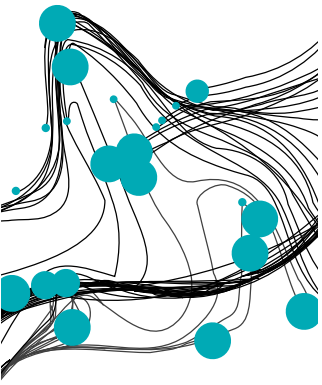
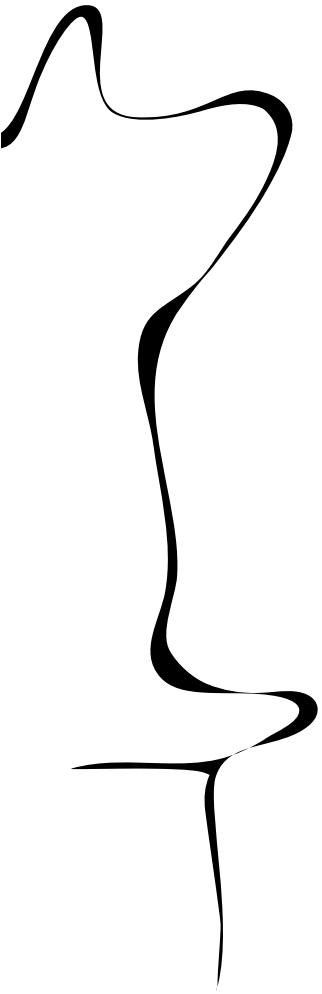


RAM ROBOTICS AND MECHATRONICS



CONTROL OF A WING FLAP USING 3D PRINTED FLOW SENSORS AND REINFORCEMENT LEARNING

T.C. (Tijmen) Hommels

MSC ASSIGNMENT

Committee:
prof. dr. ir. G.J.M. Krijnen
ir. A.P. Dijkshoorn
dr. N. Botteghi
dr. ir. F. Califano
dr. E. Mocanu

January, 2022

006RaM2022
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

The field of fluid mechanics has used reinforcement learning (RL) to increase the performance of control on objects moving in a fluid environment. One of the key requirements of RL is the ability to receive environmental observations. In this research a 3D printed flow sensor performs the environmental observations in an experimental setup.

3D printed sensors are a relatively new type of sensor and offer the ability to combine different types of polymers, allowing for a variety of uses. However, these 3D printed sensors have some drawbacks. A flow sensor measures the aerodynamic force drag using strain gauges. The observations from the sensor are used by the RL algorithm to actuate the flight control surfaces of an airfoil in a real world experimental setup. The wind tunnel is used to generate the incoming airflow and to measure the forces applied on the airfoil.

This research is performed to gain insight into the abilities of the TD3 RL algorithm to control and interact with the complex high dimensional fluid environment based on the observations of the imperfect 3D printed flow sensor. Using baseline measurements, simulations, sensitivity analyses and three experiments the research questions are answered.

First, the baseline values were obtained of the used airfoil and 3D printed flow sensor exposed to different wind speeds and different flap positions. This gave insight in the effects of the flap position on the forces the airfoil experiences. Furthermore, these baseline values showed that the 3D printed flow sensor can clearly measure the effects of the wind speed and the flap position on the airflow surrounding the airfoil. A limitation was found within the sensor because it showed hysteresis.

With the found baseline values and taking some of the shortcoming of the wind tunnel into account, a simulated RL training session was performed. The simulated training sessions formed a simplified environment for the RL algorithm. In the simulations the RL algorithm proved to have the ability to chose the appropriate action, to ensure the predetermined lift value was reached for the different simulated wind speeds.

A major drawback proved to be the wind tunnel, which required certain human based workarounds. The real world experiments were performed using this wind tunnel. The RL algorithm had more difficulty finding the optimal action to control the flight control surfaces in the real world experiments. This was especially prominent when trained on the lift goal, in comparison to the training on the normal force goal.

From the results, it was found that the 3D printed flow sensor worked sufficiently. The RL algorithm performed successful in the simulated environment, it lacked accuracy in the real world setup. This could be attributed to the more complex environment, including the drawbacks of the wind tunnel requiring manual and additional steps. Another factor was that in the real world experiments certain wind speed and action combinations provided almost the same observation from the sensor, making it more difficult for the RL algorithm to distinguish the current state.

The RL algorithm in this experimental set up could not accurately control the forces on the airfoil using the incoming 3D printed flow sensor observations. Nevertheless, the experiments did show the possibility of using an RL algorithm in combination with a 3D printed flow sensor. Therefore this combination's viability should be further explored in future research. Successful implementation could lead to versatile, energy efficient processes in fluid dynamics.

Acknowledgement

First and foremost, I would like to express my deepest thanks to Professor Gijs Krijnen for giving me the opportunity to work within the NIFTy group and providing me with his support through out my master thesis.
This project would not have been possible without my supervisors Alexander Dijkshoorn, Nicolò Botteghi, and Federico Califano. They have guided me and were always available for advice, feedback and questions.
I would also like to thank Walter Lette for his assistance in the lab with the wind tunnel. Furthermore, I offer my sincere appreciation for the learning opportunities provided by the NIFTy group from the University of Twente.
Finally I would like to thank my family and friends, for all the love and support I received.

Contents

1 Introduction

Machine learning is used in many different fields and has led to multiple innovative products such as Netflix's recommendation engine and self-driving cars. The ability to use machine learning in a large variety of settings is one of the reasons it is becoming increasingly wide spread. Machine learning is a type of artificial intelligence that allows software algorithms to predict or make decisions based on data without being explicitly programmed to do so (javatpoint, 2021). In the past few decades machine learning has made large strides solving more and more complex problems. In this research, for example, it is applied to a fluid dynamics control project.

1.1 Context

The field of fluid mechanics makes use of the advancements in machine learning. Here machine learning is used in the understanding, modeling, optimizing and controlling of fluid flows (Brunton et al., 2020). Examples include Baldi and Hornik (1989), who used neural networks (NNs) to identify the phase configurations in multi-phase flow, and Milano and Koumoutsakos (2002) who used NNs to reconstruct turbulent flow fields near walls.

Another popular form of machine learning is reinforcement learning (RL). In RL the system interacts with the environment to find an optimal solution. RL is used in control problems in the field of fluid mechanics.

Ma et al. (2018) allows an RL algorithm to control a fluid jet in a simulated 2D environment. Using this algorithm they were able to balance various ridged objects in the air, which are subject to gravity, by shooting fluids towards the object. Kim et al. (2004) used RL on an autonomous helicopter in flight. They showed that is was possible for the RL algorithm to learn how to hover the helicopter and perform difficult maneuvers used in autonomous helicopter competitions at the highest level.

Energy efficient movements are important traits in nature and man-made machines. This is also one of the key focus areas of RL based control for fluid dynamics. In nature, animals that exhibit energy efficient behavior can be found. Among these animals are flapping birds who use atmospheric thermals to generate extra lift without expending energy by flapping (Roi Harel, 2016).

This phenomena is something that Gautam Reddy (2016) tried to recreate using a glider in a simulated environment with an RL-based controller. In this paper it was shown that RL can learn to move the glider such that it ascends using the thermals, just like birds do. A key to finding the thermals was the use of certain data, such as vertical wind velocity and vertical wind acceleration. However, as it was a simulated environment no sensors were used to observe these quantities. Following up on his own research Gautam Reddy (2018) also showed that is was possible to do this with a robotic glider and that it was important to use accurate estimates of the environment observed by the sensors. This shows that it is is possible to replicate nature's method, in this specific case with the use of RL-based control.

Other papers also looked at nature to compare RL based control with traditional control methods in fluid dynamics. Guido Novati (2017) showed that RL can outperform traditional optimal flow control strategies. This was done using a deep RL algorithm on a swimming strategy for fish school formations. It was found that using this algorithm energy was saved

collectively. This swimming was performed in a simulated environment which also took into account the vortical flow fields and the complex two way interaction between the fish and the flow field.

Other studies have also looked into the use of RL for fish in a simulated environment. These studies include Gazzola et al. (2016) and Verma et al. (2018), both of these studies also show that the RL algorithm found energy saving behavior. From this it can be seen that in complex simulated environments RL can produce energy saving control solutions.

In the previous paragraph the studies only take a virtual simulated environment in account. However, Fan et al. (2020) shows that RL can also be trained and used in an experimental setup and not just in simulation environments. They show this by reducing the drag of a system in a fluid environment. This indicates that the challenges an experimental RL problem faces, such as limitations in sensing and actuation, noise in measured signals, and delay in data transmission can be overcome by an RL based controller.

A key to the success of RL is the ability to observe the environment and this can be done using 3D printed sensors. 3D printed sensors are a low-cost sensor that offers the possibility of direct integration of multi material sensor with structural components in a single build sequence without the need of an assembly step (Shemelya et al., 2015). These sensors however do have some disadvantages compared to more traditional sensors, some of these imperfections include hysteresis, drift and non-linearity (Dijkshoorn et al., 2018). To the best of our knowledge no literature could be found that uses an RL based controller in fluid dynamics in combination with a 3D printed sensor.

1.2 Problem statement

Control problems, for a fluid–solid interaction model, are very complex to be solved because of the high dimensionality of the fluid, nonlinearity of the Navier–Stokes equations, and consequent non convexity of the optimisation (Califano et al., 2021). Machine learning is well suited to deal with these high-dimensional, nonlinear problems (Brunton et al., 2020). This project looks into the control of the flight control surfaces of an airfoil with machine learning, specifically RL.

The PortWings project (PortWings, 2021) and the NIFTy group, located at the University of Twente, have shown interest in combining RL with 3D printed sensors.

The PortWings project focuses on creating a flapping robotic bird. The interest lies in the possibility of outperforming traditional model based control with the help of RL to control the robotic bird. This could allow for more energy efficient flight or more complicated flight maneuvers.

The NIFTy group is a research group that focuses on the design, fabrication and research of 3D printed sensors, actuators, meta-materials and functional structures in general. The interest of the NIFTy group lies in the prospect that an RL based controller can overcome the disadvantages of the 3D printed sensors.

This project assesses the feasibility of using a 3D printed sensor with machine learning based control in fluid dynamics. As this project is a part of the PortWings project a 3D printed flow sensor is used in combination with an airfoil with flight control surfaces. By correctly actuating the control surfaces of an airfoil, a target lift and drag can be met. Knowing the lift and drag of a flying system can lead to more efficient flights and can therefore allow a robotic bird to

fly longer distances. It also might allow the robotic bird to perform more difficult tasks such as take-off and landing. A flow sensor is chosen, as the incoming air speed is known to be an important factor in the lift and drag of wings (NASA, 2021).

This project is an extension of Dijkshoorn et al. (2021) as their 3D printed flow sensor is used. This 3D printed flow sensor was designed to be used on the wings or body of a robot moving through a flow field. This flow sensor showed potential in recognizing different flow speeds. However, it also showcased the shortcomings common in 3D printed sensors such as hysteresis and creep. An RL algorithm may overcome some of the shortcomings of the 3D printed sensor and deal with high-dimensional, nonlinear complexities seen in fluid dynamics. This would enable the Portwings project and in a larger sense the field of fluid mechanics, to make use of the many advantages of 3D printed sensors.

1.3 The research question

How can a 3D printed flow sensor in combination with RL be used to actively control the flight control surfaces of an airfoil in a real-world setup such that the airfoil reaches specific predetermined lift values?

This research question is broken down into two sub questions. These need to be answered before the research question can be solved.

1. **To what degree can the 3D printed flow sensor detect a change in flow over the airfoil due to a change in wind speed or flight control surfaces?**
2. **To what degree can the RL algorithm distinguish different readings from the 3D printed flow sensor?**

Hypothesis

It is hypothesized that the 3D printed flow sensor in combination with RL will be successful in actively controlling the flight control surfaces of the airfoil in a real world setup. This hypothesis is based on the successes of the previously performed research when combining RL in fluid dynamic systems.

However, there are still some challenges to overcome. Fan et al. (2020) found that RL in an experimental setup faces problems such as limitations in sensing and actuation, noise in measured signals, and delay in data transmission. Furthermore, while the 3D printed sensors have advantages, both Dijkshoorn et al. (2018) and Schouten et al. (2021) show that 3D printed sensors are non-ideal sensors as they showcase characteristic polymer properties such as hysteresis, drift and mechanical deformation.

Research method

To achieve the goals of this project a test setup will be designed. The setup includes an airfoil that can hold the sensor while also actuating its control surfaces. Furthermore, the machine learning algorithm and environment needs to be chosen and designed to work with the test setup. The system will be tested in a wind tunnel to ensure a repeatable laminar air flow. For simplicity only one set of flight control surfaces will be used, specifically flaps.

Once the test setup is made, the characteristics of the airfoil and that of the sensor can be found. With these characteristics predetermined lift values are used as the control goals in this project. With enough training in the wind tunnel the system will try to accomplish the chosen

control goals.

Contribution

If the feasibility of the use of 3D printed sensors in machine learning based control in fluid dynamics is shown, it could create new possibilities to implement 3D printed sensors for systems in fluid environments. The advantages of 3D printed sensors combined with the flexibility of machine learning can create a world of possibilities and is applicable to many different fields of research and engineering.

1.4 Organization of the report

Chapter 2 will give some relevant background information on the 3D printed flow sensor, the aerodynamics of an airfoil and on the basics of machine learning. Chapter 3 covers the specific RL environment for this project. Chapter 4 will outline the design of the system and how all parts work together. The experiments that will be conducted and the goal these have will be explained in Chapter 5. Chapter 6 shows the results of the proposed experiments and a sensitivity analysis is presented. Chapter 7 discusses the results and recommendations for future work are given. The conclusions is given in Chapter 8.

2 Background knowledge

This chapter gives background knowledge on topics that are relevant to the research of this thesis. First the research in traditional flow sensors are shown followed by information on 3D printed sensors. After that background information is given for the 3D printed flow sensor that is used, an explanation of the airfoil, airfoil terminology and the effects of flaps. Finally, a more in depth background is given for machine learning with a focus on RL.

2.1 Research context

2.1.1 Traditional flow sensors

A lot of research has been done on the use of flow sensors in small unmanned aerial vehicles (SUAV), an overview of this been made by Mark et al. (2019). Here, multiple methods of flow sensing have been described including: commercially available pressure sensors, capacitive sensors, hot film sensors, piezoresistive sensors, and artificial hair sensors. While no 3D printed sensors were included in the overview, the piezoresistive cantilever sensors (Seo et al., 2014) slightly resembles the 3D printed flow sensor design by Dijkshoorn et al. (2021), as both sensors rely on mechanical deformation to detect changes in local flow. The multiple types of traditional sensor mentioned in the overview still have shortcomings in areas such as durability and reliability (Mark et al., 2019). Specifically the piezoresistive cantilever sensors are susceptible to creep and fatigue. These sensors are also very sensitive to temperature (Mark et al., 2019).

Mark et al. (2019) also shows that some research has been done combining the use of these imperfect sensors with NNs to predict the lift coefficients of airfoils. Magar et al. (2016) uses an array of hair sensors placed on an airfoil with flaps and a feedforward NN to predict aerodynamic parameters of the airfoil. They found the lift and moment coefficients with a prediction error of 6 % and 10 % respectively. This shows that NNs can be combined with imperfect sensors in an aerodynamic environment to find parameters of an airfoil. Knowledge of these parameters are key to finding the real time forces and moments which paves the way for effective control design to increase flight agility, stability, and maneuverability (Magar et al., 2016).

2.1.2 3D printed sensors

In the last few years the field of 3D printing has turned its attention towards sensors.
The first article in this field is presented by Leigh et al. (2012). A good overview of the different work done in this field since the work form Leigh et al. (2012) is given by Xu et al. (2017). NIFTy has also played a part in pushing the field of 3D printed sensors forward as seen in Schouten et al. (2021).

While there are many different types of 3D printing techniques, this project uses a 3D printed sensor constructed via the Fused Deposition Modeling (FDM) technique. The interest in FDM 3D printing sensors is due to the fact that they allow for the possibility of direct integration of multi material sensors with structural components in a single build sequence without the need of an assembly step (Shemelya et al., 2015). This is made possible due to multi-material FDM and by having a wide range of sensing materials available, these include: conductive, piezoresistive, piezoelectric, and magnetic materials (Schouten et al., 2021). Another advantage is that the 3D printed sensors can be made out of many types of polymers and therefore have the possibility to be flexible, this allows sensors to be used in soft robotics as indicated by Dijkshoorn et al. (2018) and Schouten et al. (2021). However, both Dijkshoorn et al. (2018) and Schouten et al. (2021) also show that 3D printed sensors are non ideal sensors as they

showcase characteristic polymer properties such as hysteresis, drift and mechanical deformation.

Fan et al. (2020) have shown that RL can overcome challenges that present themselves when training an RL controller experimentally for fluid systems. It might then also be able to overcome some of the shortcomings of 3D printed sensors mentioned in the paragraph above. In the section below the specific 3D printed flow sensor used in combination with RL is explained in more detail.

2.2 3D printed flow sensor

As mentioned in the introduction of this paper a 3D printed flow sensor is used based on the design of Dijkshoorn et al. (2021). This design was based on the design of Schouten et al. (2019). The sensor is a drag based sensor. This 3D printed flow sensor is designed in such a way that it bends when a force is applied to it. The force applied to the sensor is the aerodynamic force known as drag. The bending of the sensor will also bend the two strain gauges located on either side of the sensor. The bending causes the strain gauges to either be compressed or elongated, this changes the resistance of the strain gauges (Cui, 2021). The strain gauges and their connection pads are made from the conductive Thermoplastic Poly-Urethane (TPU) in specific PI-eTPU (Palmiga Innovation , 2021). The other part of the sensor is made from the non-conductive TPU material NinjaFlex (NinjaTek, 2021). In Figure 2.1 a black and a grey section is visible. The black section represents the conductive material, constructed of PI-eTPU. The grey section is the non-conductive section of the sensor constructed from NinjaFlex.

