Proceedings of EOC'21:
ME8710: Engineering Optimization
Clemson University
Clemson, SC, November 30-December 2, 2020

EOC2021-13

OPTIMIZATION OF THE CONTROLLER FOR A ROCKET AIR BRAKE SYSTEM

Severin Beger
Technical University Munich / Clemson University
Pestalozzistr. 5, 80469 Munich, Germany / 702 Cambridge Creek Ct, 29631 Clemson, SC
severin.beger@tum.de / sbeger@clemson.edu

Abstract
To guide a rocket to a desired apogee an active braking system must be designed. For the Spaceport America Cup the Clemson Rocket Engineering team aims to build such a system. This paper introduces the setup of such a system and describes how Bellman's principle of optimality can be used to precisely guide the rocket with the air brakes to its destination. First, we explain our assumptions on the dynamic model of the rocket. Then we show how to calculate an optimal controller for the system. Finally, we show its application in a software in the loop environment All findings here are meant to be a proof of concept.

1. INTRODUCTION

The Spaceport America Cup is an annual prestigious competition for rocket engineering teams of colleges all over the world. Each team builds a rocket and launches it. Professional judges then rate the rockets performance. A key aspect is the prediction of the rocket's apogee. Precise simulations of the rocket's dynamics are difficult due to probabilistic aerodynamic effects like wind and highly complex behavior of airflow and drag around the rocket. An active braking system allows to bring a rocket to an intended height. Clemson Rocket Engineering (CRE) has decided to build such a system. However, the center piece of this subcomponent of the rocket is not its hardware, but its control scheme. Essentially a problem known as a terminal control problem has to be solved. Two strategies are usually implied for solving these types of problems: either some sort of model predictive control is applied, where the system tries to adapt its own model midflight to track a certain predefined trajectory. Or on the other hand some sort of optimal controller is designed, which tries to minimize a performance measure defined by the control engineer. The first concept requires high computing efforts on the onboard computer and is therefore not suitable for our system. Thus, we focus on the second method and use this paper to setup an optimal controller. As already mentioned, having a high-fidelity model of the rockets dynamics is difficult and the dynamics experience probabilistic effects. Thus, we follow a discrete optimization scheme, namely dynamic programming, based on Bellman's principle of optimality. This discrete scheme lets us consider other optimization goals as well. Therefor, we not only aim to reach the desired apogee, but we also want to minimize the load on the braking system itself. While we setup the problem, we need to make sure to consider all bounds, which are not only the physical limitations of the rockets, but also the discrete nature of the computing system as well as probabilistic aerodynamic effects.

Before we get started with the control design, we review important concepts of rocket dynamics and optimal control to give the reader an overview. After that, we present our simplified dynamic model of the rocket and explain our assumptions. The actual controller design is described next. Finally, we present results of simulations, where we tested calculated controllers within another software.

2. STATE OF THE ART

To be able to design an optimal control law for the air brake system of a rocket, we need to understand its dynamics and optimal control theory. Therefore, we use the following passages to review some well-known and some recent literature of the matter. We will start by investigating rocket dynamics and try to get an understanding for modelling aerodynamic drag. Afterwards we pose a quick overview of optimal control theory and review dynamic programming and Pontryagins' minimum principle.

A model rocket usually consists of three phases of flight. These are powered flight, coasting flight and parachute flight (cf. [1]). Figure 1 shows these in a graph. From start point 1 onwards the motor drives the rocket up to burnout at point 2. From there, the rocket goes into coasting, reaching its apogee at point 3. After that the rocket starts descending. At point 4 its parachute is deployed to bring it safely back to the ground at landing spot 5. If the parachute is not deployed, the rocket goes ballistic and flies to point 4', where it hits the ground.

The phase, that interests us, is the coasting phase from point 2 up

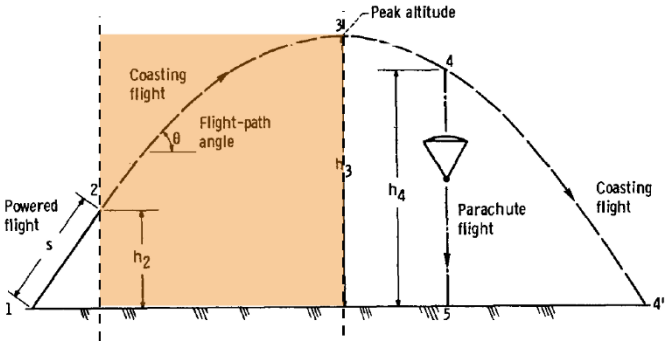

Figure 1 - Flightpath of a typical model rocket. Original from [1], slightly modified.

to the apogee at point 3. In between these points we aim to actively slow the rocket down, so it reaches the desired apogee precisely. To do so, we need to understand the dynamic behavior of the rocket. The general forces acting on the vehicle are the gravity pull of the earth as well as the aerodynamical forces of lift and drag. In Newtonian mechanics one can simply write

$$m\boldsymbol{a}_R = \boldsymbol{F}_P + \boldsymbol{F}_g + \boldsymbol{F}_D + \boldsymbol{F}_L \,. \tag{2.1}$$

This equation only considers the dynamics of a point mass. To consider rotations we would need to add momentum equations. However, to keep it simple, we focus on the translational motions of the rocket in a plane, which is a decent approximation. For more complex considerations, we refer the reader to [2]. Furthermore, we can consider the rocket to have a stable flight without a consistent angle of attack α. This removes any lift. While the precise description of the gravitational pull is rather simple, we need to investigate possible descriptions for the drag acting on the rocket. From [3] we know, that different types of drag act on the vehicle, namely friction and pressure drag. The first one is due to the viscosity of air and the resulting friction on the surfaces of a body traveling through the fluid. The second one occurs due to a disbalance of pressure, when a body moves through air or any other fluid. Figure 2 shows this for a tennis ball, moving to the left.

