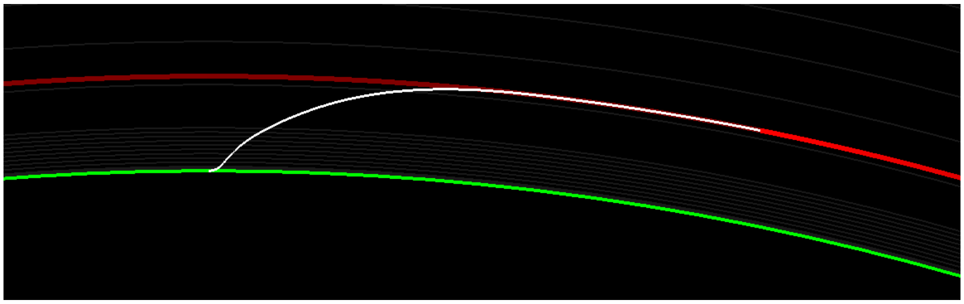

# Optimizing the ascent trajectory for an orbital class launch vehicle

Final project in SI1336

`https://github.com/eweilow/rocket-optimize`

Erik Weilow

January 20, 2019

## 1 Introduction

The goal of this project is to compare the propellant consumption for the launch of a multi-stage orbital class launch vehicle into low Earth orbit, from an equatorial and a polar launch site, with a specific payload mass $m_{payload}$. This leads to a formulation of an approximate model of the conditions encountered for the launch vehicle during ascent. To be able to compare the results, the launch will be done with a goal of reaching an orbit with an apoapsis $g_{ap}$ and a periapsis $g_{pe}$. In order to be able to reach any conclusions, any influence from a bad trajectory on the propellant consumption must be minimized. This leads to the design of a simple ascent guidance algorithm, which will expose a collection of parameters that can be optimized using a randomized optimization algorithm.

The project is split into four major parts: the equations describing the rocket, the physics simulation, the ascent guidance algorithm as well as the optimization of control parameters.

## 2 Equations of the rocket

A rocket works on the principle of momentum conservation. Mass is ejected at a rate $\dot{m}$ (mass flow) through the engines at a velocity $v_e = I_{sp}/g_0$, producing a change in momentum (thrust) $\dot{p} = F = \dot{m} \cdot v_e = \dot{m} \cdot I_{sp}/g_0$. With the calculation of these variables, a rocket can be simulated.

The naive model is that a rocket consists of a propulsion system, some structure and propellant. This is not the case for a multi-stage rocket, where stages are dropped after the propellant contained within them are exhausted. The model must therefore consider that the rocket has several stages.

This can be solved by letting the state of the rocket be described with *position* ($\vec{r}$), *velocity* ($\vec{v}$) and *expended mass* ($m_e$), where $m_e$ is the integral of exhaust mass flow (ejected mass) over time:

$$m_e(t) = \int_0^t \dot{m}(\vec{r}, \vec{v}, m_e, t)dt, \quad \dot{m} \geq 0$$

Using expended mass in the model, rather than just keeping track of total mass, lets us *relatively easily* derive the currently active stage, and in extension the necessary variables (thrust, mass and mass flow) as a function of expended propellant.

## 2.1 Input parameters

Something that will be clear in section 3.4.2 is that we first need some basic aerodynamic parameters, which will be assumed constant:

| Parameter | Description |
|---:|---|
| $c_d$ | Drag coefficient of the rocket |
| $A$ | Frontal area of the rocket |

Let each stage be a set of parameters:

| Parameter | Description |
|---:|---|
| $m_{prop,i}$ | Propellant mass of stage |
| $m_{dry,i}$ | Dry mass of stage (without propellant) |
| $F_{sl,i}$ | Thrust at sea level pressure |
| $F_{vac,i}$ | Thrust in vacuum pressure |
| $I_{sp,sl,i}$ | Specific impulse at sea level |
| $I_{sp,vac,i}$ | Specific impulse in vacuum pressure |

such that the parameters for each stage $s_0, s_1, s_2, ...$ can be written

$$s_i = (m_{prop,i}, m_{dry,i}, F_{sl,i}, F_{vac,i}, I_{sp,sl,i}, I_{sp,vac,i})$$

Furthermore, a rocket typically has an aerodynamic shell (fairing) that is deployed around the edge of space (100km). This is modelled simply by the parameters

| Parameter | Description |
|---:|---|
| $m_{fairing,i}$ | Mass of fairing |
| $h_{deployment,i}$ | Altitude of deployment |

such that each fairing $f_0, f_1, f_2...$ can be written

$$f_i = (m_{fairing,i}, h_{deployment,i})$$

Lastly, the payload of the rocket is modelled as a mass $m_{payload}$.

