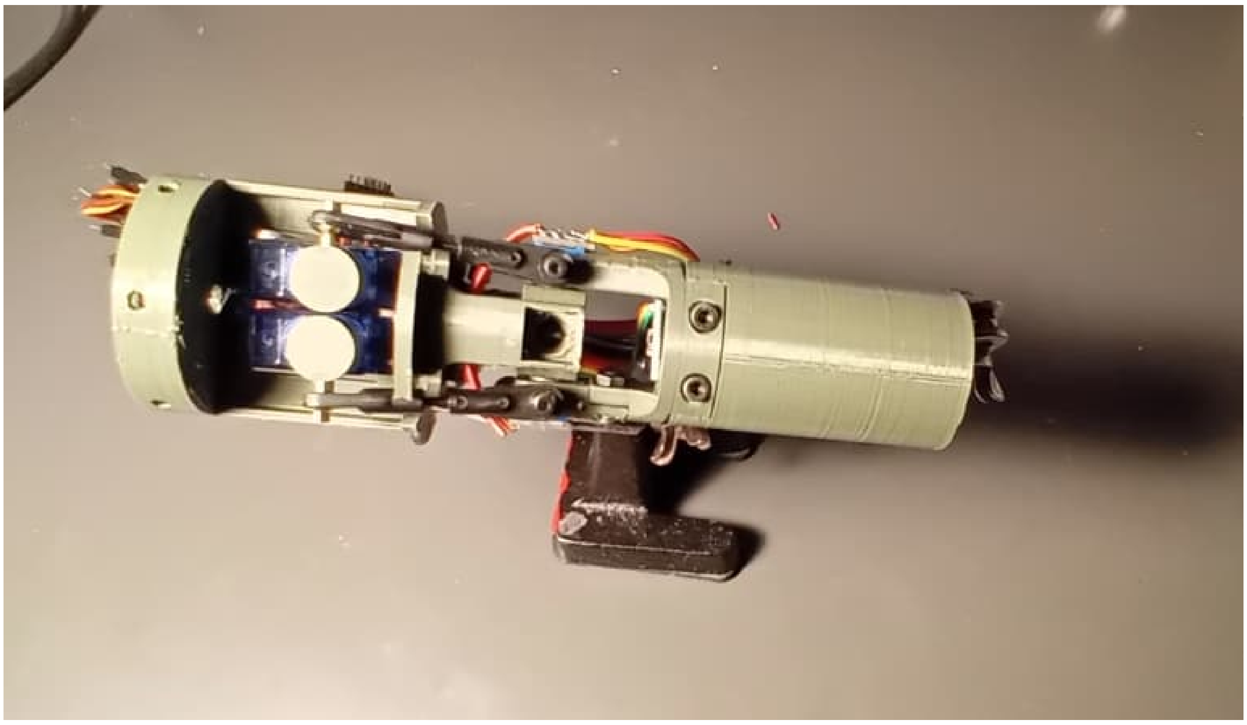
Degree project in in Mechatronics

1, 15 HP

Small-scale Thrust Vector Control System

Design and development of performance evaluation

SIMON HOLMQVIST, VILGOT LÖTBERG



Stockholm, Sweden, 2025

Small-scale Thrust Vector Control System

Design and development of performance evaluation

SIMON HOLMQVIST, VILGOT LÖTBERG

Bachelor's Programme in Vehicle Engineering

Date: 2025-03-06

Supervisors: Andreas Cronhjort (KTH), Gert Johansson (SAAB AB)

Examiner: Daniel Frede

School of Industrial Engineering and Management

Host company: SAAB AB

Swedish title: Småskaligt system för vektoriserad dragkraft

Swedish subtitle: Design och utveckling av prestandaanalyser

Abstract

This thesis treats the design, development and evaluation of the hardware and software of a thrust-vector control system for a flying vehicle built cheaply with off-the-shelf components and 3D-printing, where thrust-vector-control system refers to a machine that can angle the thrust-vector supplied by an engine to induce a torque on the vehicle. The design chosen and evaluated in this report has the entire motor mounted on a cardan-joint type gimbal actuated by electric hobby-servos via pushrods.

To properly evaluate performance, the tests were split into two categories: those intended to evaluate the system in isolation, and those to evaluate the system when integrated into the flight vehicle. Evaluation of the system in isolation meant looking at static error, rise time and settle time based on given command inputs, as well as repeatability of said results. Evaluation of the system in a flight-application was achieved by integrating the developed hardware into a hardware-in-the-loop flight simulation, where a simulation of a flight-scenario would be run on a computer, gimbal-inputs sent from the simulation to the TVC-mount, the resulting actual angle read from the TVC-mount using an array of sensors, and fed back into the simulation to further influence the flight-trajectory.

[INCLUDE RESULTS AND FINDINGS ONCE WE HAVE THOSE]

Keywords: Mechatronics, Thrust Vector Control (TVC), Raspberry PI, Control Theory

Referat

Följande rapport behandlar design, utveckling och utvärdering av hårdvara och mjukvara för ett kraftvektorstyrningssystem med avsikt att användas på en flygfarkost, utgjord till största del av 3D-printade delar och standardkomponenter. Med kraftvektorstyrning menas att på något sätt vinkla en motor eller dess gasmunstycke för att vrida på dess kraftvektorresultant, och att på så sätt inducera ett moment på farkosten. Designen som behandlas i rapporten har en krutraketmotor infäst på en kardanknutsupphängning, som i sin tur styrs av två elektriska hobbyservomotorer via två länkarmar.

För att kunna utvärdera prestanda delas testerna upp i två kategorier: de som är tänkta att utvärdera systemets prestanda i sig, och de som är tänkta att utvärdera systemets prestanda när det är integrerat i en flygfarkost. Utvärdering av prestandan av systemet sker i första hand genom att tolka det statiska felet i systemet, samt svarstid och översläng för givna input signaler. Utvärdering av systemets prestanda under flygning sker i första hand genom att koppla in systemet i en s.k. hårdvara-i-loopen simulering, där ett flygförlopp simuleras på dator, input-signaler skickas från simuleringen till systemet, och systemets är-värde mäts upp med sensorer för att återkopplas till simuleringen och influera flygförloppet.

[INKLUDERA RESULTAT OCH ANALYSER NÄR DE FINNS]

Nyckelord: Mekatronik, Vektoriserad dragkraft, Raspberry PI, Reglerteknik

Acknowledgements

We would like to extend our sincere gratitude to our supervisors Andreas Cronhjort and Gert Johansson, as well as our examiner Daniel Frede and our contact Torbjörn Crona at SAAB. They were all very helpful in answering any questions we had and guiding us in the right direction. Without them, this project would not have been possible.

Stockholm, May 2025

Simon Holmqvist, Vilgot Lötberg

Contents

List of Figures

List of Tables

Glossary

Cosine loss: If the thrust vector is the hypotenuse of a triangle where the adjacent side points through the center of mass, the cosine loss is the decrease in magnitude of this component when the thrust vector is angled and the opposite side grows, assuming that the magnitude of the thrust vector is constant.

HIGH/LOW: on/off, in the context of electronic signals

Laplace domain: A certain complex mathematical space with useful frequency properties

Nozzle: Where exhaust exits an engine

Thrust: A driving force

Transfer function: A function in signal analysis that transforms an input signal into an output signal

Acronyms

DOF: Degree(s) of Freedom

KTH: Kungliga Tekniska Högskolan

PWM: Pulse Width Modulation

SAAB: Svenska Aeroplan Aktiebolaget

SSH (Protocol): Secure Shell (Protocol)

TVC: Thrust Vector Control

Introduction

This chapter introduces the project, describes the problem at hand, the purpose of the project and the goals that need be achieved. Research methodology, delimitations and the overarching structure of the thesis are also discussed.

