



TECHNISCHE UNIVERSITÄT BERLIN

LEARNING AND INTELLIGENT SYSTEMS

# Learning-based Multirotor System Model Enhancements

*Jana Schicke*

mentored by

Akmaral Moldagalieva and Khaled Wahba

examined by

Dr. Wolfgang Hönig and Prof. Marc Toussaint

March 20, 2023

# Contents

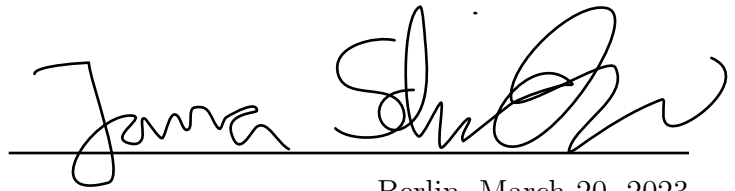
<b>Statutory Declaration</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Nomenclature</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>2</b>
2.1 Multirotor Dynamics . . . . .	2
2.1.a Aerodynamic Forces . . . . .	4
2.2 Machine Learning . . . . .	5
2.2.a Supervised Learning . . . . .	5
2.2.b Decision Tree . . . . .	8
<b>3 Approach</b>	<b>11</b>
3.1 Quadrotor . . . . .	11
3.2 Basic Forward Propagation . . . . .	15
3.3 Residual Force And Residual Torques . . . . .	17
3.4 Supervised learning . . . . .	18
3.5 Decision Tree . . . . .	20
<b>4 Results</b>	<b>21</b>
4.1 Basic Forward Propagation . . . . .	21
4.2 Residual Force and Residual Torque . . . . .	24
4.3 Supervised learning . . . . .	25
4.3.a Scaling the data . . . . .	25
4.3.b Comparison of different input features . . . . .	27
4.3.c Using different input sizes for the model . . . . .	31
4.3.d Changing the complexity . . . . .	35
4.3.e Comparing the different models of MLP . . . . .	39

4.4	Decision Tree . . . . .	40
4.4.a	Scaling the residual torque . . . . .	41
4.4.b	Comparison of different input features . . . . .	42
4.4.c	Using different sizes of input features of the model . . . . .	45
4.4.d	Changing the complexity of the decision tree . . . . .	49
4.4.e	Comparing the different decision trees . . . . .	53
4.5	Comparison of supervised learning and decision tree . . . . .	54
<b>5</b>	<b>Conclusion</b>	<b>56</b>

# Eidesstattliche Erklärung / Statutory Declaration

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

A handwritten signature in black ink, appearing to read 'Janina Schubert', is written over a horizontal line.

Berlin, March 20, 2023

# Abstract

In terms of motion planning and control, it is important to know the impact of aerodynamic effects on the quadrotor because by ignoring the effects the planned and controlled motion can be inaccurate. This thesis contains different methods to determine the impact of aerodynamic forces on the thrust and torque of the quadrotor by using machine learning. I used collected data to calculate the residual thrust and the residual force and with the calculated result I trained different multi-layer perceptrons and decision trees to determine the best parameter. For the control of the quadrotor, it is important that the model - that calculates the effects on the quadrotor - is time efficient so that the additional thrust and torque can be found within the shortest time. The decision tree turned out to be the best learning method to get an accurate prediction in a short time.

# Zusammenfassung

In Bezug auf die Bewegungsplanung und -steuerung ist es wichtig, die Auswirkungen aerodynamischer Effekte auf den Quadrotor zu kennen, da die geplante und gesteuerte Bewegung ungenau sein kann, wenn die Effekte ignoriert werden. Diese Arbeit beinhaltet verschiedene Methoden, um den Einfluss aerodynamischer Kräfte auf den Schub und das Drehmoment des Quadrotors durch maschinelles Lernen zu bestimmen. Ich habe gesammelte Daten verwendet, um das restliche Drehmoment und die Restkraft zu berechnen, und mit dem berechneten Ergebnis habe ich verschiedene Multi-Layer Perceptrons und Entscheidungsbäume trainiert, um den besten Parameter zu bestimmen. Für die Steuerung des Quadrotors ist es wichtig, dass das Modell, das die Auswirkungen auf den Quadrotor berechnet, zeiteffizient ist, damit der zusätzliche Schub und das Drehmoment innerhalb kürzester Zeit gefunden werden können. Der Entscheidungsbaum stellte sich als die beste Lernmethode heraus, um in kurzer Zeit eine genaue Vorhersage zu erhalten.

# Supplementary Material

The Code is available at [https://github.com/j-schicke/bachelor\\_thesis](https://github.com/j-schicke/bachelor_thesis)

# Nomenclature

$\dot{v}$	acceleration	$\text{m s}^{-2}$
$\dot{\omega}$	angular acceleration	$\text{rad s}^{-2}$
$\hat{\psi}_i$	force of rotor $i$	N
$\hat{b}$	normalized battery voltage	
$\hat{p}_i$	normalize PWM signal of motor $i$	
$\mathcal{B}$	body frame	
$\mathcal{J}$	inertia matrix	$\begin{pmatrix} 16.571710 & 0.830306 & 0.718277 \\ 0.830806 & 16.655602 & 1.800197 \\ 0.718277 & 1.800197 & 29.261652 \end{pmatrix} \cdot 10^{-6} \text{kg m}^2$
$\mathcal{W}$	world frame	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$\omega$	angular velocity	$\text{rad s}^{-1}$
$\omega_i$	angular velocity of rotor $i$	$\text{rad s}^{-1}$
$\tau_a$	disturbance torque	$\text{rad s}^{-2}$
$B_0$	actuation matrix	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0.046 & 0 & -0.046 \\ -0.046 & 0 & 0.046 & 0 \\ -0.006 & 0.006 & -0.006 & 0.006 \end{pmatrix}$
$f_a$	disturbance force	N
$g$	gravitation	$9.81 \text{ m s}^{-2}$
$k_F$	thrust coefficient	

$k_M$	torque coefficient	
$L$	lengths of arm of Crazyflie	4.6 cm
$m$	mass	34.7 g
$q$	quaternion	
$R$	rotation matrix	
$r$	position	m
$t$	time	s
$u_1$	total thrust	
$u_2, u_3, u_4$	body moments	
$v$	velocity	$\text{m s}^{-1}$
$z_{\mathcal{B}}$	z-axis in body frame	
$z_{\mathcal{W}}$	z-axis in world frame	

# 1 Introduction

Currently, multirotors are rising in popularity, the reason is their easy control and low pricing [1]. Some multirotors can fly autonomously. Especially in research, the multirotor with four rotors - also called the quadrotor - is popular[2]. Even when the quadrotor is flying indoors, where there are no external aerodynamic forces, the impact of the aerodynamic effects on the multirotor should not be ignored. In some research about planning and controlling the motion of the quadrotor, it is assumed that the quadrotor is a rigid body [2]. The basic model of the quadrotor ignores the impact of the aerodynamic forces and should only be used when the quadrotor is flying at a slow pace. When the quadrotor is flying maneuvers more aggressively, the residual force and residual torque should be added to the model. Without the added residual force and residual torque, the motion planning and controlling of the quadrotor will be inaccurate [3]. The goal of this thesis is to find an efficient and accurate way to estimate the residual force and residual torque of the quadrotor with the help of different machine-learning algorithms and to compare the different learning algorithms with one another. The predicted residual force and residual torque can be used to improve motion planning and controlling.



## 2 Background

In this section, I will explain the meaning of quadrotors and their dynamics. Also, I will break down the self-inflicted aerodynamic forces that can appear. Furthermore, I will go into detail about what machine learning is and the different models I can use.

### 2.1 Multirotor Dynamics

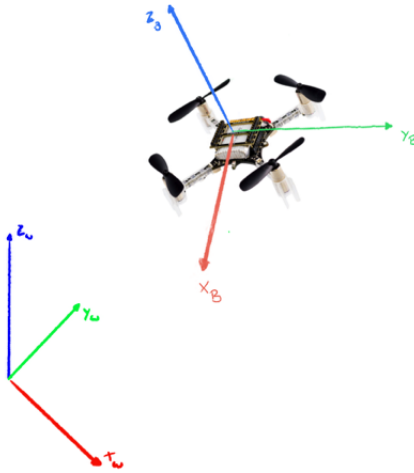


Figure 1: The Crazyflie and its reference frames[4]

A quadrotor is an aerial vehicle with four rotors. When the quadrotor is flying around, the multirotor can be represented in at least two coordinate systems at the same time as shown by 1. The first coordinate system is the world frame  $\mathcal{W}$ . It is fixed and will not be influenced by the position and orientation of the multirotor. The second coordinate is the body frame  $\mathcal{B}$ . The body frame changes whenever the multirotor moves because it is dependent on

the orientation of the multirotor  $\mathcal{B} = \mathbf{R}\mathcal{W}$ . Whereas the body frame is dependent on the orientation, the position  $\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}$  of the multirotor itself is detected with the world frame. [1, 2]. Besides the global position, the multirotor's state also displays the global velocity  $\mathbf{v}$  and the angular velocity  $\boldsymbol{\omega}$  [5]. In the thesis following multirotor dynamics are needed:

$$\dot{\mathbf{r}} = \mathbf{v} \quad (1)$$

$$m\dot{\mathbf{v}} = -mg\mathbf{z}_{\mathcal{W}} + u_1\mathbf{z}_{\mathcal{B}} \quad (2)$$

$$\dot{\boldsymbol{\omega}} = \mathcal{J}^{-1} \left[ -\boldsymbol{\omega} \times \mathcal{J}\boldsymbol{\omega} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \right] \quad (3)$$

where  $m$  is the mass of the multirotor and  $g$  presents the gravitational acceleration,  $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}$  denotes the total thrust and the body moments. The inertia matrix is depicted as  $\mathcal{J}$  and the symbol  $\times$  stands for the cross-product. The second equation (2) is Newton's second law of motion and the third equation (3) is the Euler equation [1, 2]. The Euler equation is dependent on the inertia matrix because the inertia of the multirotor can influence the angular acceleration [6]. To calculate  $u$ , there are two different methods. The first method is the Newton-Euler equation[1, 2]:

$$\mathbf{u} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_FL & 0 & -k_FL \\ -k_FL & 0 & k_FL & 0 \\ -k_M & k_M & -k_M & k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \quad (4)$$

where  $L$  presents the distance between the center and the rotor of the quadrotor,  $k_F$  is the thrust coefficient and  $k_M$  is the torque coefficient. The  $\omega_i$  stands for the angular speed for the rotor  $i$ . The other method is [5]:

$$\mathbf{u} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ -t2t & t2t & -t2t & t2t \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix}, \quad (5)$$

where  $\hat{\psi}_i$  is depicted as the force of the rotor  $i$  and  $t2t$  is the thrust-to-torque ratio.

### 2.1.a Aerodynamic Forces

In this part of the background, I am going to summarize following paper[1] and following thesis [6] on the subject of aerodynamic effects. The dynamics of the multirotor can be influenced by aerodynamic forces. The two aerodynamic forces that have the biggest impact on the multirotor are induced drag and blade flapping. Both can destabilize the multirotor since they are causing additional force in the x-y plane. An important property of multirotor is that the rotor blades are quite flexible and can bend when additional forces encounter. The flexibility of the rotor blades can cause blade flapping. However, induced drag happens because the rotor blades also have a certain rigidity to them.

Induced drag occurs when the rotor wing moves in the air because the wing is in two different pressure zones. The pressure is higher underneath the wing than on the upper side. This caused the appearance of vortices on the tip of the wing. The result is that the air is flowing down behind the wing which causes an additional lift that is proportional to the induced drag. As long as the multirotor is only hovering in the air the induced drag does not result in a drag force because the induced torque is equally distributed around the rotor; however, additional torque is generated. As soon as the multirotor moves to a direction other than the z-axis, the multirotor is generating relative wind and the blade of the forward-facing rotor encounters a higher air velocity than the opposite blade. This causes an additional force in the direction of the relative wind which is the opposite direction of the multirotor.

As for the blade flapping, the effect also happens because the rotor blades have different tip velocities. If the tip speed is increasing, so does the lift force and because one blade has a higher lift than the other one, the rotor behaves like a gyroscope when it spins at a high speed. Thus, the attacking torque effect is turned 90° and the blades can flap up and back down while flying through the wind. This can cause a balance because the advancing rotor has a higher tip velocity; however, this decreases the angle of attack and so the additional lift from the high tip velocity will be decreased through the reduced lift coefficient from the decreased angle of attack.

## 2.2 Machine Learning

Machine learning represents systems that can learn from data without the need for instruction from a human being [7]. Sometimes problems can be too time-consuming for humans to learn or the pattern of a problem is too difficult for us to solve. The machine learning algorithm can locate a pattern more easily and solve the problem faster for us. The goal of machine learning is to find patterns in the data, learn from experience or abstract new information. Machine learning has different subgroups; supervised, unsupervised and reinforcement learning. In supervised learning, The model is trained with the help of inputs and outputs, while training the model will try to find a function that matches the input with the output. The model will determine new input to matching output with that function. Unsupervised learning does not have pre-labelled data and is used to find similarities and create matching groups, also called clusters. When talking about reinforcement learning, the system learns how to act by observing its environment. The algorithm learns because it gets feedback from its environment on whether its actions have a positive or negative effect [8].

