Altitude Control of a Near Sonic Sounding Rocket

Mike Timmerman*

This document carries out a study of designing a controller to actively control the altitude of a sounding rocket using airbrakes. Two control techniques, model predictive control (MPC) and proportional integral derivative (PID) control are considered. Both controllers are tuned, and subsequently compared in their performance. It was found that, for the controller structures considered, the PID controller has a better performance as well as a more practical implementation on a real-time system.

I. Introduction

Sounding rockets have less strict requirements when it comes to following a precisely specified trajectory than rockets which enter orbit. For this, the rocket is launched nearly vertically for a few kilometers, and then progressively flattening the trajectory out at an altitude of 170+ km and accelerating on a horizontal trajectory until orbital velocity is achieved. Sounding rockets on the other had are mainly guided in order to keep the rocket and its payload within boundary lines of the launch site. Another aspect which is often controlled for a rocket is the rate at which the rocket spins, handled by the rate control system. This is especially important for stability considerations and inertial roll coupling.

As far as altitude control, this has most of its importance in amateur rocketry competitions in which more points are given if the rocket hits a certain altitude more accurately. The two major rocketry competitions are the European Rocketry Competition and the Spaceport America Cup in which mainly university teams compete. The majority of these teams fly on a solid rocket motor, which means that throttling the engine is not possible in order to achieve desired apogee targets. An alternative, inspired by the aviation industry, is the use of airbrakes to decelerate the rocket to the correct apogee. It is a type of control surface typically used on an aircraft to increase the drag when extended into the airstream.

In 2021, the university team of University of California Davis, SpaceED Rockets Team, did a preliminary design review of their rocket featuring an airbrake system. They did a comparison of a spike design in which the surfaces slid out of the rocket and an umbrella system in which the surfaces are extended in from the rocket much like an umbrella.

The rocketry team of the University of Edinburgh, Endeavour, made a design of a carbon fibre composite origami airbrakes for their rocket Darwin I [1] which to flew at the Space Port America Cup.

Pioneeer Rocketry's rocket Skybreaker also featured an airbrake system. They are actively controlling the deployment of the airbrake through a PID controller. Another team competing in the Spaceport America Cup in 2018 from the University of Ottawa used an MPC controller to actively actuate their airbrake.

The document is structured as follows: first, a detailed description of the system which is to be controlled, the rocket, is given in section II. Then, in section III, a description of two control techniques, PID control and MPC control, is given along with their control law and strucutre. section IV describes the process of tuning the parameters of the controllers to achieve desired closed-loop behaviour. section V gives the analysis of the controllers their performance. Finally, the results of the flight test are presented in section VI.

*Student, TU Delft Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft

II. Description of the System

The controlled system is a full aluminium sounding rocket reaching near sonic speeds. It has a loaded dry mass of $20.1kg$, and loaded wet mass of $24.9kg$ and spans a length of $2.48m$. A rendering is shown in Figure 1

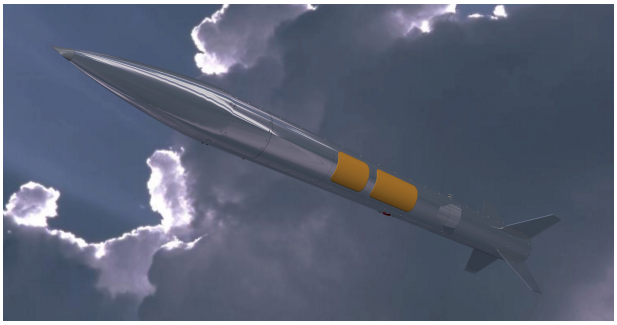

Figure 1: Rendering of the rocket.

For the purpose of simulating the trajectory of the controlled phase, the rocket is modelled as a point mass. This gives a two degree of freedom simulation, namely vertical and horizontal translation. To prevent the rocket from becoming unstable while its engine is burning, the controlled phase only initiates at burn-out, hence only gravity and drag forces act on the rocket. Moreover, the rocket has a constant mass during this phase. The equations of motion are then derived from newton's third law, giving a four state system of differential equations given in Equation 1.

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v_x} &= -\frac{1}{2m}\rho(y)A_{ref}C_d(u,M)v_x\sqrt{v_x^2+v_y^2} \\ \dot{v_y} &= -g-\frac{1}{2m}\rho(y)A_{ref}C_d(u,M)v_y\sqrt{v_x^2+v_y^2} \end{aligned} \tag{1}$$

where the y-axis points up and the x-axis is perpendicular to the y-axis. A_{ref} is a reference area taken as the cross-sectional area of the rocket without the airbrakes extended. ρ is the atmospheric density which depends on the altitude and C_d is the drag coefficient depending on both Mach number M and the linear extension of the airbrake u.