## 2.2 Derived parameters

To run simulations of defined by the parameters in the previous section, we need to derive the current mass of the entire rocket. This is not as simple as subtracting $m_e$ from the initial mass, as we want to model staging. Instead consider parameters driven by the currently active stage $s_{active}$, which will be defined below.

### 2.2.1 Mass

If the propellant of stages $s_0, s_1, ...s_{i-1}$ before stage $s_i$ is

$$m'_{prop,i} = \sum_{j=0}^{i-1} m_{prop,j}$$

then the currently active stage $s_{active}$ is the one, if staging is assumed instant, that matches the criteria

$$\overbrace{m'_{prop,active}}^{\text{propellant mass of previous stages}} < m_e(t) \quad \textbf{and} \quad m_e(t) < \overbrace{m'_{prop,active} + m_{prop,active}}^{\text{propellant mass of stages after}}$$

As we are only considering a two stage launch vehicle, the active stage can be defined by the simpler form

$$s_{active} = \begin{cases} s_0 & : m_e < m_{prop,0} \\ s_1 & : 0 < m_e - m_{prop,0} < m_{prop,1} \end{cases}$$

If the mass of stages after the stage $s_i$ is

$$m'_{stages,i} = \sum_{j=i+1} m_{prop,j} + m_{dry,j}$$

This defines the current mass of stages, when $s_i$ is active, as

$$m_{stagemass,i} = \overbrace{\left(m_e(t) - m'_{prop,i}\right)}^{\text{current propellant mass}} + m_{dry,i} + m'_{stages,i}$$

If $h_{max}$ is the maximum altitude reached up until time $t$, then the mass of fairings currently on the rocket can be described by the sum

$$m_{fairings}(h_{max}) = \sum_{i=0} m_{fairing,i} \underbrace{H(h_{deployment,i} - h_{max})}_{\text{Inverted Heaviside function}}$$

Thus, the total instantaneous mass for an active stage $s_i$ is

$$m_i(h_{max}) = m_{stagemass,i} + m_{fairings}(h_{max}) + m_{payload}$$

### 2.2.2 Thrust

Since the thrust of a rocket propulsion system is linear in the pressure difference between the exhaust and surrounding pressure (NASA Glenn Research Center, 2015a), it is assumed in our model that the thrust $F$ changes linearly between $F_{sea}$ and $F_{vacuum}$ with pressure $p$.

For a given active stage $s_i$, if pressure is written as a function of altitude $h$, thrust can be written as

$$F_i(p(h)) = F_{sl,i} + (F_{sl,i} - F_{vac,i})\,\frac{p(h)}{p(0)}$$

This only holds if the stage has remaining propellant ($m_e - m_{prop,i} > 0$) otherwise

$$F_i(p(h)) = 0$$

### 2.2.3 Mass flow

Mass flow is necessary to find $m_e(t)$, and is given in a general form by (NASA Glenn Research Center, 2015b):

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

If we assume mass flow to be constant through the propulsion system, then $I_{sp}$ must share the same linear behaviour in pressure as thrust does. Thus, for an active stage $s_i$, let

$$I_{sp,i}(p(h)) = I_{sp,sl,i} + (I_{sp,sl,i} - I_{sp,vac,i})\,\frac{p(h)}{p(0)}$$

This gives the mass flow

$$\dot{m}_i(p(h)) = \frac{F_i(p(h))}{g_0 I_{sp,i}(p(h))}$$

### 2.2.4 Summarization

To tie the equations together, we define the functions

$$m(m_e, h_{max}) = m_i(h_{max}), \quad F(m_e, p(h)) = F_i(p(h)), \quad \dot{m}(m_e, p(h)) = \dot{m}_i(p(h))$$

where the active stage $s_i$ is derived from the given $m_e$. These functions will be used later.