### 2.2.a Supervised Learning

There are main two existing supervised models that play an important role in the field of supervised learning: the classification and the regression models. Both models are divided into the same two steps: training and testing. During the training phase, the model gets a sample input and constructs a function to estimate the output. After the model predicts the output, the prediction is compared with the actual output and the error is then given to the model to improve. Throughout the testing, the model predicts the output of the test data. The difference between classification and regression models is the output. The classification model groups the input data in a predetermined number of groups. The outputs of the classification are integers. The outputs of regression models are floating points and the goal is to find a matching function with the data [8].

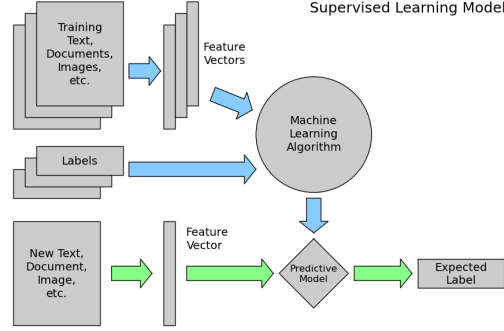


Figure 2: Supervised learning Model[9]

Figure 2 represents the process of a supervised model.

One of the learning methods is to construct an artificial neural network (ANN) which is inspired by our brain and how it is processing the data. An ANN has, like our human neural network, neurons which are connected. The neurons are divided into layers, and the number of layers can vary. In every network there are at least two layers: the input and the output layers. In the input layer, the neurons are receiving the input data. The output layer is the last one and it gives out the final result of the ANN. Besides these two layers, there can be hidden layers. These are optional and there is no fixed number of how many hidden layers that can exist. While the number of nodes of the input and the output is dependent on the size of the input and the size of the wanted output, the number of the nodes of the hidden layer can be freely selected and can be difficult to determine. This is because if the number is too small then the model may not learn but if there are too many layers then the model can result in overfitting [10]. The connections between the neurons are called edges and these edges have weights that influence the input data. Each node also has an activation function which affects the values coming through the connection [11].

The most frequently used ANN is the multilayer perceptron (MLP) and as the name suggests MLP has multiple layers. One property of MLP is that the activation functions are non-linear and that the network is a feed-forward network. A feed-forward network is a network where the neurons in the same layer are not connected, and the edges are only directed towards the next layer [12]. An advantage of MLPs is that they can be used to solve complex and non-linear problems because MLPs are used to solve complex problems the computation of

the output can be difficult. A disadvantage is that the training of a MLP is time-consuming. However, after the training the model can predict the output of new data in a short time. MLPs are able to work with large input data, but they can also reach the same accuracy with a smaller data set [13].

Overfitting is a problem that is common when training a model. Overfitting happens when the model predicts the output of the training set very accurately but when it comes to new data the prediction is not precise. In this case, the model only memorizes the training data instead of learning a pattern. [14]

Firstly, a MLP is build with the number of layers and nodes that are needed. Training a MLP is an iterative process. To train, data is feed forwarded through the network. This means he neural network receives the data and saves the output for every node  $i$  in the output layer:  $\hat{y}_i$ . The second step is to calculate the error  $\delta$  of the output:

$$\delta_i = l(y_i, \hat{y}_i), \quad (6)$$

. where  $y_i$  is how the output for the i-th node should look like. The next step is to back-propagate. The issue of finding the error in the hidden layer is that it is unknown what the output of the node should look like. A way to calculate the error for the nodes in the hidden layer is:

$$\delta_h = \sum_q w_{hq} \delta_q \quad (7)$$

where  $w_{qw}$  denotes the edge from the h-th to the q-th neuron in the hidden layer,  $\delta_q$  is the error of the output of the q-th neuron. The last step is to update the weights:

$$w_{ij} = w_{ij} - \eta o_i \delta_j \quad (8)$$

$\eta$  denotes the learning rate of the learning algorithm. The updated weight will reduce the error in the hidden layer [15].

## 2.2.b Decision Tree

Another popular model, when talking about machine learning algorithms, is the decision tree. Decision trees can be used to predict and classify data [16]. The decision tree is a directed tree and has a root node. Like the input layer of the artificial neural network, the root node has only outgoing edges. On the other side of the edges, new nodes are being created. Contrary to the neuron in the ANN, every node of the decision tree has only one incoming edge except for the root node. The nodes that have incoming and outgoing edges are called internal nodes. The internal nodes have a minimum of two outgoing edges. The splits of the nodes are determined by a discrete function of input values. The nodes that only have an incoming edge and that deliver the result are the leaves. The complexity of the decision tree is influenced by the number of nodes, the number of leaves, the depth of the tree and the number of attributes of the input data [17].

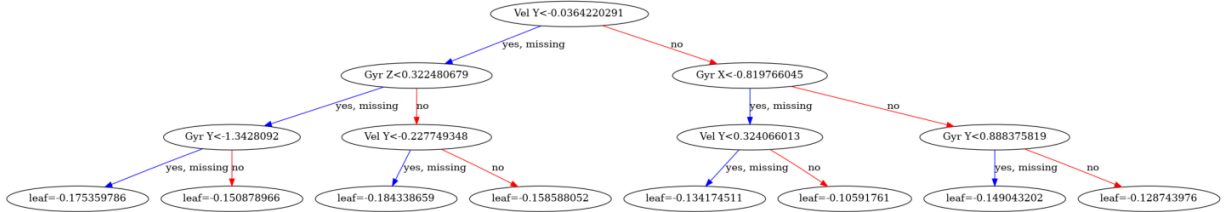


Figure 3: Example for a decision tree

Figure 3 shows a decision tree of the data that is used for this thesis. The advantages of decision trees can prevent overfitting and if the input features have missing values the model is not negatively affected. On the other hand, if the input data is noisy, the model of the decision tree can overfit. The algorithm is also difficult to understand[13].

Some of the important aspects that differentiate decision trees and other machine learning algorithms from one another are that the tree is built by the recursive splitting of the target output based on the input features. A second important feature of decision trees is that it is interpretable [18]. Additionally, the structure of decision trees does not need to be predetermined. For ANNs to decide how many hidden layers are needed and how many nodes each layer should have before the neural network is constructed. The decision tree

algorithm constructs the decision tree without having to specify the number of nodes and the tree depth [19]. A property that decision trees and other machine learning algorithms share is the goal to minimize the error between prediction and the correct output [17].

One type of decision tree is the Stochastic Gradient Boosted Decision Tree (GBDT). GBDT is an additive model which is made up of a group of regression trees. The training for stochastic GBDT is applied to a randomly selected portion of the data. It is less likely that the decision tree will overfit because the portion is randomly selected [18]. To train the decision tree, it is important to first generate the output of the training data  $\hat{y}$ . After that the error of the predicted output can be calculated :

$$\delta = l(y, \hat{y}) \quad (9)$$

where  $\delta$  is the error,  $y$  is how the output should look like and the function  $l(y, \hat{y})$  is the loss function. To minimize the error the decision tree needs to be modified. The modification can be a split to one of the leaf nodes or to update the prediction of a leaf. The equation to calculate a new decision tree is

$$f_{t+1} = f_t + \arg \min_u [\mathcal{L}_t(u) + \Omega(u)]. \quad (10)$$

In the equation,  $f_t + 1$  is the new decision tree,  $f_t$  is the current decision tree and  $u$  is the modification. The loss of the current tree with the modification  $u$  is marked as  $\mathcal{L}_t(u)$ . The  $\Omega(u)$  establish the regularization function.

$$\mathcal{L}_t(u) = \sum_{i=r+1}^t l(y_i, f_t(x_i) + u(x_i)) \quad (11)$$

$$\Omega(u) = \eta |Q_u| + \frac{\lambda}{2} \sum_{j \in Q_u} v_u^2(j) \quad (12)$$

The modification function for a new split is defined as:

$$u(\mathbf{x}) = \begin{cases} v_u(q_u(\mathbf{x})) & \text{if } \mathbf{x} \in \text{Domain}(q_u) \\ 0, & \text{otherwise} \end{cases} \quad (13)$$



In the equations,  $x_i$  denotes the leaf that will be considered to be split in iteration  $i$ ,  $t$  is the current iteration and  $r$  is the iteration where the last modification took place,  $Q_u$  is a set of unique identifiers for the new leaf nodes for the function  $u$  and  $v_u$  identifies the difference of the prediction of the new leaf nodes and the prediction of their parent nodes. The loss function is shown as  $l(y_i, f_t(x_i) + u(x_i))$ .  $\eta$  is the learning rate and  $\lambda$  is the regularization. The function  $q_u$  denotes an identifier from the current node to the new node that would result from the possible split [20].

## 3 Approach

In this section, I will go into details about the quadrotor that I used for this thesis and about the experiment that was used to collect the needed data. Furthermore, I will explain how I calculated the aerodynamic effects that affected the quadrotor. In the end, I will establish how I am going to train the different machine learning algorithms.

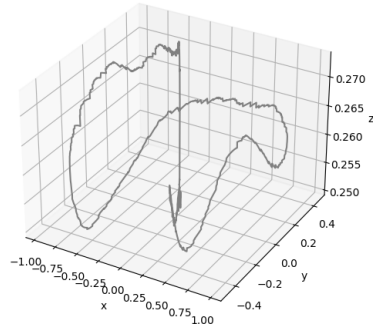
### 3.1 Quadrotor



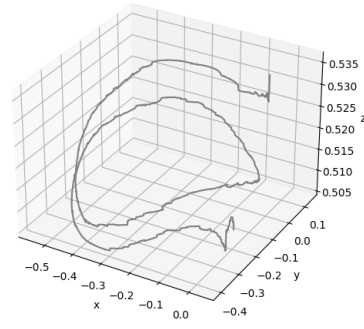
Figure 4: A picture of the Crazyflie 2.1 [4]

In this thesis, I use the Crazyflie 2.1 - as seen in 4- developed by Bitcraze, a company in Sweden. The Crazyflie is a light and small quadrotor with an arm length of 4.6 cm and a weight of 34.7 g [6]. A quadrotor is a multirotor with four rotors and the quadrotor is equipped with an accelerometer and a gyroscope. The quadrotor was collecting the following data: the position, the quaternions, the velocity, the angular velocity, acceleration, the rotational speed of each motor, the pulse-width modulation motor signal and battery voltage.

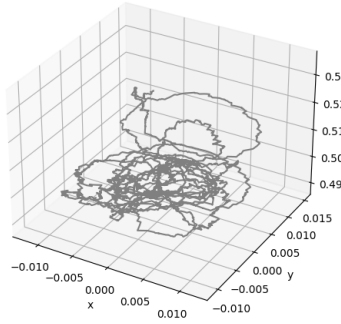
For this thesis, the research lab provided me with the data collection. It is important to note that the quadrotor was only used indoors. The Crazyflie was protected from the outside wind during the flights. The quadrotor has collected 175473 timestamps over the span of 19 flights. The first 9 flights can be classified into three different routes: figure 8, the circle and the yaw. The figures 5 and 6 show the three different routes.



(a) x-y-z plane of the flight route  
figure 8

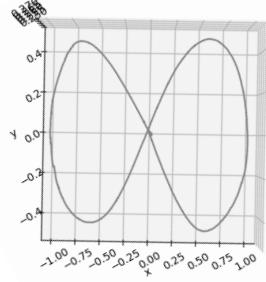


(b) x-y-z plane of the flight route  
circle

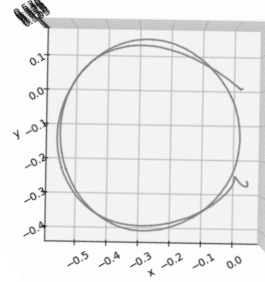


(c) x-y-z plane of the flight route  
yaw

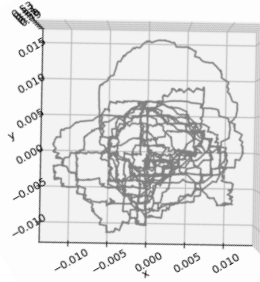
Figure 5: Different trajectories in the x-y-z plane



(a) x-y plane of the flight route figure 8



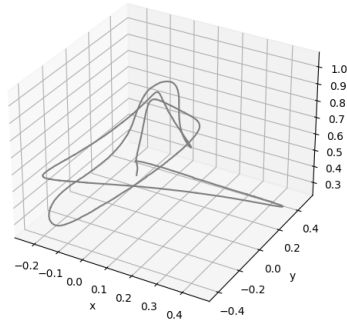
(b) x-y plane of the flight route circle



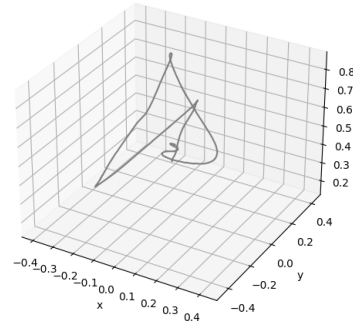
(c) x-y plane of the flight route yaw

Figure 6: Different trajectories in the x-y plane

The other 10 flights were randomly generated flights. The algorithm, that was used to generate the flight routes, was given a duration of time, the minimum and maximum speed and the boundary of the flight area. Even for the flights with the same input values, the trajectories look different because the trajectories were randomly generated.

