Extended Essay

When and How Could a Flight to Mars Be Performed?

Candidate code: jxp781

Word Count: 3947

Contents

1 Introduction

In the 1950s, an international race began to fly humans to the moon. Now that mankind has succeeded in stepping foot on the Earth's only satellite, the achievement to transport humans to interplanetary destinations is the next milestone. With the technological advancements brought to humans by means of computers, more advanced spaceships and thrusters, this is no longer a distant goal but a matter of resources and funding. This paper aims to present a path of flight to Mars, calculate fuel requirements and finally simulate the launch.

The goal of this essay will be to simulate a rocket launching from low earth orbit (LEO, approximately 1000 kilometers over Earth's surface), flying through the solar system and entering a stable orbit around Mars (ideally low mars orbit, LMO, also around 1000 kilometers). The flight will be based on the patched conic approximation, a three stage approach to interplanetary travel. Firstly, the rocket will exit the Earth's gravitational field in a hyperbolic injection orbit in such a way that it has the desired speed and direction to enter the next stage. Then, the rocket will transfer between the two planets on its own orbit around the Sun. This will be the longest part of the flight and the bulk of the simulation. Finally, the rocket will enter Mars's gravitational field and perform a hyperbolic exit (or capture) orbit which will simply be the inverse of the injection orbit. Once the rocket reaches the nearest point to Mars on its trajectory, the thrusters will be fired to enter the parking orbit around Mars, at which point the mission will be complete.

Since I lack the funds and knowledge to build an actual rocket, my experiments will be done using a computer simulation written in the programming languages C++ and Python using JupyterLab to create and download plots.

2 Prerequisites and Assumptions

2.1 Patched Conic Approximation

As mentioned above, the patched conic approximation will be utilized to provide an estimate of the flight path. This approach simplifies the interplanetary travel because each stage can be represented as a two-body problem. It is worth underlining that this is a false assumption due to the fact that gravitational attraction is universal and reaches infinitely far. Therefore, the term gravitational field will be presented as finite in below sections with complete awareness that this is subject to error. The 'range' of the fields will be labeled 'SOI', which is short for 'sphere of influence'.

The two-body simplifications are Keplerian orbits which can take the shape of a circle, an ellipse, a parabola or a hyperbola. Since the circular and parabolic orbits are specific instances of the elliptical and hyperbolic orbits, respectively, they can be omitted for simplicity's sake. These geometric primitives can be described in terms of physical quantities introduced below and are far more simple than the geometry of orbits with more than two objects.

2.2 Modeling the Solar System

The first difficult challenge I faced was accurately representing the solar system in the program. I chose to use a Cartesian coordinate system centered around the solar system barycenter (SSB), which represents the center of mass of all bodies orbiting the sun, including the sun itself. The reason for this initial setup is as follows: The Cartesian coordinate system allows the axes to be treated separately under differentiation and integration and the solar system barycenter is the point around which all planets orbit, which makes calculations and graphing much simpler and concise.

To obtain accurate data for the positions of the planets with respect to time, I decided to employ NASA's SPICE library which grants access to tools designed for parsing and using NASA's planetary databases. While verbose, the documentation allows for easy, quick, yet flexible use of planetary data (ephemerides).

The moons of the Earth and Mars will be omitted and the moons of other planets accounted for by averaging their positions with respect to their mass. This is an important limitation of this simulation and further research is needed to assure that the moons do not interfere with the flight of the rocket.

2.3 Simulating Gravity

The core of the simulation is the force of gravity between two objects first presented by Isaac Newton:

$$F = G\frac{m_1 m_2}{r^2} \tag{1}$$

Where m_1 and m_2 are the masses of both objects and r is the distance between them. Because this force influences the position of an object, I gave every relevant body in the solar system (the sun, planets and the rocket) a set of values with the minimum necessary information required to derive all other quantities. These are:

- Net force ($\vec{F}$)
- Velocity ($\vec{v}$)
- Position ($\vec{x}$)
- Mass (m)

These quantities obey the following differential equations:

$$\begin{aligned} \vec{F} &= m\frac{d\vec{v}}{dt} \\ \vec{v} &= \frac{d\vec{x}}{dt} \end{aligned}$$

While at first glance it seems that this set of differential equations has an algebraically closed form, the function $\vec{F}(t)$ depends on the position $\vec{x}$, because it is required to calculate the distance to another body as shown in (1).

Given these variables, the fourth order Runge-Kutta method (RK4) will be employed to numerically solve the differential equations above over a small time step Δt. This raises an issue however, because the differential equation is of second order, whereas the RK4 method requires a first order differential equation of the form $\frac{dx}{dt} = f(x,t)$. To remedy this, I reduce the order of the differential equations as follows:

$$\begin{aligned} F = m\frac{dv}{dt} &= m\sum_i G\frac{M_i}{r_i^2} \cdot \vec{r}_i \\ \implies \quad \frac{dv}{dt} &= \sum_i G\frac{M_i}{r_i^2} \cdot \vec{r}_i \\ \frac{dx}{dt} &= v \end{aligned}$$

Now it can be seen that $f(v,t) = \sum\limits_i G\frac{M_i}{r_i^2} \cdot \vec{r}_i$ and $f(x,t) = v(t)$. Such numerical solvers gain accuracy by reducing the time step, Δt. Hence, when the net force on the rocket is very small, i.e. the motion is nearly linear, the time step can be stretched to reduce the computational overhead.

To perform the above variation of the time step, I introduce a proportionality constant γ such that the following holds:

$$\Delta t = \frac{\gamma}{|\vec{a}|} \tag{2}$$

Where $\vec{a}$ is the net acceleration of the rocket at each step in the simulation. While based on intuition, this relationship satisfies certain mathematical requirements. Firstly, $\Delta t(|\vec{a}|)$ decreases monotonically which means that a higher force always corresponds to a smaller time step. Secondly, Δt must always be positive because passing time is one-directional, which implies that γ must be positive as well. Furthermore, to ensure that we do not lose too much accuracy, we will introduce a maximum time step, Δt_{max}. These constants will be chosen arbitrarily and will also depend on the computer used.