## 3 Physics model

### 3.1 Coordinate system

The simulation uses two coordinate systems, one cartesian and one kinematic. The cartesian system $\hat{x}, \hat{y}, \hat{z}$ is originated in the starting location and oriented such that $\hat{x}$ points towards the wanted direction of orbit, $\hat{y}$ points radially up, and $\hat{z} = \hat{x} \times \hat{y}$.

The kinematic system $\hat{r}, \hat{t}, \hat{z}$ follows the rocket and is oriented such that $\hat{r}$ is the normalized radial vector, $\hat{z}$ is the same as in the cartesian system, and the tangential vector is $\hat{t} = \hat{r} \times \hat{z}$. In a circular orbit, $\hat{t}$ is parallel to velocity $\vec{v}$.

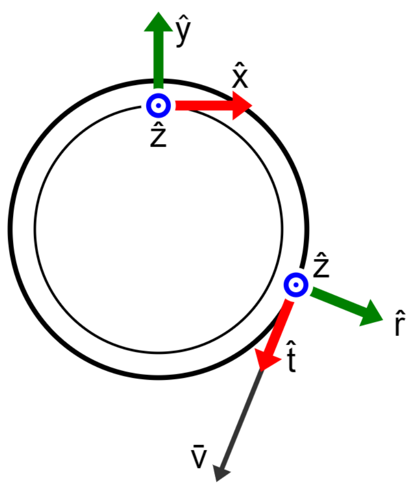


Figure 1: The instantaneous coordinates

From the kinematic system, we can derive altitude

$$h(\vec{r}) = |\vec{r}| - R_0$$

where $R_0$ is the equatorial radius of Earth, which is assumed to be spherical. From section 2.1 it is clear that we also need the maximum altitude obtained up until time $t$

$$h_{max} = \max\left\{h(\vec{r}(t')) : t' \in [0, t]\right\}$$

### 3.2 Initial conditions

For ease of simulation, the influence of launching from the equator is modelled as an initial horizontal velocity $v_{surf}$. Since the radius of Earth is denoted as $R_0$, then the initial conditions in the cartesian coordinate system are:

$$\vec{r}(0) = R_0 \cdot \hat{y}, \quad \vec{v}(0) = v_{surf} \cdot \hat{x} + 5\,{}^{m}\!/\!{}_{s}\,\hat{y}, \quad m_e = 0$$

Launch from the equator result in $v_{surf} \approx 460 m/s$ and launch from the poles result in $v_{surf} = 0 m/s$. Adding a slight upward velocity was necessary to start the simulation.

### 3.3 Atmosphere

The atmosphere in the simulation is modelled according to the U.S Standard Atmosphere (Braeunig, 2014), for altitudes below 90 km. In order to make the simulation more efficient, the model is linearly interpolated between points at altitudes spaced 1 km. The model produces the atmospheric density $\rho(h)$ and ambient pressure $P(h)$. Above 90 km, the pressure and density is linearly interpolated from $P(90\text{km}), \rho(90\text{km})$ down to 0 at 100 km.

Furthermore we assume that the atmosphere, independently of altitude, moves at a velocity

$$\vec{v}_{atm} = v_{surf} \cdot \hat{t}$$

The modelling of velocity in practice has little effect on the simulation, as the rocket will quickly lift itself to an altitude where pressure tends to zero, but it will simplify things at low altitudes.

### 3.4 Forces

In this model, it is assumed that three forces are acting on the rocket: thrust $T$, aerodynamic drag $D$ and gravity $G$.

#### 3.4.1 Gravity - G

Gravity is modelled based on the Newtonian formulation, resulting in a force

$$\vec{G}(\vec{r}, m_e, t) = -m(m_e, h_{max}(t))\frac{\mu}{r^2}\hat{r}$$

where r is the distance to the center of Earth from the rocket, and $\mu = GM \approx 3.986 \cdot 10^{14} m^3 s^2$ is the standard gravitational parameter.

