \in [-\pi, \pi\ ]\ where\ \theta\ is\ in\ radians$

$\underline{P} = \begin{pmatrix} x \\ y \end{pmatrix}\ where\ x, y \in \mathbb{R}$

$\underline{\dot{P}} = \frac{d}{dt}\ \underline{P}$

$\underline{\ddot{P}} = \frac{d}{dt}\ \underline{\dot{P}}$

Subject to the constraints:

$$\underline{\ddot{P}} = \begin{pmatrix} \left(\frac{15{,}346}{(4{,}821 - \ 5.053t)\sqrt{x^2 + y^2}}\right)(xcos\theta - ysin\theta) - x\left(\frac{4.903 \times 10^{12}}{(x^2 + y^2)^{\frac{3}{2}}}\right) \\ \left(\frac{15{,}346}{(4{,}821 - \ 5.053t)\sqrt{x^2 + y^2}}\right)(ycos\theta + xsin\theta) - y\left(\frac{4.903 \times 10^{12}}{(x^2 + y^2)^{\frac{3}{2}}}\right) \end{pmatrix}$$ *System dynamics*

$$\underline{P}_0 = \begin{pmatrix} 0 \\ 1.7381\ \times\ 10^6 \end{pmatrix}$$ *Initial position*

$$\underline{\dot{P}}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$ *Initial velocity*

$\theta_0 = 0$ *Initial angle*

$t_F \in t$ *Final time*

$\sqrt{x_F^2 + y_F^2} = 1.7912\ \times\ 10^6$ *Final radius length*

$\dot{x}_F^2 + \dot{y}_F^2 = \frac{4.903 \times 10^{12}}{\sqrt{x_F^2 + y_F^2}}$ *Final magnitude of velocity*

$x_F\dot{x}_F + y_F\dot{y}_F = 0$ *Final direction of velocity*

With objective: $\min\ (t_F)$ *The objective is to minimize the time it takes to reach orbit*

The Python program that was written with this depiction of the problem is given in the appendix.

The Solution

The final time computed for the trajectory duration was $435.298\ seconds$. The numerical solver was specified (through the code) to be accurate up to three decimal places. This level of accuracy was used since the values used to create the model have mostly been accurate to three or more decimal places. From NASA's official website, the real duration of the Apollo 11 LM ascent stage launch was $435\ seconds$[11]. This value was given with a precision of three significant figures. At this level of precision, the computed trajectory duration is 100% accurate to the real duration of the LM ascent launch. This is an astounding level of accuracy and goes to show just how well-suited mathematical models in physics can be at describing reality.

To gain a further understanding of the answer computed, extra code was produced to create plots relating some of the variables. The variation of the angle θ with respect to time was plotted:

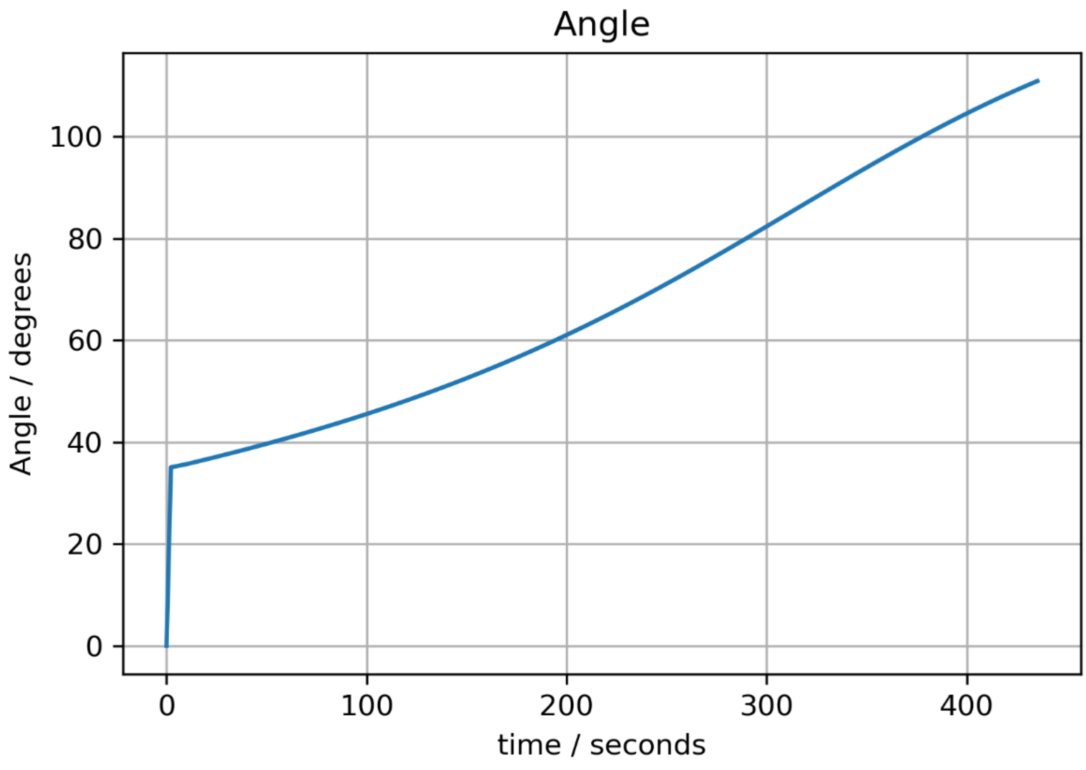


Figure 9 --The angle of the LM as it ascends to orbit.