The drag coefficient relation is obtained by determining the drag coefficient semi-empirically on a grid of varying combinations of Mach numbers and extension values of which the graph is shown in Figure 2 and the regression relation of the surface fit to these data points is given by Equation 2.

$$\begin{aligned} C_d = 0.4165+8.886u-0.3778M-10.48uM+43.25u^2 \\ +-0.9093M^2+21.67u^2M+22.7uM^2+0.5587M^3 \end{aligned} \tag{2}$$

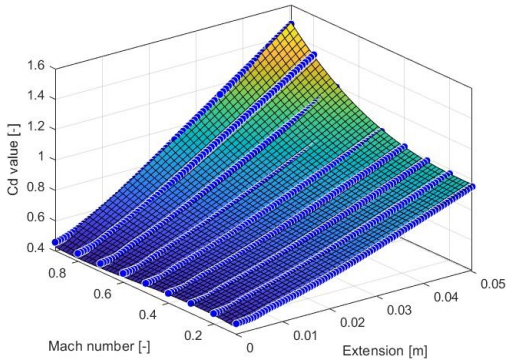

Figure 2: Variation of drag coefficient with airbrake extension and Mach number. $R^2 = 0.9985$

A. Control Actuation System

The altitude is controlled through decelerating the rocket by increasing the drag force experienced by the rocket. This is achieved through extending flat surfaces from the rocket body into the freestream flow. This results in additional surface area being exposed and an increase in the drag coefficient due to disturbing the flow surrounding the rocket. A rendering of the airbrake is shown in Figure 3.

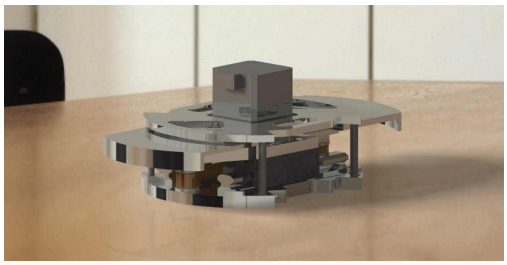

Figure 3: Placeholder airbrake

The control loop used to control the altitude of the rocket is shown in Figure 4. The controller is discussed in detail in section III. The actuator is a stepper motor used to extend the airbrake. The influence of extending the airbrake, toghether with disturbances such as wind, act on the system. The sensors register the response of the system to this input and send an updated system output compared to the reference back to the controller.

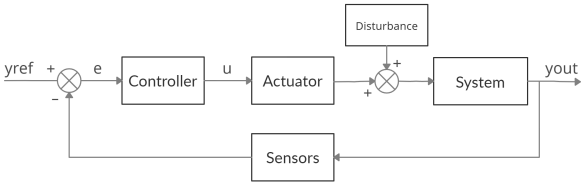

Figure 4: Rocket altitude control loop.

B. System Architecture

The system architecture of the control system consists mainly of the so-called cousin board. It is the cousin board to the main flight computer, named The Stack. The Stack is not involved with the control system and mainly functions as the state machine of the rocket. This means that it keeps track of which phase of flight the rocket is in and handles detection of events, so launch pad, launch, powered flight, burn-out, coasting flight, apogee, parachute deployment. The cousin board consists of five main sections which are shown in Figure 5 as well as how they interact with each other.

The main interface of the cousin board is the Raspberry Pi Zero 2w. It is used as the processor and memory storage of the setup. It has 512MB of SDRAM and a quad-core 64-bit ARM Cortex-A53 processor clocked at 1GHz. This processor allows four processes to run in parallel, as shown in Figure 5. The first core handles the sensor fushion and filtering of the sensor data using a Kalman filter, thereby providing an accurate estimate of the rocket's state. This is send to the thrird core which has the control algorithm on it which computes the control input based on the sensor data. The control input is send on to the fourth core which turns the stepper motor to the correct rotation. The second core handles the telemetry. It sends down data it receives from the Stack as well as from the Raspberry Pi through an antenna.

The second interface is the antenna, which sends data to the ground station using a carrier frequency of 430 KHz.

Thirdly, the power interface is there to receive power from the batteries and distribute the power to all subcomponents of the cousin board.

Fourthly, the sensors section consists of three sensors used to measure the position and attitude of the rocket. The barometer is used to measure pressure levels based on which it can make an altitude estimate. The GPS is also used to determine the position of the rocket. Lastly, the 9-axis IMU consists of three accelerometers, three magnetometers, and three gyroscopes, each type has them configured to be in perpendicular directions. The IMU also allows for attitude determination. All these sensor combined serve to give knowledge of the position and attitude of the rocket in semi-three-dimensional space.