3 Independent, Dependent and Controlled Variables

These variables will be described and explained below and represent the requirements and cost of the mission. Independent variables:

- Boost times t_1 and t_2

Dependent variables:

- Time of flight t_{f}
- Speed requirements $C_{3,\text{e}}$ and $C_{3,\text{m}}$
- Required change in velocity Δv_1, Δv_2
- Required fuel m_{f} (Directly related to Δv_1 and Δv_2)
- The launch angle ϕ

Controlled variables (using SpaceX's Starship rocket as a reference[4]):

- Rocket mass and payload $m_0 = 220\text{t}$
- Rocket thrusters' net combined force $F_{\text{t}} = 12\text{MN}$
- Fuel flow rate $\dot{m} = 3.75\frac{\text{t}}{\text{s}}$ (Positive flow rate results in decrease of total rocket mass m)

4 Launch Window

The first, and arguably hardest challenge is finding a suitable time for a launch to occur. To give us the most flexibility and room for error when executing a launch, a time window is calculated where, at any given point in time, the fuel required is under a certain threshold. These windows are short and require many calculations to obtain.

4.1 Speed Requirements

In order to understand which launch window is best for a given time, we must define multiple quantities and their meanings in terms of interplanetary missions. The first is v_∞. A rocket requires a certain amount of energy to exit Earth's gravitational field from a given distance R_{e} from the center of mass. When it is assumed that the rocket is only subjected to the force exerted by the Earth, the energy required to exit this gravitational field can be obtained by integrating over the force of gravity with respect to distance:

$$E = \int_{R_{\text{e}}}^{\infty} F_{\text{G}} ds = \int_{R_{\text{e}}}^{\infty} G\frac{M_{\text{e}}m}{s^2} ds = -G\frac{M_{\text{e}}m}{s}\bigg|_{R_{\text{e}}}^{\infty} = G\frac{M_{\text{e}}m}{R_{\text{e}}}$$

This means that, given an initial velocity v_{esc}, the kinetic energy $E_{\text{kin}} = \frac{1}{2}m \cdot {v_{\text{esc}}}^2$ is transformed entirely into potential energy. It follows that $v_{\text{esc}} = \sqrt{\frac{2GM_{\text{e}}}{R_{\text{e}}}}$. Any superfluous velocity added at the launch must therefore result in excess kinetic energy as the distance approaches infinity. As such, the velocity of the rocket after escaping the gravitational field, v_∞, can also be obtained:

$$\begin{aligned} E_{\text{i}} = E_{\text{esc}} + E_{\infty} = G\frac{M_{\text{e}}m}{R_{\text{e}}} + \frac{1}{2}m\cdot{v_{\infty}}^2 &= \frac{1}{2}m\cdot{v_{\text{i}}}^2 \\ \frac{2GM_{\text{e}}}{R_{\text{e}}} + {v_{\infty}}^2 &= {v_{\text{i}}}^2 \\ {v_{\text{esc}}}^2 + {v_{\infty}}^2 &= {v_{\text{i}}}^2 \end{aligned} \tag{3}$$

Here R_{e} represents the height of the rocket with respect to the Earth's center of mass, m describes the rocket's mass and M_{e} is the mass of the body the rocket is orbiting (In this case the Earth).

It is a common convention to use the characteristic energy $C_3 = {v_{\infty}}^2$ to represent this value. It relates to the specific energy, the energy per unit of mass, by $C_3 = 2\varepsilon = \frac{2E}{m}$. This follows from the formula for kinetic energy, $E = \frac{1}{2}m\cdot{v_{\infty}}^2 = \frac{1}{2}m\cdot C_3$.

Furthermore, one applies the same calculations as above to obtain the boost required to enter an orbit around Mars. To avoid ambiguity, $C_{3,\text{e}}$ is used as a measure of the fuel requirements for departure from Earth and $C_{3,\text{m}}$ for the entry into an orbit around Mars. With these equations, the velocity required at launch can be calculate given these two values.

4.2 Injection and Exit Times

The time at which the rocket boosts to enter the hyperbolic injection orbit will be named t_1 and the time at which the rocket boosts to enter a parking orbit after the exit orbit will be labeled t_2. The time of flight (t_{f}) of the rocket corresponds to the difference.

4.3 Δv and Fuel

Fuel is an important constraint, so being able to calculate an estimate is vital. Using equation (3), one can calculate the required change in velocity, Δv_1, to enter a hyperbolic orbit around the Earth and obtain the necessary $C_{3,\text{e}}$ as the distance approaches infinity.

First, equating Earth's gravitational pull with the centripetal force needed to stay in a stable orbit at a distance of r_{leo} away from Earth's center of mass gives a formula for calculating the speed, v_{oe}, at which the rocket will be traveling relative to the Earth while in orbit:

$$F_{\text{G}} = F_{\text{C}} \implies G\frac{M_{\text{e}}m}{{r_{\text{leo}}}^2} = \frac{m{v_{\text{oe}}}^2}{r_{\text{leo}}} \implies v_{\text{oe}} = \sqrt{\frac{GM_{\text{e}}}{r_{\text{leo}}}}$$

Now we plug this formula into (3) to obtain Δv_1:

$$\begin{gathered} v_{\text{i}} = v_{\text{oe}} + \Delta v_1 \\ {v_{\infty}}^2 + {v_{\text{esc}}}^2 = (v_{\text{oe}} + \Delta v_1)^2 \implies \Delta v_1 = \sqrt{{v_{\infty}}^2 + {v_{\text{esc}}}^2} - v_{\text{oe}} = \sqrt{C_{3,\text{e}} + 2\frac{GM_{\text{e}}}{r_{\text{leo}}}} - \sqrt{\frac{GM_{\text{e}}}{r_{\text{leo}}}} \end{gathered}$$