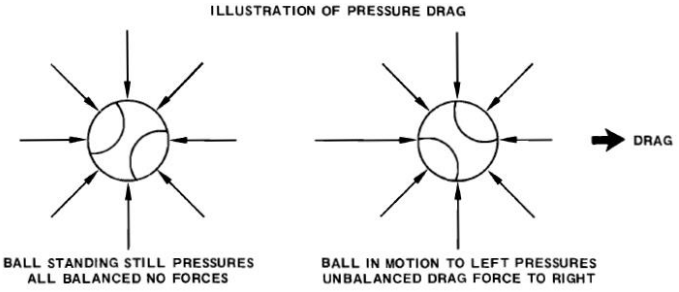

Figure 2 - Illustration of pressure drag on a moving tennis ball. Taken from [3].

These types act on any part of the rocket and must be considered for the tube, the nose cone and the fins. An especially strong type of pressure drag occurs on the base of the rocket after the motor burned out. Usually, the drag of a rocket or any flying vehicle is determined in a wind tunnel. [4] poses a good example how to do so for a model rocket. The experiments yield a function for the rockets drag coefficient C_D. This in turn may be used to calculate the resulting drag force as a function of the current air density $\rho(h)$, the rockets velocity $\boldsymbol{v}$ and the effective area $A_\perp$. The letter is a characteristic area, which the drag acts on. In the case of a rocket usually the circular area of the body tube is taken as the main reference area. Finally, the drag equation can be written as

$$F_D = \frac{1}{2}\rho(h) C_D v^2 A_\perp \,. \tag{2.2}$$

However, most often we want to simulate the rocket before we build a physical model to test in the wind tunnel. Or we don't have access to one. In these cases, we need to use experimental formulas for the drag coefficients of the single components of the rocket. In the end, the single components like the nose cone, the tube, the base or the fins are added up together with another factor for interference drag. Thus, we can get an estimation of the drag forces for simulations or event for the design of control laws. In [3] several suggestions for these empirical formulas are given. However, as drag is highly susceptible to atmospheric influences like wind and sometimes behaves randomly, these types of models cannot be very accurate. Additionally, we want to influence the drag of the rocket systematically, to slow it down and reach a specific apogee without risking the vehicles stability. Thus, we want an accurate description of the change of drag when we change our actuation system. Wind tunnel tests with different angles of the controllable air brake fins and with different speeds lead to a table of data, from which we can interpolate in the actual case of flying the rocket.

Now we want to address optimal control theory, as this is the core topic of this report. The starting point of any optimal control problem is a dynamic model of the type

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u). \tag{2.3}$$

This is a set of ordinary differential equation, possibly nonlinear, in a state space form with states $\boldsymbol{x}$ influenced by a control input *u(t)*. Now, to introduce optimality to the control, a performance measure needs to be defined. Usually it looks like this:

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \,. \tag{2.4}$$

This performance measure is a functional, a function of functions. It considers the state of the system at the final time t_f in function *h* and the development of the state and the input over the whole considered time in function *g*. Now, the goal is to find an admissible, optimal input $\boldsymbol{u}^*$, which minimizes *J* while proposing an admissible optimal trajectory of the states $\boldsymbol{x}^*$. A complete introduction to optimal control theory can be found in [5]. A few remarks concerning this search for optimality: In general, it is not guaranteed that there exists an optimal solution. However, there are few theorems to prove the nonexistence, which is why engineers try to search for an optimum solution without checking its existence. Furthermore, an optimal solution may not be unique. This is tangled with Pareto's principle and is a useful property for engineers, as it allows to introduce helpful notions of robustness and other features. However, this usually complicates computations. Another remark is, that we look for a global minimum of *J*. Only this can be considered optimal in our proposed sense. Finally, if such an approach with a performance measure is omitted or some pieces are left out, the resulting optimum may be infeasible. This is most often the case when the

boundedness of the input is not considered. Then the optimization may get to a result, which proposes infinite impulses of $\boldsymbol{u}$. These are in general physically impossible.
Different kinds of performance measures exist e.g., for time optimality, for a minimum control effort, for optimal tracking or for terminal control problems. The latter aligns well with our problem of achieving a target apogee with the model rocket precisely. A respective performance measure looks like

$$J = (\mathbf{x}(t_f) - \mathbf{r}(t_f))^T \mathbf{H} (\mathbf{x}(t_f) - \mathbf{r}(t_f)) . \quad (2.5)$$

This equation denotes a quadratic product of the difference of the state at final time $\boldsymbol{x}(t_f)$ to the desired state $\boldsymbol{r}(t_f)$. The optimum of this will be zero, as the equation is quadratic and upper or lower deviations both enlarge the performance measure *J*. The matrix **H** is symmetric, positive definite and real. It can be used to prioritize certain states or neglect other completely.
There are two main procedures to arrive at a solution for this minimization problem, namely dynamic programming and Pontryagin's minimum principle. The latter is based on the calculus of variations, which is an analogy to calculus for functionals. Basically, one tries to find a trajectory of the states for which

$$\delta J(\mathbf{x}^*, \delta \mathbf{x}) = 0 \quad (2.6)$$

for all $\boldsymbol{\delta x}$ is true. This means, any variation of the functional from of the optimal, extremal solution must vanish, similar to extrema of functions. If consequently used for the standard functional *J* one arrives at the famous Euler Lagrange equations:

$$\frac{\delta g}{\delta \mathbf{x}}(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), t) - \frac{d}{dt}\left(\frac{\delta g}{\delta \dot{\mathbf{x}}}(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), t)) \right) = 0 . \quad (2.7)$$

These equations pose some difficulties, as they are a set of nonlinear differential equations with split boundary values. Most numerical solvers can handle this type of equations, if the known conditions are on one side of the spectrum e.g., a given initial value for $t_0, \boldsymbol{x}(t_0)$ and $\dot{\boldsymbol{x}}(t_0)$. However, in most optimal control the engineer knows the state at initial time and at final time. This splits up the boundary values and makes the equations even tougher to solve.
The previously defined equations do not consider constraints, for example on input values. In reality, every system has some physical limitations and thus constraints. This is why Pontryagin and his group designed a set of equations, which allows to consider constraints. Essentially, they define a Hamilton function

$$H(\mathbf{x}(t), \mathbf{u}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \lambda(t)^T \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) , \quad (2.8)$$

which enforces the system dynamics with the Lagrange parameters $\boldsymbol{\lambda}$. From there, the three equations