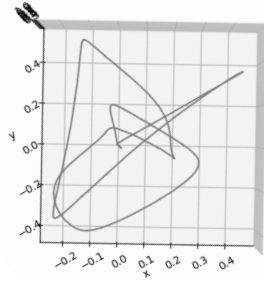


(a) x-y-z plane of a randomly generated flight route

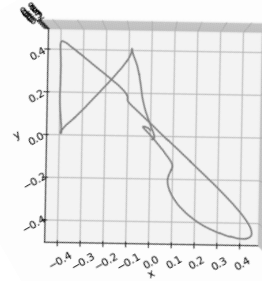


(b) x-y-z plane of a second randomly generated flight route

Figure 7: Randomly generated flight route on the x-y-z



(a) x-y plane of a randomly generated flight route



(b) x-y plane of a second randomly generated flight route

Figure 8: Randomly generated flight route on the x-y

The trajectories in figures 7 and 8 are having the same input data, but - as the images show - the trajectory is different. In contrary to the first ten flights, I cannot divide the remaining flights into subgroups like circle, figure 8 or yaw.

The data I got from the Crazyflie is encoded in a binary form. To access the encoded data I used an existing python script which was published in the Crazyflie Firmware git [21]. To be able to use the data for the thesis, the data needs to be in the SI units. The acceleration and the angular velocity are not in their SI units, the acceleration is in the g-unit and the angular velocity is in deg/s. The SI unit for the acceleration is  $\text{m/s}^2$ . To get from the g-unit to  $\text{m/s}^2$ , the values need to be multiplied by 9.81. For the angular velocity, I need to convert the data to rad/s, to do that, I need to multiply the data by 0.017453. For the equation (3), I also need the inertia matrix. The thesis [6] experimentally determined the inertia matrix for the Crazyflie 2.0. I can use the discovered inertia matrix of the Crazyflie 2.0 for the Crazyflie 2.1 as well because the Crazyflie 2.0 and Crazyflie 2.1 are the same weight and the same size.

### 3.2 Basic Forward Propagation

The goal of the basic model is to have an easy way to predict the data of the next time step of the flight. To propagate the basic model, I first need to calculate the acceleration and the angular acceleration. The acceleration and angular acceleration depend on the total thrust and body moments. To calculate the total thrust and the body moments, I can either use the equation (4) or the equation (5). I did not find the value of the thrust coefficient in the papers I used to research about the Crazyflie. To calculate the force of a rotor a polynomial function is given in the paper [5]:

$$\psi_i(\hat{p}_i, \hat{b}) = 11.09 - 39.08\hat{p}_i - 9.53\hat{b} + 20.57\hat{p}_i^2 + 38.43\hat{p}_i\hat{b}, \quad (14)$$

where  $\hat{p}_i$  denotes the normalized PWM signal of the i-th rotor and  $\hat{b}$  is the normalized battery voltage. For the equation (5) I only need to find out the length of the arm of the Crazyflie and the thrust-to-torque ratio. The battery voltage and the PWM signal is given in the data collection and I just need to normalize them. The arm length and the thrust-to-torque ratio is given in the thesis [6]. For the acceleration, I need to calculate the rotation matrix to get the body frame. The Crazyflie collected the quaternions  $\mathbf{q} = \begin{pmatrix} q_0 & q_x & q_y & q_z \end{pmatrix}$ . With

quaternions the rotation matrix  $\mathbf{R}$  can be created with the following equations [22]:

$$\mathbf{R} = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_0q_z & 2q_xq_z + 2q_0q_y \\ 2q_xq_y + 2q_0q_z & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_0q_x \\ 2q_xq_z - 2q_0q_y & 2q_yq_z + 2q_0q_x & q_0^2 - q_x^2 - q_y^2 - q_z^2 \end{pmatrix} \quad (15)$$

After calculating the rotation matrix  $\mathbf{R}$  I can use the equation (3) to calculate the angular acceleration and with the equation (2) I can calculate the acceleration. The remaining data I can calculate by integrating the acceleration and angular acceleration. To integrate the quaternion I used this equation [23]:

$$\mathbf{q}' = \exp \left( \frac{1}{2} \begin{bmatrix} -[\boldsymbol{\omega} \times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \right) \mathbf{q}_0, \quad (16)$$

where  $\boldsymbol{\omega}$  is the angular velocity of the current time stamp.

The  $[\boldsymbol{\omega} \times]$  is the cross product matrix and looks like:

$$\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (17)$$

The velocity and the position can be calculated with the integral of acceleration[24]:

$$\begin{aligned} \mathbf{v}' &= \dot{\mathbf{v}}t + \mathbf{v}_0 \\ \mathbf{r}' &= \mathbf{v}t + \mathbf{r}_0 \end{aligned} \quad (18)$$

In the equations (18)  $\mathbf{v}$  denotes the velocity and  $\mathbf{r}$  is the position of the quadrotor. With the angular acceleration, I can calculate the angular velocity [25]:

$$\boldsymbol{\omega}' = \dot{\boldsymbol{\omega}}t + \boldsymbol{\omega}_0 \quad (19)$$

In equations 16, 18 and 19 the variables with ' are the new values for position, velocity, and angular velocity. The variables with <sub>0</sub> are the positions, velocity, and angular velocity values in the last time step.

The data  $y$  consists of acceleration, angular acceleration, velocity, angular velocity, position and quaternion are to be propagated. Before the propagation I need to normalize the PMW signal of each rotor  $i$   $\hat{p}_i$  and the battery voltage  $\hat{b}$ . The algorithm that will propagate the data  $y$  will iterate through the timestamps of the flight. To calculate the data of the new timestamp  $\hat{y}_{t+1}$ , I will use collected data  $y_t$ . For the new time stamp  $t + 1$  the basic forward propagation model will first calculate the total thrust with  $\hat{p}_{it}$  and  $\hat{b}$  and afterwards I will calculate the total thrust and body moments  $\mathbf{u}_t$ . With the  $\mathbf{u}_t$  I can now calculate the acceleration  $\dot{\mathbf{v}}_{t+1}$  with the equation (2) and the angular acceleration  $\dot{\boldsymbol{\omega}}_{t+1}$  next. The velocity and the position are calculated with the equation (18). The angular acceleration is integrated and with the (??) the angular velocity and the quaternion for the time stamp  $t + 1$ .

### 3.3 Residual Force And Residual Torques

As in section 2.1.a explained, there are aerodynamic forces even if the quadrotor is flying indoors. The equations 2 and 3 are ignoring the aerodynamic forces and that can affect the accuracy of the basic model, in particular if the quadrotor has a high velocity [26, 6]. To get a more accurate prediction, I can calculate the residual force  $f_a$  and the residual torque  $\tau_a$  [27]:

$$m\dot{\mathbf{v}} = m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \mathbf{R}u_1 + \mathbf{f}_a \quad (20)$$

$$\mathcal{J}\dot{\boldsymbol{\omega}} = \mathcal{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} + \boldsymbol{\tau}_a \quad (21)$$

To calculate the disturbance, I first need to rearrange the equations to  $\mathbf{f}_a$  and  $\boldsymbol{\tau}_a$ .

$$\mathbf{f}_a = m\dot{\mathbf{v}} - m\mathbf{g} - \mathbf{R}u_1 \quad (22)$$

$$\boldsymbol{\tau}_a = \mathcal{J}\dot{\boldsymbol{\omega}} - \mathcal{J}\boldsymbol{\omega} \times \boldsymbol{\omega} - \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad (23)$$



For (20), I have all of the needed data to calculate  $f_a$  with the use of (5). For the equation (21) to calculate the residual torque, I need to find out the angular acceleration because the quadrotor has no sensor to collect the values of the angular acceleration. I cannot use the equation (3) because it was used for the basic model since residual torque could be ignored and so the residual torque would be 0. The Crazyflie collects the data of the angular velocity  $\omega$  and I can calculate the angular acceleration with the derivative of the angular velocity[25]:

$$\dot{\omega} = \frac{d\omega}{dt} \quad (24)$$

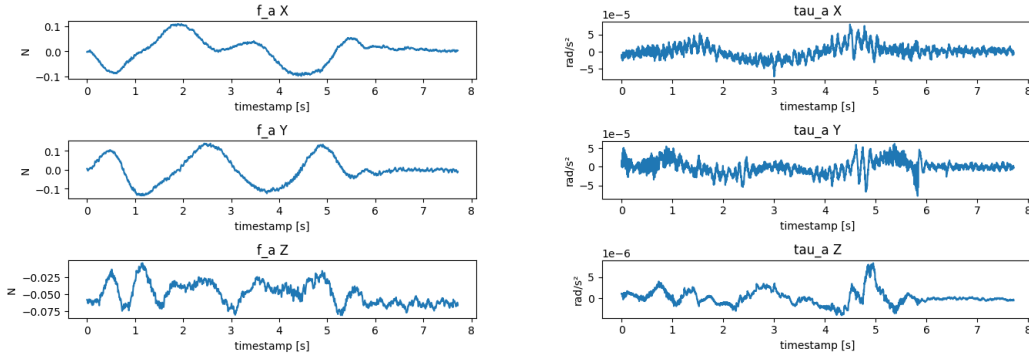


Figure 9: Example of the calculated residual force and residual torque of one flight

As figure 9 shows, the residual force and the residual torque are close to 0. But I can also notice that the force is in the range of -0.1 and 0.1 N while the residual torque is in the range of  $-0.00005$  and  $0.00005 \text{ rad/s}^2$ . The plots also show that the residual force and especially the residual torque have noise.

### 3.4 Supervised learning

For the supervised learning algorithm I used a multilayer perceptron which can solve a regression problem and learns functions to predict the residual force and residual torque. To test how well the model is trained, I separated one flight from the others. That one flight is

the test set on which I will validate my model, while the remaining flights are my training dataset. I need to scale the data, for the residual force and residual torque to be equal in size, because the residual torque is much smaller than the residual force. To program the MLP I used the programming language Python and the open-source library pyTorch. To find out how well the model performs in each training epoch, I need to use a loss function  $l(\mathbf{y}, \hat{\mathbf{y}})$ . I used the mean squared error which is calculated with the following equation[28]:

$$l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_{i=1}^n (y_i, \hat{y}_i)}{n}, \quad (25)$$

where  $y$  and  $\hat{y}$  has the size  $n$ . To update the weight in each epoch, I used the Adam algorithm which uses first-order gradients. The Adam optimization is used in the epochs of training and  $t$  is the current epoch. The optimizer calculates the gradient and with the gradient the algorithm evaluates the moving averages of the gradient  $m_t$  and the squared gradient  $v_t$ . The moving averages of the gradient and the squared gradient are influenced by hyper-parameter  $\beta_1, \beta_2$ . These hyper-parameter are in the range between 0 and 1 [29].