Background

Vehicles, such as aircrafts, rocket and missiles, need a method of maneuvering from their current position to their destination. To do this they need to be controlled and steered. While jetliners may use control surfaces on their wings to adjust their velocity, some other vehicles tilt their engines to change the line of thrust, thus inducing a torque that turns them. This is called thrust vector control (TVC) [1].

The tilting of the engine, or alternatively just the nozzle, is often achieved by mounting it on a gimbal rig (Figure 1). A gimbal rig generally has two rotational degrees of freedom which allow for precise and independent movement. Depending on the application, the mount may be controlled electronically, hydraulically, or with a combination of the two [2]. Other alternatives for TVC include installing fins or flaps, also known as thrust vanes, in the exhaust stream that deflect the particles to change the direction of the thrust [3].

TVC offers many advantages over aerodynamic control, such as higher predictability, better precision and independence from relative air speed. The last point is especially important for performance at lower velocities, or even zero velocity initial conditions. Stray winds are also consequently less of a problem.

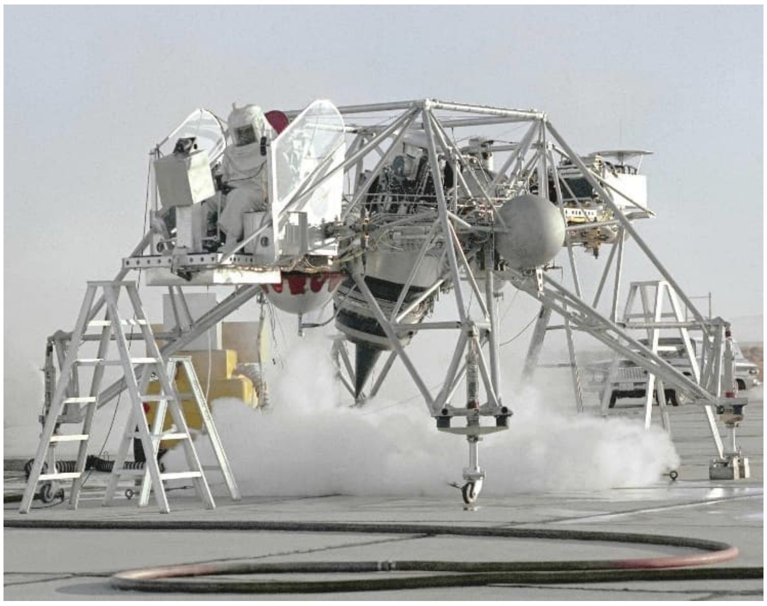


Figure 1: The Landing Research Vehicle (LLRV) developed by NASA as a simulator for the lunar landing vehicle during the Apollo missions. It uses a gimbaled jet engine to simulate lunar gravity [4]. Image: NASA.

Problem

The problem was specified by SAAB and solved by the authors of this thesis. This chapter details the problem statement and the research questions.

Original problem and definition

A TVC system for a small vehicle needs to be developed. The system must enable the vehicle to meet the following requirements:

Table 1: Project requirements.

Type	Value
Range	500 – 1000 m
Speed	50 – 100 m/s
Weight	~ 1kg
Separation	Handheld / ramp
Destination	Stationary point in 3D space within ±30° cone of initial vehicle forward direction.

Scientific and engineering issues

The TVC system needs to be designed, constructed and tested. Specifically, the following questions must be answered:

- How can a TVC mechanism be constructed to fit the requirements specified above?
- How can the transfer function (model) from command to output of the TVC mechanism be determined?
- How can the angle resolution of the mechanism be measured?
- What maneuvering performance can be achieved for the TVC unit and for a potential vehicle (simulated or tested)?

Purpose

The purpose of this project is to deliver a small-scale TVC system for SAAB that fits their requirements and to develop methods for evaluating the performance of the system. The purpose of the evaluations is to inform design decisions and to model the behavior of the TVC system once it is integrated into the vehicle. It is instrumental that the TVC enables the vehicle to achieve its purpose. Therefore, the insights gathered during analysis on the ground must also be applicable to flight operations.

Although the primary purpose is to design and test a TVC system for a specific application as described by SAAB, the methods developed aim to be generally useful for evaluating other TVC systems for different applications. The purpose of the tests is to gain deeper insight into TVC performance and the advantages and limitations of different design choices at small scale. To fulfill this general utility, the programs developed will be modular and system agnostic.

This report does not aim to give exact performance data on the final design. Instead, the report uses virtual data and other methods to test the validity of the experiments. However, a high-level description of the final mechanism is provided.

At the completion of the project, SAAB will have a design for a component that can be integrated into a vehicle. This allows them to continue with development towards a final product. The methods for evaluating the performance of the system may be useful to SAAB in future projects, but also to researchers, companies, and enthusiasts who wish to analyze thrust vector control systems.

Goals

The deliverable will be a design for a TVC system and a method for evaluating its performance. Additionally, SAAB will receive performance data for the final design. The final TVC system, and the experiments to evaluate it, should preferably have the following soft qualities:

- Stable
- Responsive
- Low vibration
- High torque
- High gimbal
- Small diameter
- Smooth motion
- Cheap
- High angle resolution
- Easy to manufacture
- Easy to assemble
- Modular testing
- Repeatable tests
- Verifiable results
- Good data quality and resolution

It is worth reiterating that these are only desirable soft qualities and not hard requirements, hence the imprecise wording.

Industry

The following goals must be achieved along the way to deliver the solution to SAAB:

- Develop a mechanical design for a TVC system
- Evaluate its performance
- Document the design and make handoff frictionless

Scientific community

These goals must be achieved to gather relevant and scientifically interesting data:

- Develop methods for testing TVC mechanisms
- Develop methods for processing the data gathered from tests
- Build a physical prototype to verify design and simulations
- Develop a simulation of a vehicle with a TVC mechanism to use for tests

Research Methodology

The performance of the mechanism will be tested with different experiments. One experiment will use gyroscopes, accelerometers and potentiometers to measure the response to a step input command sent to the servo motors that control the system's orientation. By combining data from these sensors, the angle over time can be estimated and a transfer function from input command to output angle can be calculated. The test is performed with the TVC mechanism placed upside-down on a flat surface with the motor facing up. This is done to prevent gravity from affecting one orientation disproportionally.

Knowing the transfer function allows for calculating parameters such as overshoot and rise time, and also enables the second experiment wherein a simulation of the vehicle, including the TVC mechanism modeled by the transfer function, is compared with a hardware-in-the-loop simulation. This is achieved by measuring the output angles of the mechanism when a command is sent and feeding this output back into the simulation. The purpose of this is to test the validity of the models and explore the boundary of their utility.

The last experiment also involves hardware-in-the-loop simulation but with an additional gyroscope mounted on the base of the TVC mechanism that is fixed in relation to the vehicle. The prototype is mounted on a servo motor that rotates it according to the computer simulation. The mechanism then determines its orientation from the gyroscope readings and sends an appropriate command to the servo motors. The output angles are again measured and fed back into the simulation. This then compared to the completely virtual simulation using the previously derived transfer function.

Delimitations

Only the control of the mechanism (i.e. the transfer from input command to output behavior) will be considered. Any navigation or guidance algorithms used are purely for demonstration purposes. Also, only commercial grade components will be used and the structure will be 3D printed to improve the speed of iteration. This puts physical limits on the performance, as will be discussed in later sections.

No physical flight tests will be done but we will conduct simulations of flight scenarios. This is a large limitation but a necessary one for delivering on time. Lastly, exact performance data of the final design will not be published, but the accuracy of the data and the methods will be estimated.

Structure of the thesis

Chapter 1 is an introduction to the project which describes its purpose and goals. Chapter 2 lays out the fundamental theory behind the project. Chapter 3 describes in detail the scientific and engineering methods used to obtain the results. Chapter 4 details the order and manner in which the project was carried out, and also describes the motivation behind different decisions. Chapter 5 lists the results of the project and discusses them. Chapter 6 offers the final conclusions and describes potential for future work.

Theory

A wide variety of methods, techniques and technologies were used. The report touches on electrical engineering, signals processing, control theory, and many other fields.