It can be noticed that the angle changes smoothly and continuously throughout the entire launch, which is a realistic way for the angle to change, except for the beginning of the launch, were the LM abruptly turns from 0 to $37\ degrees$. This is not possible in reality, as the LM cannot spontaneously spin about itself, and a further

[11]Loff, Sarah. "Apollo 11 Mission Overview." *NASA*, NASA, 17 Apr. 2015, www.nasa.gov/mission_pages/apollo/missions/apollo11.html. Accessed Mar. 3 2022.

extension of this investigation could be done to calculate and implement a limit on the rotation rate of the LM. This, however, seemed to be a very minor issue as it did not have a noticeable effect on the accuracy of the solution computed.

Further code was then developed to model the orbit of the LM. This was done to ensure that the computed rocket launch actually fulfilled the given success criterion of putting the LM in a circular orbit around the moon. A graph of the LM launch, along with part of the computed orbit, was then produced:

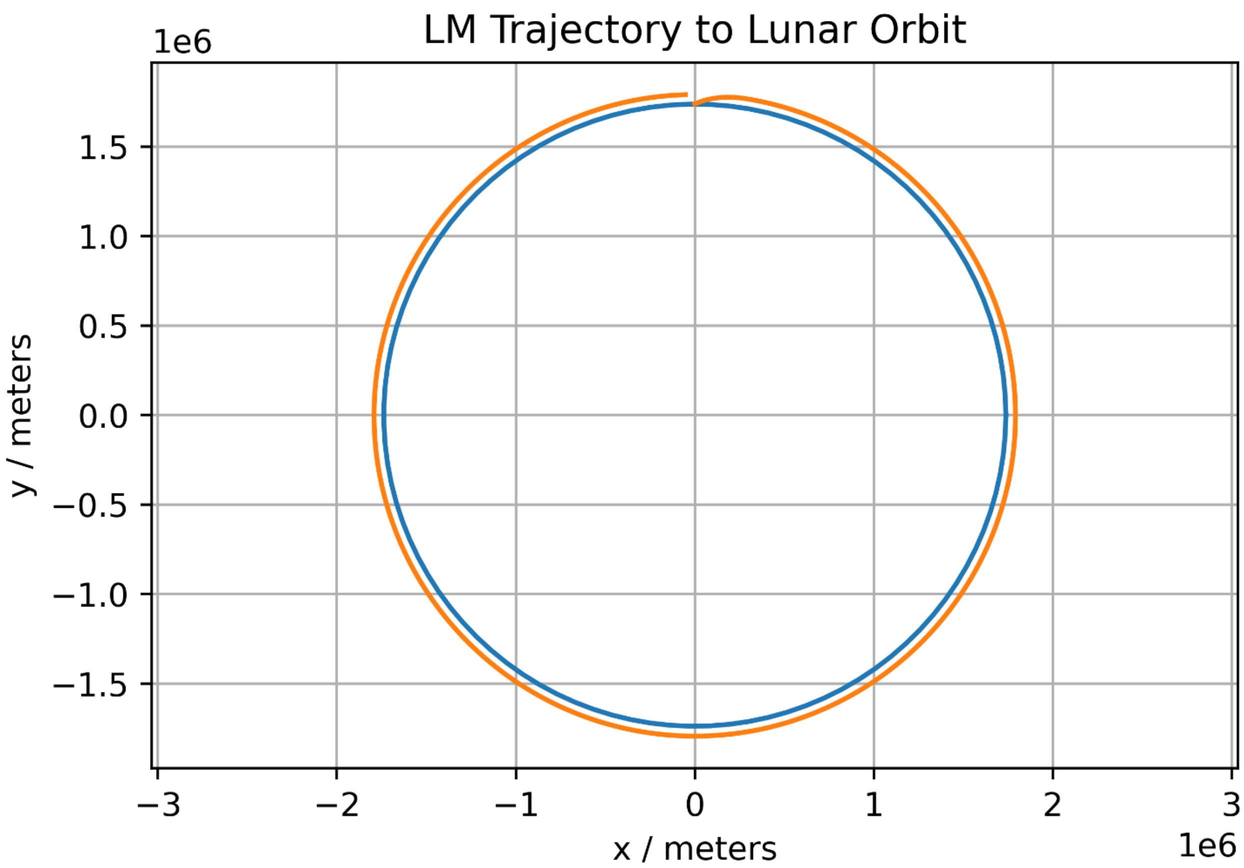


Figure 10 --The trajectory of the LM ascending and completing the majority of an orbit.

The blue circle represents the circumference of the moon (at its equator) and the orange path shows the rocket's trajectory. The graph shows how the path of the LM starts off on the moon's surface and then evolves into a seemingly circular orbit. This confirms the validity of the computed solution as the graph clearly shows the LM successfully enter a lunar orbit. The code which models the orbit is given in the appendix as the second part of the main solver and further graphs produced are also given.