---

**Algorithm 1** Adam algorithm

---

**Require:**  $\eta$ : learning rate,  $\beta_1, \beta_2$ : Exponential decay rates for the moment estimates,  $f()$ : Stochastic objective function with weights  $w$ ,  $w_0$ : Initial weight vector

```

1:  $m_0 \leftarrow 0$ 
2:  $v_0 \leftarrow 0$ 
3:  $\hat{v}_0^{max} \leftarrow 0$ 
4: while  $w_t$  not converged do
5:    $t \leftarrow t + 1$ 
6:    $g_t \leftarrow \nabla_w f_t(w_{t-1})$ 
7:    $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$ 
8:    $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$ 
9:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
10:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
11:   $\leftarrow w_{t-1} - \eta * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ 
12: end while
13: return  $w_t$ 
```

---

### 3.5 Decision Tree

Just like I did for the supervised model, I also have to prepare the data for the decision tree beforehand. I also have to scale the disturbance torque between -1 and 1. I need to divide the collected flight data into two different groups, one is the test data and the other is the training data. The test data consists of one flight and the training data consists of the remaining flights. To implement the gradient-boosted decision tree I used the XGBoost library. The goal of the decision tree is to minimize the square error. To optimize the decision tree I must use an objective function: [30]

$$obj(\theta) = L(\theta) + \Omega(\theta), \quad (26)$$

in this equation (26)  $L(\theta)$  denotes the loss, while  $\Omega(\theta)$  denotes the regularization function and  $\theta$  is the best parameter that will minimize the loss and complexity of the decision tree. The loss can be calculated with the loss function of the mean squared error:

$$L(\theta) = l(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (27)$$

where  $y$  is expected output,  $\hat{y}$  denotes the predicted output and  $n$  is size of  $y$  and  $\hat{y}$ . The complexity of the decision can be calculated with:

$$\omega(\theta) = \eta T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \quad (28)$$

where in the equation (28)  $\eta$  denotes the learning rate,  $\lambda$  is the regularization,  $T$  denotes the total number of leaves and  $w_j$  is the vector score of the leaf  $j$ . The objective function can be written with the equations (27) and (28):

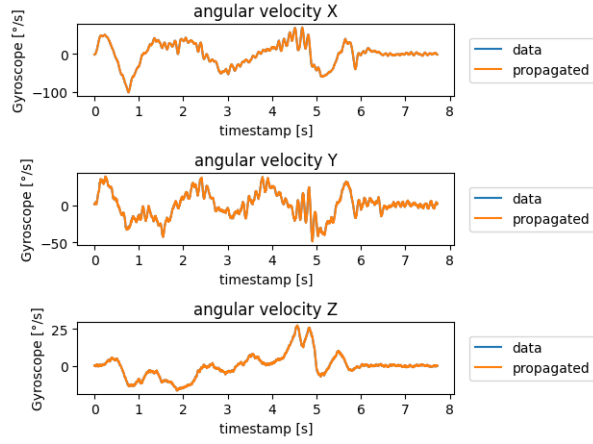
$$obj(\theta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \eta T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2. \quad (29)$$

## 4 Results

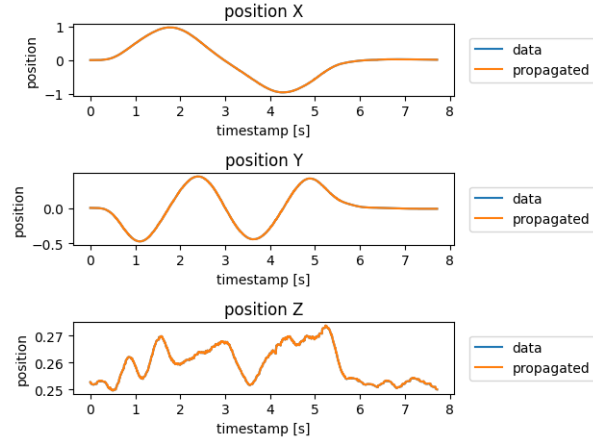
In this section, I will present the results of the propagation of the data without aerodynamic forces. Furthermore, I am going to show the different models that I used to predict the residual force and residual torque. I am going to try out different input features and I am going to change the complexity of the learning models.

### 4.1 Basic Forward Propagation

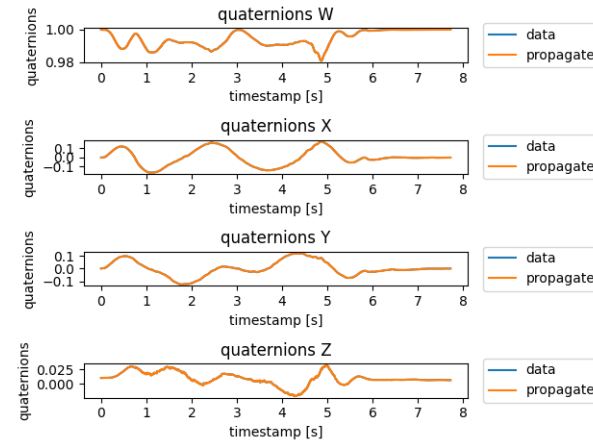
The propagation for most of the data is almost equal to the collected data. Looking at the angular velocity, position and quaternion the propagated data and the collected data are almost identical.



(a) Example of the angular velocity



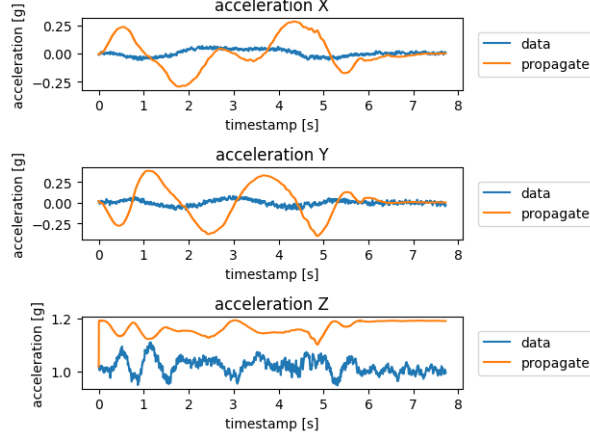
(b) Example of the position



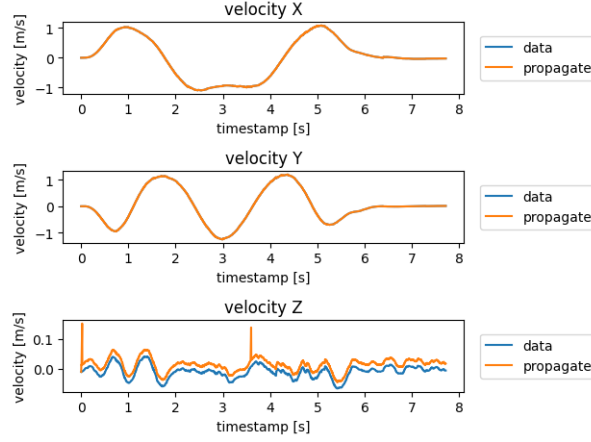
(c) Example of the quaternion

Figure 10: Comparison between the output of the propagation and the collected data

However, the propagated data of the velocity and the acceleration show some differences from the collected data.



(a) Example of the acceleration

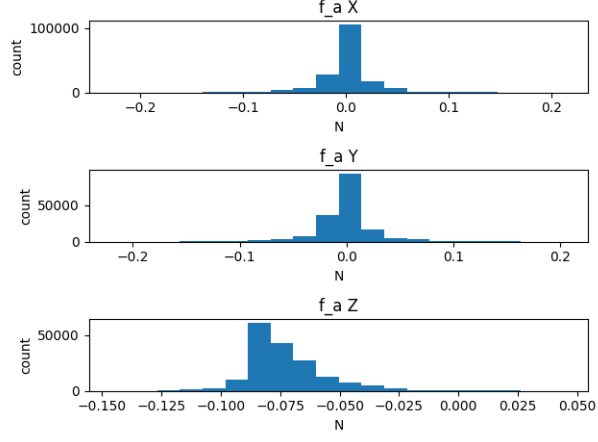


(b) Example of the velocity

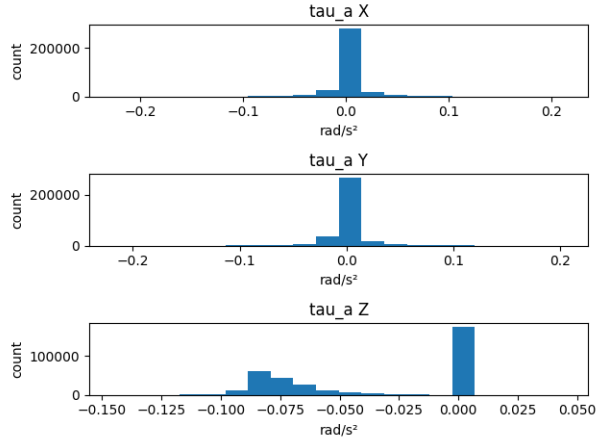
Figure 11: Comparison between the output of the propagation and the collected data

As shown in figure 11, the  $\mathbf{v}_z$ , the z-axis of the velocity, has an error and the error is for every flight around 0.025. On the other hand when it comes to the acceleration, the propagated data for every axis is completely different. When compared to the other data, the error of the acceleration is not the same for every flight.

## 4.2 Residual Force and Residual Torque



(a) Histogram of the residual force



(b) Histogram of the residual torque

Figure 12: Histogram of the complete residual force data and residual torque data

Figure 12 shows the distribution of the complete residual force data and the distribution of all the calculated residual torque. In the histogram I can see that most data is around 0.0 for  $\mathbf{f}_a X$ ,  $\mathbf{f}_a Y$ ,  $\boldsymbol{\tau}_a X$  and  $\boldsymbol{\tau}_a Y$ . Most data of  $\mathbf{f}_a Z$  is around -0.08. Meanwhile, most of  $\boldsymbol{\tau}_a Z$  are also around 0.0 but I also notice an increased number of data around -0.08. Table 1 shows that the residual torque is smaller than the residual force. The table also shows that  $\mathbf{f}_a X$  is

similar in size as  $\mathbf{f}_a Y$ , exactly as  $\boldsymbol{\tau}_a X$  has a similar size as  $\boldsymbol{\tau}_a Y$ . The  $\boldsymbol{\tau}_a Z$  has the smallest mean with  $7.8 * 10^{-8}$ , while  $\mathbf{f}_a Z$  has the biggest mean.

Table 1: Mean and standard deviation of the residual force and residual torque

	mean	standard deviation
$\mathbf{f}_a X$	$9.2 * 10^{-4}$	$2.9 * 10^{-2}$
$\mathbf{f}_a Y$	$-5.8 * 10^{-4}$	$3.1 * 10^{-2}$
$\mathbf{f}_a Z$	$-7.3 * 10^{-2}$	$1.7 * 10^{-2}$
$\boldsymbol{\tau}_a X$	$-5.0 * 10^{-6}$	$1.4 * 10^{-5}$
$\boldsymbol{\tau}_a Y$	$2.3 * 10^{-6}$	$1.1 * 10^{-5}$
$\boldsymbol{\tau}_a Z$	$7.8 * 10^{-8}$	$6.2 * 10^{-6}$

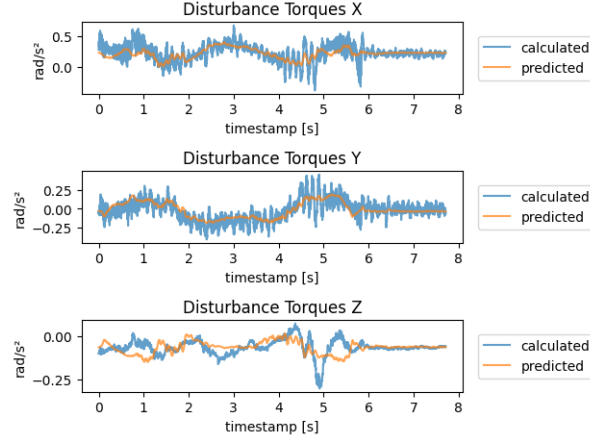
### 4.3 Supervised learning

The input of supervised learning can be acceleration, velocity, position, angular velocity and the rotation matrix. The output is the residual force and the residual torque. The objective is to find the most accurate supervised learning model that can predict the residual torque and the residual force in the shortest time.

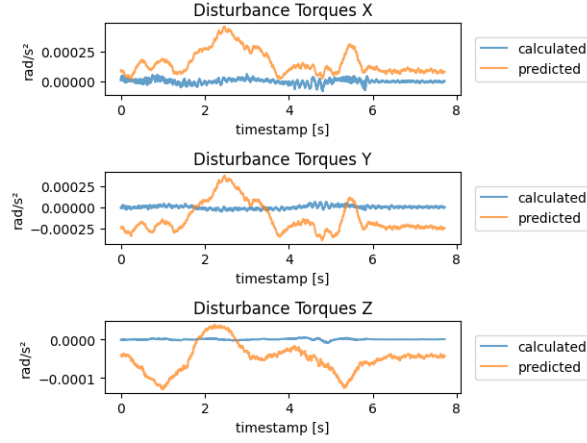
#### 4.3.a Scaling the data

As seen in 9 and 12, the residual force and torque are kind of small. In order for the model to learn, I need to scale the output. I scaled the output with a MinMaxScaler from the sklearn library and the scaler scaled the values between -1 and 1.