Thrust Vector Control

Consider a vehicle driven by a rocket motor, such as a space-launch rocket. It has a center-of-mass somewhere near its geometric center and an engine at the rear end of its long structure (Figure 2).

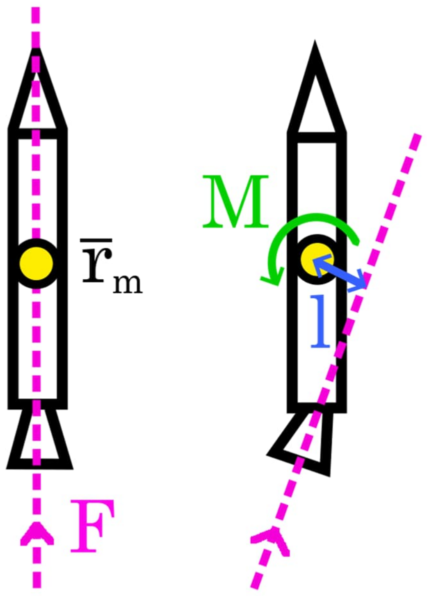


Figure 2: A diagram showing the principles of thrust vector design and how rotating the engine and the thrust line inducing a torque around the center of mass.

When the thrust F at the engine is tilted, the distance l between the thrust line and the center of mass $\vec{r}_m$ acts as a lever-arm for a torque M. This torque wants to turn the vehicle in the direction the engine is pointing: If the engine points to the right, the rocket turns to the right and vice versa. This is the method by which thrust vector control can change the orientation of a vehicle [1].

After a sufficient angular velocity (parallel with M) has been achieved, the engine may swivel to the opposite side to induce a torque that is opposite to the angular velocity. This decreases the angular velocity and when it reaches zero the engine has been rotated to once again point straight through the center of mass. The end result is that the space-launch rocket has changed its orientation. Of course, in reality this is a continuous process where the angle changes slowly over time. It is also important to consider that there is a limit to how fast and accurate the engine can be swiveled.

Servo motors

A servo motor can rotate its shaft accurately between different angles. It consists of a small DC motor coupled to the shaft via a series of gears, and a potentiometer which reads the rotation [5]. This angle data is then fed back into a board on the servo to form a closed loop describing the angle state. It is common for servo motors to be commanded using signals of different lengths. For instance, if the servo motor has a stated frequency of 50 Hz it will have 20 ms windows within which commands are processed. The type of command (i.e. which angle is to be commanded) is specified by the fraction of time that the signal is HIGH. Typically, a shorter pulse means a lower angle and a longer pulse means a larger angle. This is called pulse width modulation (PWM).

The servo motors used in this project require stable 5V DC power, which can be provided by an external power supply. A capacitor may be used for stabilization of voltage. It is especially important that the PWM signal is stable so that it is read accurately by the servo.

Raspberry Pi

This project uses a Raspberry PI 3 Model B+, which is an easy-to-use single-board computer [6]. It runs a Debian based distro of Linux called Raspberry Pi OS but for this project it was primarily controlled through SSH from a laptop. Importantly, it is equipped with multiple GPIO pins that can be used to interface with electronic component such as motors and sensors.

When sending PWM signals from the Raspberry Pi, it is possible to do so using either software PWM or hardware PWM. Software PWM is easier to implement but lacks the precision of hardware PWM. To utilize the latter, the python package *pigpio* was used [7].

Arduino Nano

The Arduino Nano is a microcontroller built on a small frame, making it suitable for applications where space is at a premium. It can be controlled through either C++ via the Arduino IDE or a version of python called micro python, built for microcontrollers. Like the Raspberry Pi, the Nano has built in hardware PWM for better control of servos or similar.

It is also possible to use a MATLAB library called 'MATLAB Support Package for Arduino', that allows for running scripts on a normal computer, and then sending commands to the Arduino to read inputs and write outputs to its peripheries, be that sensors, servos, etc. Running code locally on the Nano can often create performance bottlenecks, as the Nano's CPU, the ATmega328, has a single core with a clock speed of 16 MHz [8], compared to for example the i7 1185G7 present in some laptops, with 8 cores and a clock speed of 3.0 GHz. This makes the Nano a good choice when running heavier applications in for example MATLAB from a normal computer, and simply using the Nano as a periphery.

Gyroscopes / accelerometers (MPU6050)

Gyroscopes like the one in the MPU6050 chip measure angular acceleration along an axis by detection vibrations caused by the Coriolis force [9]. Capacitance is used for the detection and the signal is processed to produce a voltage that is proportional to the rate of rotation. The accelerometer on the MPU6050 uses three separate proof masses which are displaced when the chip accelerates. The displacement is detected differentially by capacitive sensors. An image of how the MPU6050 can be mounted is found in Figure 3.

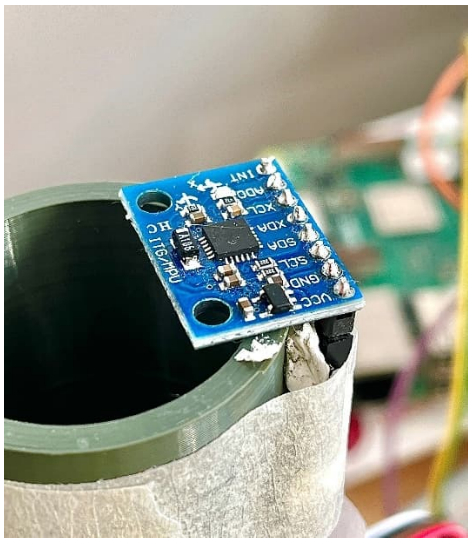
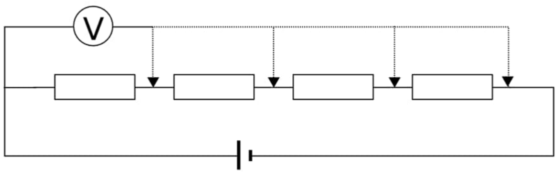


Figure 3: MPU6050 gyroscope/accelerometer mounted on the TVC mechanism.

Potentiometer

A potentiometer is a variable resistor, where the resistance can be controlled by performing some action, often turning a knob. It often comes with three pins, the ground, output and Vcc.

It works by sliding a conductive piece connected to the output, often metal, along a resistive strip connected in both ends to ground and Vcc respectively. Thus, moving the slider varies length of resistive material between the output and Vcc, allowing one to read the position of the slider based on the output voltage.



Gimbal mechanism

TVC systems often use a gimbal mechanism where gimbal rings are mounted inside each other to form a 2 degree of freedom (DOF) coupling. Such a coupling is useful for independently controlling two axes of rotation for a mounted object.

It is possible to determine if a mechanical design is over-constrained, under-constrained or perfectly constrained by counting the degrees of freedom of its constituent parts. This is done using the Chebychev-Grübler-Kutzbach criterion [10] as follows

$$M = \sum_{i=1}^{p} f_i - bq, \qquad \begin{cases} q = p - (m-1) \\ n = m - 1 \\ b = 6 \end{cases}$$

$$\Rightarrow M = \sum_{i=1}^{p} f_i - 6(p-n)$$

$$\Rightarrow M = 6n - \sum_{i=1}^{p} (6 - f_i),$$

where the parameters are defined as

- M: degrees of freedom
- m: number of bodies including the fixed body
- n: number of moving bodies
- p: number of joints
- f_i: degrees of freedom for the i:th link
- b: mobility number, which is 6 for a 3D mechanism.

Although there exists mechanism with more degrees of freedom than predicted by this equation, it does provide a useful way to analyze complex mechanisms.

Kalman filters

There is often a need to estimate a state by combining measurements from multiple sensors. For this purpose, Kalman filters are useful. They are, in a least-squares sense, the most optimal state estimator for a system with gaussian noise. Kalman filters can be conceptually divided into two steps: Prediction and Update [11]. The prediction is based on a model description of the system, and "update" is where the sensor data is introduced. It is not just the state that is estimated, but also the uncertainty (or covariance) of the estimates. This allows for weighting between more and less certain datapoints.