Lastly, between the stepper motor and the fourth core of the Raspberry Pi, there is a rotary encoder and a stepper driver. The driver serves the purpose of actuating the motor, given a voltage signal indicating an angular velocity. The rotary encoder serves the purpose of determining how many degree the stepper motor has rotated, which is a necessity since the stepper may skip steps.

Another imortant aspect not depicted in this diagram is the frequency at which each of these sections is run. This is shown in **??**.

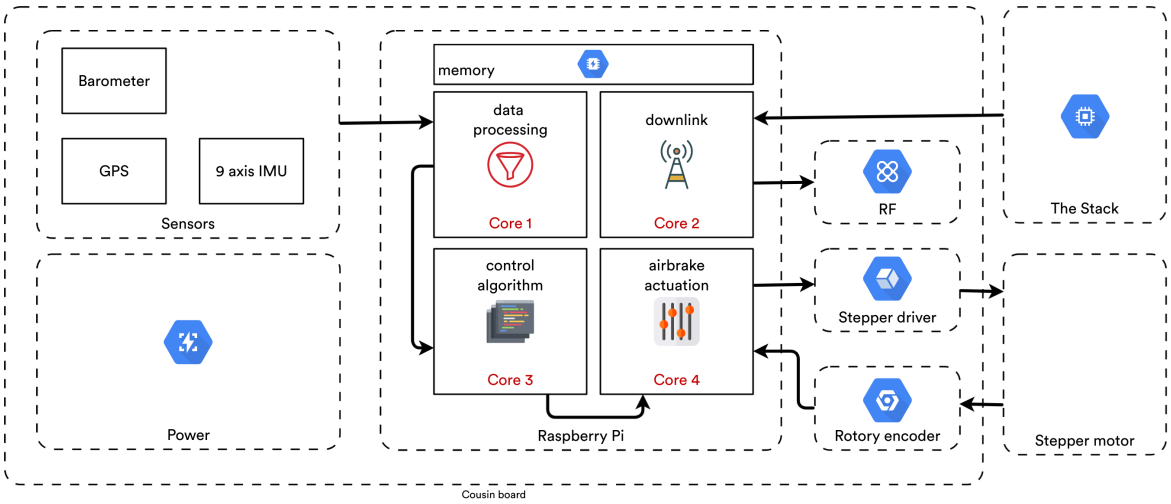

Figure 5: Control hardware system architecture.

III. Control techniques

A. Model Predictive Control

Model predictive control is an advanced control technique which first emerged in the process industries to control a process while satisfying constraints. Model predictive controllers rely on a model of the system to be controlled. It is based on a receding horizon principle, which means that it optimizes the control inputs over a certain time horizon according to an objective function. The first control input of this optimized control sequence is then injected into the system. At the next time step, the optimization is performed again. This means it has the ability to anticipate future events and can take control action accordingly.

Model predictive control has five key ingredients in order to have a working controller: system model, objective function, constraints, optimization algorithm and receding horizon implementation.

System Model

It is required that the control loop runs at a high rate, to ensure quick response time, thus the time it takes to solve the optimization problem at each time step should be short. Therefore, a linear model of the system is obtained, enabling the use of fast algorithms to solve the optimization problem. This linear model of the rocket is obtained empirically through system identification. The approach taken is outlined:

The nonlinear controlled dynamical system is represented as in Equation 3:

$$x^+ = f(x, u) \tag{3}$$

Then, the aim is the construct a linear predictor z:

$$\begin{aligned} z^+ &= Az + Bu \\ \hat{x} &= Cz \\ z_0 &= \Psi(x_0) \\ \Psi &= [\psi_1 \;\; \psi_2 \;\; \dots \;\; \psi_N] \end{aligned} \tag{4}$$

where Ψ are the basis functions which map the nonlinear state space to a linear state space with order N. The nonlinear dynamics are thus approximated by a higher order linear model. Then z represents the higher order state vector of the linear dynamics and $\hat{x}$ the prediction of the actual state vector.