(a) residual torque prediction of the test data of scaled model I,



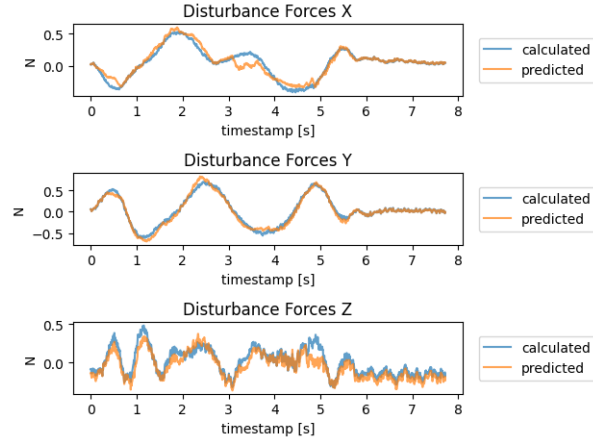
(b) residual torque prediction of test data of the unscaled model II,

Figure 13: Comparison of the predicted and the expected residual torque of the models

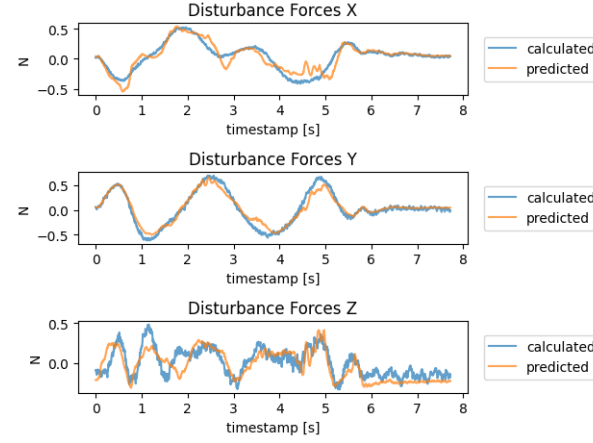
As shown in figure 13b, the predicted torque is not accurate for the model of the unscaled data. This is a possibility of outcome because the loss of torque is much smaller than the force. This is caused due to the difference in size between the residual force and residual torque. The result of figure 13b show that the residual torque is between 0.0003 and -0.00025, whereas the residual torque range is between 0.5 and -0.25 in figure 13a.

#### **4.3.b Comparison of different input features**

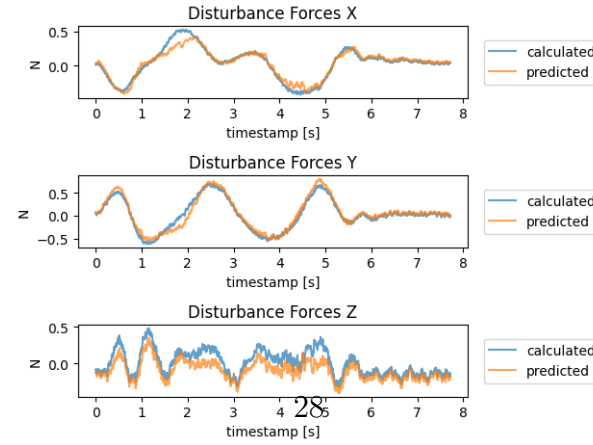
The model tries to find patterns in the input features that correlate with the output data. Too much data can extend the training duration. In figures 14 and 15 are showing the results of trying out different input features. For the first input features I tried the the complete data set. The next input features that I used were the first two columns of the rotation matrix, the velocity and the angular velocity. The two columns are sufficient enough for the model to know the complete rotation matrix. The last input features I used were the acceleration, the first two columns of the rotation matrix and the angular velocity.



(a) Model I with the input features: complete data set,



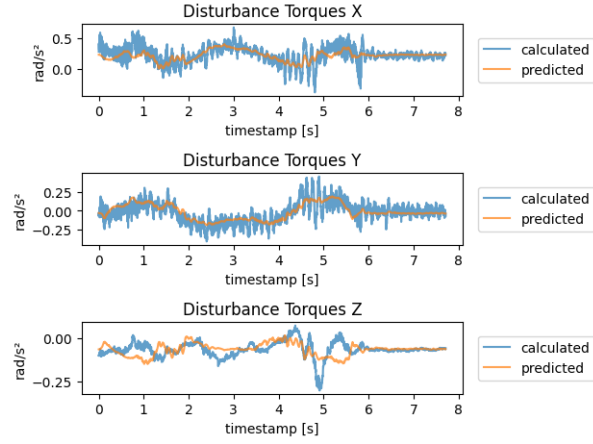
(b) Model III with the input features: rotation matrix, velocity and angular velocity,



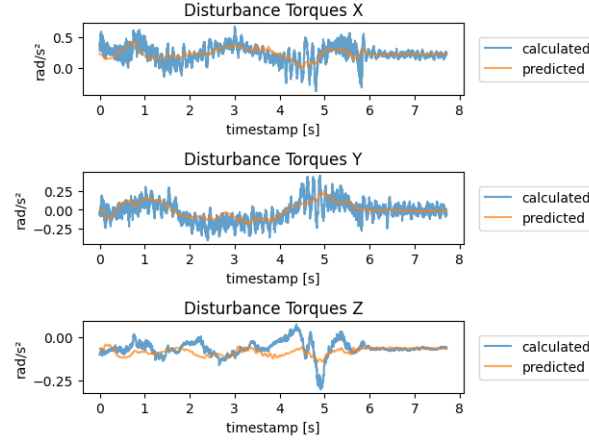
(c) Model IV with the input features: rotation matrix, acceleration and angular velocity,

Figure 14: Comparison between predicted residual force and the expected residual force of the test data

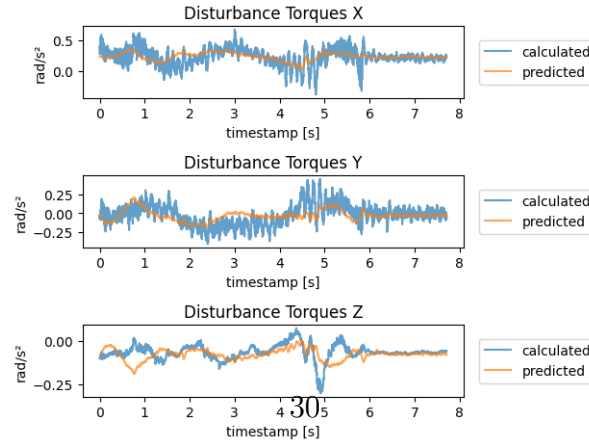
As shown in 14b, the input features velocity, angular velocity and the rotation matrix deliver the least accurate output of the residual force. Both the figures 14a and 14c yield a decent model. Figure 14c has the most accurate output for the residual force.



(a) Model I with the input features: complete data set,



(b) Model III with the input features: rotation matrix, velocity and angular velocity,



(c) Model IV with the input features: rotation matrix, acceleration and angular velocity,

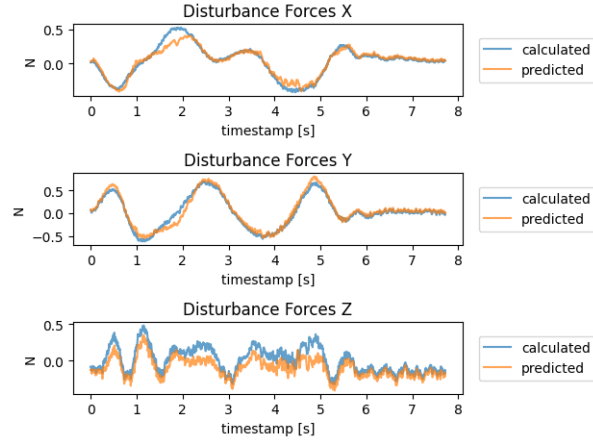
Figure 15: Comparison between predicted residual torque and the expected residual torque of the test data

As shown in figure 15, the prediction of the residual torque is not as precise as the prediction of the residual force. The model with the complete data set 15a has the highest error of the predicted torque. The most accurate result is by the model with the input features velocity, angular velocity and the rotation matrix, as is shown in figure 15b.

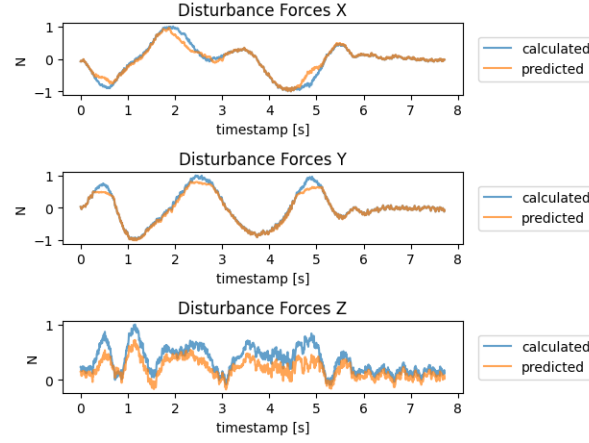
The overall most accurate model was with the input features of acceleration, the first two columns of the rotation matrix and the angular velocity. The error of the predicted residual torque was not the smallest but the model had the second smallest error when it came to the residual torque. This model had the most accurate output of the residual force whereas the model with the smallest residual torque error had by far the highest residual force error. In the following analysis of the MLP I will use the first two columns of the rotation matrix and the angular velocity as the input features.

#### **4.3.c Using different input sizes for the model**

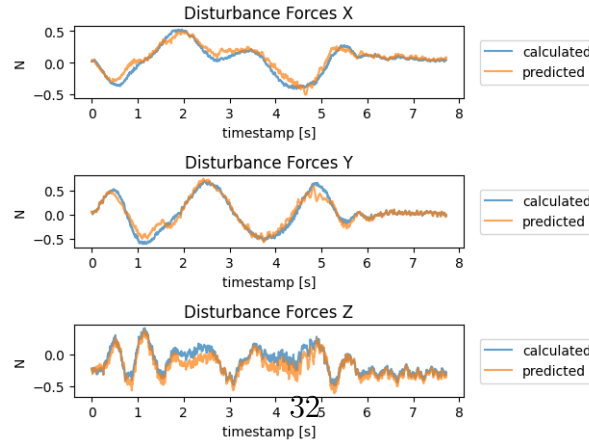
In this part of the thesis, I will look into how many flight routes the training set needs to have. The training set with all flights - except the test data - has 171555 data points. The duration of the training is dependent on how many data points exist. To reduce the data points I am going to divide the trajectories into two groups: the random trajectories and the trajectories which flight a figure. The random trajectories have 68924 data points and the second group includes 102631 data points.



(a) The training set of the model IV has 171555 data points,



(b) The training set of the model V has 68924 data points ,

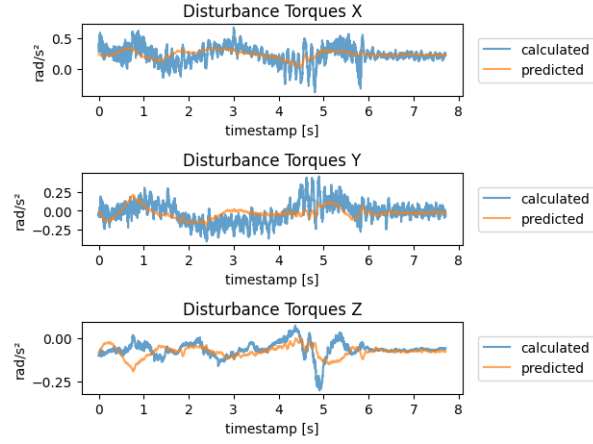


(c) The training set of the model VI has 102631 data points,

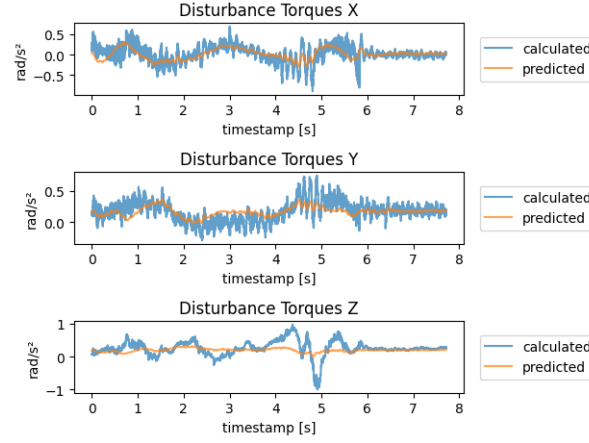
Figure 16: Comparison of the residual force output the test data and the expected residual force.

Figure 16 shows that the accuracy of the residual force is dependent on the size of the train set. Figure 16a has the most data points and the error of the residual force is the smallest. Figure 16b shows that 69894 data points are not sufficient for the model to learn the pattern of the input features and the residual force.

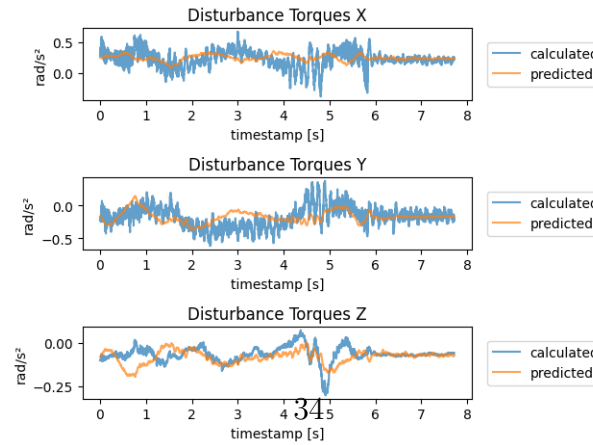




(a) The training set of the model IV has 171555 data points,



(b) The training set of the model V has 68924 data points ,



(c) The training set of the model VI has 102631 data points,

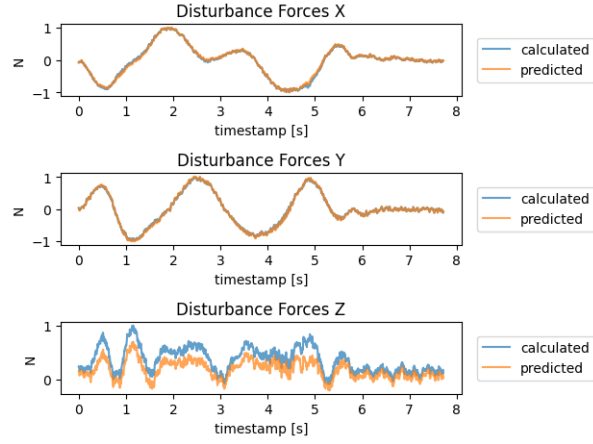
Figure 17: Comparison of the residual torque output of the test data with the expected residual torque.