Assume that a state-space system can be modelled with the following difference equation

$$x_k = Ax_{k-1} + Bu_k + w_k, \quad w_k \sim N(0, Q),$$

and that a measurement of that system can be described as

$$z_k = Hx_k + v_k, \quad v_k \sim N(0, R),$$

where

- x_k: state vector at time k
- A: state transition matrix
- B: control matrix
- u_k: control input
- w_k: process noise with covariance matrix Q
- z_k: measurement vector
- H: measurement matrix
- v_k : measurement noise with covariance matrix R.

It is important to note that covariance matrices are used instead of single scalar variances since the different states aren't independent and thus uncertainties may be correlated.

The estimation of the state $\hat{x}_{k|k}$ and the covariance $P_{k|k}$, where the first index is for the prediction step and the second for the update step, are first predicted according the model

$$\begin{cases} \hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \\ P_{k|k-1} = AP_{k-1|k-1}A^T + Q. \end{cases}$$

Incorporating the latest measurements z_k involves first calculating the innovation y_k, a measure of the amount of new information, and the innovation covariance S_k, using the following equations:

$$\begin{cases} y_k = z_k - H\hat{x}_{k|k-1} \\ S_k = HP_{k|k-1}H^T + R. \end{cases}$$

Then, the Kalman gain K_k is calculated with the purpose of evaluating how much to trust the new measurement. The expression becomes

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1} = P_{k|k-1}H^TS_k^{-1}.$$

Finally, the updated state is achieved by adding innovation transformed by the Kalman gain to the results from the prediction step, and the updated covariance matrix results from transform the predicted matrix with the identity matrix minus the measurement matrix transformed by the Kalman gain

$$\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_ky_k \\ P_{k|k} = (I - K_kH)P_{k|k-1}. \end{cases}$$

Transfer Function

The process of converting an input into an output in control theory is described by a transfer function [12]. This function, described in the Laplace domain, contains a description of the characteristics of a system. For this project, transfer functions describing the conversion from servo commands $\theta_{in}(t)$ to resulting angles $\theta_{out}(t)$ were developed.

$$H(s) = \frac{\Theta_{\text{out}}(s)}{\Theta_{\text{in}}(s)} = \frac{\mathcal{L}\{\theta_{out}(t)\}}{\mathcal{L}\{\theta_{in}(t)\}}$$

When a command is sent to a servo motor it has to physically move its load to the desired angle. This process takes time and might be subject to oscillations if the servo moves a substantial mass. If a large step change in angle is commanded it is possible to, by using a gyroscope, measure the change over time of angular velocity. The response to the step in input angle can then be integrated over time to get the rotation angle. Signal analysis software can then take time data from the input step and the integrated angles to identify parameters of a system. Then, by assuming the number of zeros and poles in the transfer function, one can estimate it. A reasonable assumption could be that there are two poles and no zeroes, since this corresponds to a standard second-degree damped system.

MATLAB:s System Identification Toolbox provides a number of tools which make analysis easier [13]. For example, the input step in angle and the integrated output angle can be encapsulated in an `iddata` object. This object can then be passed on to the `tfest()` function along with a specified number of poles and zeros to generate an estimated transfer function.

Apart from allowing for calculation of delay and oscillation magnitude, knowing the transfer function makes it possible to directly integrate it into the control loop. If the control loop knows the limitations in the system response, the controller can give better feedback and thus deliver better output.

PID-controllers and Low-Pass Filters

PID-control is a method for minimizing some error-metric in a system by use of feedback [14] (see Figure 4). The error definition can be chosen based on the application, though typically the error is formulated as the difference between some desired state and the actual state, usually given as the system input. An operator, algorithm or interface may give an input to the system in the form of a desired state; for example, the desired angle of a robot arm, and that be compared against the actual state; the actual current angle of the robot arm. The difference is then used as an error to be minimized. By varying three parameters, usually labelled P, I and D, one can change the characteristics of this minimization-action over time depending on what is desired. PID-control loops are usually desired due to their simplicity, ease of implementation and extensive record of previous application.

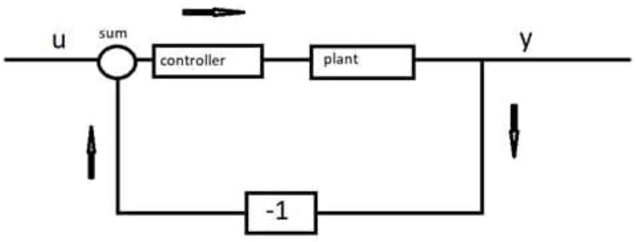


Figure 4: PID diagram

The response is usually calculated as

$$c(t) = P \cdot e(t) + D \cdot \frac{\partial e(t)}{\partial t} + I \cdot \int_t te(\tau)\,\partial\tau$$

$$e(t) = u_{input}(t) - u(t)$$

where

- $e(t)$: error to be minimized by the controller.
- $u(t)$: state of the system (either scalar or tensor).
- $u_{input}(t)$: desired state input by an operator or other process.
- $c(t)$: controller response.

In the Laplace-domain, the transfer function for the PID-controller becomes the following:

$$\frac{c(s)}{e(s)} = P + D \cdot s + \frac{I}{s}$$

Allowing one to analyze the frequency response and step response from the transfer function of the entire system by multiplying the controller's transfer function with that of the plant.

Hardware-in-the-loop simulation

Hardware-in-the-loop-simulation describes taking the normal simulation pipeline of trying to recreate the system virtually, and gradually replacing parts of the simulation with real hardware to see what changes, and what effects the hardware have on the simulation, and to evaluate how the hardware performs compared to the theoretical. The key to doing this successfully is feedback. The signal needs to be sent from the simulation to the component in a manner similar to what it would be in a real application, but the simulation also needs to know the state of the component in return. For this, the system needs sensors to measure said state, and in the case of the TVC-system this is the angle of the solid-motor.

MatRocket

MatRocket is an open-source library for MATLAB developed by the authors for simulating and modelling rockets. It was initially developed for the student rocket association AESIR, but it was modified to work effectively for the vehicle studied in this thesis. Regardless, the simulated object in MatRocket is always assumed to be, and referred to as, a “rocket”. It is built to be highly customizable when implementing new models [15]. Importantly, MatRocket supports multiple subcomponents with their own models and properties. It is modularized in such a way as to make it compatible with the MATLAB suite of ODE-solvers, and can also hence be used for Monte-Carlo simulations, parameter studies, hardware-in-the-loop simulation, state-estimation, etc. MatRocket comes with a set of finished models, among them an aerodynamics model built with testing of control-systems in mind, meant to capture the dynamics of flight and how a rocket’s body catches the wind. Specifically, it’s built to describe the process of energy dissipation from unsymmetric lift along a rocket’s body, as it works to counteract the rocket’s own rotation, a property critical to evaluation of control-system dynamics.