$$\begin{aligned} \dot{\mathbf{x}}^*(t) &= \frac{\delta H}{\delta \lambda} \; \mathbf{x}^*(t), \mathbf{u}^*(t), \lambda^* \\ \dot{\lambda}^*(t) &= -\frac{\delta H}{\delta \mathbf{x}} \; \mathbf{x}^*(t), \mathbf{u}^*(t), \lambda^*, t \\ H \; \mathbf{x}^*(t), \mathbf{u}^*(t), \lambda^*, t \; &\leq H \; \mathbf{x}^*(t), \mathbf{u}(t), \lambda^*, t \end{aligned} \quad (2.9)$$

for all $t \in (t_0, t_f)$ can be derived. The last equation states that one must find the minimum of the Hamilton function for all admissible inputs $\boldsymbol{u}$. This is a necessary but not sufficient condition. There are other things to consider, when applying these criteria of Pontryagin. They can be found in [5]. One point of interest may be how to add constraints on the states in the form of $f(\boldsymbol{x}(t), t) \geq 0$. This can be done, by adding an artificial state to the equation, which is a sum of all these constraints times a Heaviside function which is zero if the constraint is fulfilled. Thus, to minimize this artificial state, all constraints must be fulfilled.
A term, which is often put into the same context of optimal control is the bang bang controller. This type of controller is a typical result of a minimum time performance measure optimization and results in the input to be used to its maximum twice: one time in the beginning and once just before the final time in the opposite direction.
Using Pontryagins' minimum principle seems to be straight forward, but the resulting equations get difficult very fast and generally must be solved numerically. In time invariant, linear cases with dynamics like

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \quad (2.10)$$

it is possible to calculate a time variant optimal control law based on matrices $\boldsymbol{R}$ (weighting of the input), **B** (input matrix) and **P** (solution of the Matrix Riccati equation):

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\mathbf{x}(t) . \quad (2.11)$$

In [6] this linear approach is discussed extensively.

A different approach to optimal controllers was introduced by Bellman in the sixties. He stated the following, known as principle of optimality (from [5], p.53):

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Essentially, this means that one can go backwards from the optimal result in single steps and calculate costs for the transition from state $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$. The minimum value is the optimum value. If this is done recursively until the initial state is reached, an optimal path is created. This can be seen as a closed loop optimal control law. Furthermore, if the calculated step values are saved, the path for any other chosen step at a time after the initial step can be calculated to be optimal, meaning that one can calculate tables offline and react to changes in the environment online, while running the system, without a complete calculation of all possible routes.
This approach can be computed after discretizing the sets of the states and the inputs. One speaks about dynamic programming or Bellman's recurrence equation. The result of dynamic programming is a lookup table. In use, the system puts in the current state values and gets an optimal control value back. It is also possible to come up with analytical equations, the so-called Hamilton-Jacobi-Bellman equations, to calculate this numerically. We omit their exact definition here to keep this literature review short. As with Pontryagin's minimum principle, the equations become very complex differential equations and can only be solved approximately with numerical tactics. Thus,

we either get an exact solution to a discretized problem (Bellman's recurrence equation) or an approximate solution to the exact optimization equations (Hamilton-Jacobi-Bellman equations. It seems to be appealing for our problem, nonetheless. Looking for examples of aerospace problems, solved with optimality approaches, one may be interested to read [7], which

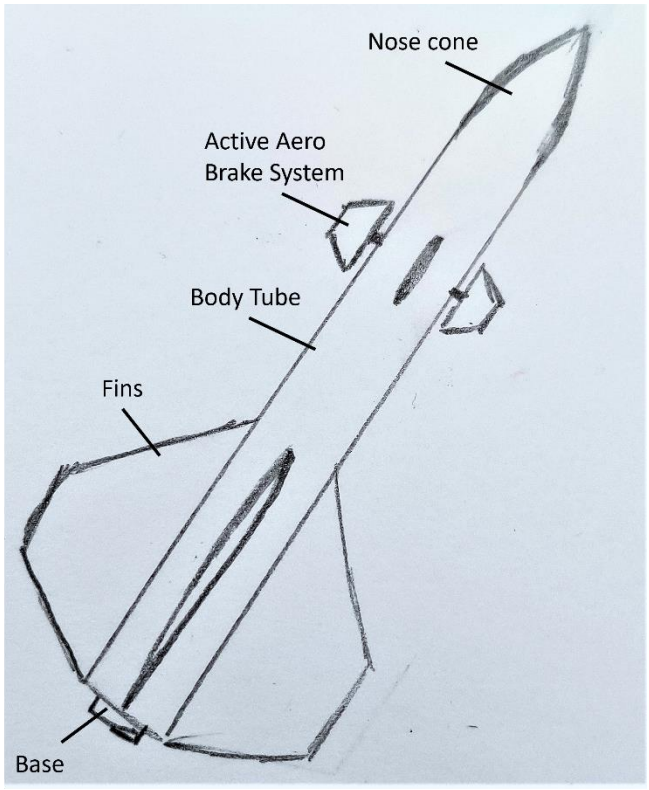

Figure 3: Sketch of the model rocket

addresses rocket landing with a fuel optimal approach. A little simpler is [8], which proposes strategies to maximize the speed of a rectilinear rocket trajectory.

When it comes to real control problems, often we have uncertainty as part of the overall dynamics. This can be considered somewhat in the optimality problem. An example can be found in [9]. This paper focuses on optimal control systems with constraints that have probabilistic features. They demonstrate their concepts on path planning, as well as on Mars and lunar landing problems.

Finally, we want to mention several papers of other rocketeers, who have inserted air brake systems into their model rockets. The teams in [10] and [11] have applied systems based on a model predictive approach. They try to track an offline calculated optimal trajectory. A very sophisticated approach can be found in [12]. It comprises three steps: firstly, a modeling step, which incorporates certain probabilistic external disturbances with the help of Monte Carlo simulations. Secondly the definition of an infinite horizon chance-constrained optimal control problem is described. Lastly, the actual application on the rocket with the help of lookup tables is explained. This last report serves as our main inspiration, although the mechanical design of the air brake is different.