Exactly as the residual force, the accuracy of the predicted residual torque is dependent on the size of the training data set as figure 17 shows. Figure 17b displays that the  $\tau_a Z$  did not learn a pattern. Throughout the three models, we see that the values for  $\tau_a Z$  are difficult to predict because the values for  $\tau_a Z$  are small.

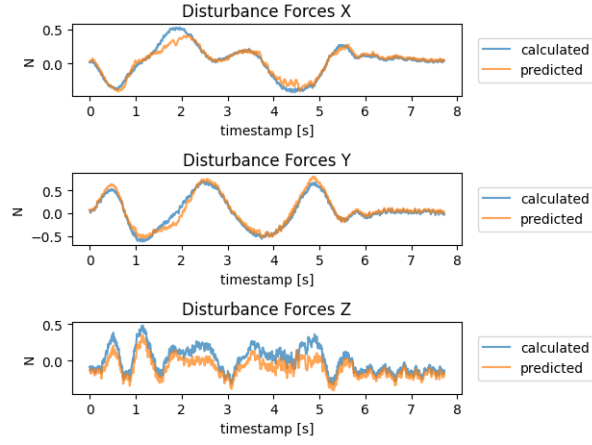
Overall the model with the largest training set is the most accurate for the prediction of the residual force and the residual torque. The error difference between the model with 171555 data points and the model with 102631 data points is too high, this is why I am still going to use the training set with 171555 data points even if the training duration is longer.

#### 4.3.d Changing the complexity

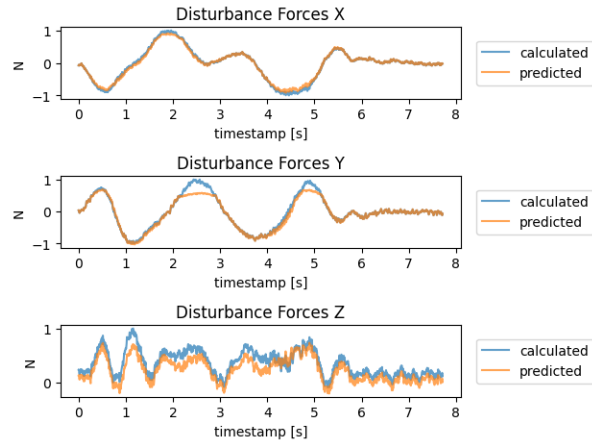
A way to have a more time-efficient algorithm is to reduce the complexity. Reducing the complexity also helps with overfitting. In this part, I will build one MLP with only the input layer and the output layer. The second model will have one hidden layer between the input and output layers. The model with the highest complexity has two hidden layers.



(a) The model VII has 0 hidden layers,



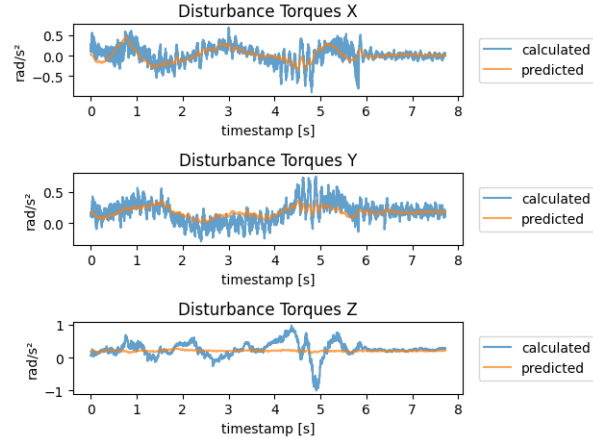
(b) The model IV has 1 hidden layer,



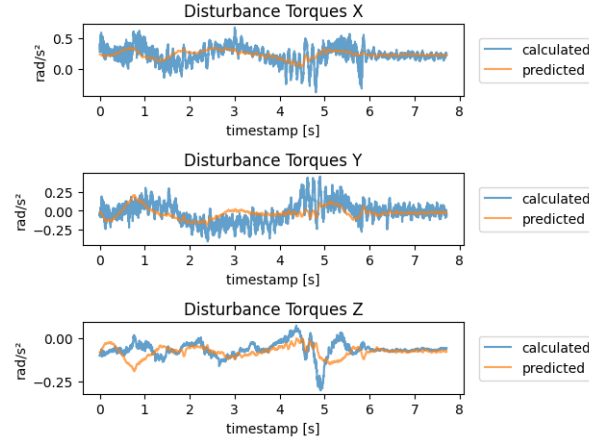
(c) The model VIII has 2 hidden layers,

Figure 18: Comparison between the predicted residual force and the expected residual force of the test data

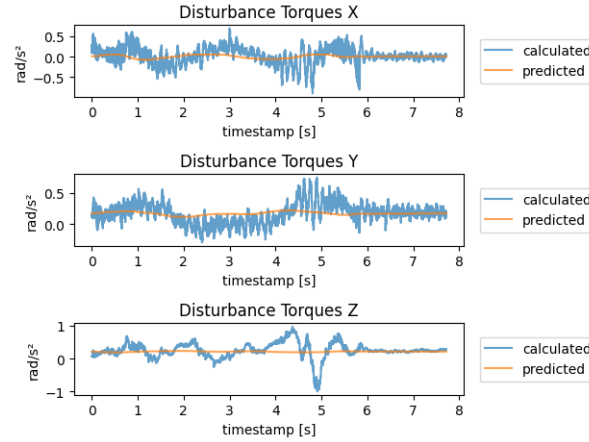
As shown in the figures 18a and 18c, the prediction is negative influenced by using no hidden layer or by using two hidden layers. The model with one hidden layer has the smallest average error.



(a) The model VII has 0 hidden layers,



(b) The model IV has 1 hidden layer,



(c) The model VIII has 2 hidden layers,

Figure 19: residual torque, l: 0 hidden layers, m: 1 hidden layer, r: 2 hidden layers

For the residual torque, I can see in the figure 19b that the model with one hidden layer is the most accurate. As shown in the figures 19a and 19c, the prediction of the residual torque is negatively influenced by using no hidden layer or by using two hidden layers.

#### 4.3.e Comparing the different models of MLP

Table 2: Supervised learning average error

	$f_a X$ in N	$f_a Y$ in N	$f_a Z$ in N	$\tau_a X$ in rad/s <sup>2</sup>	$\tau_a Y$ in rad/s <sup>2</sup>	$\tau_a Z$ in rad/s <sup>2</sup>
Model I	0.067	0.046	0.105	0.084	0.076	0.037
Model II	0.067	0.059	0.602	2.222	0.400	0.180
Model III	0.039	0.050	0.118	0.074	0.076	0.035
Model IV	0.022	0.039	0.077	0.072	0.087	0.041
Model V	0.052	0.057	0.073	0.099	0.114	0.038
Model VI	0.083	0.043	0.168	0.106	0.092	0.149
Model VII	0.086	0.053	0.143	0.078	0.088	0.045
Model VIII	0.029	0.042	0.103	0.073	0.088	0.039

Table 3: Supervised learning time in seconds

	training time	prediction time
Model I	67.86	0.28
Model II	72.99	0.29
Model III	71.08	0.29
Model IV	66.93	0.28
Model VI	38.11	0.28
Model VII	29.25	0.24
Model VII	52.17	0.16
Model VIII	80.46	0.39

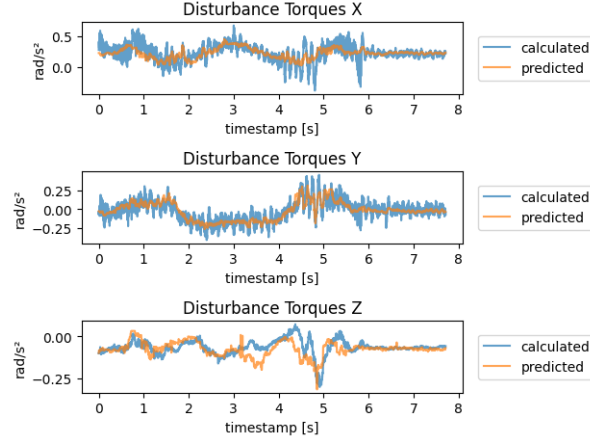
In table 2 I calculated the average error for every model that I presented in the previous sections. In table 3, I stopped the time that the model needed to train the MLP model and I also stopped the time that is needed to predict the residual force and residual torque. It is evident that model IV has the smallest average error. Model IV has one hidden layer, a scaled output and the input features acceleration, angular velocity and the two first columns of the rotation matrix. The model is trained on 171555 data points. The model with the input features velocity, angular velocity and the two first columns of the rotation matrix has a smaller error for  $\tau_a Y$  and  $\tau_a Z$ . The difference between these errors from model III and IV is small. The worst performing model was the model II where I did not scale the residual output and residual force. The highest average error was 2.222 for  $\tau_a X$ .

The fastest model to train was the model VI. This model had the smallest training set. The longest training time was 80.46 seconds for model VIII. Except when I changed the number of layers, the prediction time for the test data set took about 0.24-0.29 seconds. The model with no hidden layer took 0.16 seconds to predict the output of the test data. The longest prediction time was 0.39 seconds for model VIII. The model IV with the smallest average error took 66.93 seconds to train. In comparison with the other models, the model IV is the most accurate and is situated in the middle of the training and prediction time.

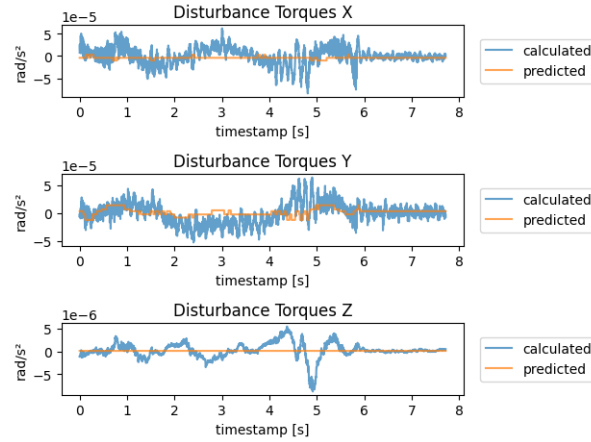
## 4.4 Decision Tree

In contrast to the MLP, the decision tree algorithm can create the optimal structure of the decision tree itself. Nevertheless, I need to give the tree specifications as to how complex it should be. But before I change the structure of the tree, I need to figure out the attributes that are not given with the optimal structure.

#### 4.4.a Scaling the residual torque



(a) scaled residual torque prediction of the test data of tree I,



(b) unscaled residual torque prediction of the test data of tree II,

Figure 20: Comparing the prediction of the residual torque and the expected residual torque

I discovered that by ignoring the size difference between residual force and residual torque, the model will not learn much about the torque and will often assume that the residual torque is 0 as it is shown in figure 20b. This can happen when the value of residual torque is



a hundred times smaller than the residual force, the error of the residual torque prediction is also much smaller and will not have a high impact on the decision tree. Figure 20a displays that the residual torque can be learned when the residual torque is scaled between -1 and 1  $\text{rad/s}^2$ .

#### 4.4.b Comparison of different input features

To get an easy model of the decision tree, it is important to know which features should be used for the training and prediction. Xgboost has a function that makes it possible to see the impact of the features on the output. While using all of the collected data, as figure 21 shows, the most important feature is the angular velocity which is the only data that has a direct impact on the residual torque as the equation (21) shows.