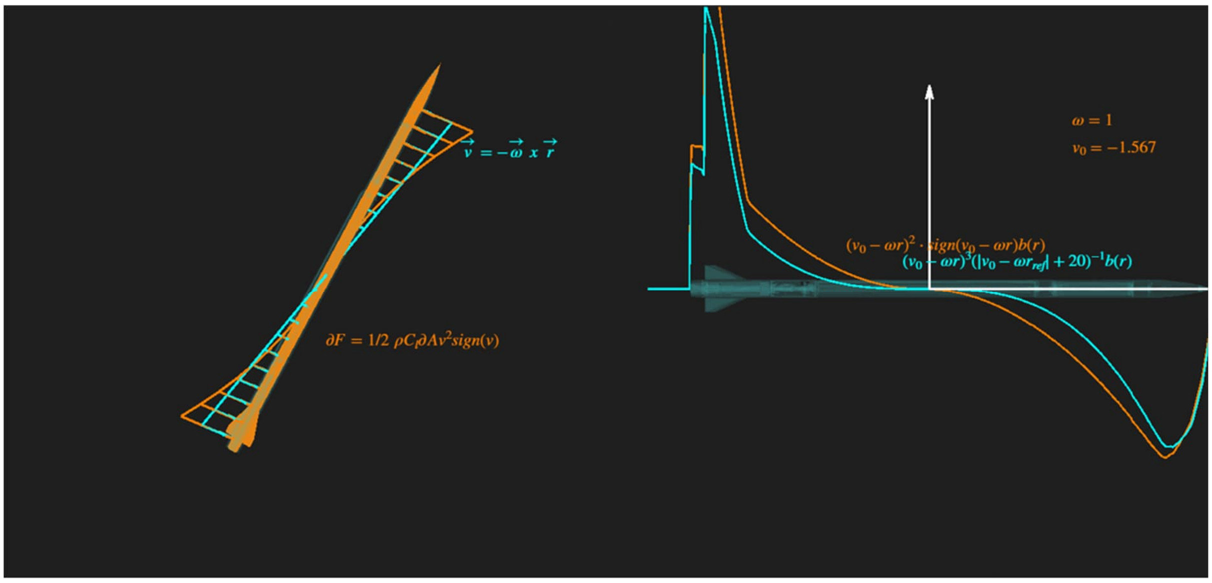

$\vec{v} = -\vec{\omega} \times \vec{r}$
$\partial F = 1/2\ \rho C_d \partial A v^2 sign(v)$
$\omega = 1$
$v_0 = -1.567$
$(v_0 - \omega r)^2 \cdot sign(v_0 - \omega r) b(r)$
$(v_0 - \omega r)^3 (|v_0 - \omega r_{ref}| + 20)^{-1} b(r)$

Figure 5: Diagram showing how the forces and torques are calculated in MatRocket.

An illustration of the calculations are found in Figure 5. It works by approximating the lift-force applied to a rocket’s broadside from the equation of drag [16], but in differential form:

$$dF = \frac{1}{2} C_d \rho v^2 sign(v) dA$$

The sign-term is added as to always map the force to the same direction as the relative air velocity.

T torque resulting from that force becomes:

$$M = \int_A r \cdot dF$$

Where:

- M: The moment acting on the vehicle.

- r: The distance along the axis perpendicular to the moment.
- dF: An infitesimal force componentent.
- C_d: The coefficient of drag.
- ρ: Density of the medium, usually air.
- v: Velocity of air perpendicular to the surface.
- dA: Infitesimal aera element of the surface.

Additionally:

$$v = v_0 - \omega \cdot r$$

$$dA = b(r) \cdot dr$$

Where:

- v_0: The component of the freestream velocity perpendicular to the surface.
- ω: The angular velocity of the vehicle, parallel with the torque vector.
- $b(r)$: The width, perpendicular to the radius and angular velocity, of the surface as a function of the distance from the center of rotation.

This gives:

$$\Rightarrow M = \int_A r \cdot \frac{1}{2} C_d \rho (v_0 - \omega \cdot r)^2 sign(v_0 - \omega \cdot r) b(r) dr$$

Here, an approximation can be made:

$$(v_0 - \omega \cdot r)^2 sign(v_0 - \omega \cdot r) \approx \frac{(v_0 - \omega \cdot r)^3}{|v_0 - \omega \cdot r_{ref}| + \partial \mathrm{v}_{\mathrm{ref}}} = (v_0 - \omega \cdot r)^3 / C_v$$

$$C_v = |v_0 - \omega \cdot r_{ref}| + \partial v_{ref}$$

Where:

- C_v: A constant to get the dimensions to agree, and to scale the approximation to fit the exact expression.
- r_{ref}: An 'average' distance from the center along the surface.
- ∂v_{ref}: A small offset to avoid singularities when $|v_0 - \omega \cdot r_{ref}|$ gets really small.

Through this approximation, the moment can be re-written on the form:

$$M \approx \int_A r \cdot \frac{1}{2} C_d \rho (v_0 - \omega \cdot r)^3 / C_v b(r) dr = \cdots$$
$$= \frac{1}{2} 1 / C_v \rho [v_0^3, -3v_0^2 \omega, 3v_0 \omega^2, -\omega^3] \cdot \int_A C_d [r^1, r^2, r^3, r^4] b(r) dr$$

This allows the integral terms to be calculated once in the beginning of the simulation, and then multiplied in during each simulation-step as a set of coefficients, each corresponding to the n'th area-moment of the vehicle [17]. This saves on compute time while still capturing the essence of the full integral, and thus the vehicles flight-dynamics.

The library implements a common technique used in game-development called parenting, wherein a component like the engine of a rocket can be said to have a 'child' relationship to the rocket, and the rocket correspondingly has a ‘parent’ relationship to the engine. This allows for design of custom subcomponents like fins, parachutes and TVC-mounts.

Functionality written for MatRocket is encapsulated as models, a model simply being a function that updates the state of the rocket in some way. All MatRocket models are written as MATLAB functions that take the rocket as input, and feed it out again as output, with an updated state.

```
function rocket = my_model(rocket)


my_parameter = % some calculation
rocket.some_parameter = my_parameter


end
```

This might be a model for the thrust, a parachute, or similar, and allows the library to be modular and to interact with other libraries.

The general layout of a MatRocket simulation is illustrated in Figure 6.

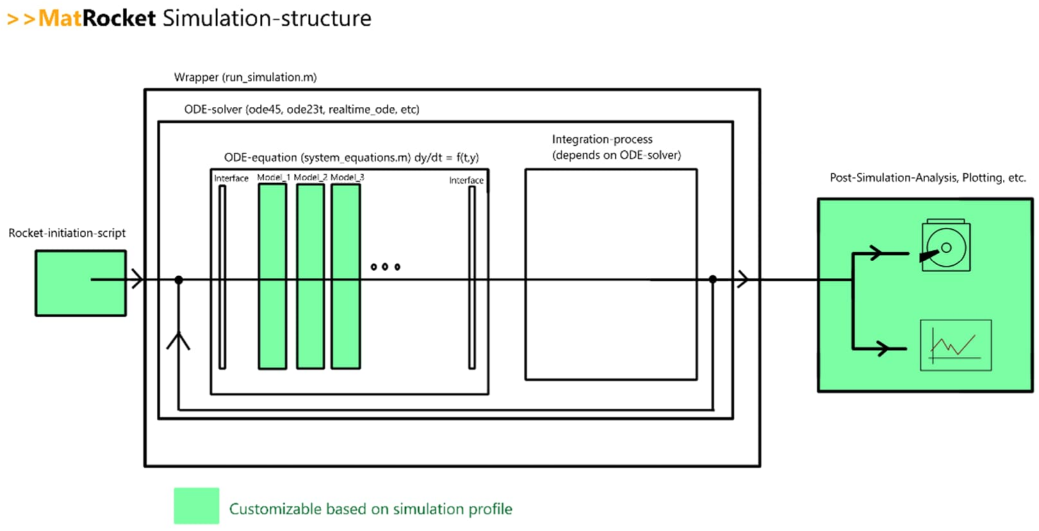

>>MatRocket Simulation-structure
Wrapper (run_simulation.m)
ODE-solver (ode45, ode23t, realtime_ode, etc)
ODE-equation (system_equations.m) dy/dt = f(t,y)
Integration-process (depends on ODE-solver)
Interface
Model_1
Model_2
Model_3
Interface
Post-Simulation-Analysis, Plotting, etc.
Rocket-initiation-script
Customizable based on simulation profile

Figure 6: A diagram explaining the layout of a MatRocket simulation.

Realtime ODE-solving

ODE-step methods

An ODE-solver takes a problem written on the form:

$$\frac{d\vec{y}}{dt} = f(t, \vec{y})$$

Where:

- $\vec{y}$: The state-vector of the ODE.
- t: the independent parameter being integrated/solved over, usually time.