### 3.4.2 Aerodynamic drag - D

From the atmospheric model described in section 3.3, we can first get the wind-relative velocity

$$\vec{v}_{atm,rel}(\vec{v}) = \vec{v} - \vec{v}_{atm}$$

Under the assumption that the the drag equation holds for the entirety of the ascent, then drag becomes

$$\vec{D}(\vec{r}, \vec{v}, m_e, t) = -\frac{m(m_e, h_{max}(t)) \cdot c_d \cdot A \cdot \rho(h(\vec{r}))}{2} |\vec{v}_{atm,rel}(\vec{v})|^2 \, \hat{v}$$

where $A$ and $c_d$ are parameters described in section 2.1.

### 3.4.3 Thrust - T

To abstract the guidance algorithms from the physics model, it is assumed that guidance controls throttle as

$$\eta = \eta(\vec{r}, \vec{v}, m_e, t) \in [0, 1]$$

and angle of thrust (AoT) from the vertical $\hat{r}$ as

$$\theta = \theta(\vec{r}, \vec{v}, m_e, t) \in [0, \pi]$$

Throttle and AoT interact such that thrust becomes

$$\vec{T}(\vec{r}, \vec{v}, m_e, t) = \left(\cos(\theta) \cdot \hat{r} + \sin(\theta) \cdot \hat{t}\right) \cdot F(m_e, P(h(\vec{r}))) \cdot \eta$$

A slight correction needs to be introduced however, to account for maximum acceleration. Let this factor be

$$\gamma = \begin{cases} \frac{m(m_e, h_{max}(t)) a_{max}}{\left|\vec{T}(\vec{r}, \vec{v}, m_e, t)\right|} & : \quad \left|\vec{T}(\vec{r}, \vec{v}, m_e, t)\right| > m(m_e, h_{max}(t)) a_{max} \\ 1 & : \quad otherwise \end{cases}$$

This defines corrected thrust (and mass flow)

$$\vec{T}_c(\vec{r}, \vec{v}, m_e, t) = \gamma \, \vec{T}(\vec{r}, \vec{v}, m_e, t)$$
$$\dot{m}_c = \gamma \, \dot{m}$$

### 3.5 Differential equations

The combination of gravity, aerodynamic drag, and (corrected) thrust give the equation for acceleration

$$\vec{a}(\vec{r}, \vec{v}, m_e, t) = \frac{1}{m(m_e, h_{max}(t))} \left(G(\vec{r}, m_e, y) + D(\vec{r}, \vec{v}, m_e, t) + T_c(\vec{r}, \vec{v}, m_e, t)\right) \tag{1}$$

Furthermore, the equation for mass is

$$\frac{dm_e}{dt} = \dot{m}_c \geq 0 \tag{2}$$

These are the equations that the simulation needs to integrate.

### 3.6 Calculation of apoapsis and periapsis

Apoapsis and periapsis are calculated in every time step, as a sort of look-ahead, that lets the control algorithm act accordingly. These can be calculated with the equations described in (Michele, 2013):

$$\begin{aligned}
\vec{h} &= \vec{r} \times \vec{v} && \text{(angular momentum)} \\
e &= \frac{1}{\mu} \left| \left(v^2 - \frac{\mu}{r}\right) \hat{r} - (\vec{r} \cdot \vec{v})\, \hat{v} \right| && \text{(eccentricity)} \\
E &= \frac{v^2}{2} - \frac{\mu}{r} && \text{(specific orbital energy)} \\
a &= -\frac{\mu}{2E} && \text{(semi-major axis)} \\
r_{ap} &= (1 + e)\, a && \text{(apoapsis)} \\
r_{pe} &= (1 - e)\, a && \text{(periapsis)}
\end{aligned}$$

# 4 Ascent guidance

The ascent guidance implemented in the simulation can be categorized into five phases: *liftoff*, *kickpitch*, *gravity turn*, *orbital insertion* and *terminal guidance*. These phases are characterized by parameters that control $\theta$ and $\eta$, which are optimized by a method described in section 6.