In order to determine the matrices A, B and C of the predictor model, a least-squares fit to data from simulations is performed. So, given a set of data

$$\boldsymbol{X} = [x^1 \dots x^M], \quad \boldsymbol{Y} = [y^1 \dots y^M], \quad \boldsymbol{U} = [u^1 \dots u^M]$$

satisfying the dynamics of the system Equation 3. Solving the least-squares problems in Equation 5 and Equation 6 yields the predictor matrices.

$$min_{A,B} ||\boldsymbol{Y}_{lift} - A\boldsymbol{X}_{lift} - B\boldsymbol{U}||_F \tag{5}$$

$$min_C ||\boldsymbol{X} - C\boldsymbol{X}_{lift}||_F \tag{6}$$

The essential dynamics of the system for the control problem are governed in the vertical motion, so the simulation data is obtained from a 1d trajectory simulation which entails following dynamics:

$$\begin{aligned} \dot{h} &= v_y \\ \dot{v_y} &= -g - \frac{1}{2}\rho v_y^2 A(u) C_d(u, M)/m \end{aligned} \tag{7}$$

where the cross-sectional area A is a function of the control input u, and the drag coefficient C_d is a function of the control input u and the Mach number M.

As basis functions Ψ, the two states, $\{h, v_y\}$, of the nonlinear dynamics and a Gaussian radial base function are selected. The extended state of the predictor z is then given in Equation 8.

$$z = \begin{bmatrix} h & v_y & exp(-(\epsilon^2 ||\boldsymbol{x} - \boldsymbol{c}||)^2) \end{bmatrix}^T \tag{8}$$

in which $\boldsymbol{x}$ is the state of the nonlinear system $\{h, v_y\}$ and $\boldsymbol{c}$ are the centers of the radial base function. The centers are randomly generated such that they can span the whole range of possible values of the nonlinear system states. ϵ is a scaling factor which ensures that the resulting predictor is stable, i.e matrices A, B and C do not contain too small or too big entries.

Model Predictive Control Law

The model predictive control law is based on an optimal control problem (OCP). The objective function used in the OCP is set up such that the system states follow a desired time trajectory. For this, its value is minimized at each time step over a finite time horizon subject to constraints with $\boldsymbol{u}_N$ the optimisation variable, thus an OCP is solved at each time step. The objective function is given in Equation 9.

$$V_N(x_0, \boldsymbol{u}_N) = \sum_{k=0}^{N-1} \{l(x(k), u(k))\} + V_f(x(N)) \tag{9}$$

where $l(x(k), u(k))$ is the stage cost and $V_F(x(N))$ the terminal cost given in Equation 10.

$$\begin{aligned} l(x_k, u_k) &= (x_k - x_{ref})^T Q (x_k - x_{ref}) \\ &\quad + (u_k - u_{ref})^T R (u_k - u_{ref}) \\ V_f(x_N) &= (x_N - x_{ref})^T P (x_N - x_{ref}) \end{aligned} \tag{10}$$

where Q is the state weighing matrix and R the control input weighing matrix. The terminal state matrix P is the solution to the discrete-time algebraic Riccati equation used to solve the infinite horizon, unconstrained optimal control problem.

The objective function is to be solved subject to constraints. The states of the linearized model must satisfy the model dynamics $z^+ = Az + Bu$, with $z(0) = z_0$. Furthermore, the control input u should be constrained as only a certain amount of extension of the airbrakes can be achieved.

The optimal control problem is the formulated as:

$$\mathbb{P}_N(x_0, t) = \begin{cases} min_{\mathbf{u}_N} & V_N(x_0, \mathbf{u}_N) \\ s.t. & z(0) = z_0 \\ & z_{k+1} = Az_k + Bu_k, \quad \forall k \\ & u_k \in \mathbb{U}, \quad \forall k \end{cases} \tag{11}$$

Closed-loop simulation

Figure 6 shows the closed-loop simulation of the rocket controlled to track a reference trajectory generated such that it has the same initial state of the rocket and hits $3500m$ 20 seconds after burn-out. The target apogee is hit with an accuracy of 0.01%, namely an apogee of $3500.7m$ is achieved. The linear extension of the airbrake is limited to and extension of $0.05m$ and the angular velocity of the stepper motor, used to actuate the airbrake, is limited to $5rad/s$.