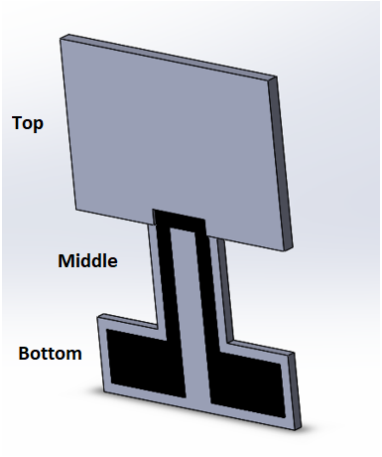


Figure 2.1: Sensor model (Cui, 2021)

The sensor can be further divided into three sections: top, middle and bottom. These sections can be seen in Figure 2.1. The top section of the sensor is designed to ensure that sufficient drag-force acts on the sensor. The middle section is the flexible section of the sensor that bends when a force is applied to the top section. The bottom part of the sensor serves as both the mechanical and electrical connection point. It is used to connect the sensor to a Wheatstone bridge (HBM, 2021), a circuit used for strain gauge measurements. It is also used for clamping the sensor to hold it in place.

The increased surface area in the top section ensures that a higher drag force is exerted on the sensor, as can be seen from Equation 2.1 (Anderson, 2016). In this equation it is shown that the drag force, F_D, increases as the cross sectional area, A increases. As more drag is applied to the sensor, due to the increased surface area of the top part, the senor will also bend further. More bending leads to more compression and more elongation of the strain gauges located in the

middle, flexible, section.

$$F_D = \frac{1}{2}\rho v^2 C_D A \tag{2.1}$$

The maximum deflection of the sensor, for a given force, is found using Equation 2.2 (Cui, 2021). Here F is the applied force on top part of the sensor. l is the length of the bending middle part. E is the Young's modulus and I represents the second moment of inertia of the bending part of the sensor. Equation 2.3 shows how the moment of inertia can be found, here w is the width of the middle part of the sensor. The center of rotation is located at $\frac{1}{2}l$. The Young's modulus is determined by the material of the sensor. Both materials used, NinjaFlex and PI-ETPU have a Young's modulus of 12 MPa (NinjaTek, 2021) (Palmiga Innovation , 2021). The sensor is placed on the airfoil to measure drag and observe environmental changes which are the input for the RL algorithm.

$$\delta_{max} = \frac{Fl^3}{3EI} \tag{2.2}$$

$$I = \frac{wl^3}{12} \tag{2.3}$$

2.3 Airfoil

An airfoil is a structure with curved surfaces that is used to generate lift while moving through a fluid. The 3D printed flow sensor is placed on an airfoil. In this section the basics of an airfoil and its definitions are explained.

An airfoil has multiple physical properties that affect its lift and drag. Figure 2.2 shows some of these properties. The leading edge and the trailing edge of an airfoil are the sections that respectively separate and recombine the incoming airflow. The chord line is a straight line drawn between these two points. This line does not have to be within the actual airfoil. The camber line is the line that can be drawn between the two outer edges of the airfoil. This line indicates how symmetric an airfoil is. The camber of the airfoil is the maximum distance between the camber line and chord line. If the camber line is above the chord line, then the airfoil has a positive camber. The angle of attack is the angle between the chord line and the incoming flow.

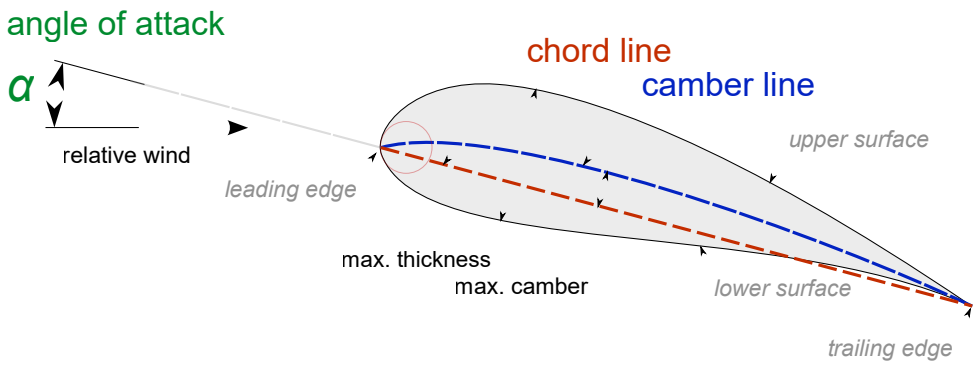


Figure 2.2: Definitions of an airfoil (Wikipedia contributors, 2021a)

An Airfoil generates lift by deflecting the air downwards. The deflection of the flow can be seen in Figure 2.3. Using Newton's third law it shows that the airflow exerts an upward force on the airfoil as the flow is directed downwards, this force is called lift (David Halliday, 1988). Airfoils with a positive camber generate lift at a 0° angle of attack.

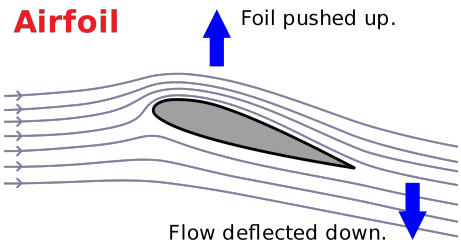


Figure 2.3: Flow around an airfoil (Wikipedia contributors, 2021b)

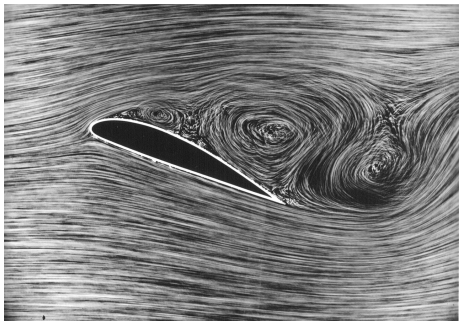


Figure 2.4: Flow separation around an airfoil (Wikipedia contributors, 2021b)

To calculate the lift of an airfoil Equation 2.4 can be used (NASA, 2021). Here L is the lift force, C_L is the coefficient of lift, this depends on the properties of the airfoil. ρ is the fluid density, V is the flow speed and S is the relevant surface area of the wing. If the wing is rectangular, S can be calculated by multiplying the span of the wing with the chord length.

$$L = C_L \frac{1}{2} \rho V^2 S \tag{2.4}$$

There are multiple ways to change the lift of an airfoil. Changes are made by, for instance, increasing the speed or altitude at which it travels through air. Another way is to change the coefficient of lift. This can be done through, for instance, changing the angle of attack or using high lift devices such as a flap.

Figure 2.5 shows that an increase in the angle of attack leads to an increase of the lift coefficient. This increase stops, once the flow going over the top of the airfoil, separates from the airfoil. An example of flow separation is shown in Figure 2.4. The vortices behind the airfoil indicates that flow separation has taken place. The graph in Figure 2.5 also shows that the flap orientation affects the lift. Downward pointing flaps increase the lift whereas upward pointing flaps decrease the lift, in comparison to the neutral flap position. A change in flap position affects the physical properties of the airfoil which in turn changes the aerodynamic properties.

As the flap moves, the trailing edge of the airfoil moves, this affects the chord line and the camber line, leading to a change in the angle of attack (Colin Cutler, 2021). This alters the coefficient of lift. The change in physical properties leads to a change in the camber of the airfoil, changing how the incoming airflow is deflected downwards, therefore also affecting the lift.

This research requires the RL algorithm to learn how to control the flaps of the airfoil based on the input of the 3D printed flow sensor to reach a specific predetermined lift value.

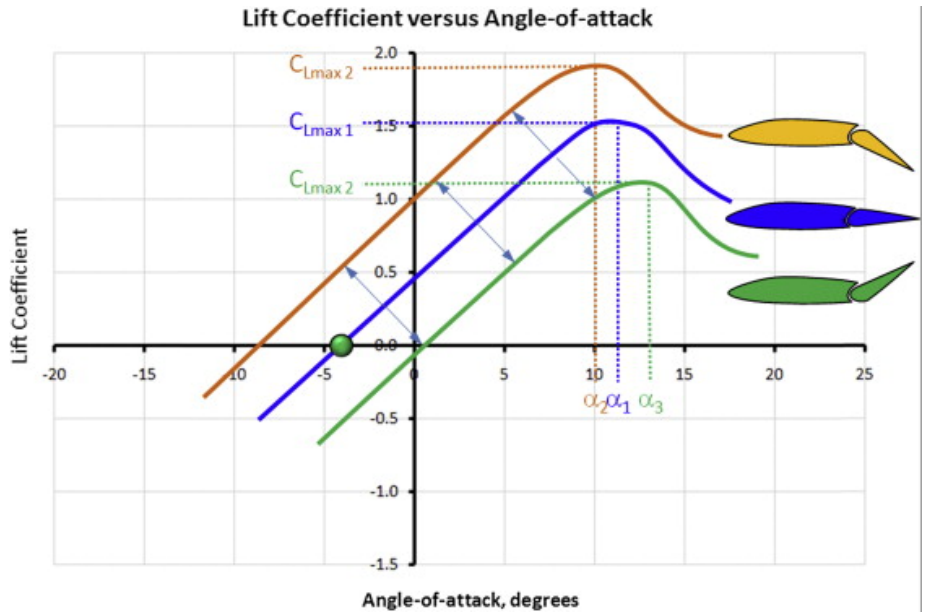


Figure 2.5: Coefficient of lift vs angle of attack and how it changes with different flap positions (Gudmundsson, 2014).

2.4 Reinforcement Learning

This section gives some background information about RL; starting with machine learning, the basics of reinforcement learning and training of the RL algorithm. Additionally the difference between on- and off policy is explained. Furthermore deep RL and in particular the twin delayed deep deterministic algorithm is discussed.

2.4.1 Machine learning

Machine learning started out in 1958 with the use of a single layer perceptron (Rosenblatt, 1958). This could only solve linear separation functions and already stranded at the XOR function (Minsky and Papert, 1969). The use of a multi-layer perceptron, a form of NN, was made possible by the development of the backwards propagation algorithm (Rumelhart et al., 1986). With an NN it was possible to solve the XOR function.

ML algorithms are divided in multiple ways. The clearest distinction is based on the data handling of the ML algorithm. This leads to four categories: supervised learning, semi supervised learning, unsupervised learning and reinforcement learning (Bushkovskyi, 2021). This thesis will focus on RL.

2.4.2 Basics of reinforcement learning

RL is a form of machine learning where an agent interacts with the environment to determine the optimal behavior to maximize its performance. The two main components are the environment, which represents the problem to be solved, and the agent, which represents the learning algorithm. This form of machine learning has been successfully used in games to perform at a strong intermediate level as shown by Tesauro (1992).

Figure 2.6 shows the basic scheme of RL. This model for RL holds under the assumption that the environment behaves like a Markov Decision Process (MDP) (Sutton and Barto, 2018). An MDP is a mathematical framework that allows the modeling of decision making in an environment were the outcome of the environment depends partially on the made decision and

partially stochastically on the current environment. At each time step the next environmental state will depend only on the current state and on the action performed by the agent.

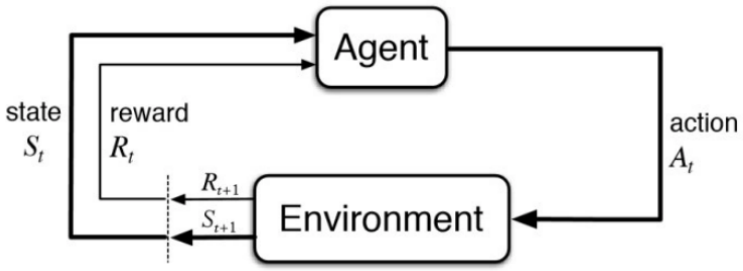


Figure 2.6: Reinforcement learning scheme (Bhatt, 2021)

On the basis of Figure 2.6 some important terms for RL are introduced. The agent is the part of the system that determines how to interact with the environment in order to maximize its performance.

The environment is the real world environment in which the agent operates, this includes all real world factors that are not the agent. Examples of these factors are for instance the agent's path that it needs to navigate, temperature and lighting. The environment can also be a simulated environment.

The state S_t represents the current state of the environment. The observability of the state can differ for different environments, states and agents. If a state is fully observable by an agent, it is called completely observable. If the agent cannot fully observe the state, it is called partial observable. Partial observability can occur due to lack of sensors and/or complex environments. Partial observable systems become an Partially Observable Markov Decision Process (POMDP) in which the agent is unsure of its current state based on the observations.

The action A_t indicates the actions the agent takes to interact with the environment. The action chosen will be part of an agent's action space which is formed by all possible actions an agent can perform. Action spaces can be continuous or discrete.

The reward R_t is the feedback from the environment based on the actions the agent takes. These rewards can be positive and negative and are used to evaluate the actions taken by the agent. An example of a reward could be the distance of a robot to a finish line. The performance of the agent can be seen as the sum of all rewards. The rewards drive the agent to the optimal solution.

A policy π is used to determine an action based on the given state. The policy can be either deterministic or stochastic. An agent determines its action based on its policy.

Another important term in regards to RL is value function. There are two main value functions: state value functions and action value functions. The latter is also known as Q-function. The state value function shows how good the current state of the agent is. The state value function is equal to the total expected reward if the agent starts in this state and follows its policy.

The Q-function evaluates the current state of the agent if it would take a certain action. The Q-value is given by the total expected reward if the agent starts in this state, performs the evaluated action and then follows its policy. The Q-value can thus have multiple values for a

given state as it also evaluates actions. The Q-function is used in the updating of the policy so that the performance of the agent improves.

2.4.3 Training of the RL algorithm

In an RL system the agent first observes the state in the current time-step. Based on this observation the agent will decide what action it wants to take by looking at its policy. This action is performed and influences the environment. This change in environment will lead to a new state that the agent can observe. This also gives a reward to the agent. The reward and the state-action pair is then stored by the agent. With the new observed state this process starts over again.

The goal of the agent is to optimize its policy in order to maximize the total reward it will receive for any starting state. This can be done by finding the optimal value functions.

The agent needs to choose between exploration and exploitation while trying to find the optimal value functions. Whilst exploiting, the agent chooses the action that returns the highest expected reward, given the current state observed. This is also called greedy policy. During exploration the agent might choose a sub optimal action to explore new features of the environment.

The policy should be updated if the agent can improve its current expected reward, by deviating from the current policy. This new policy can then be used to re-evaluate the value function. How and when the agent updates its value function depends on the chosen RL model. Another distinction between RL algorithms is how agents update their policy. This can either be done via an on- or off-policy method.

2.4.4 On-policy versus off-policy

RL algorithms can be divided into on-policy and off-policy, depending on how they update their policy (Sutton and Barto, 2018). On-policy algorithms use their current policy to evaluate and improve the policy they are currently using. An example of an on-policy algorithm is State–action–reward–state–action (SARSA).

Off-policy algorithms evaluate and improve a policy that is different from their current policy. An example of an off-policy algorithm is Q-Learning, it uses the greedy policy for evaluation.

2.4.5 Deep Reinforcement learning

Deep RL uses deep NNs to represent the policy and the value function. A deep NN is a NN with multiple hidden layers between the input and output layers. The use of a deep NN can help with more complex problems. The first use of deep RL was shown in Mnih et al. (2013), where a deep NN was combined with Q-learning. This specific algorithm is called DQN. In Mnih et al. (2013), the DQN was used to successfully complete the complex task of playing Atari video games. Also, it has been shown that deep RL can outperform traditional optimal control strategies in the field of fluid mechanics, as seen by Guido Novati (2017).

2.4.6 TD3

A newer deep RL algorithm, when compared to DQN, is called Twin Delayed Deep Deterministic Policy Gradients better know as TD3 (Fujimoto et al., 2018). TD3 is a model free, off-policy algorithm used for environments with continuous action spaces. TD3 uses the Actor-Critic framework, a method were two agents are used. The Critic estimates the value function and the Actor is used to estimate the policy (Sutton and Barto, 2018). TD3 also uses target networks to introduce stability into the system when updating the parameters of the NN of the Actor and

Critic. The combination of an off-policy algorithm with the Actor-Critic framework and target networks is also used in the RL algorithm Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015). DDPG is an improvement of the DQN used in Mnih et al. (2013) as mentioned in the previous section.

TD3 improves further on the DDPG algorithm by adding a second critic network, delaying the updates of the actor and adding noise regulation. A second critic network is added to increase the stability of the algorithm by creating a more stable Q-value approximation. The second critic network also means that another target critic network is added, due to this the TD3 algorithm uses six deep neural networks. Three of these networks represent the actor and its two critics, and the other three represent the target networks of the first three networks. Delaying the updates of the actor network is done by updating the network every other time-step ensuring a more stable algorithm as it reduced the per-update error (Fujimoto et al., 2018). Noise regularization is added to smooth the target policy in order to make it more difficult for the policy to exploit errors from the critics (OpenAI, 2021). The three changes added to TD3 substantially improved the performance of the algorithm when compared to DDPG (OpenAI, 2021).

2.5 Conclusion

In this chapter background information on different topics was introduced. The flow sensor used in this project was shown. Furthermore, the basics of an airfoil and how an airfoil generates lift in a flow field was discussed. Using this information it was also shown how flaps can affect the aerodynamic properties of an airfoil. Finally general information was given regarding RL algorithms and more specific information was given on the RL algorithm used in this project namely TD3. In this chapter it has been established that in order to reach a predetermined lift value, the RL algorithm needs to learn to control the flaps, as these change the lift of the airfoil. The 3D printed flow sensor measures the aerodynamic drag force and can observe the environment, giving feedback to the RL to learn the controls.