Similarly, we obtain Δv_2:

$$\begin{gathered} v_{\text{f}} = v_{\text{om}} + \Delta v_2 \\ {v_{\infty}}^2 + {v_{\text{esc}}}^2 = (v_{\text{om}} + \Delta v_2)^2 \implies \Delta v_2 = \sqrt{{v_{\infty}}^2 + {v_{\text{esc}}}^2} - v_{\text{om}} = \sqrt{C_{3,\text{m}} + 2\frac{GM_{\text{m}}}{r_{\text{lmo}}}} - \sqrt{\frac{GM_{\text{m}}}{r_{\text{lmo}}}} \end{gathered}$$

Where v_{f} represents the incoming velocity of the rocket as it reaches Mars before applying its second boost, v_{om} is the relative velocity required to stay in a parking orbit around Mars and M_{m} is Mars' mass.

Since the rocket uses fuel in discrete bursts, we must calculate the fuel required for the final stage, add it to the payload mass and use this new total to determine the fuel required to boost both the fuel for the final stage and the payload at the previous stage. To obtain the mass of the fuel required, we employ the famous 'Tsiolkovsky Rocket Equation' which will be derived in an appendix:

$$1 + \frac{m_\text{f}}{m_\text{p}} = e^{\frac{\Delta v}{v_\text{e}}} \tag{4}$$

Where:

- m_f is the mass of the required fuel
- m_p is the mass of the rocket at the previous stage (with or without fuel, depending on the stage)
- Δv is the change in velocity required for the given stage
- v_e is the exhaust speed generated by the thrusters (Directly related to F_t and $\dot{m}$)

To find the mass of the fuel needed to obtain the desired Δv_1 and Δv_2, we must relate v_e to the force generated and fuel flow rate required by the thrusters on the rocket. Due to conservation laws, the momentum of the rocket must be equal to the momentum of the mass exiting it:

$$p = m_\text{e} v_\text{e}$$

Differentiating both sides of the equation yields the following formula:

$$\begin{aligned}
\frac{dp}{dt} &= \frac{d}{dt}(m_\text{e} v_\text{e}) \\
F_\text{t} &= \dot{m} v_\text{e} + m_\text{e} \dot{v}_\text{e} && (\dot{v}_\text{e} = 0) \\
F_\text{t} &= \dot{m} v_\text{e} \\
v_\text{e} &= \frac{F_\text{t}}{\dot{m}}
\end{aligned}$$

Where F_t is the combined net force generated by the rocket's thrusters and $\dot{m}$ is the fuel flow rate.

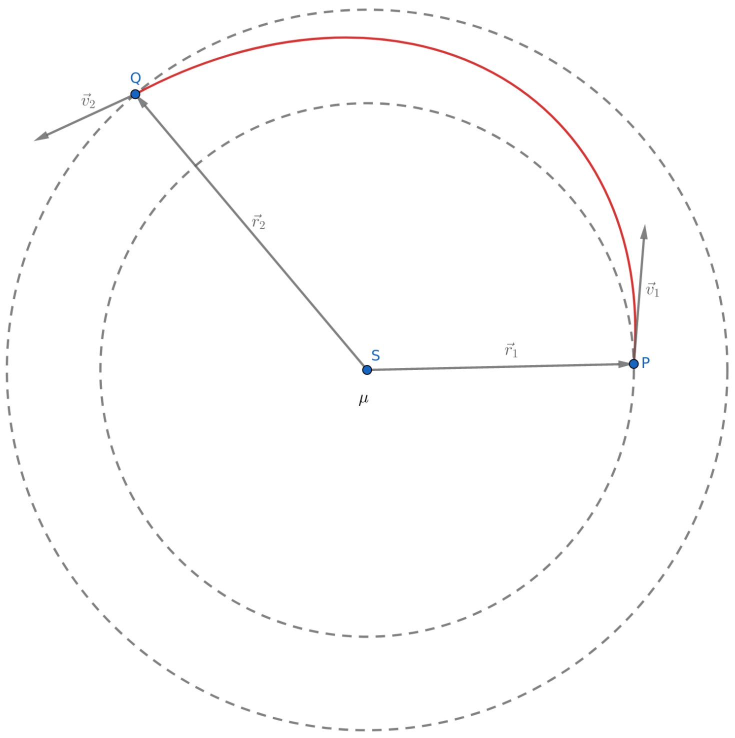
Now the total fuel requirement can be determined according to (4) and the process defined above:

$$\begin{aligned}
m_2 &= m_0 (e^{\frac{\Delta v_2}{v_e}} - 1) \\
m_1 &= m_2 (e^{\frac{\Delta v_1}{v_e}} - 1)
\end{aligned}$$

It is evident that m_1 is the fuel required to reach low Mars orbit from low Earth orbit, previously named m_f.