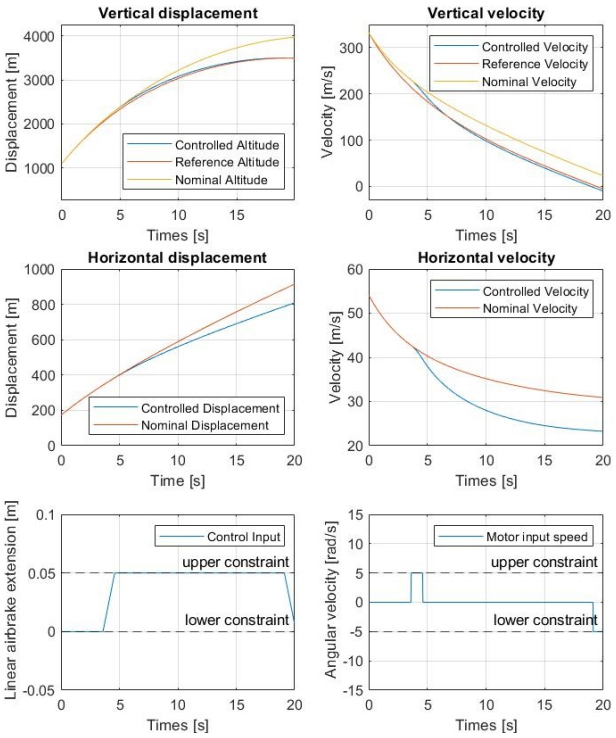

Figure 6: Closed-loop simulation of MPC controlled rocket flight with target apogee of 3500m.

B. Proportional Integral Derivative Control

A proportional-integral-derivative (PID) controller is a control loop mechanism employing feedback based on comparison of the reference trajectory with the output of the controlled system. PID controllers consist of three components; proportional, integral and derivative controls. The proportional term includes proportional changes of the error to the control output, integral term sums the error over time and corrects the output by reducing steady-state errors, derivative term tracks the rate of change of the error and thereby accounts for unusual variations. The PID controller written as a transfer function is given in Equation 12.

$$H(s) = K_p + \frac{1}{K_i s} + K_d s \tag{12}$$

A PID controller does not require an accurate model of the controlled system in its control law. However, depending on the system, the PID controller has to be tuned using a certain methods which does require a model.

Figure 7 shows the closed-loop simulation of the rocket controlled to track the same reference trajectory. The target apogee is hit with an accuracy of 0.001%, namely an apogee of $3500.0m$ is achieved.

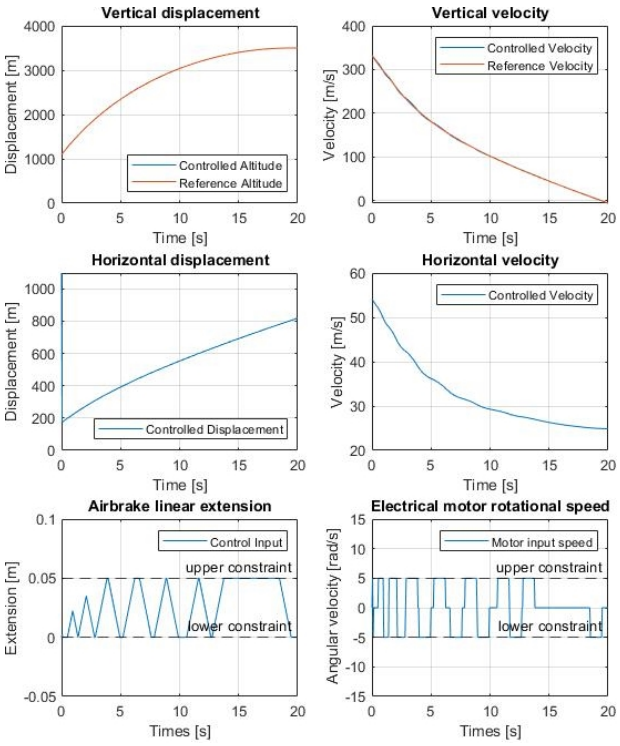

Figure 7: Closed-loop simulation of PID controlled rocket flight with target apogee of 3500m.

IV. Tuning of Controller

A. Proportional Integral Derivative Control

Tuning the PID controller consists of varying the K_p, K_i and K_d gains of Equation 12 such that desired closed-loop behaviour is obtained. This was done through varying each of these gains, simulating the response of the system and compare based on a benchmark. The benchmark is taken to be the percentage difference between actual apogee and target apogee. The set of gains delivering the lowest percentage difference in target apogee is taken as the final gains.

The tuning process was done through three different phases. In the first phase, the proportional and derivative gains for the altitude feedback were considered. These were tuned such that under a nominal flight, the apogee is as close as possible to the target apogee of $3500m$. Besides the nominal flight, the case of deviations in the initial conditions is considered. The difference in target apogee for varying deviations in initial conditions should be as close to zero for an as wide range as possible. The gains were varied between -10 and 10 with a resolution of 1, yielding gains $K_p = -3.0$ and $K_d = -7.0$ as satisfactory. The combination of these gains with altitude feedback gives a difference in target apogee of $0.0237m$. A graph showing the difference in target apogee for varying deviations in initial conditions without vertical velocity feedback is shown in Figure 8.