3 Methodology

This chapter shows the RL scheme for the project. Some of the main components of the system are introduced and it is demonstrated how the RL algorithm interacts with these components. An overview of the favorable criteria for the chosen RL algorithm, TD3, is given. Furthermore, this chapter explains the steps of the specific TD3 algorithm and how these differ from the original algorithm.

3.1 The RL scheme

Chapter 2, Figure 2.6 shows the standard RL scheme. In this chapter Figure 3.1 shows the more extensive RL scheme. This scheme contains the terms that were introduced in Section 2.4.2 and the additional major components of the systems, namely the RL algorithm, the airfoil, flaps, and the 3D printed sensor.

The agent is the RL algorithm that is explained in more depth in Section 3.2. The agent uses the observations from the 3D printed sensor as input, as seen in the RL scheme. The agent subsequently decides, based on this input, the position of the flaps.

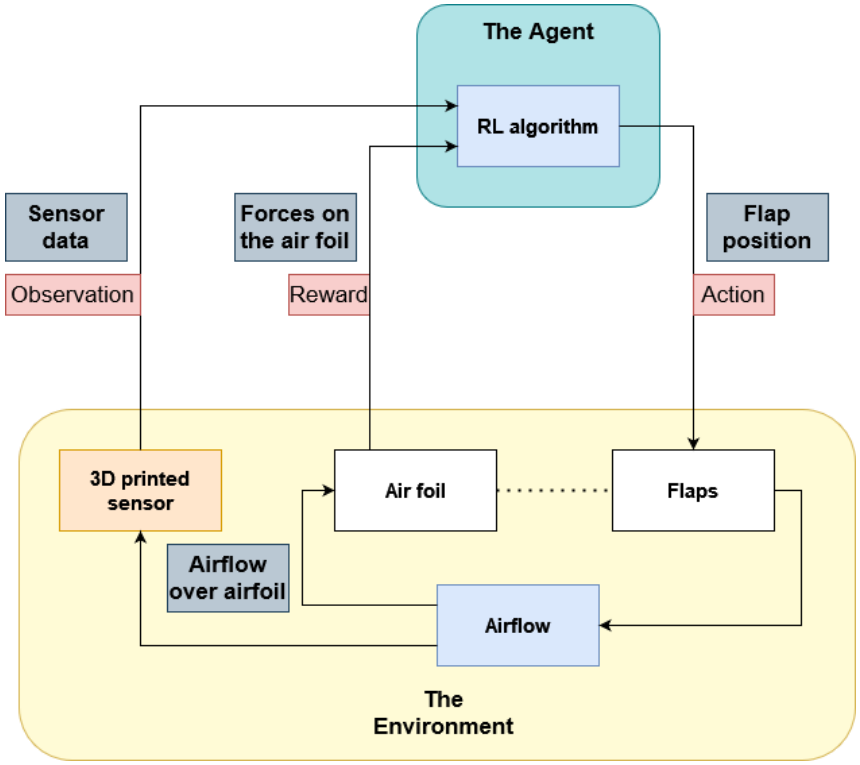


Figure 3.1: The RL scheme

The environment is an airfoil placed in an airflow that is generated by a wind tunnel. The airfoil has flaps that can be actuated and a 3D printed flow sensor which is used to observe the state of the airflow. The wind tunnel also measures the forces applied on the airfoil by the airflow.

The interactions between the agent and the environment are the actions, observed states

and rewards. The action of the agent is actuating the flap to a set position. A change in the flap position of the airfoil changes how the airflow, generated by the wind tunnel, moves around the airfoil. These changes in flap position and airflow, change the state of the environment. As the airflow affects the drag based flow sensor the change in state may be partially unobserved by the 3D printed flow sensor.

The change in the state of the environment also includes the change in forces applied by the airflow onto the airfoil. The forces applied on the airfoil are used as part of the reward for the agent. The goal of the agent is to actuate the flaps in such a way that for different environments, the force applied on the airfoil stays at a constant level.

3.2 TD3 specifications

TD3 is the chosen RL algorithm for this project (Fujimoto et al., 2018). It is a model free, deep RL algorithm that allows for a continuous action space while also being off-policy. This algorithm is more stable than other deep RL algorithms with continuous action spaces. The continuous action space is required, because the servo motor, used to actuate the flaps, has 90 possible allowed positions, corresponding with the range of the flaps. These 90 possible actions are approached as a continuous action space over the same range. The off-policy algorithm is needed as this allows training from a replay buffer. Training from a replay buffer is essential, because for this specific project the reward data is added after an episode's interaction with the environment. Episodes are usually considered all agent-environment interactions from the initial state to the final state. An episode now also includes all updating steps performed after all agent-environment interactions before the start of a new initial state and agent-environment interaction. Additionally, the off-policy algorithm offers the capability of training new iterations of the algorithm on previously obtained data. In a similar control scenario the TD3 algorithm was successfully used in training an RL controller (Fan et al., 2020).

Equation 3.1 presents the reward function. F_{desired} indicates the desired force and F_{measured} indicates the measured force on the airfoil. This is a quadratic reward function with a maximum reward at the point where F_{measured} is equal to F_{desired}. In case that the measured value differs from the desired value a negative reward is given. A larger negative reward is given if the measured value is further from the desired value.

$$r = -(F_{\text{measured}} - F_{\text{desired}})^2 \tag{3.1}$$

A key difference with the standard TD3 algorithm is that the reward data is added after the algorithm's interaction with the environment in a given episode. In addition, the updating phase occurs after the agent is done interacting with the environment. A standard TD3 algorithm adds the reward during the process and performs an update after each action. This choice is made because of the limitations of the wind tunnel, which are discussed in detail in Chapter 4.

Algorithm 1 shows the TD3 algorithm for this project. Table 3.1 lists the parameters found in Algorithm 1. The specific values for the parameters, seen in Algorithm 1, are presented in Chapter 4 Section 4.7. Three for-loops can be observed. The N_{e} loop, the N_{i} loop and the N_{j} loop. The N_{e} loop indicates an episode of the TD3 algorithm. The N_{i} loop is the interaction stage of the the episode. In this phase the algorithm uses the incoming state to decide how to interact with the environment. The N_{j} loop is the updating phase of the episode. The synchronization of the rewards is seen in between the interaction stage and the updating stage.

In Algorithm 1 the key components of TD3, previously mentioned in Section 2.4.2, are also apparent. The key components are: the extra critic network, the delayed actor updating and

the noise regulation. The extra critic network is initialized in the first step. The delayed actor updating is implemented via the if-function. The noise regulation is implemented in the updating phase, seen by the clipped noise when computing the target actions.

Algorithm 1: TD3 algorithm

Initialize critic networks Q_{θ_1}, Q_{θ_2} and the actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize the target networks with $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize the replay buffer β
$t \leftarrow 0$
for $n_e = 1$ **to** N_e **do**
  for $i = 1$ **to** N_i **do**
    Collect the observed state s_i
    Request action with exploration noise
    $a_i \leftarrow \text{clip}(\pi_\phi(\text{s}_i) + \epsilon, -1, 1), \epsilon \sim N(0, \sigma)$
    apply a_i to the environment and observe the next state s_i'
    store transition tuple, (s_i, a, r, s_i'), in the replay buffer β
  end
  Synchronize the wind tunnel data to the rewards r for episode n_e in β
  for $j = 1$ **to** N_j **do**
    sample N transition (s, a, r, s') from β
    compute target actions $a'(s') = \text{clip}(\pi_{\phi'}(\text{s}_i) + \epsilon, -1, 1), \epsilon \sim \text{clip}(N(0, \tilde{\sigma}^2) - c, c)$
    $y \leftarrow \text{r} + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
    update critics using gradient decent $\theta_i \leftarrow \text{N}^{-1} \sum (y - \text{Q}_{\theta_i}(s,a))^2$, i = 1,2
    if t *mod* d **then**
      update actor using gradient decent $\phi \leftarrow$-$\text{N}^{-1} \sum \text{Q}_{\theta_1}(s, \pi_\phi(s))$
      update the target networks
      $\theta_i' \leftarrow \tau\theta_i + (1\text{-}\tau)\theta_i'$
      $\phi' \leftarrow \tau\phi + (1\text{-}\tau)\phi'$
    end
    t $\leftarrow t + 1$
  end
end

Name	Symbol
Discount factor	γ
Standard deviation of the noise	σ
Soft updating rate	τ
Clipping value	c
Standard deviation of the noise regulation	$\tilde{\sigma}$
Batch size	N
Actor update	d
Loop sizes	N_e, N_i, N_j

Table 3.1: TD3 algorithm parameters for this project

3.2.1 TD3 structure

Figure 3.2 presents an example of the structure of TD3. This figure was used to illustrate the specific structure of the TD3 algorithm in Kim et al. (2020). In this thesis project the structure slightly differs, as no hindsight experience replay (HER) is used. This structure gives visual

insight on how the various NNs in the algorithm interact with each other and also shows the key components of Algorithm 1.

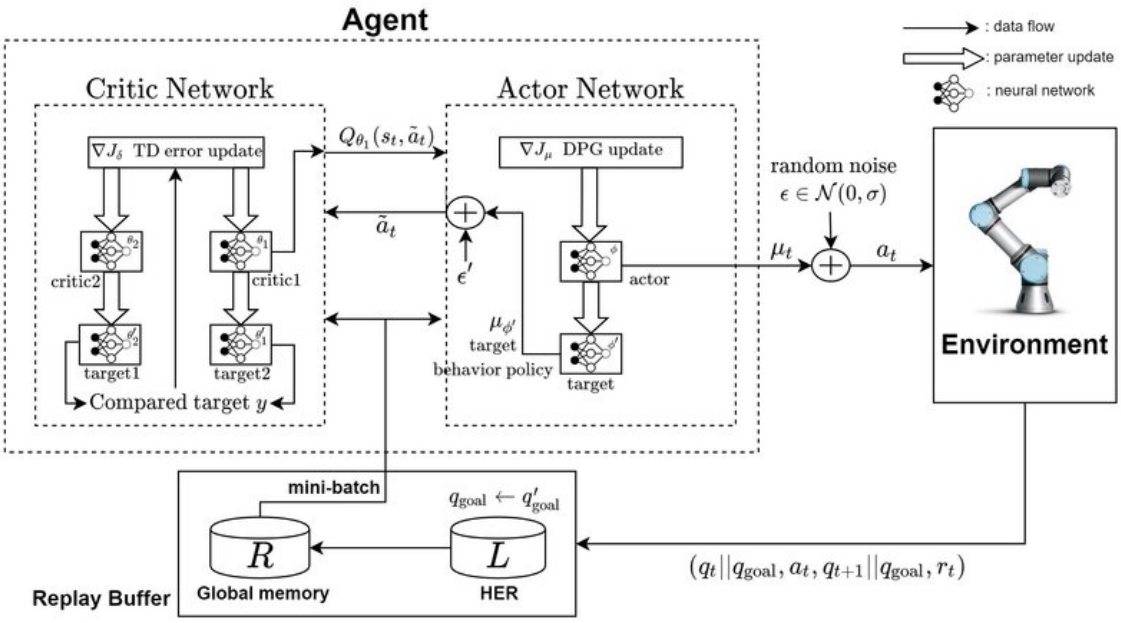


Figure 3.2: Structure of TD3 (Kim et al., 2020)

3.3 Conclusion

This chapter has shown the RL scheme for this project and the chosen RL algorithm. This RL scheme and the components mentioned in it will be used in the upcoming chapter to design the experimental system. It has been established that the RL algorithm used in this project is the TD3 algorithm with separate updating and environmental interaction phases. The reward function for the experiments is shown in Equation 3.1. The chosen RL algorithm and the introduced reward function are used in the training experiments.

4 Experimental Design

In this chapter the experimental system design of this project is discussed. The 3D printed flow sensor, the airfoil and the wind tunnel are covered. These are the hardware components used in the project. Section 4.5 explains the connective components. Section 4.6 gives an overview of how the measurements of the experiments are taken. Followed by the implementation of the RL algorithm.

4.1 The experimental system

The experimental system is based on the RL scheme seen in Figure 3.1. A more detailed overview of the components of the experimental system is given in the scheme shown in Figure 4.1. The arrows display the connection between the components. The power connections are indicated by the red arrows. The black arrows indicate the data streams. The combination of all these arrows together represent the connections between the agent and environment shown in Figure 3.1. The agent in Figure 3.1 is represented by the laptop component from Figure 4.1, that runs the RL algorithm.

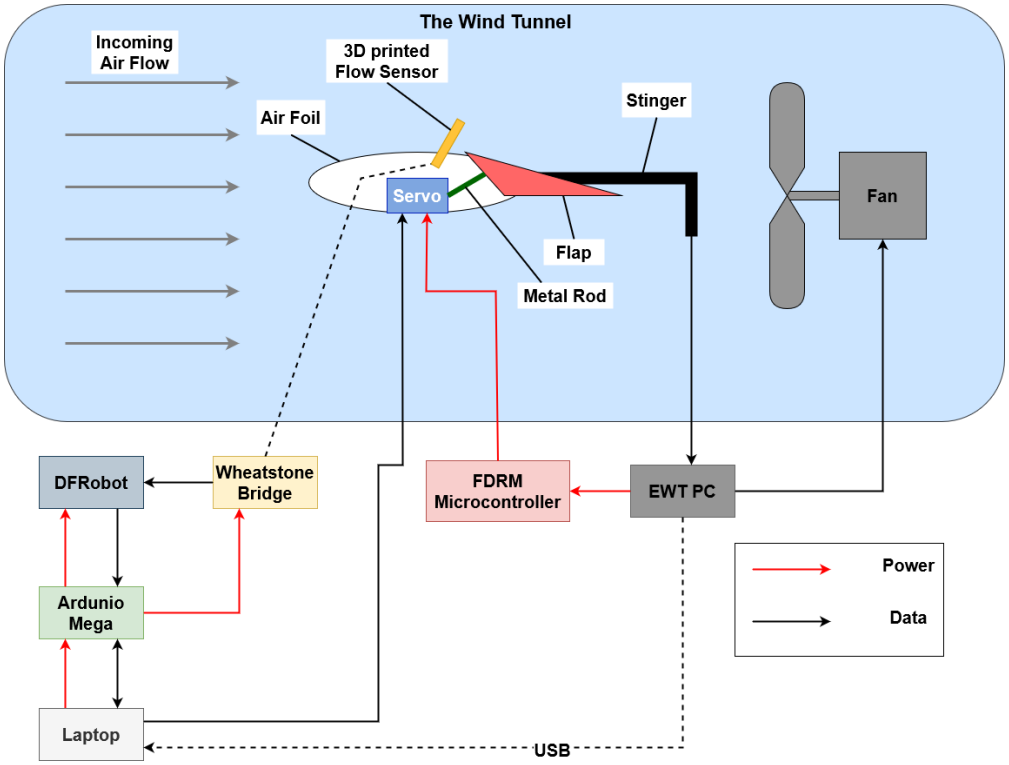


Figure 4.1: Scheme of the experimental setup

4.2 The 3D printed flow sensor

This work is a continuation of Dijkshoorn et al. (2021). Their sensors design is taken as the starting point. Their sensor is shown in Chapter 2 Figure 2.1. The specifications of the sensor in this report are, the top part: l×w×d=16×24×1.2 mm, the middle part: l×w×d=10×8×0.9 mm, the bottom part: l×w×d=7×22×0.9 mm.

During testing it was observed that the bending range of the 3D printed flow sensor was lower than desired with the applied wind speeds. This can be partially attributed to the fact that the sensor is placed at a 45° angle in the airfoil. Therefore, the angled sensor experiences

a reduced drag resulting in less bending of the strain gauges resulting in a smaller range of voltage measurements. This makes it more difficult to distinguish between the different wind speeds of the experiment.

The maximum deflection increases by decreasing the area moment of inertia of the middle part of the sensor, as seen in in Figure 4.2. To increase the bending that the sensor undergoes, for a given wind speed, a section in the middle part of the sensor is removed. The removed section creates a gap of 9.6 mm long by 2.2 mm wide. This change is based on Equation 2.2 and 2.3.

Data from Ortiz et al. (2015) shows that by placing the sensor at a 45° angle, instead of a 90° angle, will reduced the drag on the sensor by approximately 42 %. By removing a section of the middle part of the sensor the area moment of inertia is reduced by approximately 26 %. This means that the maximum deflection of the sensor at a given wind speed is reduced by approximately 22 % for the sensor with a gap placed at a 45° angle when compared to a sensor without a gap placed at a 90° angle.

Due to the 3D printing method, fully removing the non-conductive part in the middle section, without leaving an edge of non-conductive material, leads to a short circuit in the strain gauges. This short circuit occurs when two strain gauges touch. Adding an edge of non-conductive material prevents the strain gauges from touching. An example of a flow sensor with short circuited strain gauges is shown in Appendix A for illustrative purposes.

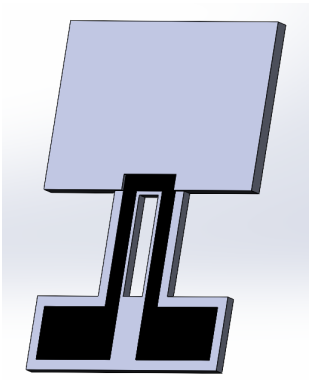


Figure 4.2: Sensor design with the removed section.

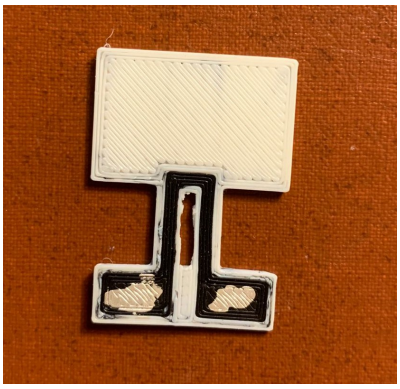


Figure 4.3: Sensor with the removed section.