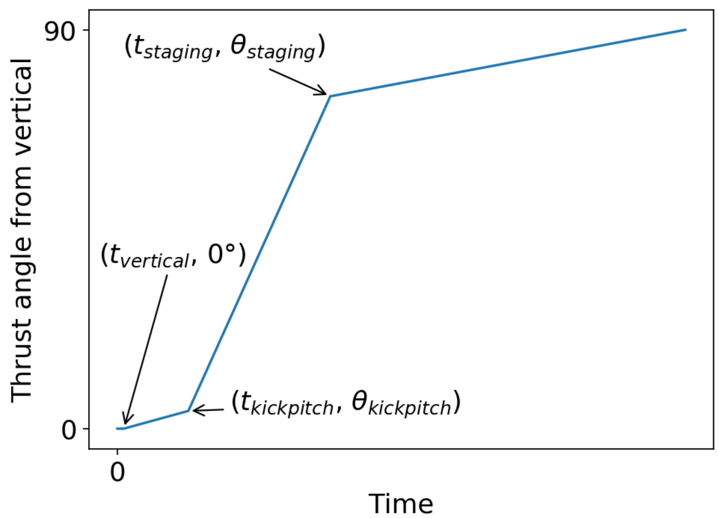


Figure 2: Typical thrust angle from vertical (intentionally without units)

### 4.1 Liftoff $(0 < t < t_{vertical})$

During this phase, the trajectory is entirely vertical: $\theta = 0$ and $\eta = 1$.

### 4.2 Kickpitch $(t_{vertical} < t < t_{kickpitch})$

During this phase, the trajectory starts pitching towards the horizon slightly. Throttle is still $\eta = 1$. The angle $\theta$ is linearly interpolated between 0 and the defining parameter $\theta_{kickpitch} = 4°$ according to

$$\theta(t) = \theta_{kickpitch} \frac{t - t_{vertical}}{t_{kickpitch} - t_{vertical}}$$

### 4.3 Gravity turn $(\dot{m}t_{kickpitch} < m_e(t) < m_{prop,0})$

During the time between the end of kickpitch and the time of staging (when the expended mass is equal to the propellant in the first stage), $\eta = 1$ and the angle $\theta$ is linearly interpolated between $\theta_{kickpitch}$ and the defining parameter $\theta_{staging}$ according to

$$\theta(t) = \theta_{kickpitch} + (\theta_{staging} - \theta_{kickpitch}) \frac{m_e(t) - \dot{m}t_{kickpitch}}{m_{prop,0} - \dot{m}t_{kickpitch}}$$

### 4.4 Orbital insertion ($m_{prop,0} < m_e(t) < m_{prop,0} + m_{prop,1}$)

This phase lasts until terminal guidance is triggered. Throttle is $\eta = 1$. The angle $\theta$ is interpolated from $\theta_{staging}$ to 0 according to

$$\theta(t) = \theta_{staging} \left(1 - \frac{m_e(t) - m_{prop,0}}{m_{prop,1}}\right)$$

### 4.5 Terminal guidance

Terminal guidance is necessary to achieve a circular orbit within this simulation and previously mentioned ascent guidance parameters. It consists of a PI-controller that attempts to cancel out any vertical velocity. It is triggered, and overrides the orbital insertion phase, once certain conditions are fulfilled:

$$(h > g_{ap} - 55) \quad \textbf{or} \quad (h > 100 \text{ and } |\vec{v} \cdot \hat{r}| < 50)$$

Within terminal guidance, $\theta$ is controlled with the regulator

$$\theta = 5 \cdot e(t) + 5 \int_{t_{triggered}}^{t} e(t')dt', \quad e(t) = \vec{v}(t) \cdot \hat{r}(t)$$

where $t_{triggered}$ is the time at which terminal guidance was triggered.

Furthermore, throttle is controlled by

$$\eta = \begin{cases} 0 & : \quad r_{pe} > g_{pe} - 2km, \quad g_{ap} - 2km < r_{ap} < g_{ap} + 3km \\ 0 & : \quad r_{pe} > g_{pe} - 2km, |\vec{v} \cdot \hat{r}| < 15km \\ 0.05 & : \quad r_{pe} > g_{pe} - 10km \\ 0.25 & : \quad r_{pe} > 0km \\ 0.5 & : \quad r_{pe} > -100km \\ 1 & : \quad otherwise \end{cases}$$

If $\eta = 0$, the simulation ends in an acceptable orbit.