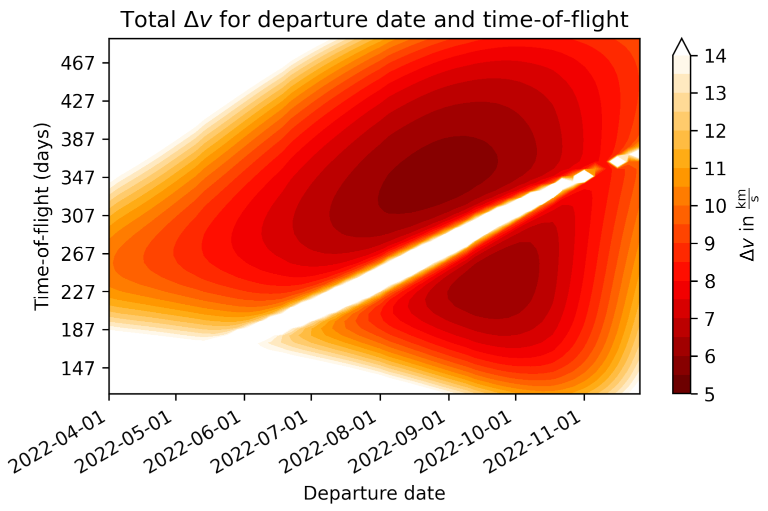
4.4 Lambert's Problem

The last step in determining the time of launch is to solve for a transfer orbit which starts at t_1 and ends at t_2 and minimizes $C_{3,\text{e}}$ and $C_{3,\text{m}}$. Since the positions of Earth and Mars are known at t_1 and t_2 (using NASA's database), the problem reduces to finding an ellipse passing through two points with one known focus. This is true because we assume the transfer orbit around the sun is a Keplerian orbit where the central body must always occupy one of the foci of the ellipse. The following diagram sketches the problem to be solved:



This problem is known as Lambert's Problem and has no algebraically closed form. In this case, $\vec{r}_1$ and $\vec{r}_2$ represent the position of the Earth and Mars relative to the Sun, respectively. $\vec{v}_1$ and $\vec{v}_2$ are the velocities the rocket is required to have at the respective planets relative to the sun. By subtracting the velocity of the current planet from $\vec{v}_1$ and $\vec{v}_2$, one finds $\vec{v}_\infty$ (to determine both $C_{3,\mathrm{e}}$ and $C_{3,\mathrm{m}}$ when applied to v_1 and v_2, respectively). Since this problem has no closed form, it must be approximated using an algorithm which can be implemented in the simulation. A modern and computationally efficient solution was found by Dario Izzo at the European Space Agency [1].

I sampled D. Izzo's algorithm many times in a window from the year 2021 to 2023, to find when a launch could occur. Once I found feasible values for t_1 and t_2, I generated a contour plot of $\Delta v = \Delta v_1 + \Delta v_2$. As shown above, $m_\mathrm{f} = m_0 \left(e^{\frac{\Delta v_1}{v_\mathrm{e}}} - 1\right)\left(e^{\frac{\Delta v_2}{v_\mathrm{e}}} - 1\right)$. It follows that $m_\mathrm{f} \leq m_0 \left(e^{\frac{\Delta v_1}{v_\mathrm{e}}} e^{\frac{\Delta v_2}{v_\mathrm{e}}} - 1\right) = m_0 \left(e^{\frac{\Delta v_1 + \Delta v_2}{v_\mathrm{e}}} - 1\right) = m_0 \left(e^{\frac{\Delta v}{v_\mathrm{e}}} - 1\right)$. Therefore, Δv can be used as a good approximate measure of the total fuel requirement.

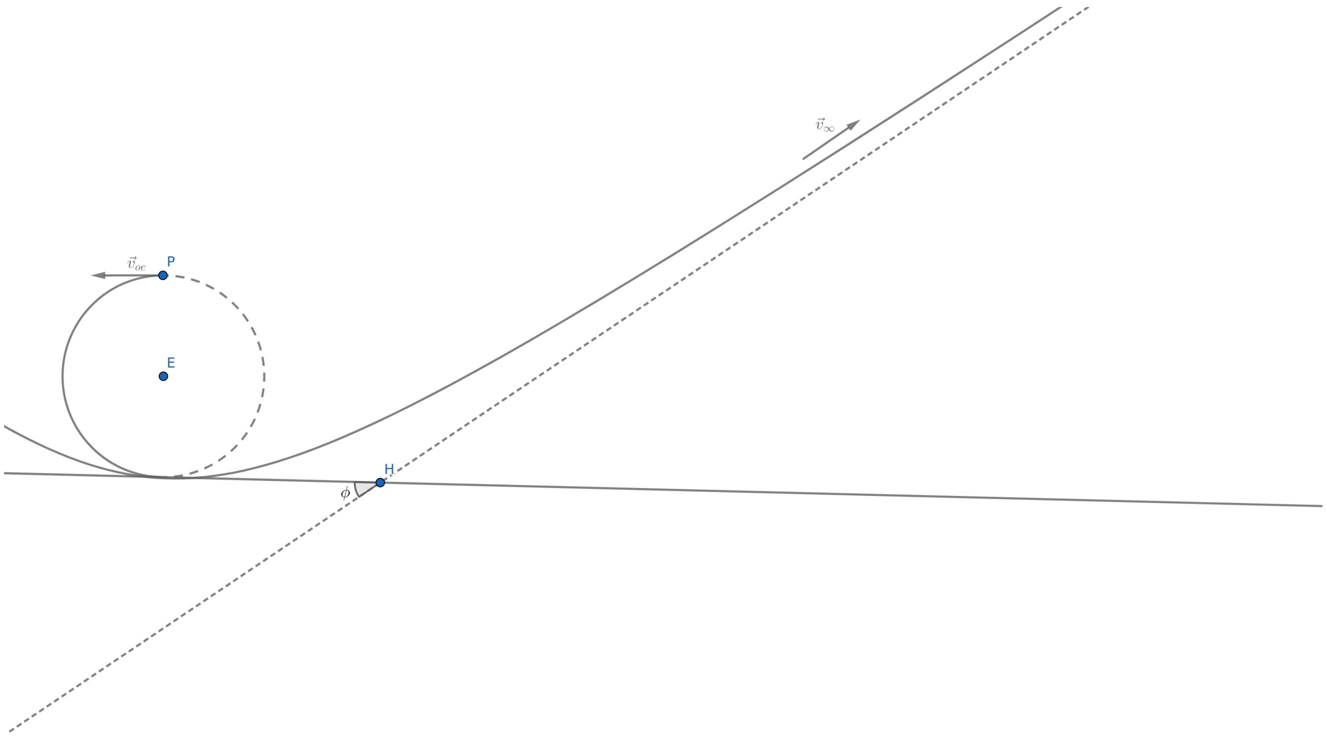


These plots are known as pork-chop plots due to their shape. This plot is easy to work with on the day of a launch because one can easily see when a launch is feasible. Computationally however, I chose the minimum to save as much fuel as possible (not including the upper region as this corresponds to a special type of fuel efficient transfer not covered here).