Manufacturing

The 3D printed flow sensor is printed using the Diabase H-series Multi- Material 3D printer (Diabase, 2021).The 3D flow sensor used in this project, seen in Figure 4.3, was constructed by cutting out a gap by hand from a sensor without a gap. This means that the specific sensor was not fully FDM made.

4.3 Airfoil design

The airfoil is made up from three parts: the main body, the flaps, and the servo-sensor holder. Each of the parts are 3D printed separately. In Figure 4.4 the grey part is the main body, the red part is one of the flaps and the blue part is the servo-sensor holder. In the final design the airfoil will have two flaps and two servo-sensor holders.

4.3.1 Main Airfoil Body

The chosen airfoil shape for this project is the NACA 2412 (AirfoilTools, 2021). This airfoil design is chosen for its simple and cambered shape. This type of airfoil generates lift at a 0°

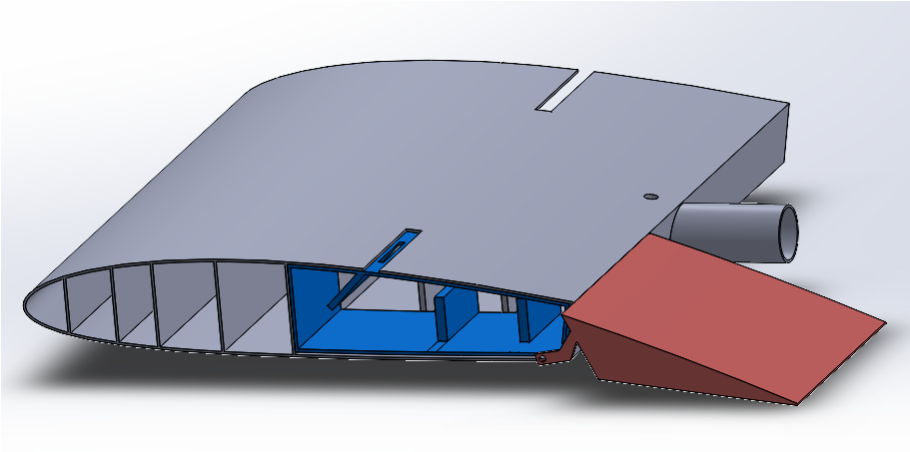


Figure 4.4: Airfoil design

angle of attack. This is an airfoil shape that is used in real airplanes such as the Cessna 172 series (Cessna Aircraft Company, 2021).

The airfoil is constructed using a 3D printer. The 3D printer used is a Prusa i3 mk3s (Prusa 3D, 2021). The printing area of the 3D printer limits the size of the airfoil, therefore the airfoil is designed with a length of 17 cm (not including the protective sheath of the holder) and a width of 20 cm. To save printing time, materials, and weight the airfoil has a hollow design. Because of the hollow design support beams are printed to ensure enough strength to prevent the airfoil from bending. The 3D printed flow sensor is placed in the airfoil and held in place by the servo-sensor holder. The cutouts on the top of the airfoil allow the flow sensor, held by the servo-sensor holder, to be slid in place.

4.3.2 Holder for the stinger

Although the force sensor of the wind tunnel is not part of the airfoil, it does affect the design of the airfoil, therefore it is discussed in this section. The force sensor of the wind tunnel is also known as the stinger and can be seen in Figure 4.5. In addition to measuring the forces on the airfoil the stinger is also the place where the airfoil is mounted in the wind tunnel. In Section 4.4 the experimental setup is explained in more detail.

The airfoil is placed on the stinger using a designated tubular hole at the the end of the airfoil, the so called holder area of the airfoil. The holder area is designed according to the specifications given by the wind tunnel manual, which is supplied by Aerolabs (Aerolab, 2021). The holder area of the airfoil for the stinger was not printed hollow in order to add strength and to ensure the screw can be firmly placed in the screw hole. The stinger area was hollowed out more to make the stinger fit. This was done by hand, using a drill press.

4.3.3 Flaps

This project uses flaps to actively change the aerodynamic properties of the airfoil. How this works is explained in Chapter 2, Section 2.3. To allow the flaps to move they are printed as separate parts from the main body. The flaps are also 3D printed using the PRUSA printer (Prusa 3D, 2021). Similar to the airfoil the flap is made form PLA. The flaps are printed with an internal matrix support structure that fills 15 % of the structure.

The flaps are connected to the main body of the airfoil with hinges, to enable the movements

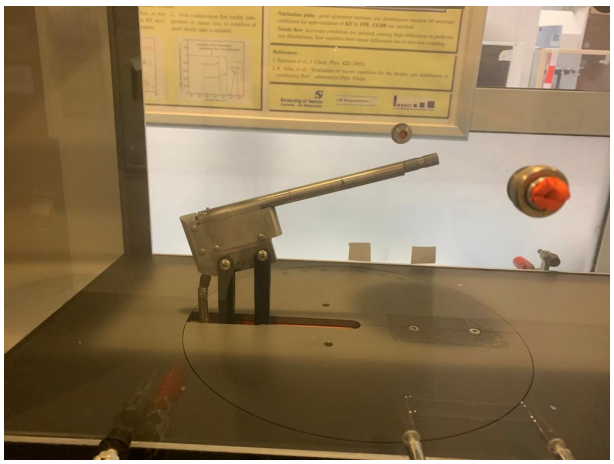


Figure 4.5: Stinger of the wind tunnel

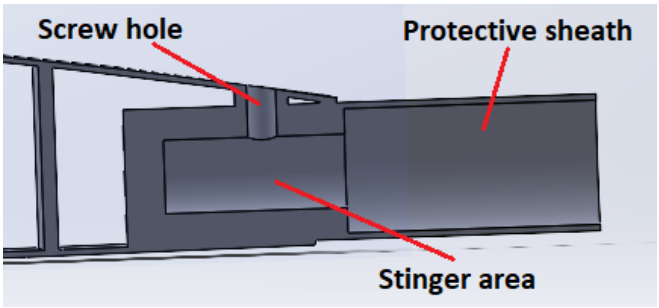


Figure 4.6: Cross section of the holder for the stinger

of the flaps. Tape is a popular hinge design, often used in 3D printed hobby aircrafts (Eclipson, 2021). The advantages of tape hinges over more traditional hinges, such as pin hinges used in most doors, is the ease of use and low complexity. The disadvantage of the use of a tape hinge design is the vulnerability to weather effects. These can have influence on the tape, resulting in reduced adhesiveness of the tape, resulting in detached hinges. However, this is not a concern for this project as experiments are performed in a controlled environment with no weather effects. For this reason it was decided to connect the flaps to the main body of the airfoil using tape. This can be seen in Figure 4.7.

Servo motors

Servo motors are used to actuate the flaps. These are connected to the flaps using a metal rod. The rods need to be connected to the flaps in such a way that the flaps have an adequate range of motion. To translate the rotational movements of the servo motors to that of the flaps, arms are also used in combination with the metal rods. The arm is part of the flap and translates the movement of the rod into a rotation of the flap around the hinge point. The hole in the arm is used as a connecting point for the metal rod. How all components connect can be seen in Figure 4.7. Figure 4.8 shows the full range of the flaps.

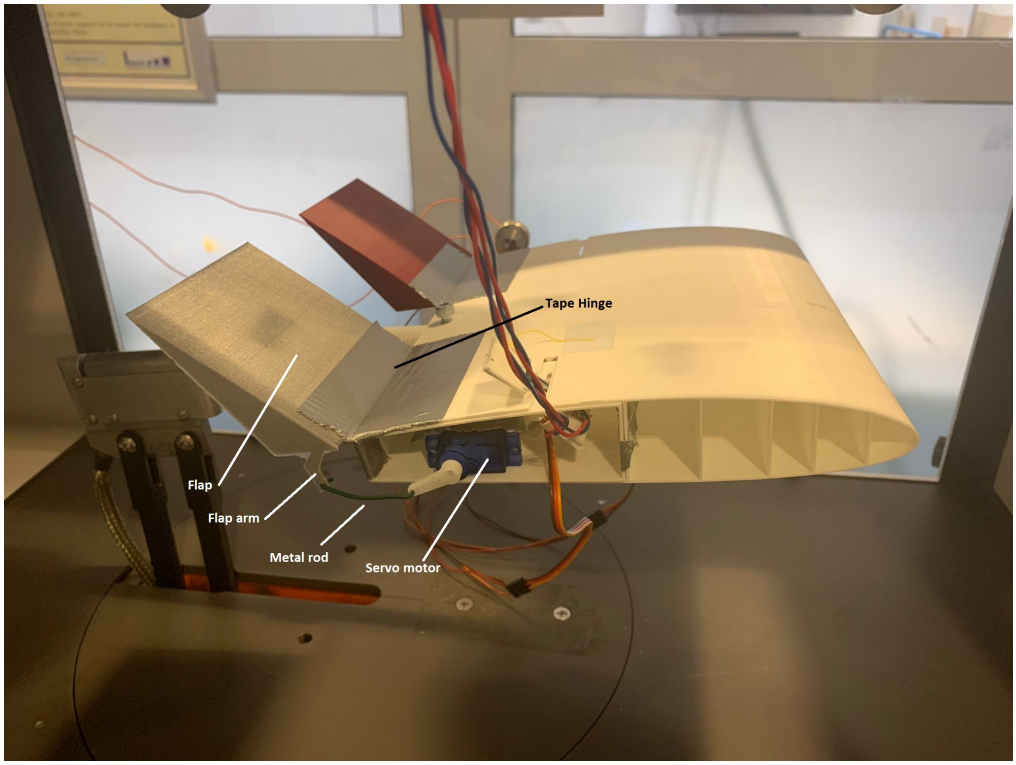


Figure 4.7: Components of the complete airfoil used in the actuation of the flap

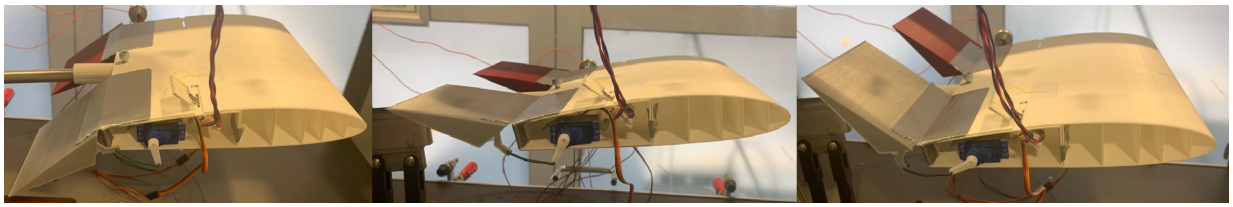


Figure 4.8: Left: lowest flap position, Middle: middle flap position, Right: highest flap position

4.3.4 Servo-Sensor holder

The sensor and servo motors are held in place in the servo-sensor holder. This is a section that can be inserted into the main body of the airfoil and is shown in Figure 4.9.

The sensor is placed towards the back of the airfoil. In this location the protruding sensor will have less affect on the airflow over the airfoil than a forward placement. Furthermore, the effect of the flap position will be measurable. The noise of the non-insulated servo motor on the sensor signal was negligible.

The sensor is held at a 45° angle to reduce flow separation over the airfoil, which may be induced by the sensor. The sensor would act as a flat plate if placed under an angle of 90°. A flat plate separates incoming flows whereas an inclined plate shows properties of flow re-attachment (Shademan and Naghib-Lahouti, 2020). In Chapter 2 it was shown that flow separation on the airfoil greatly reduces its lift. The limited bending of the sensor due to the reduction of drag on the sensor, due to the inclination, is compensated by an increased flexibility due to the gap in the middle section.

To hold the sensor in place it is clamped in between two walls. These walls each contain two strips of copper tape. This method is also used in Cui (2021), Prakken (2019) and Kosmas

(2020). The strips of tape form a connection with the conductive sections in the bottom part of the sensor. Wires are also soldered to the copper tapes on the other side of the clamping walls to allow the sensor to be placed in a Wheatstone bridge circuit.

The clamping mechanism of the sensor is part of the servo-sensor holder. The servo-sensor holder can be seen in blue in Figure 4.4. It was made as a separate removable section to provide access to the clamping mechanism. This allows for easy placement of the sensor and soldering of the wires. The closed shape provides stability to the part.

The servos are also placed in the servo-sensor holder. They are placed in between the two walls at the bottom. This prevents movement of the servos. Extra movement can be prevented by placing tape over the walls and servo, as seen in Figure 4.7. The servos are flipped between the left and right side of the airfoil such that the rotation point is pointing outwards of the airfoil for both sides. This allows the flaps on both side of the airfoil to be actuated. The walls are shifted between the left and right side to allow the axis of rotation of both servos to be on the same axis.

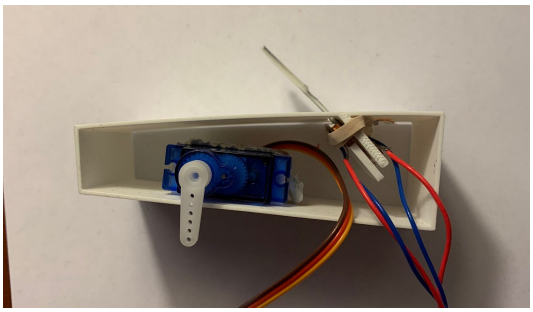


Figure 4.9: the servo-sensor holder

4.4 The wind tunnel

In this section a short explanation will be given as to why the wind tunnel was used. Then the shortcomings of the wind tunnel will be shown and how these are overcome.

Design decision for wind tunnel

One of the first decision made in this project was the choice between using either the available educational wind tunnel (EWT), or a self-made setup. In order to make this decision the advantages and disadvantages of each setup were compared.

The advantage of the EWT is that it offers the possibility of a repeatable controlled flow, it comes with a three dimensional force sensor that is already calibrated, and is designed for testing of airfoils. The software of the EWT is however lacking. Data cannot be recorded while the wind speed changes because the cursor needs to click to record one data point. The data can also not be read out in real time.

The advantage of the self-made setup is that it can solve the software issues of the EWT. The disadvantage of the self-made setup is that it will take a considerable amount of time to research, design, build and program a system that can offer repeatable controllable flows. In order to focus on the research question of this project the EWT is chosen as part of the setup in this project.

Figure 4.10: The EWT

Specifications of the EWT

The EWT is a wind tunnel designed by Aerolabs (Aerolab, 2021). The wind tunnel is shown in Figure 4.10. The force sting balance, known henceforth as the stinger, sends its data to the NI SCXI box (National Instruments, 2021) that controls the wind tunnel. This communicates with the executable program on the EWT PC. These parts are further discussed below.

The stinger is the three dimensional force sensor of the wind tunnel that measures normal force, axial force and pitching moment (Aerolab, 2021). This sensor is calibrated by Aerolabs and its load limits are designed for optimal measurements in the wind tunnel. The stinger also forms the attachment point for the airfoil, using the holder area in the main body of the airfoil. The angle of the stinger can be manually changed by turning a knob.

The wind tunnel PC runs on the Windows XP operating system. The executable that communicates with the NI SCXI box (National Instruments, 2021) in the wind tunnel was designed to run on this Windows XP operating system. The older operating system leads to compatibility issues with potential hardware that only would run on newer operating systems. A glimpse of the user-interface of the executable can be seen in Figure 4.11.

The executable offers a range of options. Zero preset allows the force readings to be set to zero. Furthermore, two speed control options are present, one is by setting the fan speed and the other is setting the wind speed. By setting the wind speed the wind tunnel will try to remain at that wind speed by varying its fan speed. Data is recorded when the ‘take data’ button is

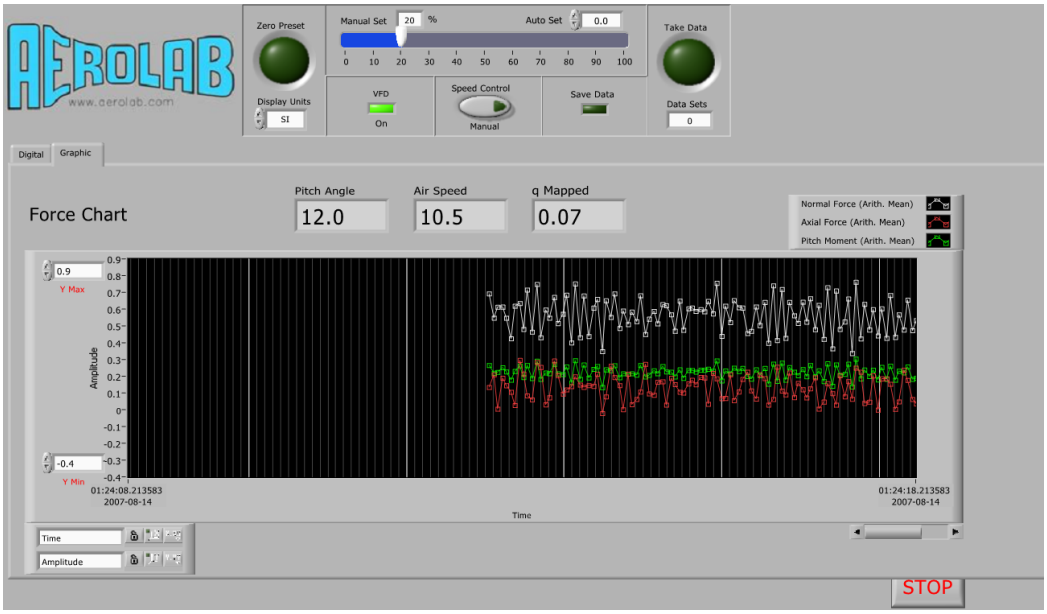


Figure 4.11: The executable of the EWT

clicked. This data is saved in the memory of the executable. To extract the data the 'save data' button needs to be clicked.