## 5 Integration

The integration of equations (1) and (2) described in section 3.5 is done using a 4th order Runge-Kutta integrator. This choice was made, because even though RK4 is not stable in terms of energy, it is simple to implement and produces low errors for the relatively short time spans simulated during a rocket launch (in the order of 500 seconds). Also, relatively few large time steps are required to get a stable simulation compared to other methods, which is very useful when running an optimization algorithm that requires many (several tens of thousands) simulations.

The choice of time step for the optimization step is detailed in the next section, but the final simulation is run in scaled real-time to run an animation: such that we do $N$ steps per animation frame $\Delta t_f = {}^{1s}/_{60}$, resulting in a time step of

$$\Delta t = \frac{\Delta t_f}{N} = \frac{1}{60\,N} s$$

The simulation is then run as an animation with speed-up $n$, resulting in $n \cdot N$ time steps taken per frame. Going much higher, up to 10x more steps per frame, would be computationally possible but introduces numerical inaccuracies. Being able to run the simulation in real-time means that it could also, in theory, be run on a real launch vehicle.

# 6 Optimization of parameters

To optimize parameters, an error function has to be established. From section 1, we know that the goal is to reach a specific periapsis ($g_{pe}$) and apoapsis ($g_{ap}$). Let the periapsis from each simulation be $r_{pe}$, and the periapsis be $r_{ap}$. As the terminal guidance algorithm works by eliminating vertical velocity, after reaching maximum altitude, we also want to make sure to use the maximum altitude reached ($h_{max}$). To make sure the terminal guidance algorithm works well, we also want to use the altitude at which orbit is reached ($h$). Since the project goal is to look at propellant usage for equatorial versus polar launch, it would be good to also use the propellant remaining after orbit insertion ($m_{final}$).

This lets us define the error function

$$E(r_{pe}, r_{ap}, g_{pe}, g_{ap}, h, h_{max}, m_{final}) = \frac{(r_{ap} - g_{ap})^2}{(10)^2} + \frac{(r_{pe} - g_{pe})^2}{(10)^2} + \frac{(h_{max} - g_{pe})^2}{(10)^2} + \frac{(h - g_{pe})^2}{(10)^2} + m_{final}$$

This error function is used to score control algorithm parameters using a randomized optimization method, in which the ascent control parameters described below are randomly perturbated from the currently best choice of parameters using specific random radii and scaling, such that for each one: $parameter_{i+1} = parameter_i + scale \cdot \text{random}(-radii, radii)$

| Parameter | Random radius |
|---|---|
| $t_{vertical}$ | 1 |
| $t_{kickpitch}$ | 4 |
| $\theta_{staging}$ | 2 |
| $a_{max}$ | 0.1 |

For every perturbation iteration $i$, a simulation is run at a specific timestep $\Delta t$ and iteration-specific scaling $scale = s_{par}(5 - (i \bmod 5))/5$.[1] If the result is scored lower than the currently best score, the currently best choice of parameters are set to the perturbated ones. If the result is not scored lower, the perturbation is discarded. This is done for different sets of time steps and scaling factors, progressively going to a finer time step, until either a max count of attempts have been reached or a minimum score is achieved according to the following sequence:

| Min score | Max attempts | $\Delta t$ | Parameter specific scale ($s_{par}$) |
|---|---|---|---|
| 10 | 10000 | 1 | 30 |
| 5 | 10000 | 1 | 15 |
| 5 | 10000 | 0.5 | 4 |
| 2 | 10000 | 0.5 | 2 |
| 1 | 10000 | 0.2 | 1 |
| 0.2 | 1000 | 0.1 | 1 |

The parameters that is considered best at the end of optimization is then used in a more accurate simulation to produce results.

# 7 Results

## 7.1 Used parameters

To get a sense for what the parameters should be, the values for SpaceX's Falcon 9 Block 5 rocket was used as a starting point [2]. The values that had to be corrected was thrust on the second stage, which was increased by 20%, in order for optimization to converge.