- $t_1 = $ 2022 Sep 28 00:00:00
- $t_2 = $ 2023 Jun 01 00:00:00
- $t_\text{f} = 246\,\text{days}$
- $C_{3,\text{e}} = 25.7\frac{\text{MJ}}{\text{kg}}$
- $C_{3,\text{m}} = 5.60\frac{\text{MJ}}{\text{kg}}$
- $\Delta v_1 = 4.27\frac{\text{km}}{\text{s}}$
- $\Delta v_2 = 1.79\frac{\text{km}}{\text{s}}$
- $m_\text{f} = 460\text{t}$

5 Hyperbolic Injection Orbit

While scalar quantities were used above, velocities must be expressed as vectors to also obtain the correct directions since we are working with three-dimensional space. The following diagram will provide a basic explanation of the values needed:



As $\vec{v}_\infty$ is the rocket's relative velocity with respect to the Earth, $\vec{v}_\infty = \vec{v}_1 - \vec{v}_{\text{earth}}$ must hold. Since we only apply Δv_1 in a short burst, the rocket must be launched at a certain angle ϕ relative to $\vec{v}_{\infty,1}$ to assure that it exits in the right direction.

Since only the Earth and the rocket will be taken into account, this problem simplifies to one of a two-body hyperbolic Keplerian orbit. Using equation (3), the required velocity at the periapsis of the orbit can be obtained. The following equation determines the value of ϕ [2]:

$$\phi = \arcsin\left(\frac{1}{1 + \frac{r_{\text{leo}} {v_\infty}^2}{GM_{\text{e}}}}\right) = 0.663 \tag{5}$$

It follows from the geometric properties of a hyperbola's eccentricity and can be expressed in terms of physical quantities like mass and velocity.

To perform the rotations necessary for accommodating the angle ϕ, one constructs a matrix $R(\vec{u}, \alpha)$ which represents the rotation around $\vec{u}$ by α radians. The product of this matrix with any given vector $\vec{w}$ can be computed as follows [5].

$$\vec{w} \cdot R(\vec{u}, \alpha) = \vec{u}(\vec{u} \cdot \vec{w}) + \cos(\alpha)(\vec{u} \times \vec{w}) \times \vec{u} + \sin(\alpha)(\vec{u} \times \vec{w}) \tag{6}$$

Using ϕ, the vector $\vec{v}_\infty$ and (6), the starting position and velocity required by the rocket at launch can be calculated (assuming instantaneous application of Δv_1).

Firstly, one rotates the velocity vector $\vec{v}_\infty$ around a normal, $\vec{n}$, by ϕ to obtain the starting velocity with the correct planar direction. $\vec{n}$ can be any vector perpendicular to $\vec{v}_\infty$ because as long as the rocket reaches $\vec{v}_\infty$ at Earth's SOI, the minor displacement due to the rocket flying around the Earth in a different (e.g. opposite) direction becomes negligible.

$$\vec{v}_{\text{i}} = \vec{v}_\infty \cdot R(\vec{n}, \phi)$$

Similarly, one can construct a normalized perpendicular vector to $\vec{v}_\infty$ as follows: $\vec{x}_\text{n} = \frac{\vec{n} \times \vec{v}_\infty}{|\vec{n} \times \vec{v}_\infty|}$. After adding this displacement normal, rotated and scaled to the desired distance from the Earth, $r_\text{leo}\vec{x}_\text{n} \cdot R(\vec{n}, \phi)$, to the Earth's absolute position, the absolute starting position of the rocket, $\vec{x}_\text{i}$, can be obtained:

$$\begin{aligned} \vec{x}_\text{n} &= \frac{\vec{n} \times \vec{v}_\infty}{|\vec{n} \times \vec{v}_\infty|} \\ \vec{x}_\text{i} &= r_\text{leo}\vec{x}_\text{n} \cdot R(\vec{n}, \phi) + \vec{x}_\text{e}(t_1) \end{aligned}$$

Since this paper assumes the rocket is already in orbit around the Earth, this construction is valid, but additional calculations would have to be done to assure that the rocket ends up in this position after a launch from the Earth's surface.

6 Arrival

Now that the position, velocity and fuel requirements of the rocket at launch have been determined, the only missing step is the arrival orbit around Mars.

6.1 Aerobraking

At this point, a mechanical failure or underestimation of the required fuel could cause the rocket's Δv capacity to be lower than required. While one hopes that this may never be the case, preparations can still be made for such a scenario. Aerobraking lets the rocket use the frictional force generated by moving through the atmosphere to reduce its kinetic energy, which, when planned correctly, will lower the fuel requirements drastically. This method is mentioned here due to completeness' sake, but it requires a very accurate knowledge of the atmosphere and weather to execute [6], so I will not use it in this paper.

6.2 Hyperbolic Capture Orbit

Similarly to section 5, the rocket follows a hyperbolic path when entering the SOI of Mars and must be accelerated to reach the parking orbit around the planet.

In 4.3, I already calculated the boost needed to enter an orbit around Mars, but there is a caveat: if the rocket slightly strays from the expected distance and velocity calculated here, it will either over- or undershoot depending on the sign of the error. This problem can be easily remedied by calculating the necessary Δv_2 on the fly. Using this method also simplifies the problem, because no predictions need to be made.