Conclusion

Throughout the investigation, I was able to formulate the equations that describes the dynamics of the Apollo LM as it launches to lunar orbit. A set of constraints that the spacecraft would need to satisfy to ensure that it follows a trajectory that starts on the surface of the Moon and ends in a desired orbit were also successfully formulated. The problem was then expressed in code and passed to an NLP solver, with objective to optimise the angle of the rocket throughout its trajectory as to minimize the amount of time it takes to reach orbit and hence minimize fuel costs. The result was then compared to the real duration of the Apollo 11 LM launch and shows astonishing accuracy as the computed optimal trajectory had exactly the same duration as the real Apollo 11 LM launch, correct to three significant figures. This demonstrates that the rocket trajectory was successfully modelled and optimised, fulfilling the aim of the investigation. Through this investigation, I've learned a lot about how abstract mathematical concepts such as differential equations, vectors and trigonometric functions can all be used to model complex real-world phenomena to a degree of incredible precision.

Bibliography

List of Tables and Figures

Table 1 – Data From:

Hooper, J.C. "Performance Analysis of the Ascent Propulsion System of the Apollo Spacecraft." NASA Lyndon B. Johnson Space Center, 1 Dec. 1973. Accessed Nov. 16 2022.

Williams, David R. "NASA - NSSDCA - Spacecraft - Details." *Apollo 11 Lunar Module / EASEP*, NASA, 7 Jan. 2022, nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1969-059C. Accessed Jan. 10 2022.

Figure 1 – Image from: Loff, Sarah. "SpaceX's Falcon 9 Rocket Launches Dragon to the Space Station." *NASA*, NASA, 29 June 2018, www.nasa.gov/image-feature/spacexs-falcon-9-rocket-launches-dragon-to-the-international-space-station. Accessed Jan. 11 2022

Figure 2 – Self-made using PowerPoint

Figure 3 – Self-made using PowerPoint

Figure 4 – Self-made using PowerPoint, with lunar module clipart from:

Smith, Taylor. "Apollo Lunar Module Clipart PNG Icons." *Iconspng*, Iconspng.com, 6 June 2019, www.iconspng.com/image/132552/apollo-lunar-module-clipart. Accessed Jan. 7 2022

Figure 5 – Self-made using PowerPoint

Figure 6 – Self-made using Desmos (online graphing calculator)

Figure 7 – Self-made using Desmos (online graphing calculator)

Figure 8 – Self-made using PowerPoint

Figure 9 – Self-made using Python with Matplotlib library

Figure 10 – Self-made using Python with Matplotlib library

List of Equations

Equation (1) – Standard equation for magnitude of a two-dimensional vector

Equation (2) – Taken from: Hall, Nancy. "Specific Impulse." *NASA*, NASA, 13 May 2021, www.grc.nasa.gov/www/k-12/airplane/specimp.html. Accessed Jan. 11 2022.

Equation (3) – Self-derived

Equation (4) – Taken from: "6.2 The law of gravitation." *Physics for the IB Diploma*, by Mark Farrington and K. A. Tsokos, 6th ed., Cambridge University Press, 2014, pp. 259–262.

Equation (5) – Self-derived

Equation (6) – Self-derived

Equation (7) – Self-derived (with value from *Table 1* used)

Equation (8) – Self-derived

Equation (9) – Self-derived

Equation (10) – Self-derived

Equation (11) – Self-derived

Equation (12) – Self-derived

Equation (13) – Taken from: "Centripetal Forces." *Physics for the IB Diploma*, by Mark Farrington and K. A. Tsokos, 6th ed., Cambridge University Press, 2014, pp. 253–256.

References

"Apollo 11 (AS-506)." Apollo 11 (AS-506) | National Air and Space Museum, Smithsonian Air and Space Museum, airandspace.si.edu/explore-and-learn/topics/apollo/apollo-program/landing-missions/apollo11.cfm. Accessed Jan. 7 2022.

C., Hooper J. "Performance Analysis of the Ascent Propulsion System of the Apollo Spacecraft." NASA Lyndon B. Johnson Space Center , Dec. 1 1973.

Hall, Nancy. "Specific Impulse." *NASA*, NASA, 13 May 2021, www.grc.nasa.gov/www/k-12/airplane/specimp.html. Accessed Jan. 11 2022.

Jones, Eric M, and Ken Glover. "Landing Site Coordinates." NASA, NASA, www.hq.nasa.gov/alsj/alsjcoords.html. Accessed Jan. 7 2022.

Kelly, Matthew. "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation." SIAM Review, vol. 59, no. 4, 2017, pp. 849–904., doi:10.1137/16m1062569.

Loff, Sarah. "Apollo 11 Mission Overview." NASA, NASA, 17 Apr. 2015, www.nasa.gov/mission_pages/apollo/missions/apollo11.html. Accessed Mar. 3 2022.