[1]This led to faster convergence, since some iterations are more precise

[2]These values are easily found on Wikipedia

| Parameter | Unit | Value |
|---|---|---|
| $m_{prop,0}$ | kg | 410900 |
| $m_{dry,0}$ | kg | 22200 |
| $F_{sl,0}$ | kN | 7605 |
| $F_{vac,0}$ | kN | 8226 |
| $I_{sp,sl,0}$ | s | 282 |
| $I_{sp,vac,0}$ | s | 311 |
| $m_{prop,1}$ | kg | 107500 |
| $m_{dry,1}$ | kg | 4000 |
| $F_{sl,1}$ | kN | 1200 |
| $F_{vac,1}$ | kN | 1200 |
| $I_{sp,sl,1}$ | s | 282 |
| $I_{sp,vac,1}$ | s | 348 |
| $m_{fairing,0}$ | kg | 1700 |
| $h_{deployment,0}$ | km | 100 |
| $m_{payload}$ | kg | 10000 |
| $c_d$ | 1 | 0.2 |
| $A$ | $m^2$ | 19.63 |
| $R_0$ | km | 6374 |
| $\mu$ | $m^3 s^2$ | $3.986004418 \cdot 10^{14}$ |

### 7.2 Analysis

We'll start by looking at how well the guidance algorithm worked. Three goal orbits was optimized in the program, and we'll look at both how close it guided the rocket into the goal orbit as well as how much propellant was consumed.

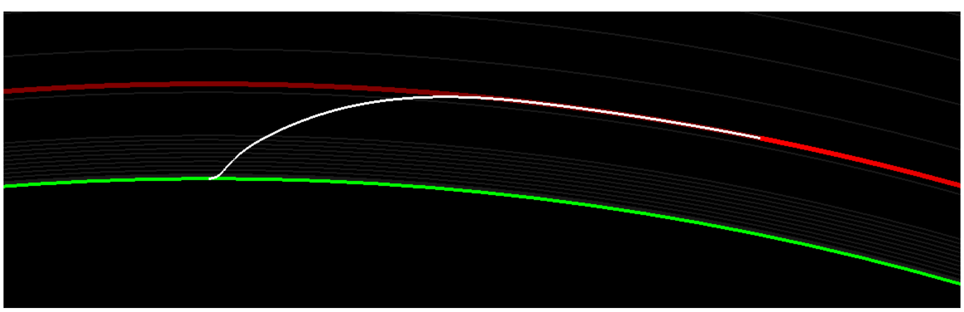

Figure 3: Typical ascent to 220 km in a non-rotating Earth-centric reference frame. Equatorial launch. Trajectory shown in white. Final orbit shown in red.

It's clear from the trajectory in figure 3 that the trajectory starts out mostly vertical, where the atmosphere is densest. This prevents too much losses due to drag.

### 7.3 Orbit insertion accuracy

| | | Achieved | |
|---|---|---|---|
| Surface motion ($v_{surf}$) | Goal altitude | Perigee | Apogee |
| 0 m/s | 170 km | 168.45 km | 169.52 km |
| 460 m/s | 170 km | 168.03 km | 170.26 km |
| 0 m/s | 220 km | 218.49 km | 220.77 km |
| 460 m/s | 220 km | 218.56 km | 219.21 km |
| 0 m/s | 275 km | 274.41 km | 275.25 km |
| 460 m/s | 275 km | 275.30 km | 275.78 km |

It's clear that the algorithm works astonishingly well, with insertions just a few km in difference from the goal. It also works well for both equatorial and polar launch.