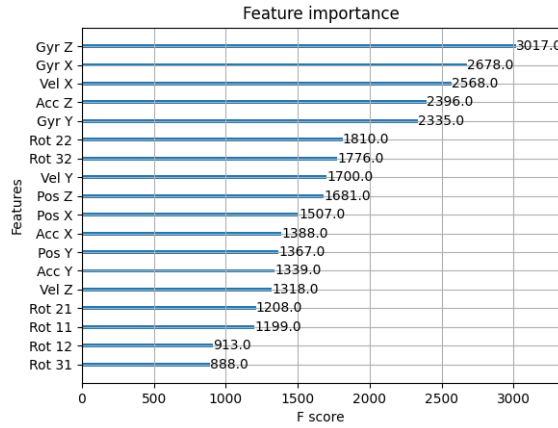
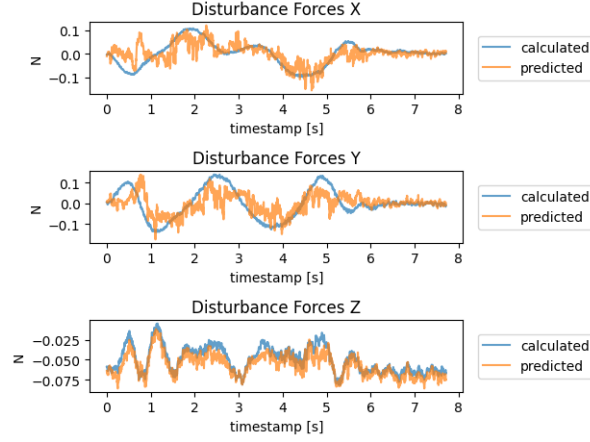
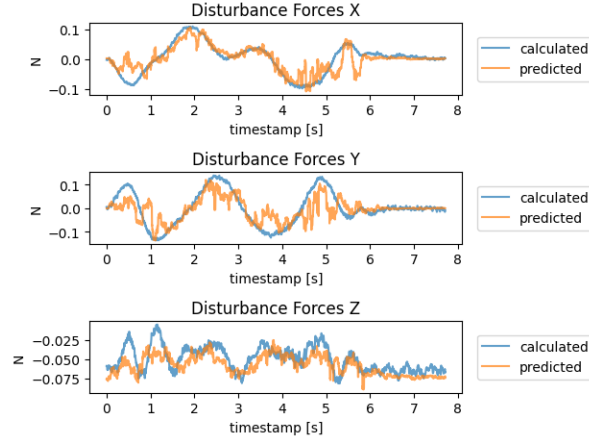


Figure 21: importance of features when all data is used as input

As the figure 21 shows, the next important features are acceleration and velocity. The equation (20) also shows that both attributes are influencing the residual force.



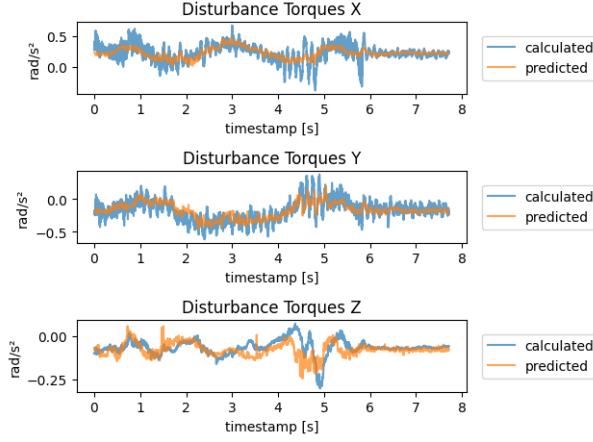
(a) Tree III with the input features: acceleration and angular velocity,



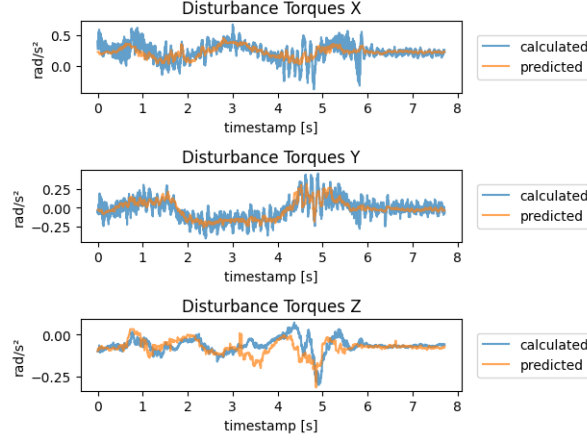
(b) Tree I with the input features: velocity and angular velocity,

Figure 22: Comparison of the predicted residual force of the different input features of the decision tree

As figure 22a and figure 23b show, the predicted residual force is quite noisy. By only looking at the figures, it is difficult to see which of the decision trees predicts a more accurate residual force. after calculating the average error I am able to notice that tree I has a smaller average error with 0.021 N in comparison to the error of model III with 0.024 N.



(a) Tree III with the input features: acceleration and angular velocity,



(b) Tree I with the input features: velocity and angular velocity,

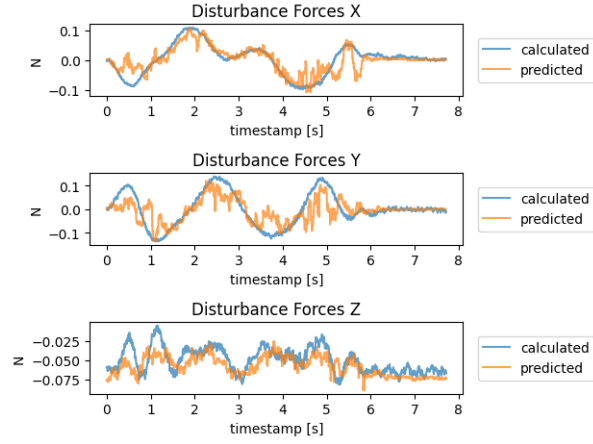
Figure 23: Comparison of the predicted residual torque of the different input features of the decision tree

Similar to the residual force, it is difficult to detect in the figures 23 which of the two decision trees predicts a more accurate residual torque. The average error of the residual torque for tree I is  $0.053 \text{ rad/s}^2$  and the average residual torque error for the decision tree III is  $0.054 \text{ rad/s}^2$ .

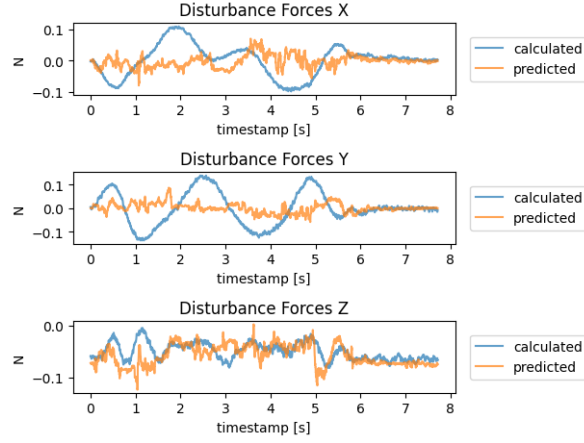
Both of the decision trees have a similar mean average error but the decision tree with the velocity and angular velocity as input features is a little bit more accurate. The time it takes to train the decision tree I is shorter than for III.

#### **4.4.c Using different sizes of input features of the model**

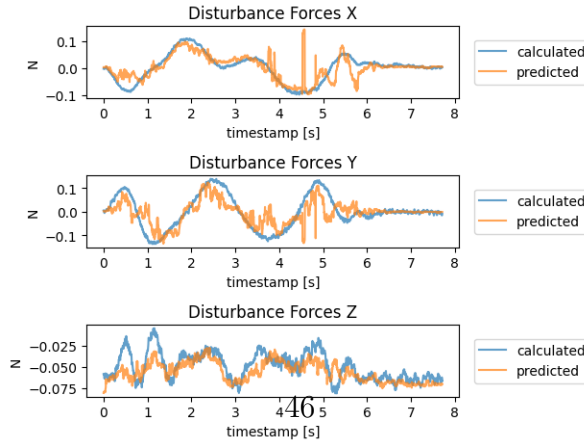
The training time is dependent on the size of the input data. To reduce the training time, I divide the trajectories into two subgroups: the randomly generated flights and the pre-planned trajectories.



(a) The training set of the decision tree I has 171555 data points,



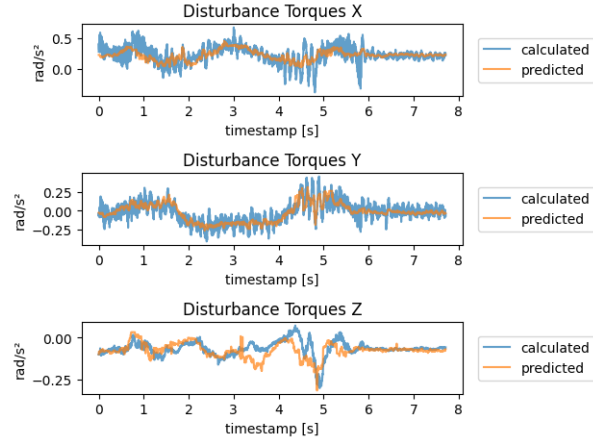
(b) The training set of the decision tree IV has 68924 data points ,



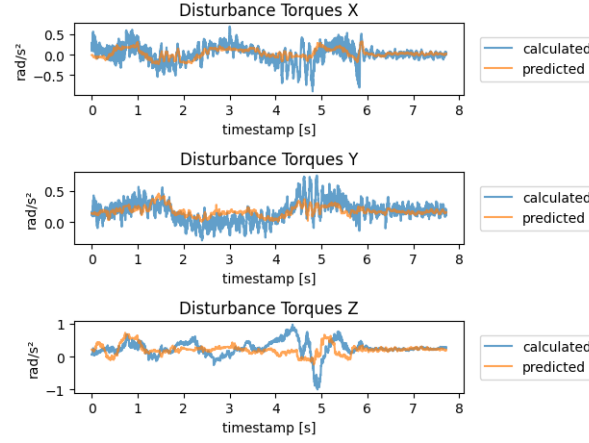
(c) The training set of the decision tree V has 102631 data points,

Figure 24: Comparison of the residual force output of the test data with the expected residual force

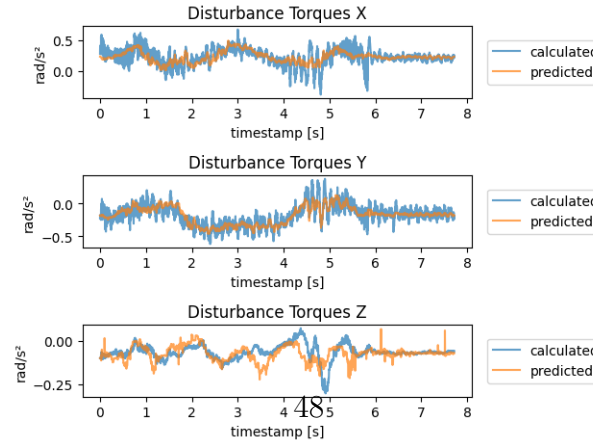
As shown in figure 24a, the predictive residual force is the most accurate. For the residual force, figure 24 displays that the higher the data size of the input the more precisely the decision tree. Decision tree IV has the least amount of data points and it does not learn the pattern of the residual force as seen in figure 24b.



(a) The training set of the decision tree I has 171555 data points,



(b) The training set of the decision tree IV has 68924 data points ,



(c) The training set of the decision tree V has 102631 data points,

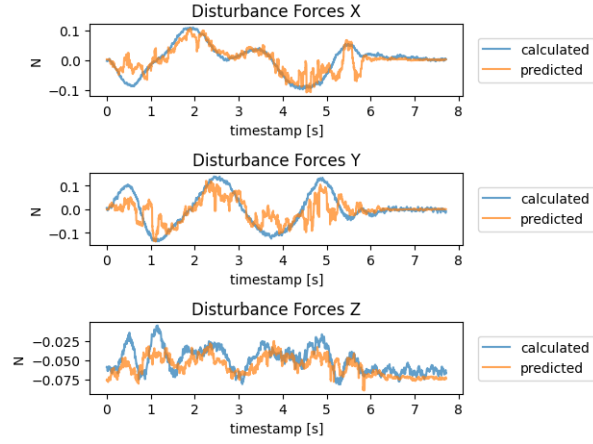
Figure 25: Comparison of the residual torque output of the test data with the expected residual torque.

Figure 25 shows that while the accuracy is dependent on the size of the training set, the average error of the predicted residual torque does not look as big as the average residual. The decision tree I predicts the most accurate residual force and residual torque.

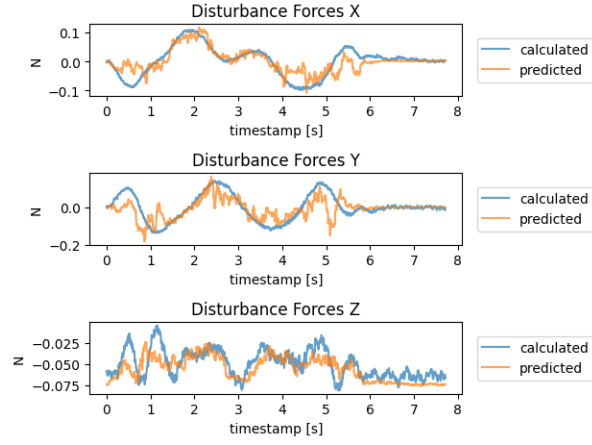
#### **4.4.d Changing the complexity of the decision tree**

Since I know the input features that predict the residual force and residual torque most accurately, I can experiment with the tree structure and detect which structure will predict most precisely. With the help of a function from the XGBoost library, I can plot the structure of the decision tree I. The decision tree has a depth of 6 and 31 leaves when I do not give a limit for the complexity. In this section, I compare the unlimited decision tree with a decision tree that has a depth of 3 and has 6 leave and a decision tree with a depth of 1 and 3 leaves.