3. MODEL OF THE ROCKET WITH THE AIR BRAKE SYSTEM

In this section we discuss the dynamic models and assumptions on the system, which we need to simulate the model rocket of the CRE club with a braking system. These equations underly all further calculations of the controller and its validation with a software in the loop approach. We have already introduced the main forces acting on a rocket in equation (1.1). When we validate our controller in the end, we also account for a turn rate of the rocket around its center of mass. This creates an angle Θ with respect to the ground level. To keep things simple, we base the turn rate on the ratio of the rockets heading velocity and the gravity part perpendicular to it. The equation then becomes

$$\dot{\Theta} \approx \tan^{-1}\left(\frac{\cos(\Theta) g \Delta t}{v}\right). \tag{3.1}$$

Equation (3.1) can now be added to a set of state space equations. Thus, we can simulate the rocket with three DOFs.

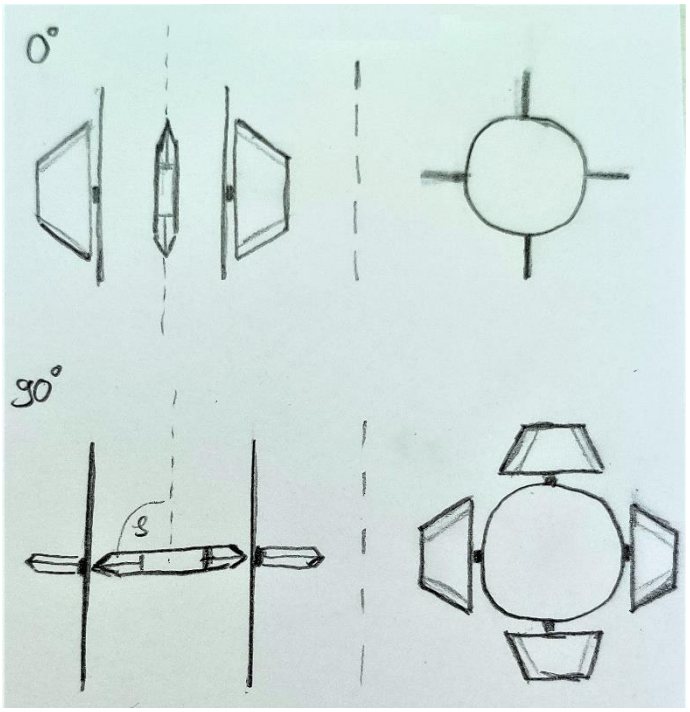

Figure 4: Sketch of the braking system in its inactive state (0 degree, top) and fully extended state (90 degree, bottom)

Now, let us focus on the dynamics while the braking system is active. Figure (3) shows a sketch of the rocket with the most important parts labeled. The Active Aero braking system is at the upper section of the body tube. The fins can be rotated to create higher or lower drag. This is the mechanism we want to exploit. The fins maximum ratings can be seen in figure (4) with a front view of the rocket on the left and a top view on the right. We need to find a suitable model of the rocket dynamics with the braking system, to simulate it and to build a controller on top of this data. To simplify further analysis and calculations for the controller development, we assume that our rocket has no side motion and simply flies vertically upward with close to a 90 degree angle to the ground. Now the system only has one degree

of freedom (going up or down) and the second order ODE from (1.1) can be reduced to a scalar formular. Additionally, we neglect any rotation of the rocket, as we are always vertical in our motion. In a further step we assume the stabilizing fins on the bottom of the rocket to quickly pull the rocket back into a streamlined flight mode if it ever gets displaced and builds up an angle of attack. This assumption is also valid for the bigger simulation with three degrees of freedom (horizontal, vertical and rotational motion). Additionally, we plan to use the Active Aero fins in a way that each pair cancels their respective induced lift. This lets us omit all lift terms from the dynamic equations. When it comes to propulsion, we know that it is only active for a certain time. The air brakes will not be activated before the burnout of the motor occurs. Thus, for calculating the rocket dynamics with the air brakes on, the dynamic system reduces to

$$ma_r = F_g + F_D \quad (3.2)$$

The drag force still is calculated by the drag equation (2.2), but what changes is the drag coefficient, as we can increase drag with angles of the Active Aero fins. The following
Equation (3.2) can easily be reformulated into a state space system, where the two states are the current height and velocity of the rocket, while the input is the Active Aero fins angle. Higher angles of the fins mean more drag. The system looks like follows:

$$\boldsymbol{x} = \begin{bmatrix} h \\ v \end{bmatrix} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u) = \begin{bmatrix} v \\ \frac{1}{m_R}(F_g + F_D(\boldsymbol{x}, u) \end{bmatrix}$$
$$u = \varphi \quad (3.3)$$

Note that the input is a singular angle. This makes the system easy to use, but in application, the same value for the angle has to be applied either positively or negatively to the four fins. This differentiation is necessary to cancel out lift and to not cause a rotation of the rocket around its own angle.
The desired state of the rocket can be denoted as

$$\boldsymbol{x}_d = \begin{bmatrix} 10,000ft \\ 0\frac{ft}{s} \end{bmatrix}. \quad (3.4)$$

This is ultimately the state we want to reach with our rocket. One can see in equation (3.3) how the drag force depends on the current state and the current angle of the braking fins. To model the dependency, usually experimental data from wind tunnels is used. Carefully calculated CFD data can work for a first shot and then be updated with real measurements on a test flight. As we have neither a wind tunnel and a rocket model of the right size nor time and expertise to set up the CFD simulation, we follow an analytic approach to calculate drag. The used equations are of empiric nature and based on data (see [3, 13]). As the derivation of the equations is rather lengthy, we omit it here. What these papers don't cover are models for an active moving fin. Said element creates primarily friction drag, when not used, but is intended to create high pressure drag when turned into the flow of air. The fins side is airfoiled, thus it creates close to no pressure drag, while it is at 0 degrees. A typical value for the coefficient of drag of airfoil shaped objects $C_{D0} = 0.045$. When the fins are fully displaced an are perpendicular to the stream of wind, a fitting drag value would rather be an average drag coefficient of a flat plate $C_{D1} = 1.28$. To model a function of drag coefficients based on the angle of attack of the Active Aero fins, we considered the effect of stalling and the following unpredictable disturbances. Thus, we modelled the function like follows.