To begin with, I convert to a Mars-centric coordinate system to ease the calculations and obtain the rocket's relative velocity, $\vec{v}_\text{rel}$:

$$\vec{v}_\text{rel} = \vec{v} - \vec{v}_\text{m}(t_2) \tag{7}$$

Secondly, I determine the velocity required for a parking orbit as presented in 4.3:

$$v_\text{p} = \sqrt{\frac{GM_\text{m}}{|\vec{x} - \vec{x}_\text{m}(t_2)|}} \tag{8}$$

Note that I recalculate the distance between Mars and the rocket in the denominator because r_lmo will likely not be the correct distance anymore.

Finally, using (7) and (8), the actual Δv_2 can be obtained by subtracting these velocities' magnitudes:

$$\Delta v_2 = |\vec{v}_\text{rel}| - v_\text{p} \tag{9}$$

This boost is actually positive, even though v_p is smaller than $|\vec{v}_\text{rel}|$, because Mars's velocity is larger than the rocket's so it must speed up to match this velocity.

7 Accounting for Errors

Now that all of the dependent variables have been determined, the launch can be performed. One issue is, however, that most above calculations have been based on assumptions that require a simplified model of gravity. In a real situation, the following maneuvers could be performed to account for the errors arising due to the simplifications.

7.1 Correction Burn

The most simple solution of the aforementioned problem is to require an additional Δv buffer while launching to perform a correction burn. Once the rocket has (mostly) exited the influence of Earth's gravitational field and is in a fairly stable transfer orbit around the sun, the thrusters are fired to regain the necessary orbital energy to reach $\vec{x}_{\mathrm{m}}(t_2)$. This method only requires us to keep track of only two variables to perform: The rocket's velocity and position.

After the rocket enters the transfer orbit, calculations are run simulating the path taken by the rocket when various amounts of correction Δv are applied. The path which ends up closest to $\vec{x}_{\mathrm{m}}(t_2)$ is chosen and boost is applied accordingly. This method allows for very flexible launches and guaranties decent safety for the passengers onboard. However, this does require a mostly unknown amount of additional boost which might be out of range of the rocket's capacity. Therefore I will not be using this approach.

7.2 Sampling and Simulated Annealing

Simulated annealing is a max- or minimization algorithm which attempts to find the global max- or minimum of a function. This algorithm can be used to vary the launch variables Δv_1 and ϕ and to simulate a flight to determine which set of variables results in the closest distance to Mars (More precisely, the flight matching the distance r_{lmo} to Mars the best). This method is fairly simple and can be implemented using the technology already at hand. By sampling the simulations and decreasing γ the algorithm will converge on the launch variables which will result in the closest orbit around Mars.

7.3 Hohmann Transfer Orbit

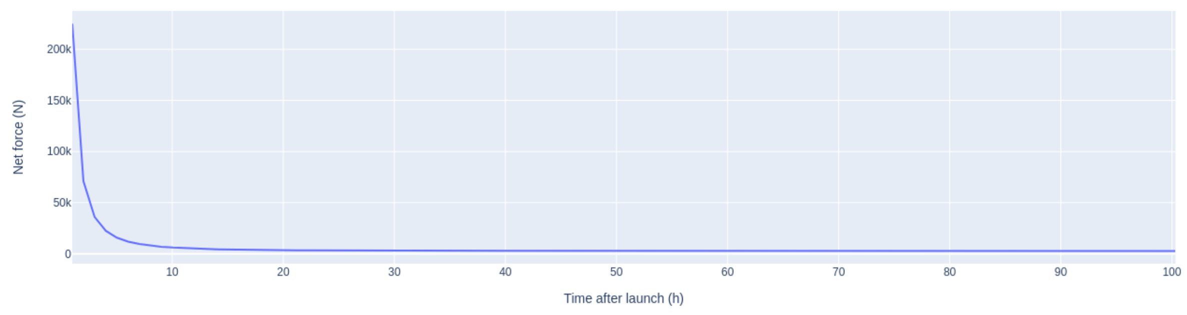
Due to the inaccuracy of the approximations I have made, it is unavoidable that the rocket will not be captured at the desired height of r_{lmo} above Mars. If the rocket can be captured at a height of h however, a transfer orbit can still be performed to lower the height of the rocket over Mars. One can simply use the Lambert solver as described above to solve for an orbit which transitions the rocket from its higher orbit around Mars to the lower one. In this case, the vectors $\vec{r}_1$ and $\vec{r}_2$ are parallel (facing in opposite directions) and the velocities will be perfectly tangent to the starting and ending orbits. This particular case of the Lambert problem is known as the Hohmann transfer and does have an algebraically closed form, so it does not need to be approximated. That being said, this goes outside of the scope of this paper so I will not present it here.

8 Results

Finally, after having done many calculations and approximations, I can simulate the launch and gather some experimental results.

8.1 Determining the Optimal γ

Before I start doing many computationally heavy calculations, I will determine the optimal γ to save time and assure the accuracy of my results. I found that a simulation took approximately 3.5 minutes to complete at a constant time step of 10 seconds. Additionally, I generated a plot showing the net force acting on the rocket with respect to time:



Here the x-axis represents the hours after launch and the y-axis shows the net force in newtons ($\frac{\text{kg}\cdot\text{m}}{\text{s}^2}$).