And approximates a solution by means of iteratively stepping the vector $\vec{y}$. The most straight forward method of these is the Euler-forward method. It performs poorly in most metrics, but it has the redeeming quality that it is easy to understand for anyone getting into solving differential equations. It solves the problem by approximating:

$$\overrightarrow{y_{n+1}} = \overrightarrow{y_n} + \frac{d\vec{y}}{dt}(t_n, \vec{y}_n) \cdot \Delta t$$

Where:

- Δt: Is a discrete timestep, usually as small as possible.
- n: Denotes the iteration-number.

The problem with Euler's method is that the error of the solution can be shown to decrease linearly with the timestep Δt, and it is thus said to be of order 1. Methods with better error-characteristics do exist, and MATLAB provides a great number of custom ODE-solvers like ode45, ode23, ode15t, etc. [18] with very good error characteristics. Unfortunately, a lot of them cannot be used for hardware-in-the loop simulation, as they vary the timestep based on the sensitivity of the problem at any given time, and some of them can even go back in time if it realizes that the error starts to become too large, like ode23t [19]. Hence, simulation time and world time are constantly out of sync. For an ODE-solver to be usable for hardware-in-the-loop simulation, it needs to be strictly chronological in when it queries $d\vec{y}/dt$.

The Adam-Bashforth methods

The Adam-Bashforth family of methods can provide of order anywhere from 1 to s (with s being any positive integer), while being strictly chronological in when it queries $d\vec{y}/dt$. They achieve this by reusing previous timesteps to gain insight into the tendencies of the system.

A step using an Adam-Bashforth method of order s generally takes the form [20]:

$$\overrightarrow{y_{n+1}} = \overrightarrow{y_n} + \sum_{i=0}^{i=s-1} \alpha_i \frac{d\vec{y}}{dt}(t_{n-i}, \overrightarrow{y_{n-i}}) \cdot \Delta t$$

Where:

- α_i: Is a coefficient/weight for a given timestep.

For example [21]:

$$s = 2 \Rightarrow \alpha_0 = \frac{3}{2}, \alpha_1 = -\frac{1}{2}$$

$$s = 3 \Rightarrow \alpha_0 = \frac{23}{12}, \alpha_1 = -\frac{16}{12}, \alpha_2 = \frac{5}{12}$$

$$s = 4 \Rightarrow \alpha_0 = \frac{55}{24}, \alpha_1 = -\frac{59}{24}, \alpha_3 = \frac{37}{24}, \alpha_4 = \frac{-9}{24}$$

Related work area

Other gimbal systems

The model rocket community has developed a few TVC designs, but most are useful for diameters larger than those which are relevant for this project. For example, BPS space [22] has developed a TVC mount where two servo motors are mounted next to the solid propellant rocket motor. This severely limits the gimbal range and goes against the diameter (soft) requirements.

(Images will be added if usage rights can be obtained)

Fins and flaps

TVC systems utilizing fins and flaps have been used before [3], but these are more difficult to model since they involve interactions between the plume and the fins. Additionally, the fins will ablate, introducing testing difficulties.

Summary of related work

To summarize, Table 2 shows the differences between different designs.

Table 2: Comparison between different TVC methods.

	Our gimbal	**Existing gimbal**	**Fins / flaps**
Diameter	Small	Medium	Large
Modeling difficulty	Medium	Medium	Difficult
Testability	Good	Good	Worse (ablative)
Thrust loss	Cosine loss	Cosine loss	Drag

Method

To answer the research questions, it was important to properly evaluate performance, and to establish the validity of the results. This focus on data collection and processing influenced the design process on both the hardware and software side. Additionally, rapid iteration was a core part of the methodology from the beginning, with the purpose of achieving early progress and discovering unknowns.

Research Progress

The choice was made early on to manufacture hardware as early as possible (see Discussion).

To carry out the project and answer the research-questions, the following steps were taken:

1. Initial design drafts on pen and paper combined with a study of previous work to gain an overview of design-considerations.
2. Initial rough design drafts in CAD, manufacturing of basic design-components and mechanisms.
3. Iteratively redesigning and printing parts, integrating electronics step by step to work out kinks, in parallel with integration of sensors for data collection (potentiometers, gyros).
4. Design of a calibration mount with limit-switches to sense the TVC-mounts position in space.
5. Writing of scripts and hardware drivers.
6. Measurement of sensor-data to create transfer function.
7. Initial evaluation of overshoot, settle time.
8. Integration of transfer function into simulation to evaluate predicted flight performance.
9. Integration of hardware into simulation to evaluate flight performance with hardware in the loop.
10. Evaluate simulation and result validity.

Research Paradigm

A case study in implementation and characterization of a TVC mechanism on a flying vehicle. Data and insights are collected and documented.

Experimental Design / Planned measurements

The project was divided up into four main experiments:

1. Calibration of the maximum and minimum angles of the nozzle
2. Find the system’s transfer function from servo command to resulting nozzle angle, and evaluate overshoot and rise time.
3. Use the transfer-function model in the flight simulation.
4. Implement and analyze direct sensor-feedback for hardware-in-the-loop simulation.

Each of which used a similar experimental setup, with slight differences in which signals were sent where and for what.

Calibration

Due to the design of the TVC-mechanism, every output angle is a linear combination of the servo-angles. To find the maximum and minimum values of the transformation between servo-angle and output-angle, calibration was necessary.

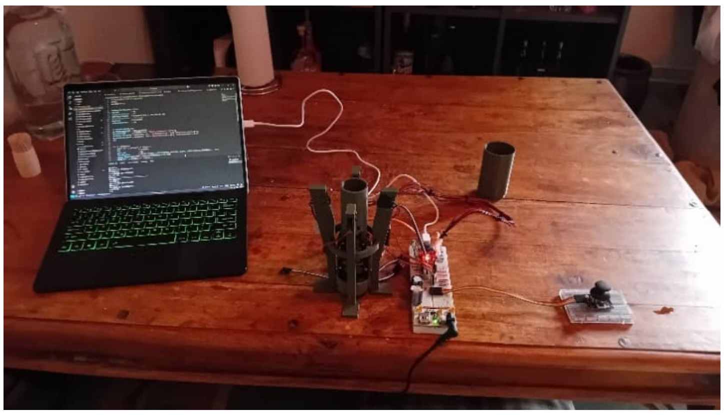


Figure 7: Calibration setup, with the calibration rig in the center, and a joystick to the right that can control the mechanism after calibration which allows for quick testing.

Test bed – Calibration rig

A calibration-rig (see Figure 7) was used with limit-switches placed at known angles and positions in space. Running the servos until they hit one of the limit-switches gave the servo-actuations for that particular point in space. Then, once the servo-actuations for all four limit-switches were known, the average of all the servo-actuations could be taken to get the zero of the system. Also, knowing the positions of the limit-switches, a transfer-matrix could be built between desired output-angles, and the servo-actuation required to achieve it. The same process could be used to find a transfer-matrix between measured voltage on the potentiometers, and the corresponding angle in space. The calibration does not account for inertia in the system or its behavior over time, which is studied in the transfer function analysis.

The main circuit was built on a breadboard, with an Arduino Nano microcontroller, and connections to all the sensors via their respective analogue or digital pins. The main exception being the servos. Due to concerns about the servos using too much current, a separate power supply was added, and the servos connected to that instead. The PWM-signal from the Nano to each respective servo was fed through a transistor, as to run even the servos logical current from the power-supply, and to amplify the signal if the signal from the Nano for some reason were to drop in voltage.

Hardware/Software to be used

The code to be run was written in MATLAB, and the input-output commands sent to the Arduino Nano. From there, the Arduino sent commands to the servos mounted on the TVC-mount to actuate it, and read the data back using potentiometers and a gyroscope. What happened with the data thereafter depended on the experiment

Transfer function estimation

The transfer function from commanded servo angles to the resulting rotation of the nozzle holder over time was estimated with a step response analysis. First, one servo was sent commands to step back and forth between its maximum and minimum allowed angles and the angular acceleration was measured by the gyroscope. Then, the same procedure was done for the second servo, which yields motion in a plane perpendicular to the plane from the first servo. Using the assumption that the outputs are independent, a transfer matrix can be constructed from the results.