(3.4)

Essentially, we have a quadratic rise of the drag coefficient with the angle up to 11 times C_{D0} until stalling at $\varphi_{lim} = 16^o$. After that we rise linearly up to C_{D1}. In simulation we also put on some white noise in the height of ±5% of the current drag coefficient value. The corresponding graph can be seen in figure 5. Note, that this value is a pure guess to be able to calculate a controller and proof the applicability of the concept. In reality, we need to get a good CFD simulation of the complete rocket with different angles of the air brakes system at different speeds. This can then be interpolated and used instead of this arbitrary model.
The overall drag coefficient, as it is used in equation (2.2), is a linear combination of the drag coefficient of the other body parts of the rocket, such as the big fins, the nose cone and the body tube, and the just calculated coefficient for the air brake system. To consider some interfering of geometries, we use the factor $K_F = 1.04$.

$$C_D = C_{D-R} + K_F C_{D_A A} \quad (3.5)$$

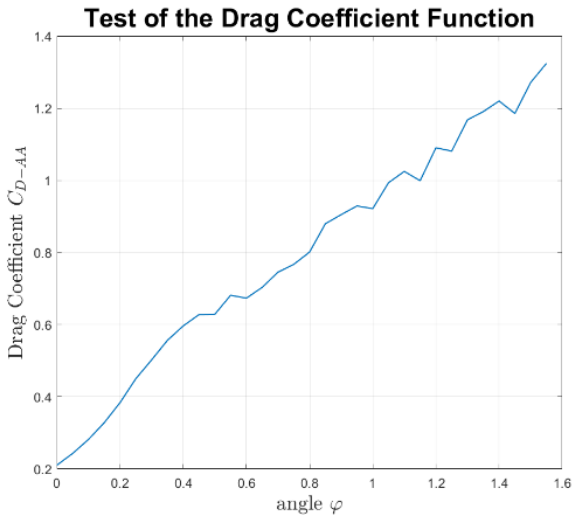

Figure 5 – Function of the fin angles drag coefficient C_{D-AA} over the angle φ

4. DESIGNING THE CONTROLLER

In this chapter we explain how we derived an optimal controller with dynamic programming for the air braking system Active Aero.
We start by setting up bounds for the problem. Then we discretize the feasible region of states and finally we iterate over all states backwards from the final state to possible starting states to get the optimal cost and the optimal fin angle for each state. In the end, we expect to have a lookup table, where we can find optimal angles for the air brakes for a measured height and velocity. The final control scheme in the final rocket looks like the block diagram in figure 6. We use the data of a barometer to

get a current height and speed value for the rocket. The latter is obtained by numerical differentiation. We also have an IMU on board, which may be used to improve the measurements of the barometer. The measured height and velocity are given to the controller and an angle is feedback to the rocket system. The controller for the actual servo motors, which adjust the angle of the air brakes, are not displayed. For the sake of simplicity, we assume that an angle can be set instantaneously.

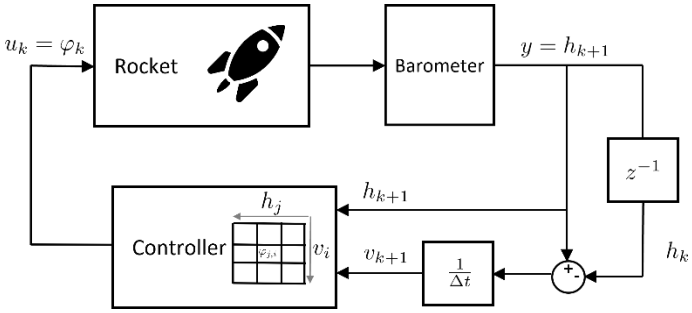

Figure 6 - Control circuit of the rocket

To generate velocity and height bounds for our problem we make use of a simulation without Active Aero and get a starting height after the burnout. From this point on, the air brake system is active. We determined $h_0 = 3280.8\ ft$ to be a good value. Now, we use the simplified dynamics from the previous section with the air brake system active. Our goal is to determine a minimum and a maximum bound for possible states of the system. If the rocket drops below the minimum bound in operation, the Active Aero system goes into the lowest possible braking state, meaning to 0 degrees. This ensures, that the rocket flies as high as possible and gets as close as possible to the desired height. The same idea applies for the upper bound, only that the system sets the air brake fins to 90 degrees, once it is above this line. Thus, it slows down with the maximum rating and gets as close to the goal as it can. Within the two bound is where the controller actually works and decides on the optimal angle of the brake fins. As we know, no simulation is perfect and we are aware that we have made several assumptions and simplifications, which are not true to the physical reality. Additionally, we did not account for probabilistic effects like wind. Thus, we need to widen our feasible region of states, giving the rocket a broader range of states to work on, even if we didn't model its behavior exactly. We decided to allow a deviation of the final height of ±5%. Thus, our lower bound marks the lowest possible speed at h_0, from which we can reach *9500ft* when the Active Aero system stays at 0 degrees. The upper bound on the other hand defines the sequence of states, which lead to a height of *10500ft* from a maximum velocity with maximally displaced braking fins. As we apply a discrete scheme, we need to separate these newly found bounds into a discrete grid of states. Thus we choose a number of velocities *N* for which we find a number of height values *H* between the respective height bounds of the discrete velocities. An exemplary plot with $N = 21$ and $H = 20$ can be seen in figure 7.

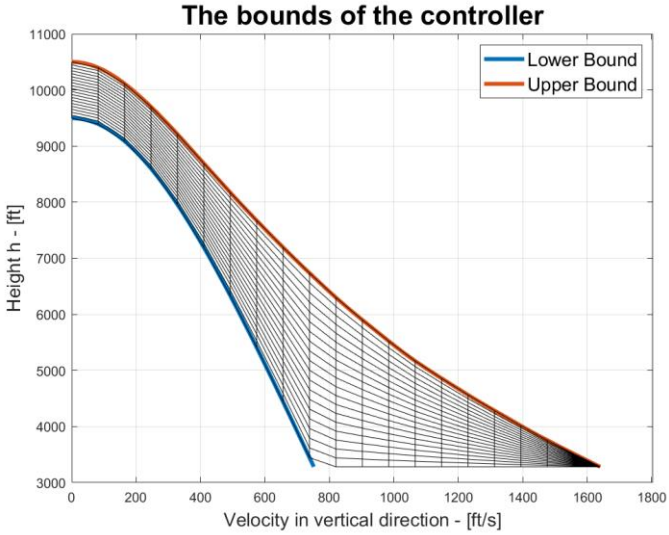

Figure 7 – Discretization of the feasible region for an exemplary rocket to reach the target apogee of 10,000ft ± 500ft