Data is sent to the executable from the wind tunnel via a SCXI system from national instruments (National Instruments, 2021). This is made possible by a specific PCI Express card (Farnell, 2021) placed in the wind tunnel PC.

The important limitations of the wind tunnel

The main limitation of the wind tunnel for this project is the data acquisition. The data of the wind tunnel needs to be recorded and moved to the laptop running the RL algorithm. Using only the provided executable the data acquisition will not suffice for the project. Furthermore, due to a battery issue of the wind tunnel PC the time on the PC needs to be set manually.

Working with the limitations

Multiple methods have been tried to minimize the limitations of the EWT. These methods included among others, looking into the possibility of direct data readouts from the sensor and upgrading the executable on the EWT PC. However all of the tried methods did not succeed. The final solution reached includes the use of an auto clicker and USB flash drive.

USB flash drive

A method for computers to transfer files between them is called file sharing. This was not recommend by Microsoft for the windows XP operating system and new operating systems such as the one on which the RL algorithm runs. Therefore the only method that remained to share the data between the PC and laptop is using a USB flash drive.

Auto clicker

As mentioned previously the executable program records a data point when the 'take data' button is clicked. However, an RL algorithm will take multiple steps per episode. In the project the RL algorithm takes 1000 steps at a fixed frequency of around 10 Hz. Ideally the data gathered from the wind tunnel would also be taken each time the RL algorithm performs a step. Manually clicking a button at a fixed frequency at least 1000 times is not humanly possible. To

solve this issue an auto clicker was installed on the wind tunnel PC. An auto clicker can automatically click at the location of the cursor for any set frequency. The auto clicker used for this project was GS Auto Clicker (goldensoft, 2021). This auto clicker is chosen as it is compatible with windows XP. As the executable program has a built in delay the auto clicker clicks every 0.07 seconds, this translates to a frequency of around 14.29 Hz.

Summary wind tunnel

The wind tunnel setup is as follows: the EWT with the already calibrated stinger is the designated wind tunnel for this project. In order to work with the limitations multiple methods have been tried to read out the data differently. However, the final solution was a USB flash drive in combination with an auto clicker.

4.5 Connective components

In this section the connective components of the system are discussed. These components include the Arduino, DFRobot, Servo Motors, Wheatstone bridge and the micro controller. All of these components can be seen in Figure 4.1.

Arduino

An Arduino Mega 2560 (Arduino, 2021) is used to communicate with the laptop and the rest of the system. The Arduino is plugged into the laptop, with the RL algorithm, via a USB cable, this supplies power and signal connection. The Arduino also acts as a power supply and ground for the circuit part of the system. The Arduino reads out the flow sensor measurement via the DFRobot and Wheatstone bridge. These will both be discussed in the following sections. The data received from the DFRobot is then sent to the laptop with the RL algorithm. Furthermore, the Arduino uses serial communication to receive signals from the laptop to actuate the servo motors. The signal is translated such that both servo motors move in synchronization, this is needed as the servo motors are flipped. The translated signal is sent to the servo motors. The Arduino code is compiled using the Arduino IDE (Arduino, 2021).

DFRobot

DFRobot ADS 1115 is a high precision ADC, it is 16 bit versus 8 bit for the Arduino (Dijkshoorn et al., 2021) (DFRobot, 2021a). Using this ADC allows the signal from the Wheatstone bridge to be measured with a higher precision than measuring with just the Arduino. The communication between the Arduino and DFRobot is made possible by using the external software library provided by DFRobot (2021b).

Wheatstone bridge

The Wheatstone bridge is the circuit in which the strain gauges of the flow sensor are measured. This is done using a half bridge configuration as seen in Figure 4.12. This allows for precise measurements of the strain gauges (HBM, 2021). Furthermore this configuration is the recommended for strain measurement on a bending beam (HBM, 2021). As the sensor is a bending sensor this configuration was chosen. The chosen half bridge configuration also allows for checking of the power supplied to the circuit. The fixed resistors of the circuit have a value of 5.53 kΩ and 5.98 kΩ. In Figure 4.12, V1 and V2 are the locations where the DFRobot measures the voltage via single-ended voltage measurements. The voltage and ground used in this circuit are supplied by the Arduino.

Servo Motor

The servo motors, used to actuate the flaps, are two SG90 Tower pro servos (TowerPro, 2021). This is a 9 g servo motor with a rotational range of 180°. This project uses 90° of this allowed

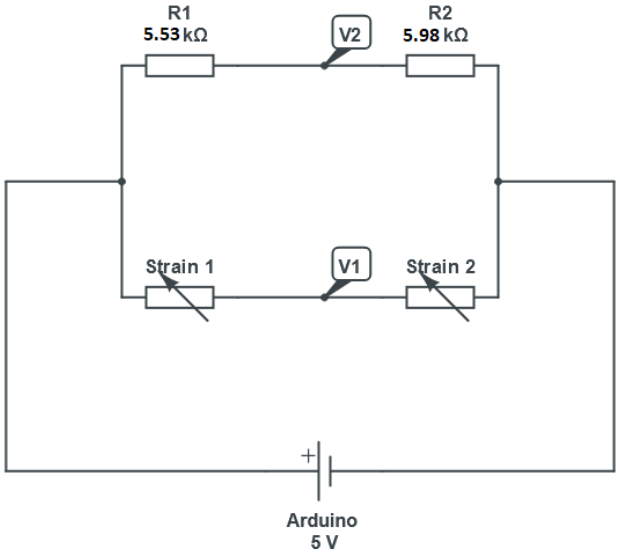


Figure 4.12: Wheatstone bridge

range, the lowest used servo motor position is at 50° and the highest is at 140°. The servo motors are placed in the airfoil's servo-sensor holders as seen in figure 4.7.

The servo motors are each connected to the Arduino via a Pulse Width Modulation (PMW) pin (ArduinoGetStarted, 2021). This output is used by the servo motor to determine the amount of rotation. Even though the Arduino can supply the necessary power and ground needed to run the servos, it will be supplied by an external power supply. This is done as a noise reduction measure, being further described in the next section.

The microcontroller

The FRDM microcontroller (NXP, 2021) is used to supply the voltage and ground needed to power the servo motors. This board is powered via USB cable that is plugged in to the wind tunnel PC. This is done as a noise reduction measure and therefore increases the precision of measurement for this project. Would the microcontroller not been used, the voltage supplied to the Wheatstone bridge by the Arduino would fluctuate each time the servo motor is actuated. Even using this microcontroller board this problem still exists, as the power supply by the USB cable from the laptop still fluctuates. This occurs due to the high voltage the servo motors use when being actuated, which in turn requires the voltage supplied to the Arduino to be compromised. This then also leads to a compromised voltage supply to the Wheatstone bridge, as the laptop USB voltage decreases when one of the USB ports is taking too much power. However, when powering the microcontroller by the wind tunnel PC, voltage fluctuations in that circuit will have no affect on those other components connected to the RL laptop.

4.6 Measurement and actions process

In this section an overview of how the measurements of the experiments are taken is given in more detail. First the sensor measurements will be discussed, followed by the software used and finally the wind tunnel measurements are briefly highlighted.

Sensor measurements

The flow sensor that is placed in the servo-sensor holder, which is located in the main body of the airfoil, is connected to the copper tapes in the servo-sensor holder via clamping. Wires are soldered to these pieces of copper tape which allows the flow sensor to be connected to

the Wheatstone bridge. The Wheatstone circuit is constructed on a breadboard using fixed resistors and the wires coming from the copper tape. The power and ground of the circuit are supplied by the Arduino. When the flow sensor bends due to airflow, the strain gauges of the sensor also bend accordingly. This changes the resistive values of the strain gauges, which changes the voltage at point V1 in the Wheatstone bridge. This change represents a voltage change, relative to the reference voltage V2, due to bending of the flow sensor, which is measured with the DFRobot. These measurements are sent to the laptop with the RL algorithm via the Arduino.

Measurement code

To increase the measurement frequency of the DFRobot, the provided library of the DFRobot (DFRobot, 2021b) on the Arduino was changed. A change was made in the delay value of the `readVoltage` function. The original delay value of 100 ms was changed to 10 ms. This leads to a maximum measurement frequency of 22 Hz for the measurements sent out by the Arduino. The measured values of V1 and V2, by the DFRobot, are sent to the laptop with the RL algorithm, via the Arduino, every time both measurements are taken.

Software Environment

A programmed environment and pySerial (Chris Liechti, 2021) facilitate communication between the Arduino and the RL algorithm. The signals can be requested and are used as the observation inputs for the RL algorithm. In turn the RL algorithm can send its actions via the programmed environment.

Wind tunnel measurements and actions

The EWT, with the already calibrated stinger, is used to measure the normal and axial forces on the airfoil. These forces can be used to calculate the lift and drag on the airfoil. The forces measured by the stinger are used in the reward function of the RL algorithm. The wind tunnel speed is manually set per experiment. The wind tunnel and auto clicker are turned on manually. The measurement data is transferred from the EWT PC to the laptop via a USB flash drive.

4.7 The RL algorithm

The RL algorithm, TD3, has certain parameters that need to be set. These parameters are listed in Algorithm 1. In this project all six neural networks have two hidden layers. Hidden layer one contains 400 neurons and hidden layer two contains 300 neurons. The Adam optimizer is used to update the critic NNs and the actor NN. The Adam optimizer is used with learning rates α for the actor NN and δ for the critic NNs. The other parameters of the TD3 algorithm can be seen in Table 4.1.

The experiments all have a warm up period. The length of the warm up period will either be either 2000 or 3000 steps, depending on the specific experiment. The system chooses random actions, whilst steps are performed in the warm up period of a training run.

4.7.1 Normalization and moving average

To make it possible for the TD3 algorithm to interact with the environment it needs to be able to communicate with the servo motors, the sensor and the wind tunnel. This is done using a section of code that translates and transmits incoming and outgoing data. The used TD3 algorithm only performs effectively with observation and action values in the range of -1 and +1. The 90 possible servo motor positions are scaled to fit in this range. The incoming observation data is also scaled. This is done by a normalization test performed before every training run. This normalization step was done by bending the sensor all the way down to the airfoil in the direction of the air flow from the wind tunnel, by hand. The range and average of the measured

Name	Symbol	Value
Discount factor	γ	0.99
Standard deviation of the noise	σ	0.04
Learning rates	α, δ	0.001
Soft updating rate	τ	0.005
Clipping value	c	0.5
Standard deviation of the noise regulation	$\widetilde{\sigma}$	0.2
Batch size	N	100
Actor update	d	2
Loop sizes	N_i, N_j	1000

Table 4.1: Values for the TD3 algorithm parameters for this project

values are used to scale the observation data.

Due to the the wind tunnel limitations the rewards for the TD3 algorithm, obtained from the environment, are added after the algorithm is done interacting with the environment. The rewards are synchronized with the existing data via timestamp comparison. As the EWT records data by the second, and not in milliseconds, synchronization of the rewards with the existing data is done in seconds. The rewards given to the algorithm are the same for every time step in a given second, this is the average reward of that given second. The reward is inserted into the system via a USB flash drive.

To increase the resolution and decrease the noise, a moving average (MVA) of the sensor observations is used as the observation input for the RL algorithm.

This MVA uses 10 sensor measurements to create the observation input for the RL algorithm. A MVA of 10 can reliably increase the resolution of the signal with one bit (Atmel, Seen 2021). The noise is reduced by a factor of $\sqrt{10}$ (Smith, 1985). In case the system receives a measurement error from the environment the previous observation will be used.

To allow the servo motors to reach the desired position, every other time step will repeat the action, chosen by the RL algorithm, from the previous time step.

RL algorithm implementation

The RL algorithm was programmed in Visual Studio (Microsoft, 2021) with programming language Python 3 (Van Rossum and Drake, 2009) using the tensorFlow package (Abadi et al., 2015) and openAI Gym (Brockman et al., 2016).

4.8 Conclusion

In this chapter the experimental setup is established and how each of the components play their part and interact with one another. The setup presented is used in the experiments, which are further discussed in the upcoming chapter. This chapter also explains the reasons behind the decisions taken for the components. It can be concluded that in order for an experimental setup to work many adaptations are required. Furthermore, certain hardware components can cause limitations which require workarounds and leave drawbacks in the research.

5 Experiments

In this chapter the initiation step is discussed, this step is performed before every experiment session. Here the baseline exploration is also included. Then the basis for the training of the RL algorithm is explained, followed by the simulation of the training experiments. Finally the three real world training experiments are featured.

5.1 Initiation

After connecting all the hardware and mounting the airfoil, the EWT is manually turned on to start the experiments and the time is synchronized to the laptop with the RL algorithm. The servo motors attached to the flaps of the airfoil are set to the 95 position, halfway between the minimum and maximum angles of the servomotor, which then allows the force balance stinger to be set to zero for the start of the measurements. This is done so the airfoil weight is not affecting the measured forces.

5.1.1 Baseline values

After the initiation steps, the baseline values of the set-up, airfoil and sensor can be measured. This step is not performed at the start of each experiment, but is performed to find the baseline properties of the previously mentioned components. The baseline values are found for different wind speeds for incremental flap positions which together covers the whole range of possible flap position. The baseline values for the airfoil are measured in normal force and axial force. From these the lift and drag force are derived using the manually set angle of the stinger. For the flow sensor, the effects of the changing resistive value of the strain gauges due to bending under the airflow, is measured in mV via the Wheatstone bridge. As mentioned previously the voltage at V1 in the Wheatstone bridge is measured with the DFRobot in combination with the Arduino. The baseline values for the fixed resistors side, V2, of the Wheatstone bridge are measured in the same fashion. V1 and V2 can be seen in Figure 4.12.

Method

The steps to measure the baseline values are as follows. The flaps are moved at a set path with the stinger set to an angle of 6°. The flaps are moved after a fixed amount of data points have been taken for that corresponding flap position, at a set wind speed. After a small time interval has passed (10 seconds) the measurements are turned on. This is done to allow the airflow to settle over the airfoil before starting the measurements. The measurements are done for 300 data points at a frequency of approximately 20 Hz. This was repeated for each flap position in the set path. Meanwhile, the auto clicker was turned on constantly during both the settling and the measurement period to obtain constant measurements from the EWT. This process was performed for wind speeds: 3, 5, 7, 9, and 11 ms^{-1}. This resulted in a baseline for each of those wind speeds for the airfoil, sensor and Wheatstone bridge resistors. Using Equations 5.1 and 5.2, the lift and drag can be calculated from the measured normal and axial force (Aerolab, 2021). Here N_{force} and A_{force} represent the normal and axial force. α is the angle of attack which is determined by the angle of the stinger. In this project the angle of the stinger and thus the angle of attack, α, is kept constant at 6°. This angle is chosen as it ensures a certain amount of generated lift by the airfoil.

$$F_{\text{lift}} = N_{\text{force}} * \cos(\alpha) - A_{\text{force}} * \sin(\alpha) \tag{5.1}$$

$$F_{\text{drag}} = N_{\text{force}} * \sin(\alpha) + A_{\text{force}} * \cos(\alpha) \tag{5.2}$$

5.1.2 The training process

In this subsection the followed training process for each of the experiments discussed below, is explained in more detail. As the process is approximately the same for each of these training experiments, the standard method is explained in this section. A scheme of the training can be seen in Figure 5.1. The measurement session component is performed for each measurement session. The initializing learning component is performed at the start of each training session. The orange component shows what happens each episode. The red and blue sections in the orange component are the interaction and updating phases of the TD3 algorithm, which can be seen in Algorithm 1. Each episode in the experiments has 1000 steps.

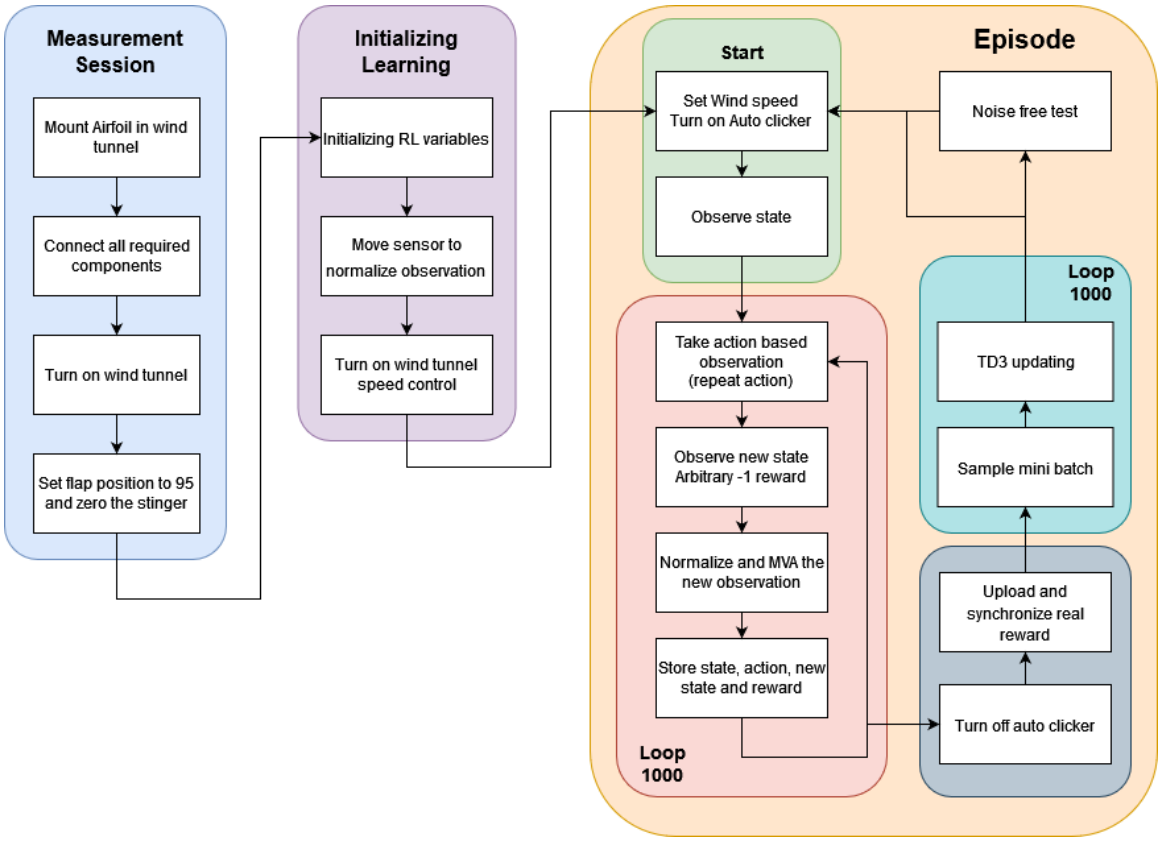


Figure 5.1: Scheme of the training

Initializing software

Before starting any of the training, the TD3 algorithm initializes its policy parameters, Q-function parameters, target policy parameters and target Q-function parameters. Additionally, an empty replay buffer is created.