Loff, Sarah. "SpaceX's Falcon 9 Rocket Launches Dragon to the Space Station." *NASA*, NASA, 29 June 2018, www.nasa.gov/image-feature/spacexs-falcon-9-rocket-launches-dragon-to-the-international-space-station. Accessed Jan. 11 2022.

Mingsheng Zhan, Xincheng Xie. National Science Review, Volume 7, Issue 12, December 2020, Pages 1803–1817, https://doi.org/10.1093/nsr/nwaa165.

Tsokos, K. A., and Mark Farrington. *Physics for the IB Diploma*. 6th ed., Cambridge University Press, 2014.

Williams, David R. "Moon Fact Sheet." NASA, NASA, 20 Dec. 2021, nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html. Accessed Jan. 7 2022.

Williams, David R. "NASA - NSSDCA - Spacecraft - Details." *Apollo 11 Lunar Module / EASEP*, NASA, 7 Jan. 2022, nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1969-059C. Accessed Jan. 10 2022.

--Thanks to Danylo Malyuta for offering guidance on formulating the problem as a solvable NLP problem--

Appendix

Python Trajectory Solver Program:

```
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO
from gekko import *

# create GEKKO model
m = GEKKO()

nt = 200 #number of timesteps
# scale 0-1 time with tf
m.time = np.linspace(0,1,nt)

# options
m.options.NODES = 2 #The number of collocation points between each timestep
m.options.SOLVER = 3 #1 = APOPT, 3 = IPOPT
m.options.IMODE = 6  #Tells gekko the problem is an optimal control problem
m.options.MAX_ITER = 20000
m.options.MV_TYPE = 0 #How the interpolation between Manipulated variables (MVs) is done
#m.options.COLDSTART= 2 #Should help find bad constraint
m.options.OTOL = 1e-3 #allows for margins of error in solution, default of 1e-6
m.options.RTOL = 1e-3 #same as above
m.options.DIAGLEVEL = 0

# Parameters of the problem
G =  m.Const(6.674*10**(-11), name='G') #Gravitational Constant
M =  m.Const(7.346*10**(22), name='M')  #Mass of the Moon
R0 =  m.Const(1738100, name='R0')       #Radius of the lunar surface

Ft =  m.Const(15346, name='Ft')          #thrust force of engine
M0 =  m.Const(4821, name='M0')            #Wet Mass of rocket
M_dot =  m.Const(5.053, name='M_dot')       #Mass flow rate of propellant

Rfmin = m.Const(53108.4, name ='Rfmin') #final height of orbit
```

```

orbital_v = ((6.674*10**(-11)*7.346*10**(22))/(1738100+53108.4))**(1/2)
#print(orbital_v)
mflow = 5.053/2376

# final time
final_time = 470 #470
tf = m.FV(value=0,lb=0,ub=1) #FV is a fixed value variable
tf.STATUS = 1 #STATUS = 1 means the optimzer can minimize the function

#Mass
mass = m.Var(value=0, lb = 0, ub = 1, name='mass') #lb, ub are upper and lower bounds respectively

# Position variables
y = m.Var(name='y', value = 0)
ydot = m.Var( name='ydot')
ydoubledot = m.Var(name='ydoubledot')

x = m.Var(name='x', value = 0)
xdot = m.Var( name='xdot')
xdoubledot = m.Var(name='xdoubledot')

```

```python
#Distance scale:
Scalar = m.Const(53108.4, name = 'distance Scale')
mass_scalar = m.Const(2576, name = 'mass Scale')

angle = m.MV(name='angle', value = 0, lb = 0, ub = np.pi/3)
#angle.DPRED
angle.STATUS = 1 #Allows computer to change theta
angle.DCOST = 1e-5 #Adds a very small cost to changes in theta
angle.REQONCTRL = 3 #tells solver whether to change MVs or run as simulator

# differential equations scaled
m.Equation(y.dt()==tf*ydot*final_time) #Expression for y velocity
m.Equation(ydot.dt() == tf*ydoubledot*final_time) #Expression for y acceleration


m.Equation(x.dt()==tf*xdot*final_time) #Expression for x velocity
m.Equation(xdot.dt() == tf*xdoubledot*final_time) #Expression for x acceleration

m.Equation(mass.dt() == mflow*final_time*tf) #Expression for mass

#system dynamics

m.Equation(ydoubledot == ( ( (Ft/((M0-mass_scalar*mass)*((x*Scalar)**2 + (y*Scalar+R0)**2)**(1/2)))*\
                            ((y*Scalar+R0)*m.cos(3*angle)+(x*Scalar)*m.sin(3*angle)) )\
            - (y*Scalar+R0)*(G*M/(((x*Scalar)**2 + (y*Scalar+R0)**2)**(3/2))) )/ Scalar\
            )


m.Equation(xdoubledot == ( ( (Ft/((M0-mass_scalar*mass)*((x*Scalar)**2 + (y*Scalar+R0)**2)**(1/2)))*\
                            ((x*Scalar)*m.cos(3*angle)-(y*Scalar+R0)*m.sin(3*angle)) )\
            - (x*Scalar)*(G*M/(((x*Scalar)**2 + (y*Scalar+R0)**2)**(3/2)))))/ Scalar\
            )