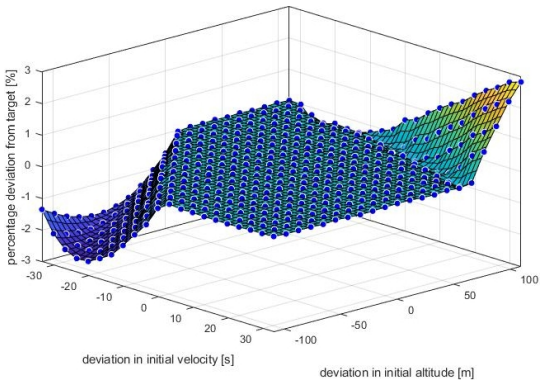

Figure 8: Percentage difference in target apogee for varying deviations in initial conditions without vertical velocity feedback

The second phase considers further improving target apogee difference in the case of deviations in the initial conditions. For this, the vertical velocity feedback is used. Again, the proportional and derivative gains are tuned such that the difference in target apogee for varying deviations in initial conditions is as small as possible. The gains $K_p = -3.0$ and $K_d = -7.0$ were found to achieve this goal. An updated graph showing the difference in target apogee for varying deviations in initial conditions with vertical velocity feedback is shown in Figure 9.

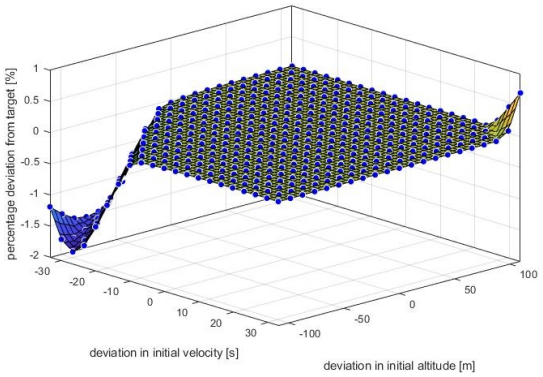

Figure 9: Percentage difference in target apogee for varying deviations in initial conditions with vertical velocity feedback

Finally, the third phase is to consider the integral gains of both the altitude and velocity feedback. It was observed that, even though including integral feedback could have a positive effect on the nominal target apogee difference, it has a negative effect when considering deviations in the initial conditions. Especially since the improvements in nominal target apogee difference is minuscule compared to the already achieved accuracy.

B. MPC

The tuning parameters of the MPC controller are its weighing matrices, giving the ability to put more importance on certain outputs following their trajectories or more importance on the control effort. Since the goal of controlling this rocket is to hit a certain apogee, more importance is set on following the altitude reference trajectory and none is put on the control effort since there is no cost in extending the airbrakes. The following weight matrices are obtained:

$$Q = \begin{bmatrix} 500 & 0 \\ 0 & 10 \end{bmatrix} \qquad R = \begin{bmatrix} 0 \end{bmatrix} \tag{13}$$

and P is the solution to the discrete-time algebraic Riccati equation used to solve the infinite horizon, unconstrained optimal control problem. Another parameter is the prediction horizon length. A longer horizon should lead to a better trajectory tracking, however in this case it was noticed that a shorter horizon leads to more accurate tracking. This is likely due to the fact that at each step, the next reference point is given as a constant reference to be tracked throughout the whole horizon, instead of the actual time-varying reference.

V. Controller Comparison

A. Robustness

The robustness of a controller is a measure of its ability to deal with parameter variations while maintaining stability and performance goals. Stability of the rocket has been ensured at all extension levels throughout the coasting flight, hence only performance goals are considered. Parameter variations might occur in initial conditions, atmospheric density and drag coefficients.

All uncertainties of the powered phase, such as wind and engine performance, are captured by considering a variation in initial conditions of the controlled phase. Figure 9 shows the percentage difference in target apogee for the PID controller. The deviations range from -1.80% to 0.70% or an apogee range of $3437m$ to $3525m$ for variations of 10% in both initial vertical velocity and altitude.

Table 1 shows similar information for the MPC controller. Only a variation in initial condition of 5% in a square around the nominal initial conditions is considered. It is observed that this MPC controller performs worse in terms of robustness performance w.r.t. the PID controller.