Normalization sensor

After the initialization of the software, the program request that the normalization test is performed. The normalization test is done by manually bending the flow sensor in the direction of the air flow, until it touches the airfoil. This normalization step is explained in more detail in Chapter 4.7. The found values of this step are used to normalize the measurements from the Wheatstone bridge between the ranges -1 and +1.

Training

Now the training episode can begin. The RL algorithm gives a grace period of ten seconds to set the EWT to a desired speed and to turn on the auto clicker. Then the data used in the

reward function can be recorded. Following this, the RL algorithm receives the normalized MVA observation from the software environment, which receives data from the Arduino. Using this observation an action is chosen by the RL algorithm. This action is performed by the servo motors to move the flaps. The flaps only move every other step because of the repeated actions in alternating time steps. As mentioned before this is done to ensure that the servo motors can reach the desired position before receiving a new command. After the action is performed the new state is reached. The RL algorithm receives the normalized MVA observation from the software environment. An arbitrary reward is given which will later be overwritten by the rewards from the EWT data. The observation, action, reward and the new observation are stored in the previously mentioned replay buffer. This process is repeated 1000 times.

Reward data
Once all observations have been collected the auto clicker is manually turned off. The data from the EWT is stored on an USB flash drive and transferred to the laptop with the RL algorithm. From this the reward is found and by using time synchronization, matched to the state and observation of the RL algorithm. This replaces the arbitrary rewards.

Learning and noise free tests
After the step mentioned in previous paragraph the RL algorithm performs its learning process as seen in the pseudo code in Algorithm 1. Once the learning is completed the next training episode can be started at a different wind speed. The process mentioned in this section, Section 5.1.2, is repeated. Before a new episode is started the RL algorithm can be tested. This is done by letting the algorithm choose actions without exploration noise based on incoming normalized MVA observations. In the experiments below exploration noise free tests are performed to assess the performance of the algorithm.

5.2 Simulation of training

In this section the simulations of the experiments will be explained in detail. The simulations of the training will be used to determine the number of episodes the actual experiments will be trained for. The RL algorithm, in the simulation runs, will be trained to reach the maximum lift and reach a specific lift. The reward functions in both simulations reflect these goals.

5.2.1 Method

In the simulation only the RL algorithm is used. No wind tunnel, air foil or sensor is used. The RL algorithm will only perform the orange component as seen in Figure 5.1. However, there are a few changes made for the simulation run. Instead of a sensor observation found at the set wind speed, the RL algorithm receives a normalized observation at a predetermined level with added noise.

This predetermined level is based on observed values of the observations in real experiments and changes per episode. The levels for V1 are set to -0.129 and -0.34. These levels for V1 represent normalized observations for 3 and 9 ms^{-1} wind speeds.

The level for V2 is set to 0 for all episodes. The lift, used in the reward, is determined by the baseline experiments. For the wind speeds of 3 and 9 ms^{-1} a fit for the lift, per flap position, is found. From this fit the lift can be assumed for each action the RL algorithm chooses in the simulations. To further simulate the shortcomings of the wind tunnel the rewards for every 10 steps are the same. This reward depends on the actions taken of the previous 10 steps. From the actions of the previous 10 steps lifts are calculated. These lifts are averaged and this average lift is used to calculate a reward based on the reward function. This is then the reward that is given to the current 10 steps. This is used to simulate a delay in the reward. It also simulates

the effect that in the real experiments all time steps, within a second, receive the same reward.

The reward function for the maximum lift simulation will be equal to the lift, thus a higher lift gives a higher reward. For the simulation that needs to reach a specific lift goal the reward function will be that of Equation 3.1. Here the desired lift is set to 0.15 N.
For both simulations the warm up period will be two episodes and each episode will be 1000 steps.

5.3 The training experiments

In this section the training experiments performed are explained in more detail. First learning the maximum lift value is explained, followed by learning a specific normal force and lift at different wind speeds.

5.3.1 Experiment 1: learning maximum lift/normal force value

In this first experiment the goal was for the RL algorithm to learn how to position the flaps in such a way that the maximum lift can be reached for different wind speeds. The reward function in this experiment was set to be equal to the lift. A higher lift leads to a higher reward, therefore the RL algorithm was taught to maximize lift. This experiment was done to assess if the RL algorithm and system can match the maximum lift. The optimal action to maximize the lift is the same for each wind speed.

Method

The training for this system was done at two different wind speeds, namely 3 and 9 ms^{-1}. Each wind speed was used for 3 episodes, together totalling 6 training episodes. The warm up period was set to be equal to two training episodes. The training steps were performed as described in section 5.1.2. The exploration noise free test was performed between some of the episodes for each training wind speed and an unknown wind speed, for the RL algorithm, of 7 ms^{-1}. Table 5.1 shows all these parameters.

Parameter	Value
Wind speeds	3 ms^{-1} and 9 ms^{-1}
# of episodes per wind speed	3
Total amount of episodes	6
Warm up period	2
Unknown wind speed	7 ms^{-1}

Table 5.1: Parameters for experiment 1

5.3.2 Experiment 2: learning a set lift/normal force value using two wind speeds

In the second experiment the goal for the RL algorithm was to learn how to position the flaps in such a way that a specified lift/normal force value is reached. Unlike in experiment 1, the optimal flap position differs with wind speed. The reward function for this experiment is seen in Equation 3.1.

F_{measured} fluctuates depending on the wind speed and flap position. F_{desired} was set to be equal to a lift/normal force of 0.15 N. The goal of this experiment is to assess the ability of the RL algorithm to reach the desired value at two different known wind speeds and one unknown wind speed.

Method

The training in this experiment was done in the same method as for experiment 1 but with various different parameters. The parameters are shown in Table 5.2.

Parameter	Value
Wind speeds	$3\,ms^{-1}$ and $9\,ms^{-1}$
# of episodes per wind speed	6
Total amount of episodes	12
Warm up period	2
Unknown wind speed	$7\,ms^{-1}$

Table 5.2: Parameters for experiment 2

5.3.3 Experiment 3: learning at three different wind speeds

The third experiment was similar to the second experiment. The goal was again for the RL algorithm to learn how to position the flaps in such a way that a specified lift/normal force value is reached. This experiment uses the same reward function as experiment 2. However, in this experiment the RL algorithm was subjected to three different wind speeds. The goal of doing this additional experiment was to assess if the RL algorithm performed differently when trained with either two or three known wind speeds. This assessment was done by comparing the reached lift by the RL algorithm on the known and unknown wind speeds.

Method

The training in this experiment was done in the same method as for experiment 1 and experiment 2 but with various different parameters. The parameters are shown in Table 5.3.

Parameter	Value
Wind speeds	$3\,ms^{-1}$, $9\,ms^{-1}$ and $11\,ms^{-1}$
# of episodes per wind speed	6
Total amount of episodes	18
Warm up period	3
Unknown wind speed	$7\,ms^{-1}$

Table 5.3: Parameters for experiment 3

5.4 Conclusion

In this chapter the three types of experiments were introduced: baseline, simulations and training experiments. How each experiment is conducted and the goals of the experiments are discussed. From creating the experiments the necessity of the initiation steps was established. Furthermore, it can be concluded that due to limitations in the experimental design, running of the experiments requires many manual steps and processes. The results of the experiments are presented in the next chapter.

6 Results

This chapter contains the results of the experiments. The sequence is the same as the chapter above and therefore first discusses initiation, then the experiments one through three.

6.1 Baselines

In this section the initiation is discussed. During the initiation steps, performed before the experiments, the baseline values for normal, axial, lift and drag forces are found. The effects of the changing resistance, in the sensor strain gauges, on the voltages in the Wheatstone bridge are also found.

6.1.1 Baseline values

Plot 1: normal force

In Figure 6.1, presented below, the normal force baseline is shown. The plot has normal force plotted against time. Normal force is presented in N and time is presented in steps, each representing a measured data point. The total amount of time is approximately 7 minutes and 10 seconds. The right y-axis is to represent the black line of the plot, which showcases the position of the flap, with the unit representing the flap position. In the plot each colour represents the measurement done at a different wind speed. Where blue is $3\,ms^{-1}$, red is $5\,ms^{-1}$, yellow is $7\,ms^{-1}$, purple is $9\,ms^{-1}$ and green is $11\,ms^{-1}$. These colours are consistent with their respective wind speeds for all of the upcoming baseline plots.

From Figure 6.1 it can be seen that the wind speed and flap position greatly affect the measured normal force. The wind speed versus normal force is explained in Equation 2.4. The highest normal force can be seen at the lowest flap position. When normal force is at its minimum the flap position is at its maximum. At a flap position of approximately 100 the airfoil produces the same amount of normal force regardless of the wind speed. In the graph this is the point where the different wind speed lines intersect.

Plot 2: axial force

Figure 6.2 shows axial force versus time steps. The x-axis shows the time steps and the y-axis, on the left side, shows the axial force in N. The y-axis on the right side represents the flap position. This is similar to Figure 6.1. From this graph it can be seen that when the flap position increases, an increase in axial force is observed. Furthermore, a higher wind speed always has a higher axial force at a set flap position for this set up.

Plot 3 and 4: lift and drag

Lift and drag are derived from the normal and axial force as seen by Equation 5.1 and Equation 5.2. A strong resemblance between the graphs of lift and normal force and the drag and axial force can be seen. Both plots have force in N on the left hand y-axis, the time steps on the x-axis and flap position on the right y-axis, similar to Figure 6.1. The black line in the graph represents the path taken by the flaps. The coloured lines are the lift and drag forces at each measured wind speed. In Figure 6.3 the force measured is lift, while in Figure 6.4 the drag force is measured. The notable points in the graphs for the lift and drag forces correspond with the Figure 6.1 and the Figure 6.2 observations, respectively. The lift is higher for low flap positions, this corresponds to the theory as seen in Figure 2.5.