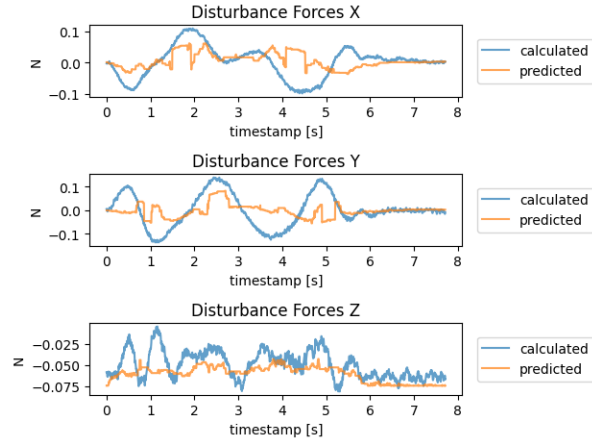




(a) Tree I with the depth of 6 and 31 leaves



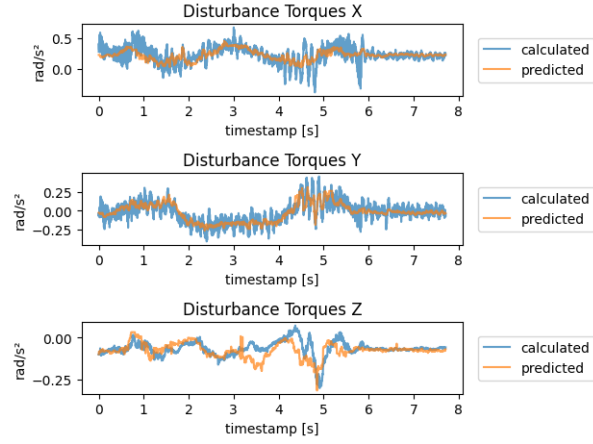
(b) Tree VI with the depth of 3 and 6 leaves,



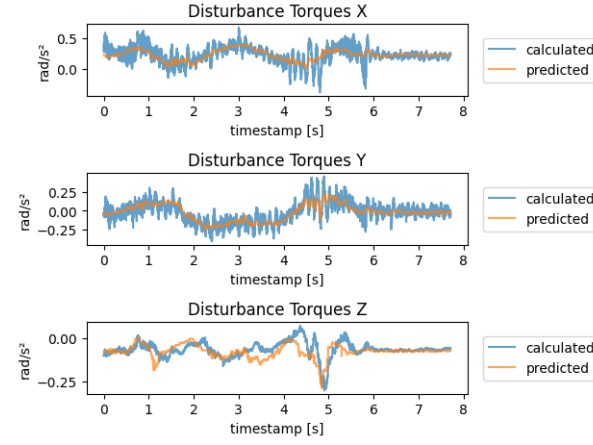
(c) Tree VII with the depth of 1 and 3 leaves,

Figure 26: Comparison of the predicted residual force of trees with different complexities

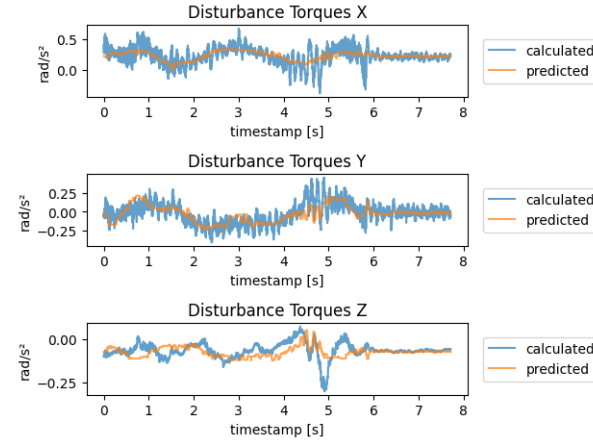
Regarding the residual force, there is not a huge difference between tree I and tree VI, by looking at the figures 26a and 26b. These two figures also show that the residual force error of tree VI is smaller for  $\mathbf{f}_a X$  than on tree I. Figure 26c clearly shows that a decision tree with a depth of 1 and 3 leaves is too simple to predict the residual force accurately. This shows that even when forcing the decision tree to reduce its complexity, it can still predict a relatively good output. However, I also have to be careful not to reduce the complexity too much as shown in the third model 26c.



(a) Tree I with the depth of 6 and 31 leaves



(b) Tree VI with the depth of 3 and 6 leaves,



(c) Tree VII with the depth of 1 and 3 leaves,

Figure 27: Comparison of the predicted residual torque of trees with different complexities

As shown in figure 27b, the residual torque prediction is better with the model that has a reduced complexity, unlike the residual force. Figure 27c shows, that while it has a positive effect to reduce the complexity of the decision tree when the change of the decision tree is too drastic, the model cannot predict the residual torque precisely.

#### 4.4.e Comparing the different decision trees

Table 4: Decision tree average error

	$\mathbf{f}_a X$ in N	$\mathbf{f}_a Y$ in N	$\mathbf{f}_a Z$ in N	$\boldsymbol{\tau}_a X$ in rad/s <sup>2</sup>	$\boldsymbol{\tau}_a Y$ in rad/s <sup>2</sup>	$\boldsymbol{\tau}_a Z$ in rad/s <sup>2</sup>
Tree I	0.018	0.035	0.012	0.069	0.057	0.032
Tree II	0.018	0.035	0.012	0.318	0.074	0.043
Tree III	0.025	0.041	0.008	0.069	0.065	0.030
Tree IV	0.042	0.058	0.016	0.115	0.096	0.189
Tree V	0.019	0.035	0.010	0.068	0.067	0.034
Tree VI	0.021	0.037	0.011	0.067	0.057	0.030
Tree VII	0.035	0.051	0.014	0.072	0.500	0.034

Table 5: Decision tree time in seconds

	training time	prediction time
Tree I	81.31	0.04
Tree II	53.77	0.06
Tree III	88.79	0.08
Tree IV	36.33	0.03
Tree V	29.89	0.04
Tree VI	41.90	0.03
Tree VII	20.25	0.02

In table 4 I calculated the average error for every decision tree that I presented in the previous section. Table 5 shows that I stopped the time to know the training duration and the duration of the prediction. When looking at table 4 it is noticeable that the decision tree I and VI have the smallest average error. The worst average residual error has tree IV and the worst average residual torque has decision tree II.

The fastest training time is found in the measurements of V and tree III achieved the longest training duration. While the two most accurate models have very similar mean average errors, decision tree VI needs almost half of the time for the training that tree I needs.

## 4.5 Comparison of supervised learning and decision tree

The best-performing MLP model was model IV and the best-performing decision tree was tree VI. Firstly, I am going to compare the input data and the structure of the model. After this comparison, I am going to compare the results of the model with one another.

The MLP model has three input features, whereas the decision tree has two input features. The angular velocity is the one input feature that both models have in common. As equation (21) shows, the residual torque is dependent of the angular velocity. The other two input features of the MLP are the acceleration and the two first columns of the rotation matrix. The second input feature of the decision tree is the velocity. One thing that both of the models have in common is that the model predicts better when we have more data points. Both - the MLP model and the decision tree - are trained on a scaled output where the scaled data is between -1 and 1.

Table 6: Comparison of the most accurate MLP model with the most accurate decision tree

	$\mathbf{f}_a X$ in N	$\mathbf{f}_a Y$ in N	$\mathbf{f}_a Z$ in N	$\boldsymbol{\tau}_a X$ in rad/s <sup>2</sup>	$\boldsymbol{\tau}_a Y$ in rad/s <sup>2</sup>	$\boldsymbol{\tau}_a Z$ in rad/s <sup>2</sup>
Decision tree	0.021	0.037	0.011	0.067	0.057	0.030
MLP	0.022	0.039	0.077	0.072	0.087	0.041

	training duration in s	prediction duration in s
Decision tree	41.90	0.03
MLP	66.93	0.28

As table 6 shows the average error of the decision tree is significantly smaller than the mean average of the MLP. Especially the average error of the  $\mathbf{f}_a Y$  is seven times more than that of the decision tree of the MLP. The duration of time needed by the decision tree to train and predict is much smaller than for the MLP.

I would recommend using the decision tree because the model is more time-efficient and the prediction is more accurate. The decision tree is also easier to program because I did not need to define the structure before finding decent input parameters.

## 5 Conclusion

In this thesis, I showed different methods of building a machine learning algorithm. The algorithm can predict the residual force and residual torque which were created by the Crazyfly. In comparison to the other thesis - that ignored the aerodynamic forces - the motion planning will be more accurate. The model to propagate the data - while ignoring the aerodynamic forces - delivered an accurate output for the angular velocity, quaternion and position. The calculated acceleration and velocity of the model yield an output which is different to the collected data. The calculated residual force and residual torque are quite small and because the residual torque is much smaller the propagated angular velocity is quite accurate. Whereas the residual force is larger and I am able to notice the influence of the residual force on the quadrotor. In the section result, the different models show that machine learning algorithms can predict the aerodynamic effect. The decision tree creates a more accurate output than the MLP.

## References

- [1] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, 2012.
- [2] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.
- [3] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. NeuroBEM: Hybrid Aerodynamic Quadrotor Model. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, jul 2021.
- [4] Crazyflie 2.1 — bitcraze. <https://www.bitcraze.io/images/Crazyflie2.0/Crazyflie2.0-585px.JPG>. Accessed: 2023-03-03.
- [5] Guanya Shi, Wolfgang Hoenig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-swarm2: Planning and Control of Heterogeneous Multirotor Swarms using Learned Interactions. 12 2020.
- [6] Julian Förster. System Identification of the Crazyflie 2.0 Nano Quadcopter. Bachelor thesis, ETH Zurich, Zurich, 2015-08.
- [7] Machine Learning — Cambridge Dictionary. <https://dictionary.cambridge.org/dictionary/english/machine-learning>. Accessed: 2023-03-16.
- [8] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons*, 4:51–62, 2017.
- [9] [https://radimrehurek.com/data\\_science\\_python/](https://radimrehurek.com/data_science_python/). Accessed: 2023-03-04.
- [10] Suhner MC Thomas, P. A New Multilayer Perceptron Pruning Algorithm for Classification and Regression Applications. *Neural Processing Letters*, 42:437–458, 2015.



- [11] Roza Dastres and Mohsen Soori. Artificial Neural Network Systems. *International Journal of Imaging and Robotics*, 21:13–25, 03 2021.
- [12] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastrokakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009.
- [13] Berke Akkaya and Nurdan Çolakoglu. Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases. 09 2019.
- [14] Xue Ying. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168:022022, 02 2019.
- [15] Oliver K Ernst. Stochastic gradient descent learning and the backpropagation algorithm. *University of California, San Diego, La Jolla, CA, Tech. Rep*, 2014.
- [16] Wei-Yin Loh. Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.
- [17] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.
- [18] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic Gradient Boosted Distributed Decision Trees. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, page 2061–2064. Association for Computing Machinery, 2009.
- [19] Barry de Ville. Decision trees. *WIREs Computational Statistics*, 5(6):448–455, 2013.
- [20] Henry Gouk, Bernhard Pfahringer, and Eibe Frank. Stochastic gradient trees, 2019.
- [21] crazyflie-firmware/cfusdlog.py. <https://github.com/bitcraze/crazyflie-firmware/blob/master/tools/usdlog/cfusdlog.py>. Accessed: 2023-02-23.
- [22] Chao Hu, Max Q.-h. Meng, Mrinal Mandal, and Peter Xiaoping Liu. Robot Rotation Decomposition Using Quaternions. In *2006 International Conference on Mechatronics and Automation*, pages 1158–1163, 2006.

- [23] Michael Andrieu and John Crassidis. Geometric Integration of Quaternions. *Journal of Guidance Control Dynamics*, 36:1762–1767, 11 2013.
- [24] 3.8: Finding Velocity and Displacement from Acceleration. <https://phys.libretexts.org/@go/page/3983>. Accessed: 2023-02-15.
- [25] 10.3: Rotation with Constant Angular Acceleration. <https://phys.libretexts.org/@go/page/4028>. Accessed: 2023-02-16.
- [26] Bo Li, Haichao Zhang, Yuxiao Niu, Dechao Ran, and Bing Xiao. Finite-time disturbance observer-based trajectory tracking control for quadrotor unmanned aerial vehicle with obstacle avoidance. *Mathematical Methods in the Applied Sciences*, 46(1):1096–1110, 2023.
- [27] Guanya Shi, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Neural-swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, 2020.
- [28] Pytorch MSELoss. <https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html?highlight=mse#torch.nn.MSELoss>. Accessed: 2023-03-09.
- [29] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014.
- [30] Introduction to Boosted Trees. <https://xgboost.readthedocs.io/en/stable/tutorials/model.html>. Accessed: 2023-03-09.