In a dynamic programming algorithm we determine states as optimal by the cost it takes to get to the final state. So, before we can apply this optimization scheme, we need to define a performance measure.

$$J(h_k, v_k, \varphi_k) = w_F \frac{(h_N - h_d)^2}{\Delta h_{max}^2} + \sum_{i=k}^{N-1} \left(w_1 \frac{\varphi_i v_i}{\varphi_{max} v_{max}} + w_2 \frac{\varphi_i^2}{\varphi_{max}^2} + w_3 \frac{(h_i - 0.5(h_{i,max} + h_{i,min})^2}{(0.5(h_{i,max} - h_{i,min})^2} \right) \tag{4.1}$$

We use four terms in total, all of which are normed. The first one is outside of the sum and is calculated for the final value only. Its' weight is the highest, as it is our foremost goal to reach the exact height of 10,000ft. The three terms in the sum have different purposes. The first punishes combinations of high velocities and high displacement angles of the brake system. This may seem odd at first, as we have the most potential to slow the rocket down at high speeds. However, this term reduces the load, which the fins must endure, significantly, which is our second optimization goal. The third term squares the current angle and thus also provides a punishment for high angles and potentially high loads. To make this even more suitable, we chose to make its weight based on the distance from the current step to the final step.

$$w_2 = \frac{N-i}{N} \tag{4.2}$$

This ensures that the rocket is punished for heigh fin angles at heigh initial speeds, but not as much at later stages when we get towards the final point and being able to brake is crucial. The

final term can be seen as a tracking term towards the middle of the feasible set, as it punishes states for their distance to the middle of the feasible heigh range at a velocity v_k.
With the performance measure defined, we have everything in place to calculate our optimal controller. The usual procedure is to go backwards from the last reachable state. The costs for the final node are calculated and saved in an array or matrix. Then one moves a level downward – in many cases the level are discrete times, but in our case, we move down in velocity levels. You can imagine going from left to right in the feasible set of image 7, one vertical line at a time. At every vertical line we loop over all states, in our case all H discrete heights of a velocity level v_k. For each we need to find the transition that leads to the state on the next higher velocity level, which has the lowest total costs to reach the final state h_F. To do this in an efficient manner, we sort all possible states of the next level with respect to their total costs. Then we can begin to check, if the current state can reach the state with the lowest cost. This means, we use the same strategy as when we determined the bounds of our overall controller: we simulate the system from a velocity v_k to v_{k+1} with the minimum input $\varphi = 0$ and the maximum input $\varphi = 90^o$. If our intended height value h_{k+1} lies within the two determined height, we know that reaching it is feasible. We then start an algorithm, similar to the golden section algorithm, which adapts the input and simulates the dynamics until we land within a defined limit around h_{k+1}. Then we consider the transition as determined. We save the necessary angle φ in a matrix as well as the new costs, which is the transition cost for a single step (the three terms of the sum in equation (4.1) for the current step) for our state k plus the cost to the goal state from the newly reached state $k + 1$. If the intended point was not reachable in the first place, we try the next state in the list. Since the list is ordered in ascending order of total cost to the goal, we know that we have found the optimal value once we determined a possible transition. By honoring this scheme, we follow Bellman's principle of optimality: the costs from each state to the goal are the lowest possible costs to get there. By iterating over all possible states, we end up with a matrix of sizes $[N - 1, H]$ with the saved angles φ in it. This is the controller and provides an optimal policy for a state of height and velocity. The cost matrix can be discarded after calculations, since we don't need it, while

```
FOR all velocity levels from N-1 to 1
    INITIALIZE list of states with costs of the previous
            velocity level
    SORT the list of states with respect to their total costs
    FOR all height levels from 1 to H
        FOR all states in the list
            DETERMINE if the state in the list is reachable
            IF YES
              FIND correct angle phi
              Break
            IF NO
              Continue
            END IF
        END FOR
        ADD previous cost value and transition cost for the
            current state
        SAVE new cost in cost matrix and new phi value in
            Controller
    END FOR
END FOR
```

Figure 8 - Pseudo code for the dynamic programming algorithm. FIND is a line search optimization.