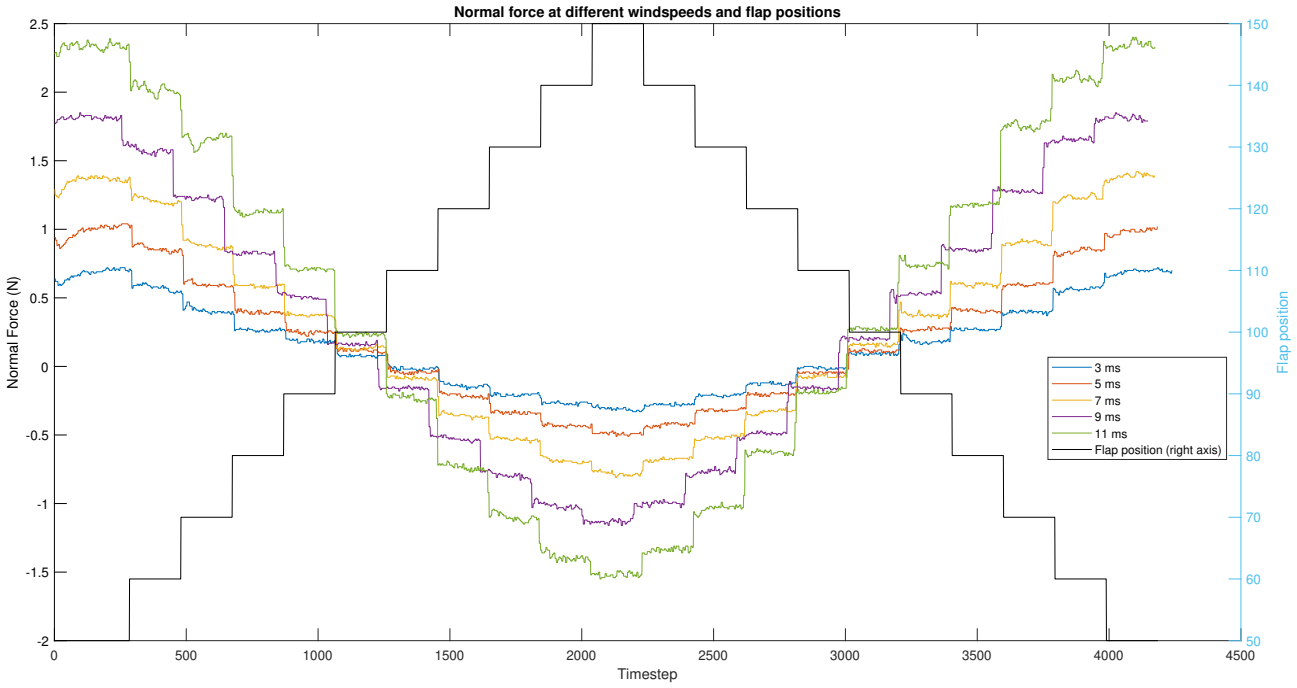


Figure 6.1: Experiment baseline Normal force

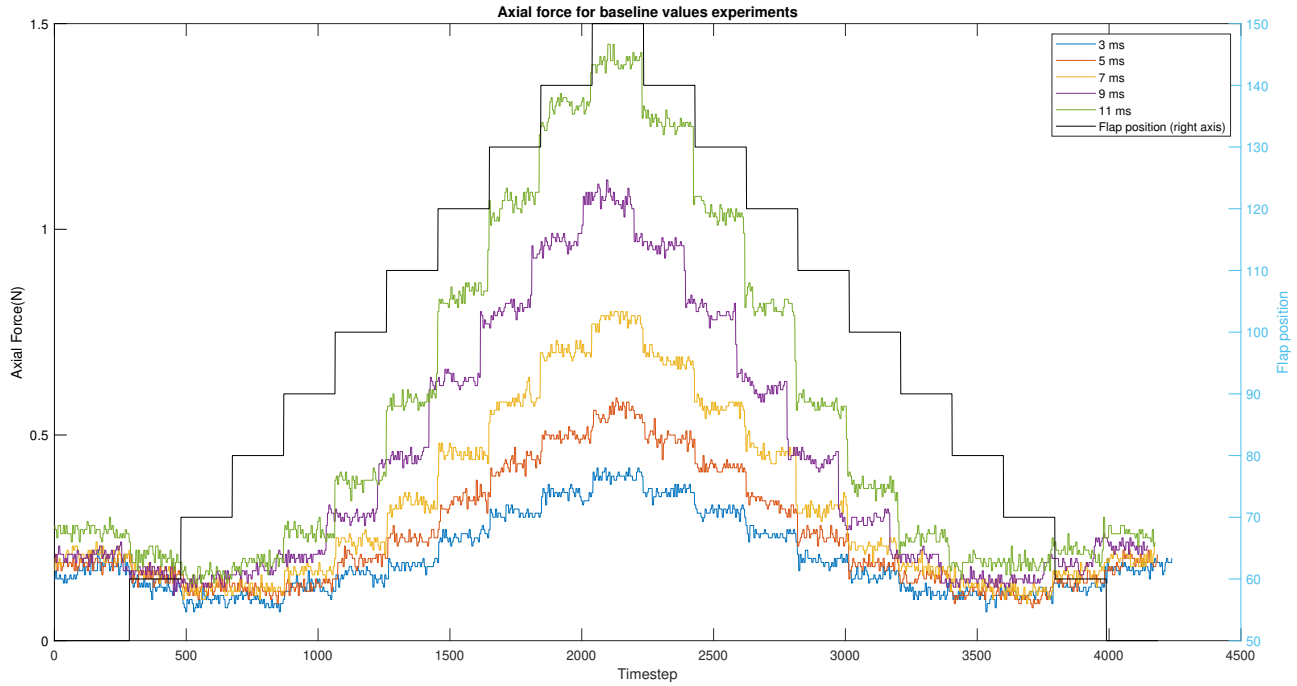


Figure 6.2: Experiment baseline Axial force

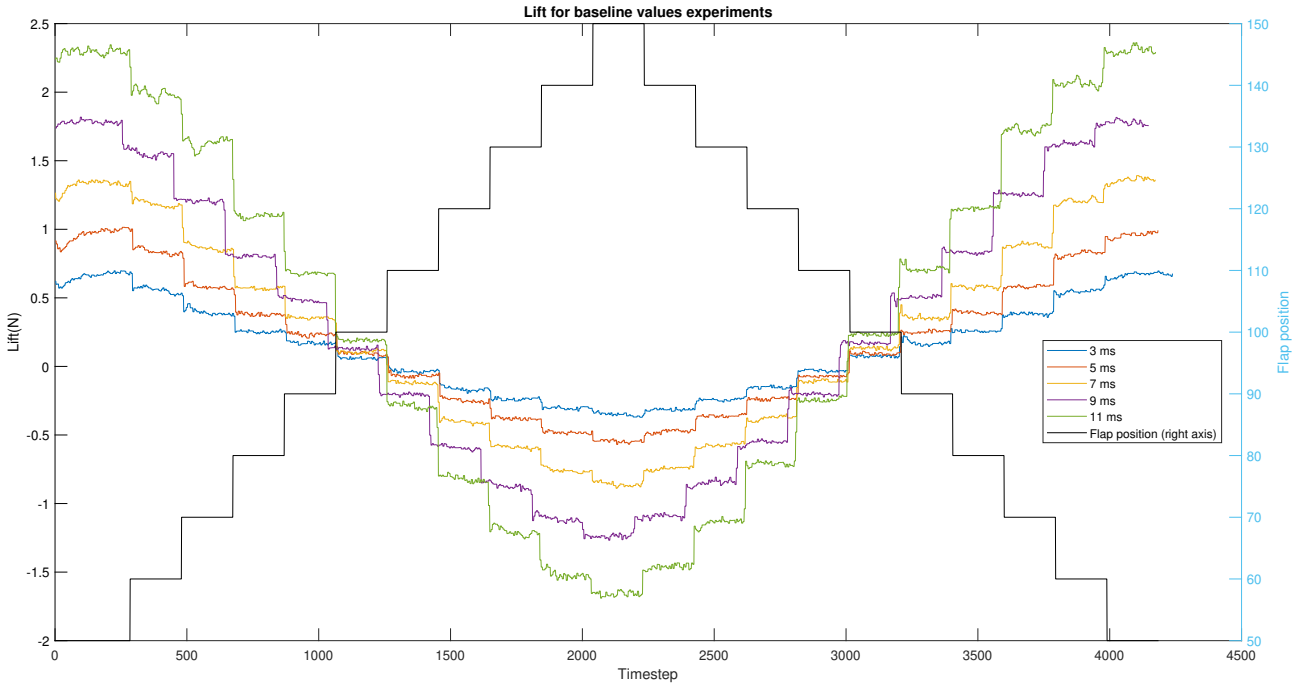


Figure 6.3: Experiment baseline Lift force

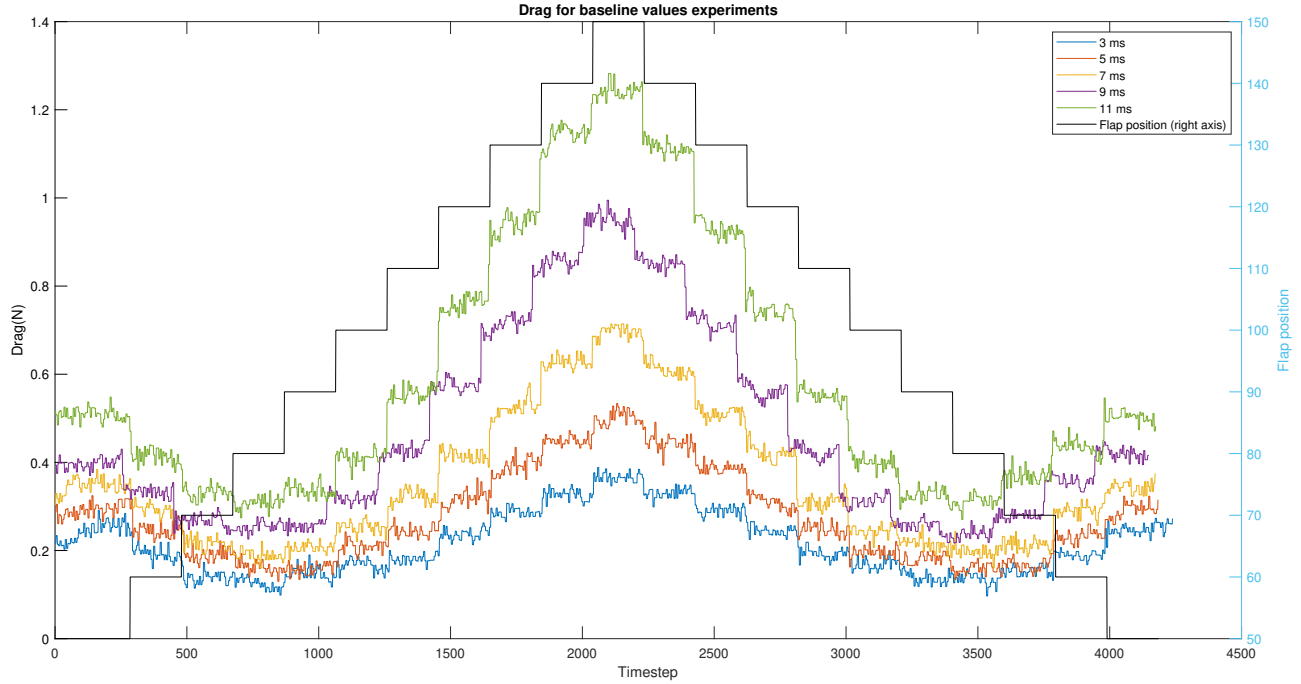


Figure 6.4: Experiment baseline Drag force

6.1.2 Wheatstone bridge measurements

In this section the sensor output during the baseline measurements is shown with three plots.

V1 (sensor)

The graph in Figure 6.5 shows the measured values by the DFRobot over the sensor side, V1, of the Wheatstone bridge, known onwards as sensor measurement.

In Figure 6.5 the sensor measurement in mV is plotted against time steps, with a right hand y-axis representing the flap position. The colours of the wind speeds are the same as in the figures above, explained in Section 6.1.1. Fluctuations between whole numbers in the sensor measurements in the graph show the limits of the resolution of the DFRobot.

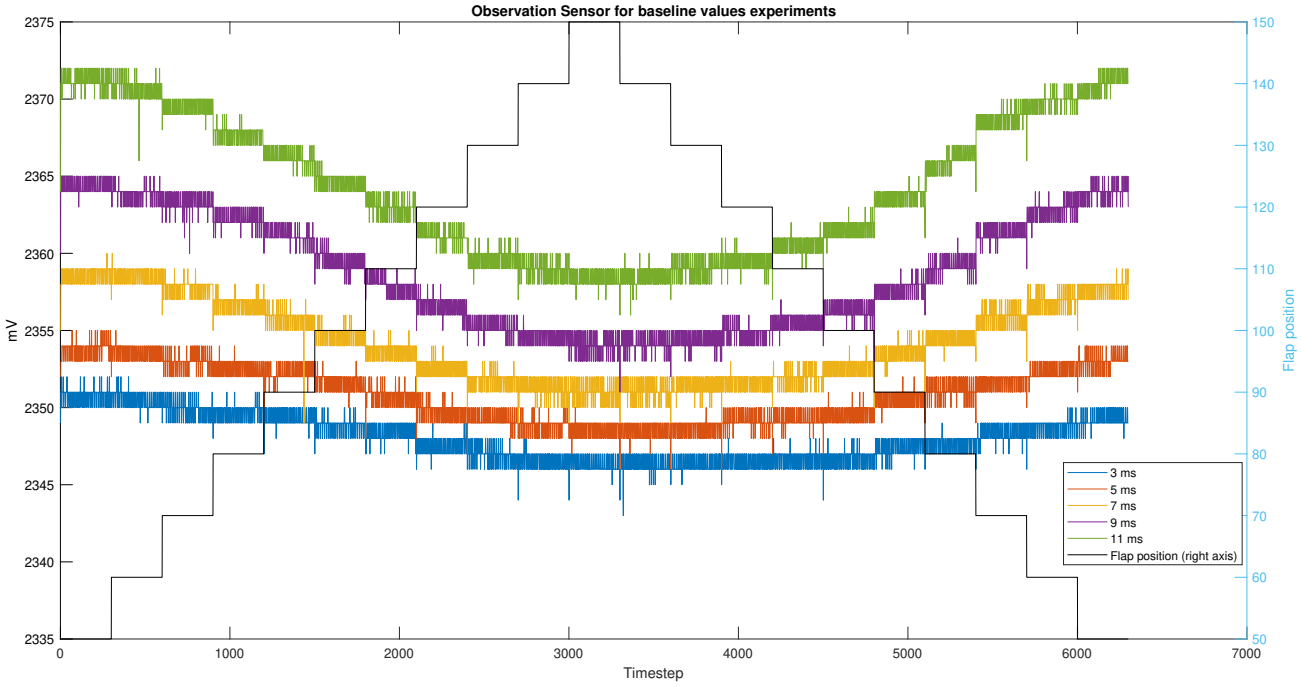


Figure 6.5: Experiment baseline Raw sensor

Moving average of V1

In Figure 6.6 the previous plot from Figure 6.5 has been transformed using a moving average of ten data points. This graph shows the input for the RL algorithm. The moving average was taken for each respective flap position, this signifies that a new flap position leads to a restarting of the moving average process. The y-axis presents the measured values from the sensor measurement in mV, the x-axis presents the time steps and the right hand y-axis presents the flap position.

V2 (resistors)

Figure 6.7 shows the measured output of the resistor side of the Wheatstone bridge by the DFRobot. The axes present the same values as above. A great number of the output was measured to be between 2212 and 2213 mV regardless of the wind speed and flap position.

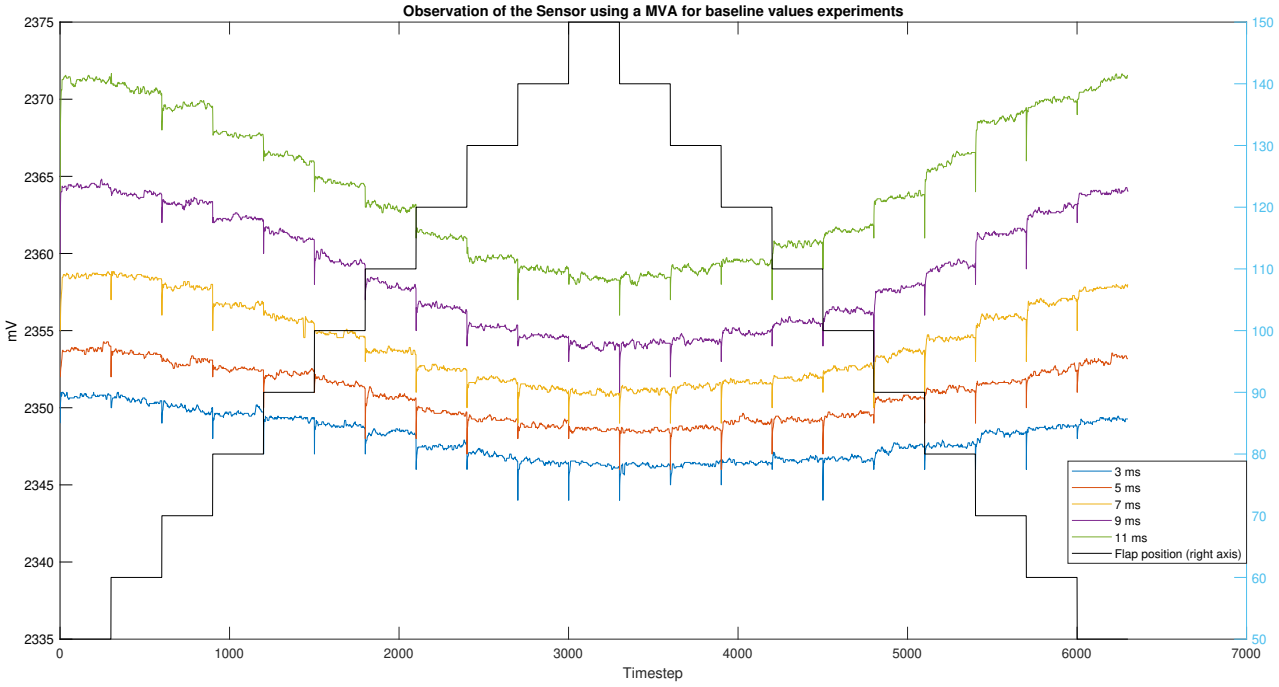


Figure 6.6: Experiment baseline MVA(10) sensor

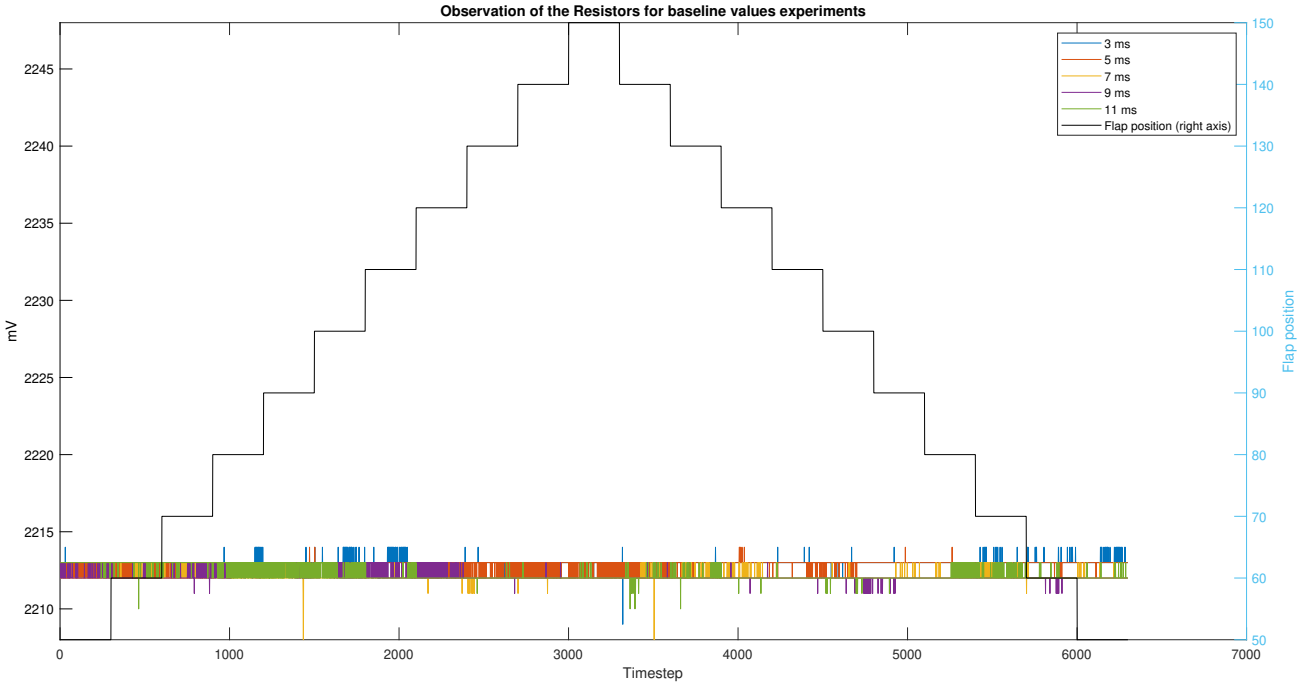


Figure 6.7: Experiment baseline Raw Resistors

6.1.3 Hysteresis

In this section hysteresis is checked for the lift around the airfoil and the flow sensor.

Folded average for hysteresis lift

Figure 6.8 show a plot with on the y-axis the lift in N and the flap position on the x-axis. This plot is created by taking the average lift for the flap positions. This specific graph showcases the average lift for the 9 ms^{-1} wind speed measurements. These averages are differentiated between the upward and downward movement of the flap seen in blue and red, respectively. This graph was plotted to check for hysteresis in the lift of the airfoil. The upward and downward movements of the flap are close together, which is why sometimes in the graph the colours are difficult to differentiate. The standard deviation of the lift is shown by the error bars surrounding the data points.

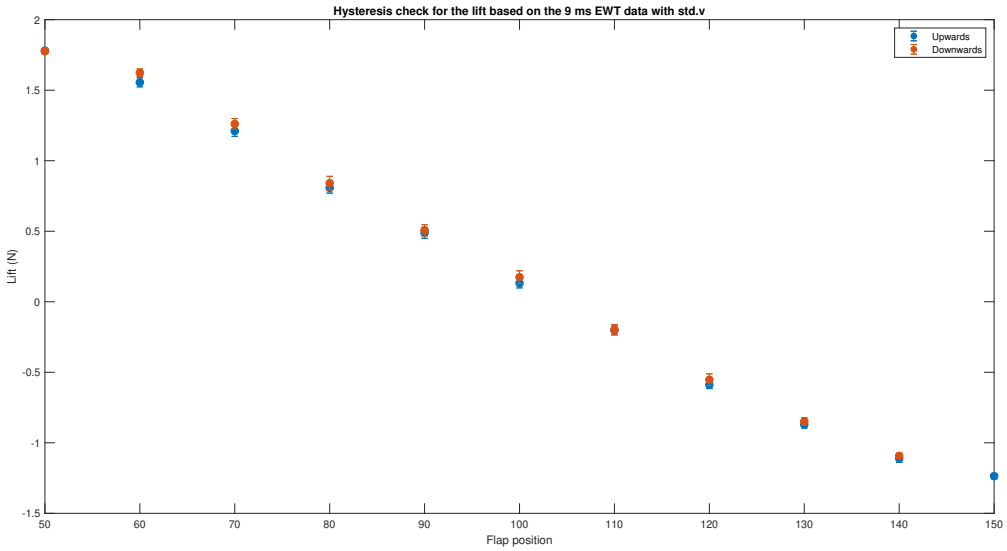


Figure 6.8: Hysteresis check Lift

Folded average for hysteresis sensor

The Figure 6.9 is similar to Figure 6.8, except that the y-axis now represents the average observed sensor measurements per flap position in mV. Unlike the graph in Figure 6.8, in Figure 6.9 the data points are easily distinguished from another as the upward and downward data points follow a different path. The standard deviation of the sensor output is shown by the error bars surrounding the data points. This graph was plotted to check for hysteresis in the sensor measurement.