Table 1: Performance robustness of MPC controller

Altitude variation [m]	Velocity variation [m]	Apogee deviation [m]
55 (5%)	17 (5%)	-135 (-3.84%)
55 (5%)	0 (0%)	-205 (-5.87%)
55 (5%)	-17 (-5%)	-202 (-5.75%)
0 (0%)	-17 (-5%)	-32 (-0.92%)
-55 (-5%)	-17 (-5%)	47 (1.35%)
-55 (-5%)	0 (0%)	-66 (-1.88%)
-55 (-5%)	17 (5%)	-85 (-2.41%)
0 (0%)	17 (5%)	-155 (-4.41%)

The performance of the MPC controller is already severely degraded by only allowing the initial conditions to vary from the nominal initial conditions. Thus, atmospheric density and drag coefficient variations are only investigated for the PID controller.

The atmospheric density variation is investigated by considering a 10% variation in the standard sea-level density of $\rho_0 = 1.225kg/m^3$. The effect is shown in Figure 10 and Figure 11.

Similarly, a 10% variation in drag coefficient could be considered. However, considering the equations of motion, this leads to the same effect as for a 10% variation in the sea-level atmospheric density.

Essentially, these figures show a 10% uncertainty margin of the atmospheric drag which will be experienced by the rocket.

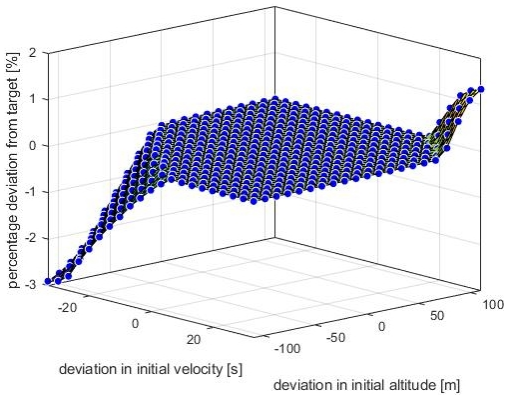

Figure 10: Effect on target apogee differences for sea-level density of $\rho_0 = 1.04kg/m^3$

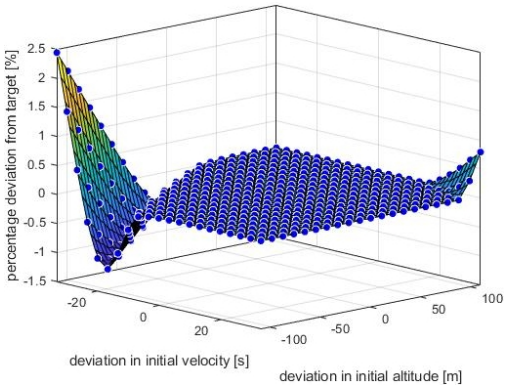

Figure 11: Effect on target apogee differences for sea-level density of $\rho_0 = 1.41kg/m^3$

B. Adaptability

The adaptability of the controller is its ability to track different reference trajectories without changing the control law. This is relevant in case of a change in apogee target, but also as a way to see how to controller reacts to a different reference. This is tested by setting different target apogees, given in Table 2.

Table 2: Performance robustness of MPC controller

Controller	Target Apogee [m]	Error [m]
PID	3700	0 (0%)
MPC	3700	-151 (4.06%)
PID	3600	1 (0.029%)
MPC	3600	-63 (1.77%)
PID	3400	1 (0.029%)
MPC	3400	54 (1.61%)
PID	3300	3 (0.086%)
MPC	3300	72 (2.18%)

C. Sensors and actuators

The effect of sensors and actuator noise and bias is considered. Firstly, only considering the effect of actuator noise and bias. Figure 12 and Figure 13 shows the influence of a 10% noise level with a, respectively, negative and positive bias in the linear airbrake extension of $1cm$. The noise and bias are well-contianed, resulting in a difference in target apogee of -0.01% and 0.02%, respectively.

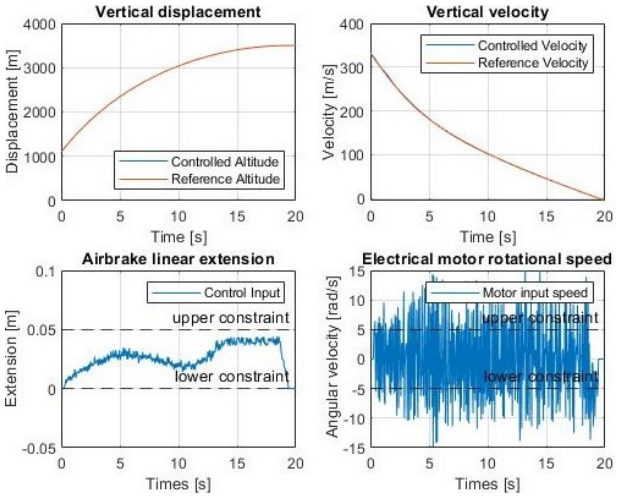

Figure 12: Trajectories with a 10% noise level and a $-1cm$ bias on the control input