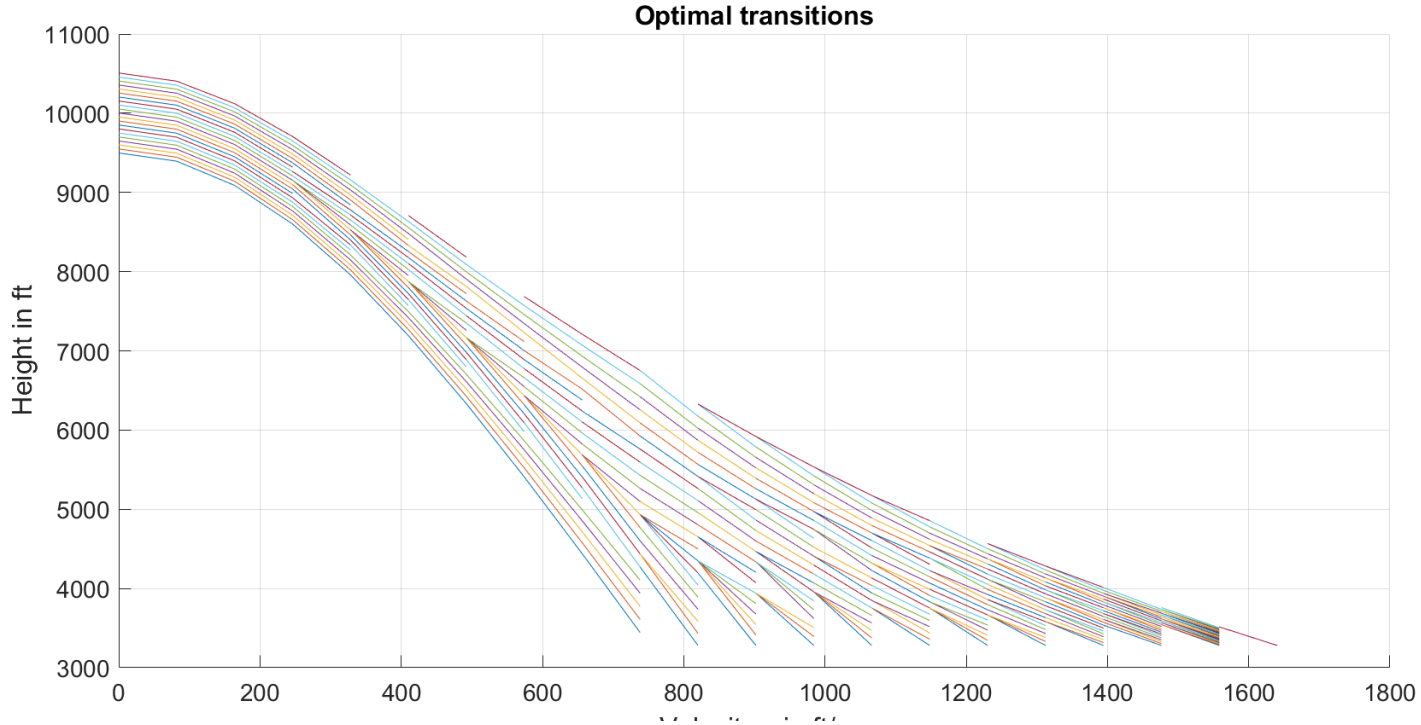

Figure 9 – Optimal transitions determined by the proposed dynamic programming algorithm for an exemplary rocket and a discretization of 21 steps for each velocity and height

running the real system.
When using the optimal controller on the actual system, we choose a new value for the Active Aero fins every time we reach on of the discrete v levels. The current height may not be exactly at one of the discrete points. Therefor we interpolate between the two closest values in the controller matrix. If the rocket height is already outside of the feasible region, we set the fin angle to the maximum or the minimum value, based on the height.
Figure 9 shows the path choices, which the algorithm determined to be optimal. When observing it closely, one can see how choice in the middle of the feasible set of states lead toward more centered solutions. Another noticeable behavior is the

5. SIMULATION RESULTS OF THE OPTIMAL CONTROLLER

As shown above in figure 9 we created an optimal controller for an exemplary rocket with *Matlab* 2021a. It's model data can be seen in the following table. This controller was integrated in a more sophisticated simulation of the rocket with three degrees of freedom (two translational ones and one rotational one). We used *Simulink* to integrate the controller and simulated the complete dynamical system. One additional difference of the simulation to the one used to calculate the optimal controller is the calculation of the coefficient of drag. This time, we did not use the analytic approach from before, but used drag coefficient data of another simulation software *RASAero* II. For each time step, we interpolated its values to generate a new C_{D-R} value. Our analytical approach gave greater values than this one. We chose to use a different approach to test the controller for robustness.
The coefficient of drag for the Active Aero system stayed the same. The results of these simulations are promising. Figure 10

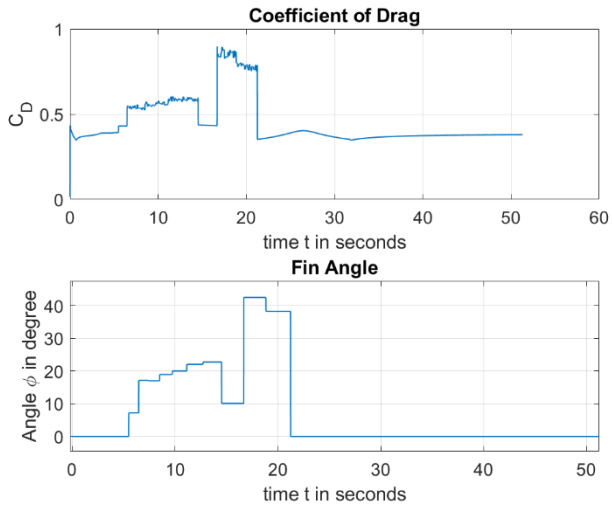

Figure 11 – Coefficient of drag and angle over time of the validation simulation of the optimal controller

shows the path of a simulated rocket with the braking system off (in red) and on (in blue). Without the air brakes the system overshoots the desired apogee and fly to 11043ft. When we make use of the optimal controller, the system is brought down to an apogee at 9973.5 ft. Therefore, the error from the desired apogee is merely 0.265%. When we run the system with higher or lower motor input, the controller reliably brings the rocket to apogee values around 9970ft. However, we also see, that we quickly land outside of the feasible region, when we have to high discrepancies in the starting velocity. We conclude that the detected bounds of the system are only valid for the simulation used to determine the controller. For a real system with