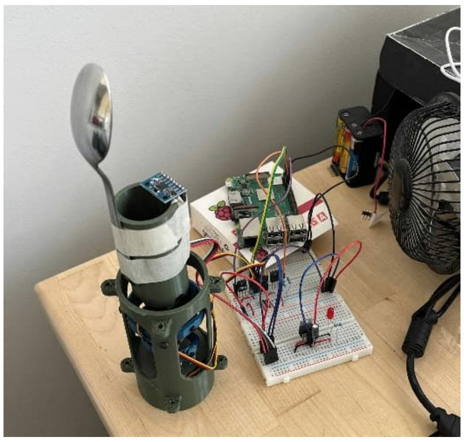


Figure 8: The test setup for the transfer function estimation. The battery is temporarily disconnected in the image, but it is connected during the test.

Test bed

The TVC mechanism was placed upside down, with the aft side facing up (see Figure 8). An MPU6050 sensor with a gyroscope was mounted at the end of the nozzle holder. To simulate the weight and inertia of a hobby motor, a small spoon was taped to the nozzle holder.

Hardware/Software to be used

The hardware used for this experiment where, apart from the TVC mechanism, only an MPU6050 gyroscope and a mass simulator / spoon. As for software, a Python script running on the Raspberry PI was used to collect data which was then processed after the fact in MATLAB.

Model based simulation (with transfer function)

Model based simulation involves using the model of the TVC system created with the transfer-function, and implementing it with the flight simulation in MatRocket, to see how the flight is affected by the performance of the TVC.

Hardware-in-the-loop simulation

Hardware-in-the-loop simulation similarly involves using the TVC-system in the MatRocket flight simulation loop, but instead of modelling the TVC-system, the hardware itself is the model, where the state of the system is updated by reading it from the suite of sensors on the hardware, and command-outputs are sent from the simulation to the hardware to emulate flight-conditions.

Assessing reliability and validity of data collected

To properly identify sources of error, a distinction needs to be made between sources of error and sources of noise. There is a difference between having bad data and not knowing if the data and results are bad, and the methods described below aim to identify and, whenever possible, mitigate the latter.

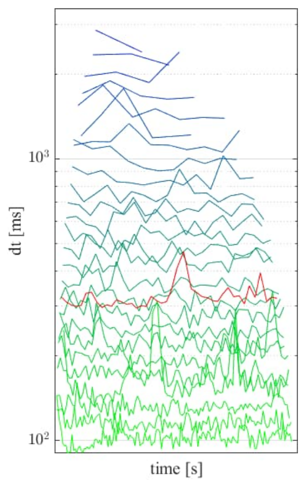
Validity

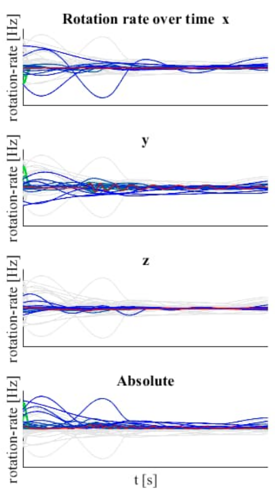
The main sources of error that were identified were:

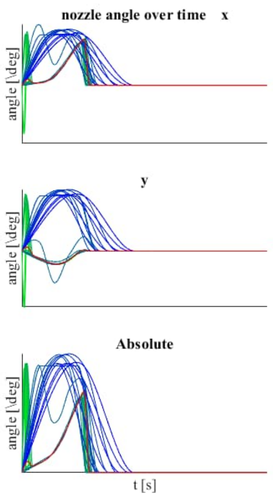
1. **Numerical stability**; the timestep achieved during real time simulation needs to be small enough to give a good solution.
2. **Implementation/coding/model errors in MatRocket**; MatRocket is largely unvalidated up until this project.
3. **Implementation errors in reading the data from the TVC-mount to the simulation**; errors in this step would be difficult to catch.

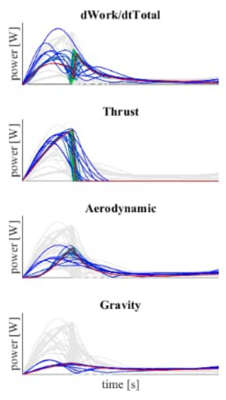
Numerical stability
One such source of error is the numerical stability of the simulation, as the need to run the simulation in real time to keep in sync with the hardware places a hard limit on how small of a timestep can be used; the timestep must be the actual compute-time it takes the computer to run one iteration of the simulation. If it turns out that that timestep is not sufficiently small and the solution does in fact not converge, this mediately invalidates the results gotten from the simulation. To make sure that the timestep was in fact small enough, a convergence-study was performed, to make sure that as the timestep was varied, the actual timestep used in a time-synced simulation was within the regime where the solution had converged. Due to security liability reasons, exact performance data cannot be disclosed, though that is not necessary for the purpose of the convergence-study, as it is only the timestep which is of interest.

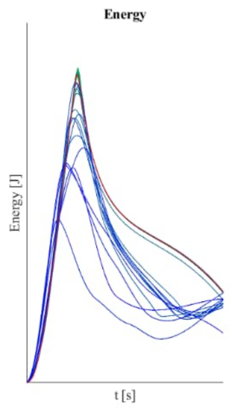
Convergence baseline: no hardware in the loop, no transfer-function for TVC-system.

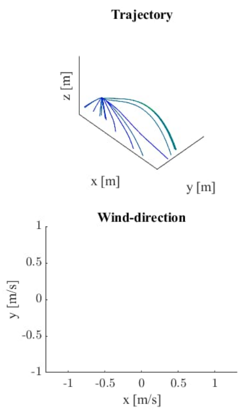


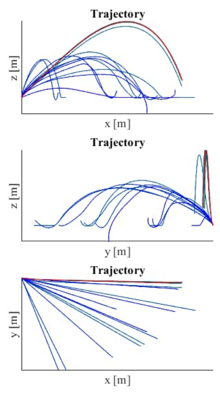












Where the timestep is at its largest in the blue lines, and decreasing in size as the lines get progressively more green. The red line is the actual timestep achieved when running the simulation in real time, the goal being for the red line to overlap with the green lines that are bunched together, indicating a converged solution.

The convergence-study and simulation were conducted on a computer with the specifications described in Table 3.

Table 3: Convergence study computer specifications.

Device model:	Microsoft Surface Pro 8
Processor:	11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz
RAM:	16.0 GB (15.8 GB usable)
OS:	Windows 11 Pro 24H2

Implementation/coding/model errors in MatRocket
MatRocket is largely unvalidated, and it is impossible to validate a model without performing actual flight tests. Two main projects are in the works that will help mitigate this; the project described in this thesis, and the Mjollnir project performed by AESIR at KTH, a student association with which both authors are active, which aims to send a roughly 80 kg rocket to 10 km of altitude [23]. The results from both these projects will be instrumental to validating MatRocket, and the results of the work following on from this thesis will be of huge interest to the authors and for MatRockets future. As of writing this thesis however, MatRocket is stuck in the situation of the chicken and the egg, as both MatRocket and its applications require the other to be validated.

Implementation errors in reading the data from the TVC-mount to the simulation
The simulation trusts that the data coming in from the sensors is indeed accurate, although noisy, and if there’s an implementation-fault in the sensor-reading, the simulation will take those readings at face value. This is not a problem if the fault is easy enough to spot, but if the error is small and invisible, say a case where the sensors give the right values 80% of the time but only sometimes glitches out (one example of this could be a short happening somewhere in the circuit or some component drawing an extra-large current during certain time intervals, decreasing the voltage in the entire circuit and giving faulty values on the analogue sensors), the simulation could continue without giving an error, giving an output that seems reasonable and physically consistent enough to go undetected.