```

```python
#Initial Boundary Conditions:

m.fix(y, pos=0,val=0)       #Initial y position
m.fix(x, pos=0,val=0)       #Initial x position
m.fix(ydot, pos=0,val=0)    #Initial y velocity
m.fix(xdot, pos=0,val=0)    #Initial x velocity

m.fix(angle, pos=0, val=0)  #Initial angle
m.fix(mass, pos=0,val=0) #Initial mass


#Final Boundary Conditions

#Creates a list that satisfies final condition at all time except at final time for constraint_1
constraint_1 = np.full(nt, Rfmin+R0+1)
constraint_1 [-1] = 0
final_radius = m.Param(value = constraint_1 )

m.Equation(((y+R0/Scalar)**2+(x)**2)**(1/2)+final_radius >= ((R0+Scalar)/Scalar))#ensures final radius is greater or equal to rfmin

#Creates a list that satisfies final condition at all time except at final time for constraint_2
constraint_2 = np.full(nt, 0)
constraint_2 [-1] = 1
final_velocity = m.Param(value = constraint_2 )

m.Equation((xdot**2 + ydot**2) >= (orbital_v/Scalar)**2*final_velocity)#ensures final velocity is orbital

# dot product of velocity and position should be zero
m.Equation(((y*Scalar+R0)*(ydot*Scalar)+(x*Scalar)*(xdot*Scalar))*final_velocity == 0)
```

```python
m.Minimize(tf)

#Solving the problem
try:
    m.solve(disp=True)    # solve
except:
    print('Not successful')
    from gekko.apm import get_file
    print(m._server)
    print(m._model_name)
    f = get_file(m._server,m._model_name,'infeasibilities.txt')
    f = f.decode().replace('\r','')
    with open('infeasibilities.txt', 'w') as fl:
        fl.write(str(f))



print('Optimal Solution (final time): ' + str(tf.value[0]))



# scaled time
ts = m.time * tf.value[0]
pos_factor = 53108.4
pos_offset = 1738100

print('final y',y.value[-1]*pos_factor)
print('final x', x.value[-1]*-pos_factor)
print('final ydot',ydot.value[-1]*pos_factor)
print('final xdot', xdot.value[-1]*-pos_factor)
print('final ydoubledot',ydoubledot.value[-1]*pos_factor)
print('final xdoubledot', xdoubledot.value[-1]*-pos_factor)
print('final time', tf.value[0]*final_time)
```

```python
#SECOND PART OF THE CODE: produces graphs and computes the orbit with the trajectory information

y_pos_list = [0]*len(x.value)
x_pos_list = [0]*len(x.value)
theta_list = [0]*len(x.value)
y_v_list = np.zeros(len(x.value))
x_v_list = np.zeros(len(x.value))
y_a_list = np.zeros(len(x.value))
x_a_list = np.zeros(len(x.value))

for i in range(len(x.value)):
    x_pos_list[i] = -x.value[i]*pos_factor
    y_pos_list[i] = y.value[i]*pos_factor+pos_offset
    y_v_list[i] = ydot.value[i]*pos_factor
    x_v_list[i] = -xdot.value[i]*pos_factor
    y_a_list[i] = ydoubledot.value[i]*pos_factor
    x_a_list[i] = -xdoubledot.value[i]*pos_factor
    theta_list[i] = 3*angle.value[i]*(360/(2*np.pi))

#Constants
Gs = 6.67*10**-11
m_1 = 2000
m_2 = 7.346*10**(22)
f_y = y.value[-1]*pos_factor+pos_offset
f_x = x.value[-1]*-pos_factor
f_ydot = ydot.value[-1]*pos_factor
f_xdot = xdot.value[-1]*-pos_factor
```