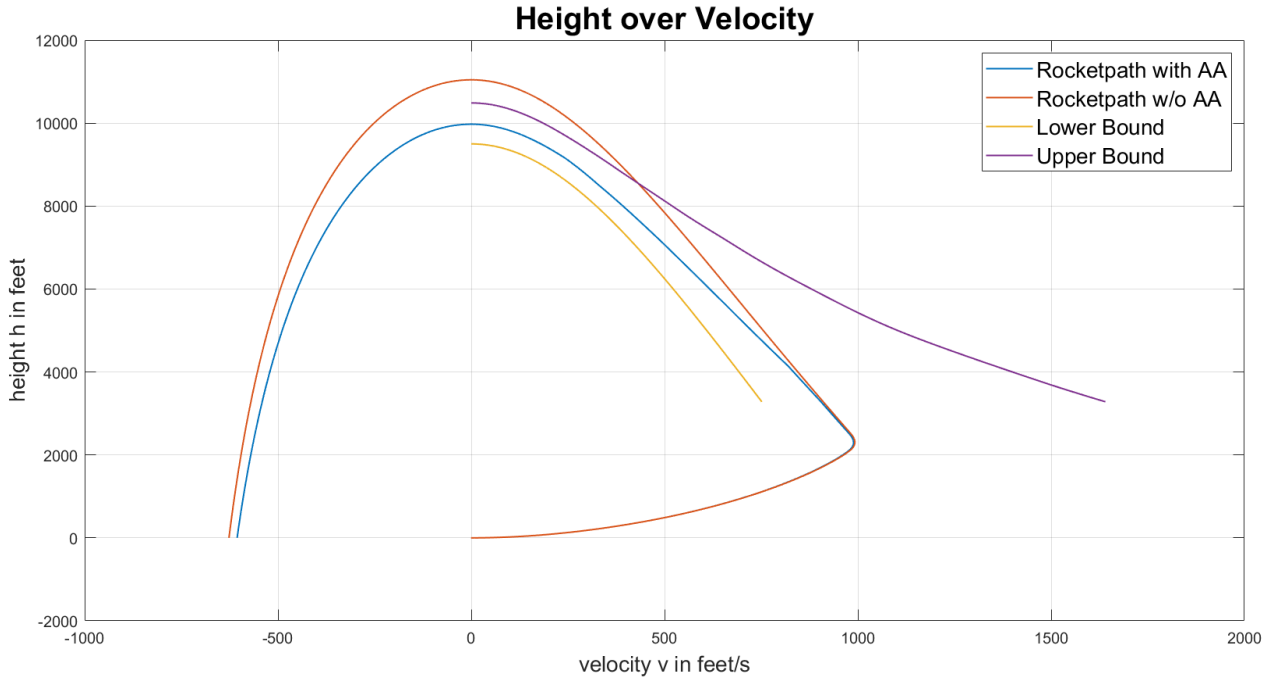

Figure 10 – Simulation of the rocket with and without the active air brake system activated.

additional nonlinear and probabilistic influences to their dynamics, the actual feasible region to reach the apogee is probably smaller than what we have determined. However, we talk about differences of about 100 ft/s, at which the rocket only reaches the defined bounds of the apogee. With an accurate enough simulation, we should be able to predict the rockets velocity at starting height with a higher accuracy. Figure 11 shows the course of the coefficient of drag on the top and the corresponding angle of the air brake fins on the bottom. We see, that we get sudden angle jumps up to 40 degrees. These jumps can be eased with a higher discretization. In reality, the controller for the servo motors of the fins takes care of too sudden changes in input.

Data of the simulated rocket	
Geometry	Mass: $m_0 = 60\ lbs$ Nosecone: Type: Ogive Length: $l_{nc} = 24.5\ in$ Diameter: $d_{nc} = 6.125$ Bodytube: Length: $l_{BT} = 66\ in$ Diameter: $d_{BT} = 6.125$ Fins: Type: swept Number: $n_F = 4$ Thickness: $t_F = 0.5\ in$ Span: $l_{F,sp} = 6\ in$ Root chord: $l_{F,rc} = 13\ in$ Tip chord: $l_{F,tc} = 5\ in$ Sweep distance: $d_{F,sweep} = 8\ in$ Airfoil – Leading edge: $LE = 0.5\ in$ – Trailing edge: $TE = 0.25\ in$ Active Aero Fins: Type: trapezoidal Number: $n_{F,AA} = 4$ Thickness: $t_{F,AA} = 0.5\ in$ Span: $l_{F,AA,sp} = 2\ in$ Root chord: $l_{F,AA,rc} = 4\ in$ Tip chord: $l_{F,AA,tc} = 2\ in$
Motor	Type: AeroTech M2000R Motor mass: $m_M = 9.162\ kg$ Propulsion mass: $m_P = 5.368g$ Burn time: $t_{BO} = 4.0s$ Initial Thrust: $F_{T0} = 1891.0\ N$ Average Thrust: $F_{Tavg} = 2000.0\ N$ Maximum Thrust: $F_{Tmax} = 2327.0\ N$

Table 1 - Rocket data used for the simulation

6. CONCLUSION

Within this paper we have proofed the idea of using an optimal controller determined with dynamic programming for a given active air brake system for a model rocket. The results are promising, as we manage to get errors as low as 0.265% for the reached apogees with the system in validation simulations. Not only is the controller effective, but it also consumes only 58kb of memory. Thus, it is highly usable on a small-scale flight computer.

In further steps, a simulation of the rocket with higher fidelity needs to be setup along with reasonable CFD simulations of the drag coefficient of the rocket with the Active Aero fins. This enables us to come up with a more suitable optimal controller. Additionally, it should have a higher rate of discretization and thus more states. A test flight in the upcoming year may provide helpful data on the drag of the rocket and can be used to gain first experience with the air brake system.

REFERENCES

[1] R. W. Luidens, “Exploring in Aerospace Rocketry - 9. Rocket Trajectories, Drag and Stability,” Exploring in Aerospace Rocketry, 1968.

[2] U. Walter, Astronautics. Cham: Springer International Publishing, 2018.

[3] G. M. Gregorek, “TR-11 Aerodynamic Drag of Model Rockets: Model Rocket Technical Report,” Estes, 1970.

[4] J. S. DeMar, “Model Rocket Drag Analysis using a Computerized Wind Tunnel: National of Rocketry Research & Development Report,” 1995.

[5] D. E. Kirk, Optimal control theory: An introduction. Englewood Cliffs, N.J.: Prentice-Hall, 1970.

[6] J. Lunze, Regelungstechnik 2. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[7] X. Liu, “Fuel-Optimal Rocket Landing with Aerodynamic Controls,” Journal of Guidance, Control, and Dynamics, vol. 42, no. 1, pp. 65–77, 2019, doi: 10.2514/1.G003537.

[8] M. Aliane, N. Moussouni, and M. Bentobache, “Optimal control of a rectilinear motion of a rocket,” Stat., optim. inf. comput., vol. 8, no. 1, pp. 281–295, 2020, doi: 10.19139/soic-2310-5070-741.

[9] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, “Chance-constrained dynamic programming with application to risk-aware robotic space exploration,” Auton Robot, vol. 39, no. 4, pp. 555–571, 2015, doi: 10.1007/s10514-015-9467-7.

[10] C. Gearhart, “Documenting Rocket Science: Cardinal Heavy Design Overview,” 2017.

[11] University of Ottawa Team of Aeronautics and Rocketry, “It's Not Rocket Science; It's Simple Latte: Team 55 Project Technical Report for the 2018 Spaceport America Cup,” 2018.

[12] T. Lew, F. Lyck, and G. MULLER, “Chance-Constrained Optimal Altitude Control of a Rocket,” 2019, doi: 10.13009/EUCASS2019-388.

[13] n.a., Drag Coefficient Prediction: No source available. [Online]. Available: http://www.braeunig.us/space/pdf/Drag_Coefficient_Prediction.pdf