### 7.4 Propellant consumption

| | | Propellant | |
|---|---|---|---|
| Surface motion ($v_{surf}$) | Goal altitude | Consumed | Remaining |
| 0 m/s | 170 km | 512414 kg | 5986 kg |
| 460 m/s | 170 km | 509838 kg | 8562 kg |
| 0 m/s | 220 km | 513645 kg | 4755 kg |
| 460 m/s | 220 km | 511933 kg | 6467 kg |
| 0 m/s | 275 km | 513562 kg | 4838 kg |
| 460 m/s | 275 km | 512208 kg | 6192 kg |

The propellant consumption was highest for the highest orbit goal, a reasonable expectation, but the difference is the greatest for the lowest orbit goal simulated. This is shown in the following table, comparing propellant usage for equatorial launch relative to polar launch:

| | Propellant | |
|---|---|---|
| Goal altitude | Consumed | Remaining |
| 170 km | 99.40% | 143 % |
| 220 km | 99.67% | 136 % |
| 275 km | 99.74% | 128 % |

## 8 Discussion

By the results achieved, it's clear that orbital launch from the equator is better than launch from the poles. It is however interesting that the propellant consumption, for the orbit insertions simulated, is higher for polar launch to 220 km than it is for polar launch to 275 km. This is likely an optimization problem, rather than related to the physics of the problem. A better error function might solve this, but it could also be that the optimization function finds local minimums rather than a global minimum. A solution to this would be to sometimes run the algorithm for entirely random parameters, or increase the random scaling temporarily for every nth iteration.

One must be careful just looking at the results, as one might say that it is not by a huge margin that equatorial launch is better, but that's where the tyranny of the rocket equation comes into play. This can be shown by introducing Tsiolkovsky's rocket equation

$$\Delta v = v_{exhaust} \ln \frac{m_{final}}{m_{initial}}$$

Solving for $m_{initial}$ gives

$$m_{initial} = m_{final} \exp\left(-\frac{\Delta v}{v_{exhaust}}\right)$$

If the goal is to accelerate 10000 kg of payload into a new velocity of 6 km/s with the second stage defined in the simulations, then reducing the propellant consumption by 2000 kg allows the usage of 10000 kg less propellant in total.

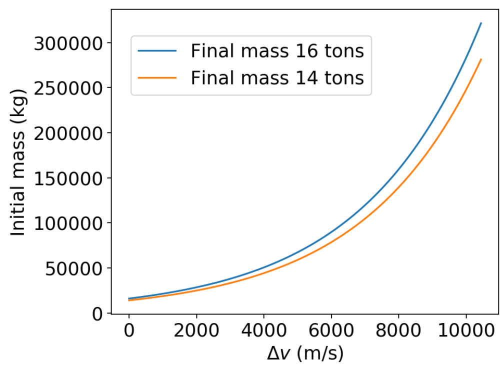


Figure 4: The initial mass necessary for the $v_{exhaust}$ used in simulation, for different final masses

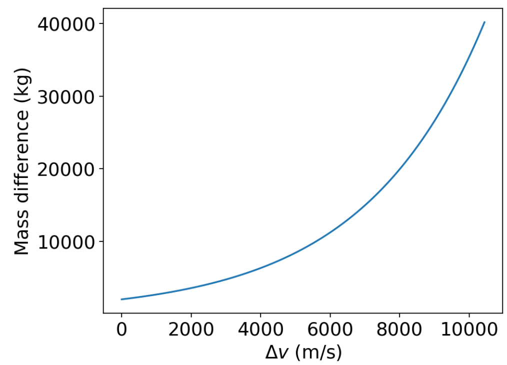


Figure 5: The difference in initial mass for final masses of 14 and 16 tons

It was also interesting to see that the *very* simple guidance algorithm was able to guide the rocket into an orbit very close to the goal. It is definitely just scraping the surface of what is possible.

## References

Braeunig, R. A. (2014). *Basic of Space Flight: Atmospheric Models.* Retrieved 2019-01-03, from `http://www.braeunig.us/space/atmmodel.htm#USSA1976`

Michele, R. (2013). *Determining Orbital Elements.* Retrieved 2019-01-14, from `http://www.aerospacengineering.net/?p=1638`

NASA Glenn Research Center. (2015a). *General Thrust Equation.* Retrieved 2019-01-17, from `https://www.grc.nasa.gov/www/k-12/rocket/thrsteq.html`

NASA Glenn Research Center. (2015b). *Specific Impulse.* Retrieved 2019-01-17, from `https://www.grc.nasa.gov/www/k-12/airplane/specimp.html`