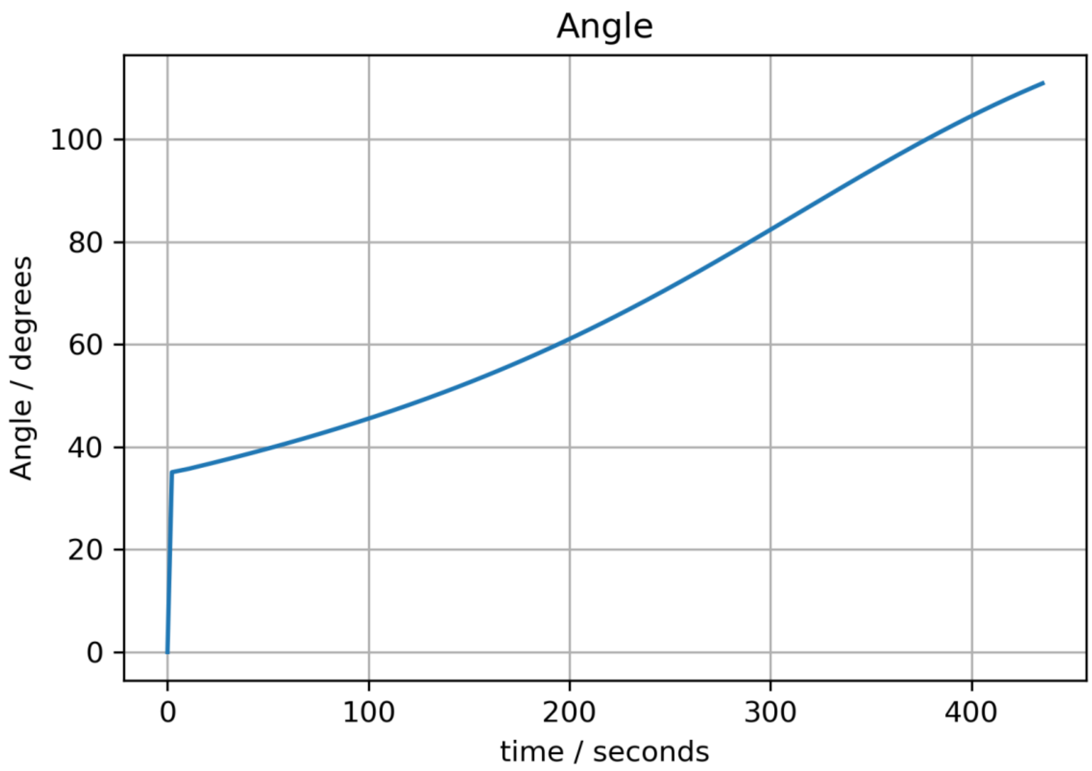
```python
#Initial Conditions
radius = 1738100

position_1_0 = [f_x, f_y] #on surface
position_2_0 = [0,0] #yeeted out
position_1_dot_0 = [f_xdot, f_ydot]
position_2_dot_0 = [0,0]
#Defenition of ODE
def get_position_1_double_dot(pos_1, pos_2):
    distance =np.sqrt((pos_2[0] - pos_1[0])**2+(pos_2[1] - pos_1[1])**2)
    #print(distance)
    x_1_direction = pos_2[0] - pos_1[0]
    y_1_direction = pos_2[1] - pos_1[1]
    x_1_double_dot = Gs*m_2*(x_1_direction/distance)*(1/(distance**2))
    y_1_double_dot = Gs*m_2*(y_1_direction/distance)*(1/(distance**2))
    #if distance < 1737400:
        #print("ground hit")

    return [x_1_double_dot,y_1_double_dot]

#Solution to DE
def position(t):
    pos_1_x_list = []
    pos_1_y_list = []
    pos_1_double_dot_list = []

    position_1 = position_1_0
    position_2 = position_2_0
    position_1_dot = position_1_dot_0
    position_2_dot = position_2_dot_0

    delta_t = 0.001 #time step
    time_list = np.arange(0,t,delta_t)
    for time in np.arange(0,t,delta_t):
        position_1_double_dot = get_position_1_double_dot(position_1,position_2)
        pos_1_double_dot_list.append(position_1_double_dot)

        position_1[0] += position_1_dot[0]*delta_t
        position_1[1] += position_1_dot[1]*delta_t
        position_1_dot[0] += position_1_double_dot[0]*delta_t
        position_1_dot[1] += position_1_double_dot[1]*delta_t
        pos_1_x_list.append(position_1[0])
        pos_1_y_list.append(position_1[1])
    return [time_list, pos_1_x_list, pos_1_y_list]

test = position(6600)

theta = np.linspace(0, 2*np.pi, 100)
a = radius*np.cos(theta)
b = radius*np.sin(theta)

pos_1_x_array = np.array(test[1])
pos_1_y_array = np.array(test[2])

#plt.style.use('ggplot')
x_pos = np.concatenate((x_pos_list, pos_1_x_array))
y_pos = np.concatenate((y_pos_list, pos_1_y_array))


plt1 = plt.figure()
ax = plt1.add_subplot()
plt.plot(a, b)
plt.plot(x_pos , y_pos)
plt.title("LM Tradjectory to Lunar Orbit")
plt.xlabel("x / meters")
plt.ylabel("y / meters")
ax.set_aspect(1,adjustable='datalim')
ax.grid()
plt.savefig('assembled.png', dpi=500)
```