To mitigate this, multiple sensors are used that can be compared against each other. For example, each axis on the TVC-mechanism has two redundant potentiometers to reduce noise, as well as a gyroscope for dissimilar redundancy. The data from these are meant to be fused using the Kalman-filter sensor fusion technique, but the data can also be examined in isolation to check if it matches.

For example, the gyroscope uses I2C, a digital communications protocol, thus the data transmitted is not contained in the analogue voltage itself, and is thus more robust to voltage drops in the circuit. Potentiometers on the other hand are more robust to drift, and are less disturbed by noise when someone drops something on the table or accidentally shakes the setup.

Reliability
To know that the results are reliable, the hardware-in-the-loop simulation is run multiple times to show that the simulation gives similar results each time. Also, the transfer function analysis uses many step responses for the estimation. This results in an average step response, which is then used to get the transfer function.

Planned Data Analysis

The only data queried from the TVC-mount (post-Kalman-filter) being raw angle-data, and to gain any interesting insights, this had to be processed further. After finding the transfer function, as well as integration with the simulation-loop, the following data could be queried:

- The systems overshoot.
- The systems settle time.
- The trajectory and flight performance.

Data Analysis Technique

Below are described the techniques for getting and analyzing the data.

Overshoot and settle-time

Once the transfer-function was known, and a model was set up for the system based on it, the overshoot can be measured by giving a step input, and analyzing the resulting graph in the time-domain. A low settle-time is desirable, as well as a low overshoot.

Trajectory and flight-performance

Analysis of trajectory and flight-performance involved setting up a set of flight scenarios with MatRocket, simulating the scenarios through both transfer-function based model of the TVC, as well as actual hardware-in-the-loop, and then observing how the vehicle behaved.

Software Tools

Below is listed the set of software tools used to make the data analysis possible:

- MatRocket for simulating the flight.
- MATLAB Data Analysis Toolbox for finding the transfer-function.
- MATLAB Support Package for Arduino to communicate with the Arduino Nano and receive sensor-data.
- Blender for final trajectory animation and visualization.

Evaluation Framework

Below is described on what basis the data was evaluated, and how success was measured.

Overshoot and settle-time

What qualifies as a low settle time and a low overshoot could be difficult to quantify without understanding how much they affect the flight dynamics, which is where the simulation was of help. Measuring settle-time and overshoot was done via modelling, and then by simulation, insight could be gained on whether or not the settle-time and overshoot were sufficient for good flight performance.

Trajectory and flight-performance

Analysis of trajectory and flight-performance can be less quantitative than settle-time and overshoot; a decreased settle-time is almost exclusively a good thing, and so is overshoot. Evaluation of flight-performance was therefore done with qualitative assessment of stability, turning-radius, et.c. Final evaluation was done through discussion both between the authors, and with the industry partner SAAB to decide if the observed behavior was desirable, or if something needed to be adjusted.

System Documentation

Sketches and simulation results will be used for documentation.

What we did

What we did

Transfer function estimation

Here we get more detailed about the design of the tests

Hardware/Software design

Hardware/Software design

Implementation

Implementation

Some examples of coding

Some examples of coding

Some examples of figures

Some examples of figures

Results and Analysis

Results and Analysis

Major Results

Major results

Reliability Analysis

Reliability analysis

Validity Analysis

Validity analysis

Discussion

Discussion

Conclusions and Future work

Conclusions and Future work

Conclusions

Conclusions

Limitations

No real motor was used for safety and simplicity reasons.

Future work

Future work

What has been left undone?

What has been left undone?

Cost analysis

Cost analysis

Security

No actual solid motor was used. We only ever used simulators.

Next obvious things to be done

Test with motor.

Reflections

Reflections

References

[1] Glenn Research Center, NASA, "Gimbaled Thrust Interactive," [Online]. Available: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/gimbal/. [Accessed 07 March 2025].

[2] Honeywell, "Thrust Vector Control Systems," 2010. [Online]. Available: https://performanceaccel.honeywell.com/~/media/Files/PDF/thrust-vector-control-systems.ashx. [Accessed 07 March 2025].

[3] Cape Canaveral Space Force Museum, "V-2 JET VANE MOTOR," [Online]. Available: https://ccspacemuseum.org/artifacts/v-2-jet-vane-motor/. [Accessed 07 March 2025].

[4] NASA, "ECN-448," 30 October 1964. [Online]. Available: https://images.nasa.gov/details/ECN-448. [Accessed 06 March 2025].

[5] Hitec Commercial Solutions, "HS-85BB," [Online]. Available: https://www.hiteccs.com/public/uploads/data_sheet/HCS_HS-85BB_Specsheetv2.2_10-1729888315.pdf. [Accessed 07 March 2025].

[6] Raspberry Pi Ltd, "Raspberry Pi 3 Model B+," 14 March 2018. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. [Accessed 06 March 2025].

[7] abyz.me.uk, "The pigpio library," 01 January 2023. [Online]. Available: https://abyz.me.uk/rpi/pigpio/. [Accessed 06 March 2025].

[8] Arduino, "Arduino Nano," [Online]. Available: https://store.arduino.cc/en-se/products/arduino-nano?srsltid=AfmBOoqwr3WxW8QxSGwTUU5FeRbrfsZO8PhR76aglLeXhjUGdDbZ7thk. [Accessed 07 April 2025].

[9] Invensense, "MPU-6000 and MPU-6050 Product Specification Revision 3.4," 19 August 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. [Accessed 06 March 2025].

[10] G. Gogu, "Chebychev–Grübler–Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations," *European Journal of Mechanics - A/Solids,* vol. 24, no. 3, pp. 427-441, 2005.

[11] G. B. Greg Welch, "An Introduction to the Kalman Filter," 2002.

[12] K. A. U. C. Hans Bodén, Applied Signal Analysis.

[13] MATLAB, "System Identification Toolbox," [Online]. Available: https://se.mathworks.com/help/ident/index.html?s_tid=CRUX_topnav. [Accessed 07 March 2025].

[14] L. L. M. E. Torkel Glad, Reglerteknik, grundläggande teori, vol. 5, Lund: Studentlitteratur, 2024.

[15] V. Lötberg, "MatRocket," 29 March 2025. [Online]. Available: https://github.com/spiggen/MatRocket. [Accessed 07 April 2025].

[16] N. Hall, "Drag Equation," 15 July 2024. [Online]. Available: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/drag-equation/. [Accessed 07 April 2025].

[17] V. Lötberg, "Aerodynamics model," 29 March 2025. [Online]. Available: https://github.com/spiggen/MatRocket/blob/master/Documentation/aerodynamics_model.md. [Accessed 07 April 2025].

[18] MathWorks, "Choose an ODE Solver," [Online]. Available: https://se.mathworks.com/help/matlab/math/choose-an-ode-solver.html. [Accessed 07 April 2025].

[19] MathWorks, "ode23t," [Online]. Available: https://se.mathworks.com/help/matlab/ref/ode23t.html. [Accessed 07 April 2025].

[20] N. S. Dattani, "Linear Multistep Numerical Methods for Ordinary Differential Equations," 2008 October 28. [Online]. Available: https://arxiv.org/abs/0810.4965v1. [Accessed 07 April 2025].

[21] B. Seidu, "A Matrix System for Computing the Coefficients of the Adams Bashforth-Moulton Predictor-Corrector formulae," *International Journal of Computational and Applied Mathematics.,* vol. 6, no. 3, pp. 215-220, 2011.

[22] BPS Space, "TVC," [Online]. Available: https://bps.space/products/thrust-vector-control. [Accessed 07 March 2025].

[23] AESIR, "Mjollnir," [Online]. Available: https://www.aesir.se/projects/mjollnir. [Accessed 07 April 2025].

MATLAB Code

MATLAB Code

Raspberry PI code

Raspberry PI code

Datasheets

Datasheets