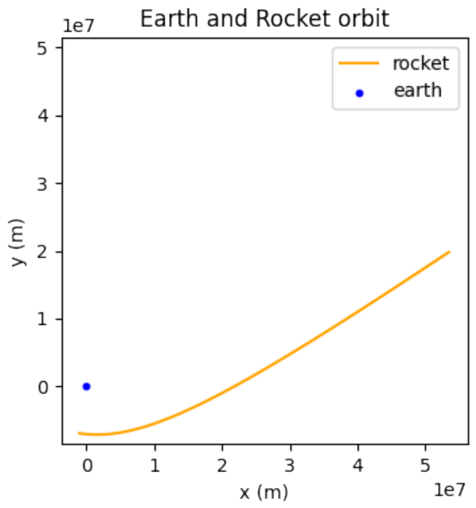
This graph only shows the first 100 hours for convenience's sake, but the global maximum is 224 kN, which occurs at t_1. Furthermore, the mean lies at mere 1.38 kN, which shows that the global maximum is an anomaly and we can settle for a relatively high Δt_{max}. Using this information, I opted for $\gamma = 0.02\frac{\text{N}}{\text{kg}} \cdot \text{s}$ and $\Delta t_{\text{max}} = 20\text{s}$. This means that the smallest time step, occurring at the entry and exit orbits, would be approximately 44.6ms. While simulating, this turned out to result in acceptable waiting times but the choice of these numbers ultimately depends on the hardware being used.

8.2 The Launch

The launch is fairly straightforward and does not require any other maneuvers except for boosting. The amount of time needed to boost by a given Δv is simply the mass of the fuel being burnt divided by the fuel flow rate: $t_{\text{b}} = \frac{m}{\dot{m}}$. Pairing this formula with Tsiolkovsky's rocket equation, one can figure out how long the thrusters need to fire at t_1 and t_2.

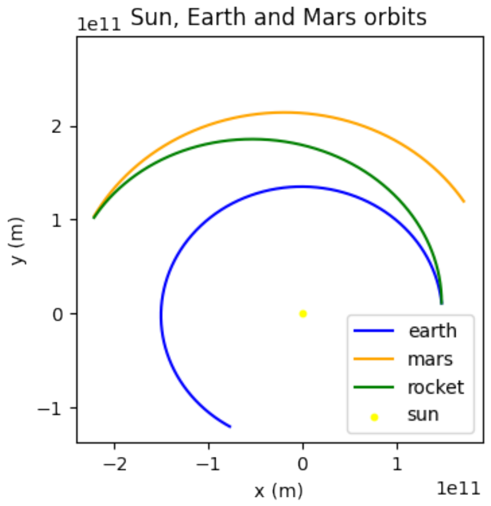
8.2.1 Escaping Earth's Gravitation Field

After running the simulation, the rocket manages to exit Earth's SOI successfully as seen below (in the reference frame of the Earth):



8.2.2 Transfer Orbit

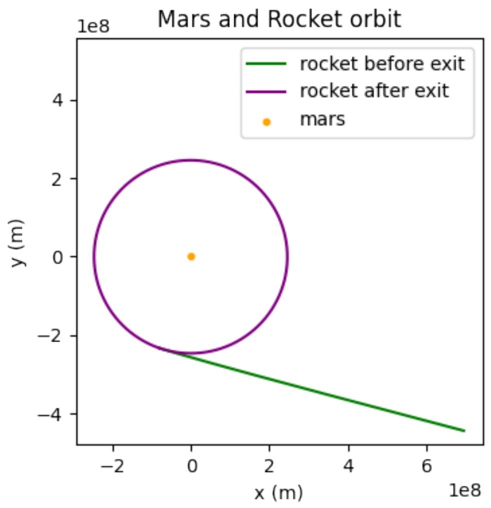
Now that the rocket has exited Earth's SOI, it begins an 8 month journey around the sun. This long flight has also completed successfully as can be seen in the following (reference frame of the SSB):



Here we can see the expected elliptical orbit generated by the Lambert solver.

8.2.3 Capture and Parking Orbits

Finally the arguably most exciting phase of the launch lakes place: the capture and parking orbits. Here the rocket must be boosted so that it successfully stays in orbit for as long as necessary. The graph below shows this process in the reference frame of Mars:



The green and purple paths represent the hyperbolic capture and parking orbits, respectfully. It looks successful at a glance but the orbit may be unstable due to the fact that the rocket orbits at a very wide radius, meaning the Sun's influence becomes relevant. To assure that the orbit is in a stable state, I simulated 100 days further into the simulation to assure that the rocket is able to stay in orbit for a reasonable amount of time (to perform a Hohmann transfer, for instance). The rocket did not stray from the orbit in 100 days so it is safe to say that the parking was successful.

It is important to keep in mind that these graphs are all projections of three-dimensional space. The planes in which the orbits lie actually differed by a quite large amount. If the rocket reaches Mars's surface, this is not an issue, but it could result in problems if the rocket were supposed to fly back from the same parking orbit it ended in here.

9 Conclusion

Using the patched conic approximation and a solver for the Lambert problem, I have shown when and how a launch to Mars could be possible in the year 2022. While my methods have certain limitations due to approximations in the simulation (Euler integration) and in the theoretical gravity calculations, the assumption that space is a vacuum and that the moon does not affect the rocket, the simulation offers a starting point for institutions like NASA or the ESA to continue research and more accurately plan a manned mission to Mars.

Such a project is invigorating because it gives me the chance to explore (if only fictionally) the entire solar system using only a computer. Technology allows this kind of research to become more accessible to students and for interest to more easily be developed.

References

[1] Dario Izzo. "Revisiting Lambert's problem". In: *Celestial Mechanics and Dynamical Astronomy* 121.1 (Jan. 2015), pp. 1–15. DOI: 10.1007/s10569-014-9587-y.

[2] Craig A. Kluever. "Spaceflight Mechanics". In: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. by Robert A. Meyers. Third Edition. New York: Academic Press, 2003, pp. 507–520. ISBN: 978-0-12-227410-7.

[3] Josh Lengel. *Rocket Flight Simulator for Extended Essay*. Version 1.0.0. Feb. 2022. URL: https://github.com/joshlengel/ExtendedEssay.