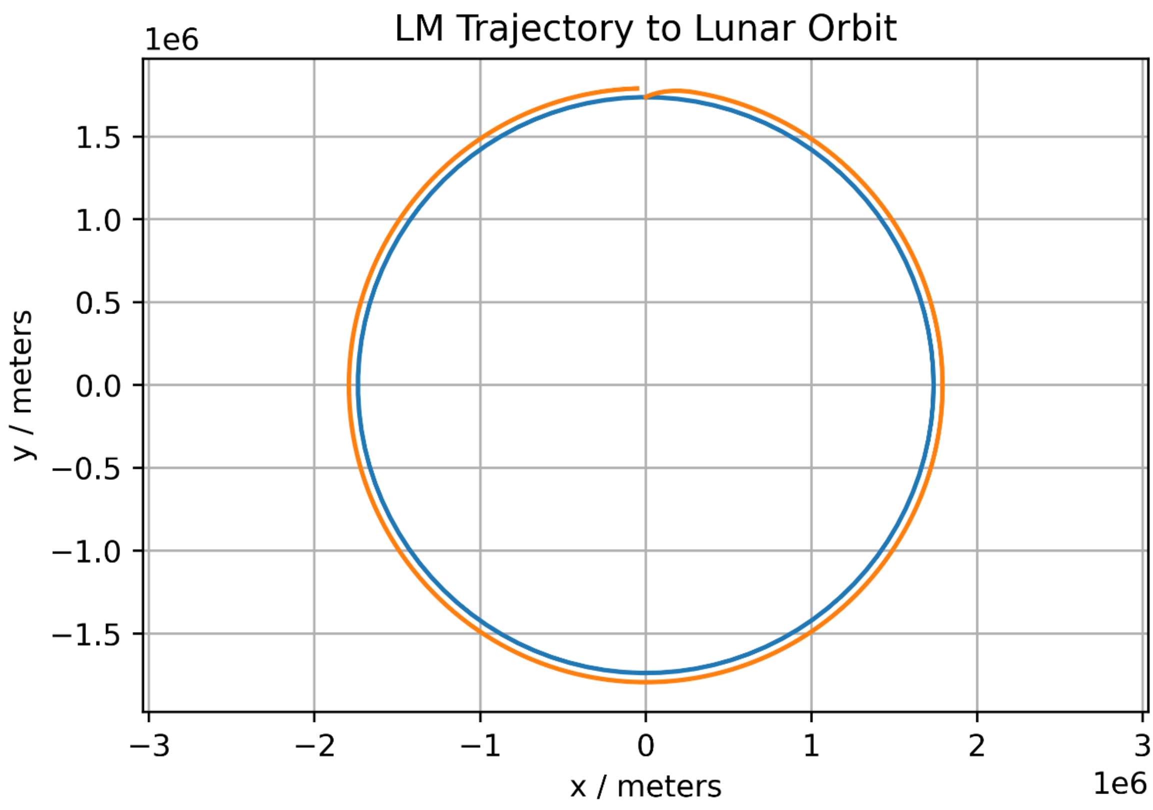
```python
plt2 = plt.figure()
ax = plt2.add_subplot()
plt.plot(470*ts,theta_list)
ax.set_title('Angle')
plt.ylabel('Angle / degrees')
ax.set_xlabel('time / seconds')
ax.grid()
plt.savefig('angle.png', dpi=300)

plt3 = plt.figure()
ax = plt3.add_subplot()
plt.plot(x_pos_list,y_pos_list)
ax.set_title('LM Trajectory Path')
plt.ylabel('y / meters')
ax.set_xlabel('x / meters')
ax.set(xlim=(0, 300000), ylim=(pos_offset, pos_offset+50000))
ax.set_aspect('equal')
ax.grid()
plt.savefig('takeoff_context.png', dpi=300)

plt4 = plt.figure()
ax = plt4.add_subplot()
plt.plot(x_v_list,y_v_list)
ax.set_title('LM X and Y Velocity')
plt.ylabel('y velocity / ms-1')
ax.set_xlabel('x velocity / ms-1')
ax.set_aspect('equal')
ax.grid()
plt.savefig('takeoff_velocity.png', dpi=500)

plt5 = plt.figure()
ax = plt5.add_subplot()
plt.plot(x_a_list,y_a_list)
ax.set_title('LM X and Y Acceleration')
plt.ylabel('y acceleration / ms-2')
ax.set_xlabel('x acceleration / ms-2')
ax.set_aspect('equal')
ax.grid()
plt.savefig('takeoff_acceleration.png', dpi=500)

plt.show()
```

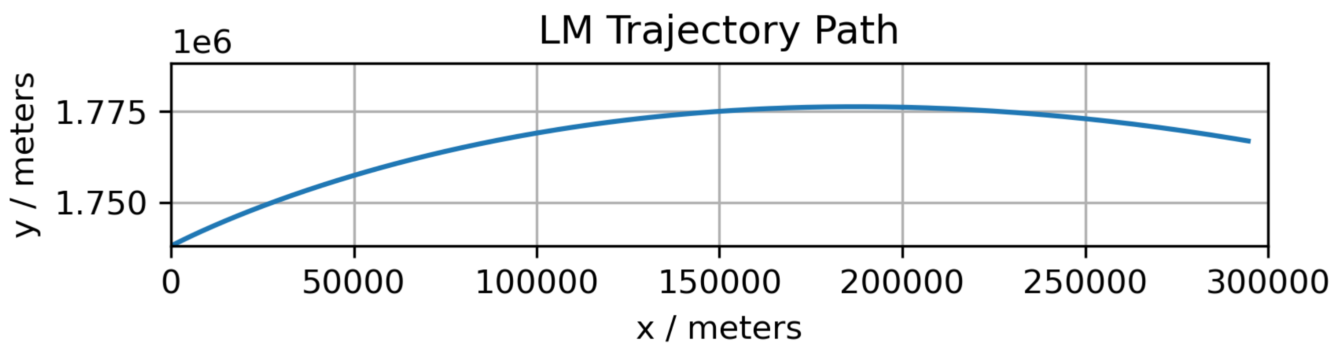
Results from running the code:

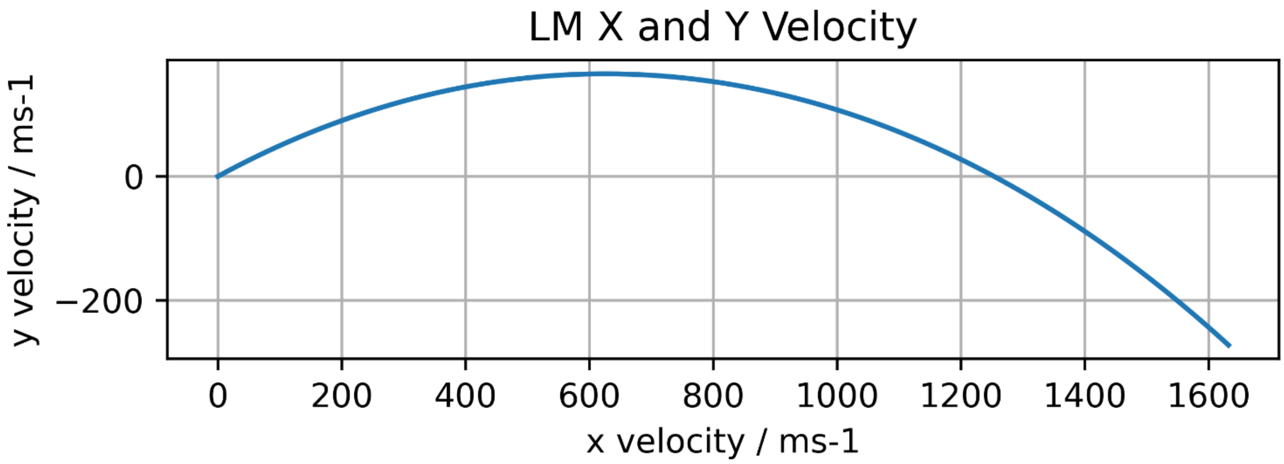
```
Optimal Solution (final time): 0.92616537474
final y 28716.160349635127
final x 294598.36483519967
final ydot -272.0993356840796
final xdot 1631.8810994353596
final ydoubledot -4.689807224722499
final xdoubledot 5.184464927832862
final time 435.29772612780005
```

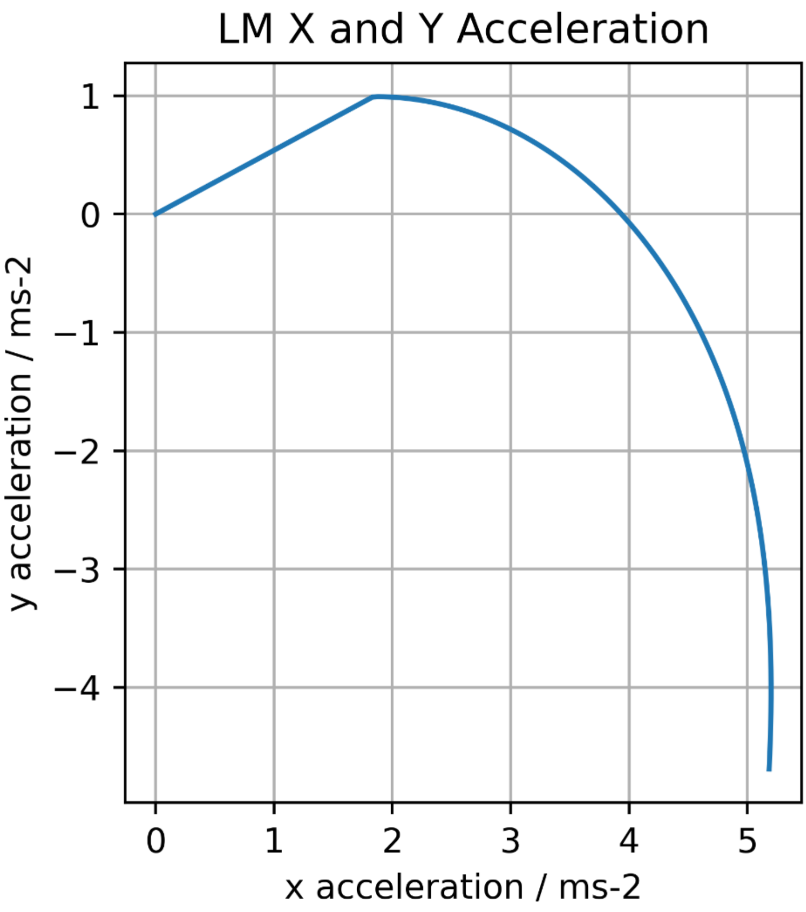
Graphs produced:






LM Trajectory Path
1e6
1.775
1.750
y / meters
0
50000
100000
150000
200000
250000
300000
x / meters


LM X and Y Velocity
0
−200
y velocity / ms-1
0
200
400
600
800
1000
1200
1400
1600
x velocity / ms-1


LM X and Y Acceleration
1
0
−1
−2
−3
−4
y acceleration / ms-2
0
1
2
3
4
5
x acceleration / ms-2