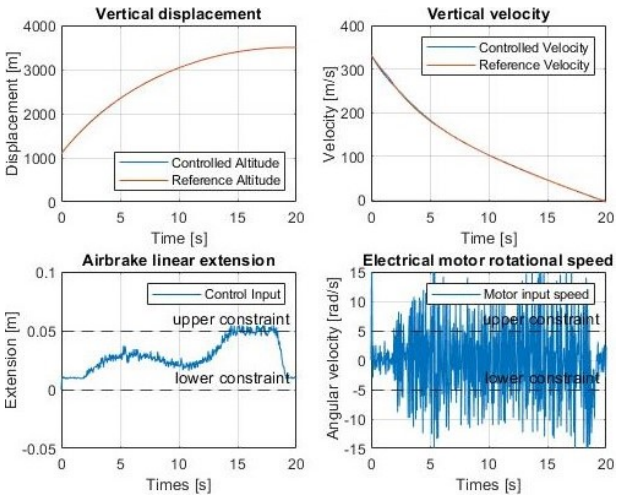

Figure 13: Trajectories with a 10% noise level and a $1cm$ bias on the control input

Secondly, considering the effect of sensor noise and bias. Two output measurements are to be considered, namely altitude and vertical velocity. A combination of different noise and bias for each of the outputs is obtained and collected in Table 3 such that the impact on the target apogee difference can be seen.

Table 3: Effect on target apogee difference for different noise level and bias combinations on both outputs.

Output	Bias [-]	Noise level [-]	Actual Apogee [m]	Measured Apogee [m]
Altitude	-30.0 [m]	0.001	3562.53	3535.09
Vertical velocity	-10.0 [m/s]	0.01	(1.78%)	(1.00%)
Altitude	-30.0 [m]	0.001	3507.11	3480.28
Vertical velocity	10.0 [m/s]	0.01	(0.2%)	(-0.56%)
Altitude	30.0 [m]	0.001	3447.35	3480.65
Vertical velocity	-10.0 [m/s]	0.01	(-1.50%)	(-0.55%)
Altitude	30.0 [m]	0.001	3400.88	3434.13
Vertical velocity	10.0 [m/s]	0.01	(-2.832%)	(-1.882%)
Altitude	30.0 [m]	0.0	3469.45	3499.45
Vertical velocity	0.0 [m/s]	0.0	(-0.87%)	(-0.0157%)
Altitude	0.0 [m]	0.005	3555.75	3572.32
Vertical velocity	0.0 [m/s]	0.0	(1.59%)	(2.07%)
Altitude	0.0 [m]	0.0	3509.68	3509.68
Vertical velocity	-10.0 [m/s]	0.01	(0.27%)	(0.27%)
Altitude	00.0 [m]	0.0	3489.54	3489.54
Vertical velocity	10.0 [m/s]	0.01	(-0.30%)	(-0.30%)

Since the knowledge of the rocket's apogee is limited to the measurements taken, the output altitude is taken as a benchmark of how close the rocket hit its target apogee. Therefore, looking at the 5th column of Table 3, noise on the altitude measurement has the most impact on deviations from the target. Furthermore, a positive bias in both or a negative bias in both output also leads to worse performance.

D. Practicalities

The practicalities pertain to the fact that these controllers need to be implemented and run on embedded systems and run in real-time. Hence the implementability, loop-cycle time and memory usage are compared.

For implementability, the PID controller does not require many external packages, while the MPC controller relies on quadratic programming solvers which in turn have linear algebra packages as dependencies. The other option is to use code generation which is offered by the package "Acado", but the few files which would be required for a PID controller is still easier to implement, hence less error prone.

The loop-cycle time is the time it takes for the controller to calculate the next control input. The solve time for the PID controller is on average $60\mu s$. Thus, the PID controller can calculate control inputs at a rate of $16500Hz$. The MPC controller requires a calculation time of $87.9ms$ which is a rate of $11Hz$.

The PID controller requires very little memory as it does not require any complex calculations or storing of big matrices. The MPC controller on the other hand is completely matrix-based and requires a model of the system which is represented by a matrix, hence a lot of matrix multiplications are performed requiring a lot of memory.

VI. Flight test

TBD

References

[1] Lee, H. A., and Alam, P., "The Design of Carbon Fibre Composite Origami Airbrakes for Endeavour's Darwin I Rocket," *Journal of Composites Science*, Vol. 5, No. 6, 2021. https://doi.org/10.3390/jcs5060147, URL https://www.mdpi.com/2504-477X/5/6/147.