[4] Elon Musk. *Starship Update*. URL: https://youtu.be/sOpMrVnjYeY?t=1229. (accessed: 07.09.2021).

[5] Olinde Rodrigues. "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire". In: *Journal de Mathématiques Pures et Appliquées* 5 (1840), pp. 380–440. URL: http://sites.mathdoc.fr/JMPA/PDF/JMPA_1840_1_5_A39_0.pdf.

[6] David A. Spencer and Robert Tolson. "Aerobraking Cost and Risk Decisions". In: *Journal of Spacecraft and Rockets* 44.6 (2007), pp. 1285–1293. DOI: 10.2514/1.24303.

Appendices

Deriving Tsiolkovsky's 'Rocket Equation'

The conservation of momentum will be employed to derive the 'Rocket Equation'. Firstly, the rocket's momentum will be equated with itself after an infinitesimally small time step dt:

$$(m + dm)v = m(v + dv) + dm \cdot (v - v_\mathrm{e})$$

Where $(m + dm)$ is the mass of the rocket before expelling part of its exhaust, $(m + dm)v$ is the momentum of the rocket, $(v + dv)$ is the rocket's velocity and $m(v + dv)$ the rocket's increased momentum after boosting and finally $dm \cdot (v - v_\mathrm{e})$ is the exhaust's momentum:

$$\begin{aligned}
(m + dm)v &= m(v + dv) + dm \cdot (v - v_\mathrm{e}) \\
mv + v \cdot dm &= mv + m \cdot dv + v \cdot dm - v_\mathrm{e} \cdot dm \\
m \cdot dv &= v_\mathrm{e} \cdot dm \\
\int_{v_0}^{v_\mathrm{f}} \frac{1}{v_\mathrm{e}} dv &= \int_{m_\mathrm{i}}^{m_0} -\frac{1}{m} dm && (dm > 0 \text{ results in a decrease of mass}) \\
\frac{v_\mathrm{f} - v_0}{v_\mathrm{e}} &= \ln(m_\mathrm{i}) - \ln(m_0) \\
\frac{\Delta v}{v_\mathrm{e}} &= \ln\left(\frac{m_\mathrm{i}}{m_0}\right) && (\Delta v = v_\mathrm{f} - v_0) \\
\frac{m_\mathrm{i}}{m_0} &= e^{\frac{\Delta v}{v_\mathrm{e}}} \\
\frac{m_0 + m_\mathrm{f}}{m_0} &= e^{\frac{\Delta v}{v_\mathrm{e}}} && (\text{Initial mass is payload + fuel}) \\
1 + \frac{m_\mathrm{f}}{m_0} &= e^{\frac{\Delta v}{v_\mathrm{e}}}
\end{aligned}$$

Here the mass is integrated from m_i (initial mass) to m_0 (payload) since the mass decreases. The minus sign is introduced in the mass integral for the same reason. All other variables are either temporary or have been explained in above sections.

Code Snippet

This is a code snippet of my repository which is used to calculate all of the values needed to generate a porkchop plot.

Porkchop.cpp

```cpp
#include"Porkchop.h"
#include"Lambert.h"
#include"Vec.h"

#include<SpiceUsr.h>

PorkchopPlot::PorkchopPlot(double dt1, double dt2, double tof1, double tof2,
    size_t num_values, double rleo, double rlmo):
     m_dt1(dt1), m_dt2(dt2),
     m_tof1(tof1), m_tof2(tof2),
     m_num_values(num_values)
```

```
{
    m_values.reserve(num_values * num_values);

    int dim; double gm;

    bodvrd_c(”Sun”, ”GM”, 1, &dim, &gm);
    double mu_sun = gm * 1e9;

    bodvrd_c(”Earth”, ”GM”, 1, &dim, &gm);
    double mu_earth = gm * 1e9;

    bodvrd_c(”Mars_Barycenter”, ”GM”, 1, &dim, &gm);
    double mu_mars = gm * 1e9;

    for (size_t i = 0; i < num_values; ++i)
    for (size_t j = 0; j < num_values; ++j)
    {
        double t1 = (dt2 - dt1) * static_cast<float>(i) / num_values + dt1;
        double tof = (tof2 - tof1) * static_cast<float>(j) / num_values + tof1
            ;
        double t2 = t1 + tof;

        double lt;

        double earth_state[6];
        spkezr_c(”Earth”, t1, ”J2000”, ”NONE”, ”Sun”, earth_state, &lt);
        Vec3 earth_pos = Vec3(earth_state[0], earth_state[1], earth_state[2])
            * 1000;
        Vec3 earth_vel = Vec3(earth_state[3], earth_state[4], earth_state[5])
            * 1000;

        double mars_state[6];
        spkezr_c(”Mars_Barycenter”, t2, ”J2000”, ”NONE”, ”Sun”, mars_state, &
            lt);
        Vec3 mars_pos = Vec3(mars_state[0], mars_state[1], mars_state[2]) *
            1000;
        Vec3 mars_vel = Vec3(mars_state[3], mars_state[4], mars_state[5]) *
            1000;

        Vec3 v1, v2;
        LambertDIzzo(earth_pos, mars_pos, tof, mu_sun, v1, v2);

        Vec3 rel_1 = v1 - earth_vel;
        Vec3 rel_2 = mars_vel - v2;
        double C3_e = Vec3::Dot(rel_1, rel_1);
        double C3_m = Vec3::Dot(rel_2, rel_2);
        double dv_1 = sqrt(C3_e + 2 * mu_earth / rleo) - sqrt(mu_earth / rleo)
            ;
        double dv_2 = sqrt(C3_m + 2 * mu_mars / rlmo) - sqrt(mu_mars / rlmo);
        m_values.push_back(dv_1 + dv_2);
    }
}
```