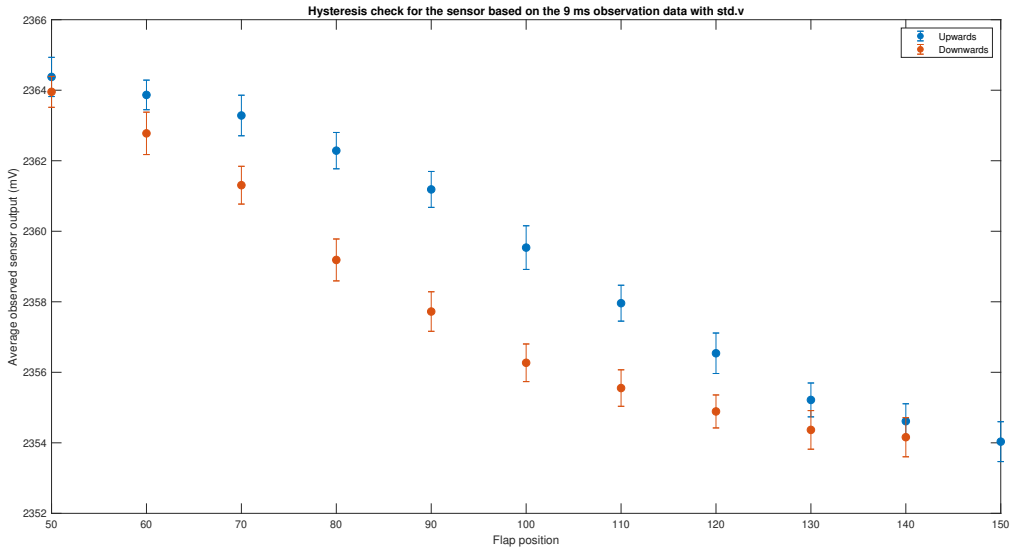


Figure 6.9: Hysteresis check Sensor

6.1.4 Analyses of the forces

In this subsection a more classical analysis approach of the forces upon the sensor is carried out by comparing the lift to different variables.

In Figure 6.10 the average lift at a given flap position is plotted against the average V1 sensor output. This is done for all the measured wind speeds. These are represented each by their respective colours. Larger wind speeds have a larger lift range, therefore leading to a larger state versus lift curve.

Using the information found in Figure 6.9 it can be concluded that the lowest data point of each curve represents the data measured at the flap position of 150. From this point two lines, per wind speed, are seen moving upwards. The left line of these two lines contains the data points measured in the upward movement of the flaps. The right line contains the data points measured during the downward movement of the flaps. The end data points of each line represent the average data taken at the flap position of 50.

In Figure 6.11 the lift is plotted against the drag, these values were found during the baseline experiment. Each data point in the curve represents the average lift and drag for that respective flap position and specific wind speed. The wind speeds are represented by the different coloured lines.

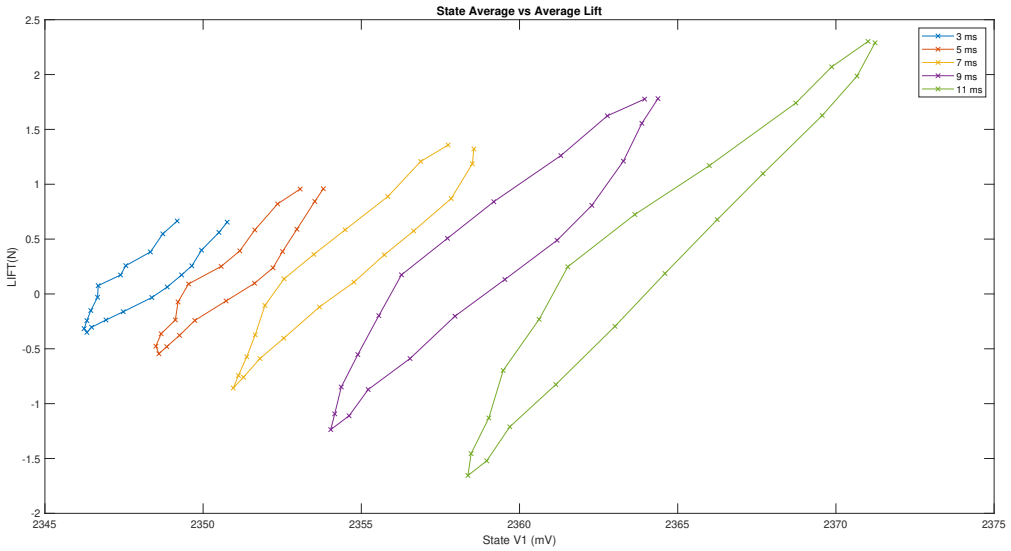


Figure 6.10: Average State of V1 plotted versus Average Lift

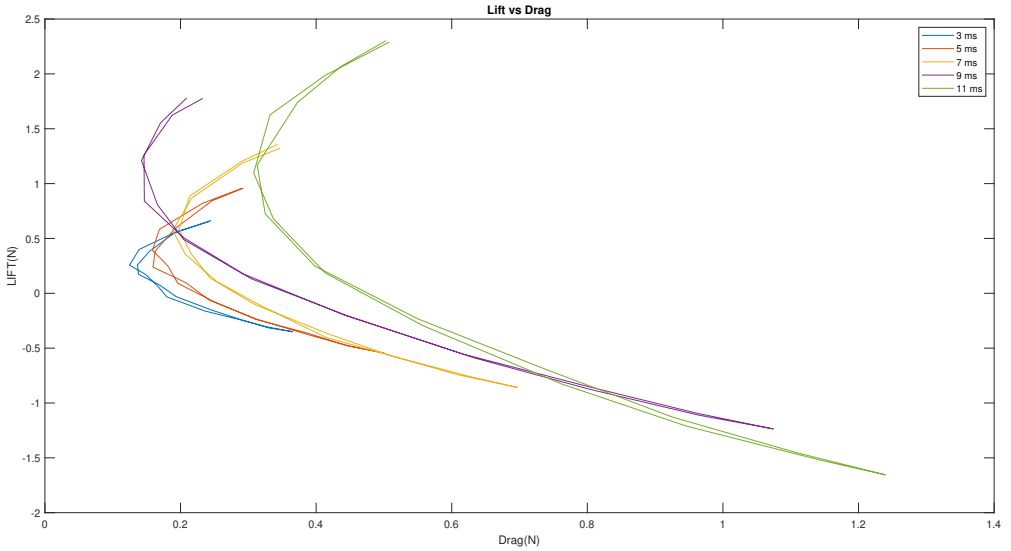


Figure 6.11: Average Lift versus Average Drag

6.2 Simulations

In this section the results of the simulation are shown. Both simulations were trained using two predetermined baseline values for the observations.

6.2.1 Maximum lift

The results of training the system in the simulation can be seen in Figure 6.12. The x-axis of the plot shows the steps of the system, each episode contains 1000 steps. This plot contains four y-axes.

The leftmost y-axis, seen in red, presents the simulated wind speed in ms^{-1}. The other left y-axis, seen in yellow, presents the MVA of the set sensor observations in mV. The rightmost

axis, seen in green, represents the action in flap position. The other right y-axis, seen in blue, represents the reward of the system. The red line in the plot corresponds with the red y-axis and shows the simulated wind speed per step. The yellow line in the plot corresponds to the yellow y-axis and shows the MVA of the set sensor observation per step. The green line and green area corresponds to the green y-axis. The green line is the average action taken in that episode. The green area is the standard deviation of the action in that given episode. The blue line in the plot corresponds to the blue y-axis and shows the average reward per step for the episode. In this simulation the reward was set to be equal to the lift.

It can be seen, from the large standard deviation in the flap position, that the system explores the environment in the first two episodes of the training. After the first two episodes the average flap position is set to almost 50. This it the flap position that reaches the theoretical maximum lift for all wind speeds.

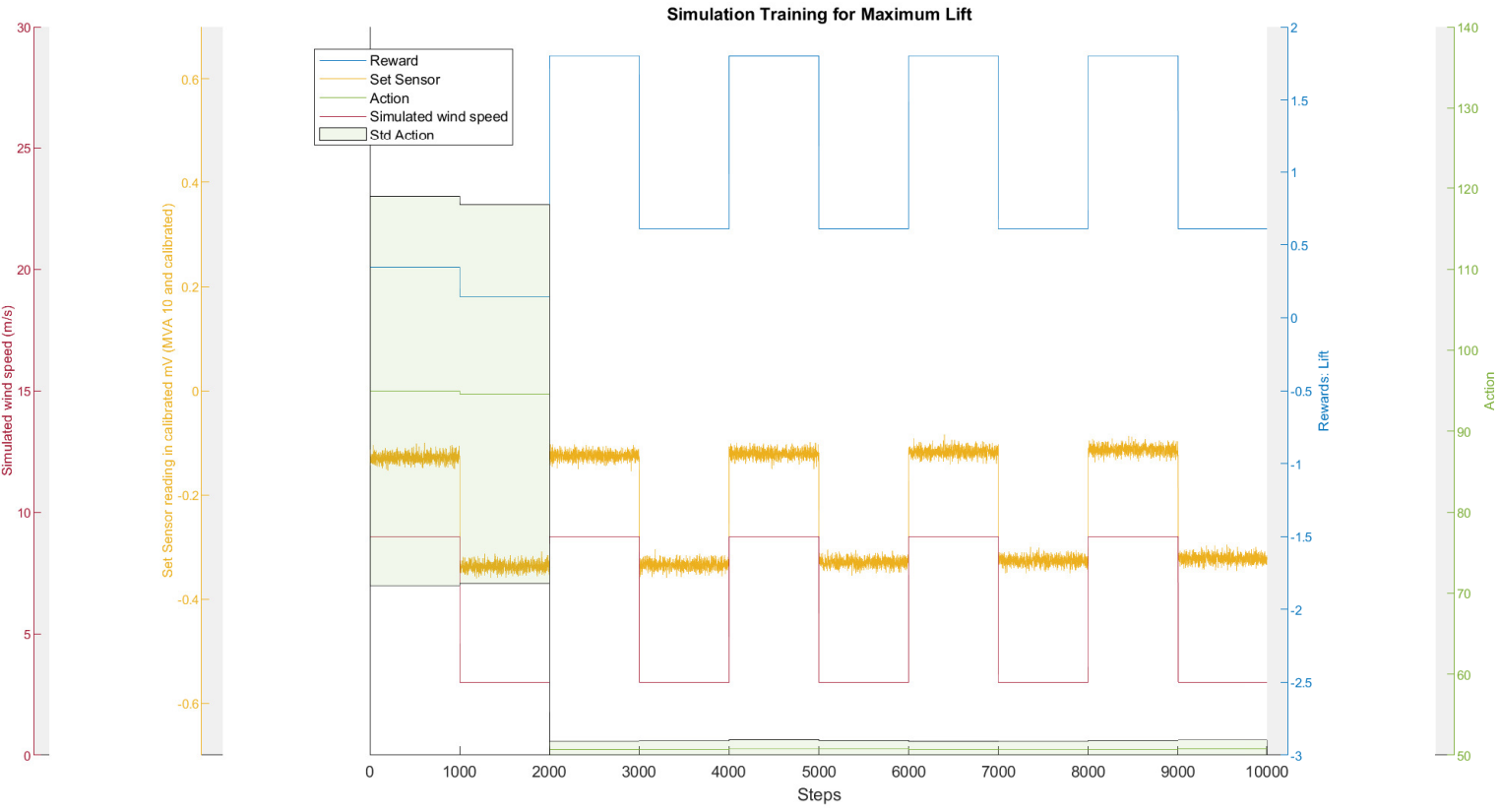


Figure 6.12: Finding maximum lift in the simulation

6.2.2 Target lift

In Figure 6.13 the same axes as in the figure above are used. The standard deviation of the reward is added, as seen by the blue area. From this plot it can be seen that the RL algorithm learns two different positions for the flaps in regards to the different input observations. These positions were used to simulate the different wind speeds. Although slight variations can be seen in the average flap position and reward, after episode 4, it is relatively constant for episodes with similar input observations. The positions found, for the different input observations, are close to the optimal position as seen by the near zero reward.

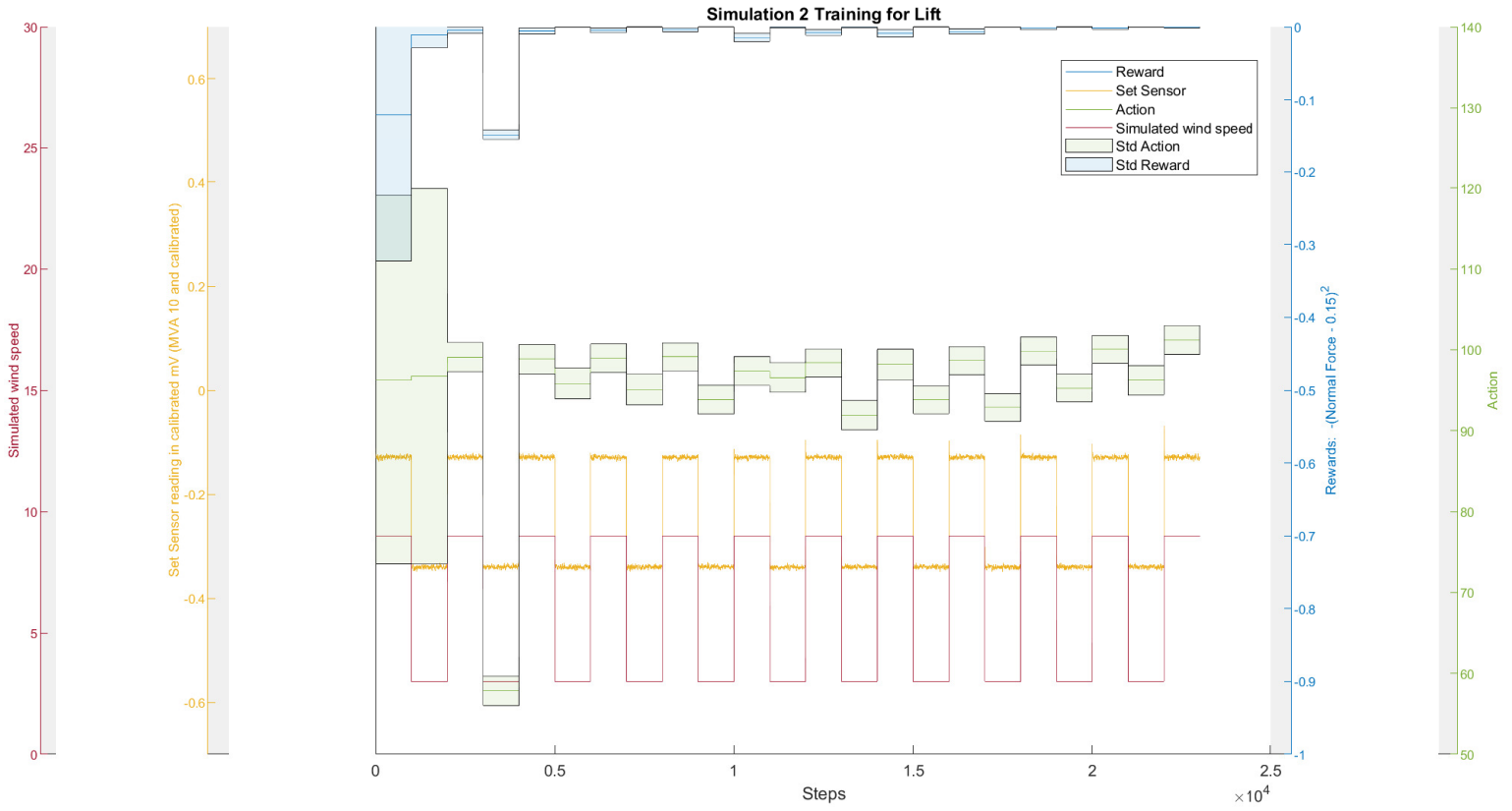


Figure 6.13: Finding a lift of 0.15 N in the simulation

6.3 Experiments

The following section presents the results from the experiments one through three. The first experiment performed was for the RL algorithm to learn the optimal flap position for maximum lift. Experiment two and three were performed for the RL algorithm to match a predetermined lift/normal force at various wind speeds. Experiment two trained at two wind speeds and experiment three trained at three wind speeds.

6.3.1 Experiment 1

Experiment one trained the RL algorithm to learn the optimal flap position for maximum lift.

Plot of the training

The training was performed using the moving average of the mV measured by the DFRobot in the Wheatstone bridge. In Figure 6.14 the plot of the training for the experiment is shown. This plot has one x-axis and four y-axes. The x-axis of the plot shows the steps of the system, each episode contains 1000 steps. The leftmost y-axis, seen in red, presents the set wind speed in ms^{-1}. The other left y-axis, seen in yellow, presents the normalized MVA of the sensor observations. The rightmost y-axis, seen in green, presents the action in flap position. The other right y-axis, seen in blue, presents the reward of the system. The red line in the plot corresponds with the red y-axis and shows the set wind speed per episode. The yellow line in the plot corresponds to the yellow y-axis and shows the normalized MVA of the sensor observation per step. The green line and the green area corresponds to the green y-axis. The green line is the average action taken in that episode. The green area is the standard deviation of the action

in that given episode. The blue line in the plot corresponds to the blue y-axis and shows the average reward per step for the episode. In this simulation the reward was set to be equal to the lift.

Every 1000 steps the set wind speed, average reward and average action with its standard deviation all change due the start of a new episode. The plot shows that in the first two episodes the system explores the environment. This can be seen from the large standard deviation of the action. Furthermore, the sensor observation show clear differences between the episodes with different set wind speeds.

In addition, it can be observed that the average action also changes with the set wind speed. The average action also drops between episode 3 and 5, these two episodes have the same set wind speed. For episodes 4 and 6 the average action stays the same. The average reward also changes as the set wind speed changes.

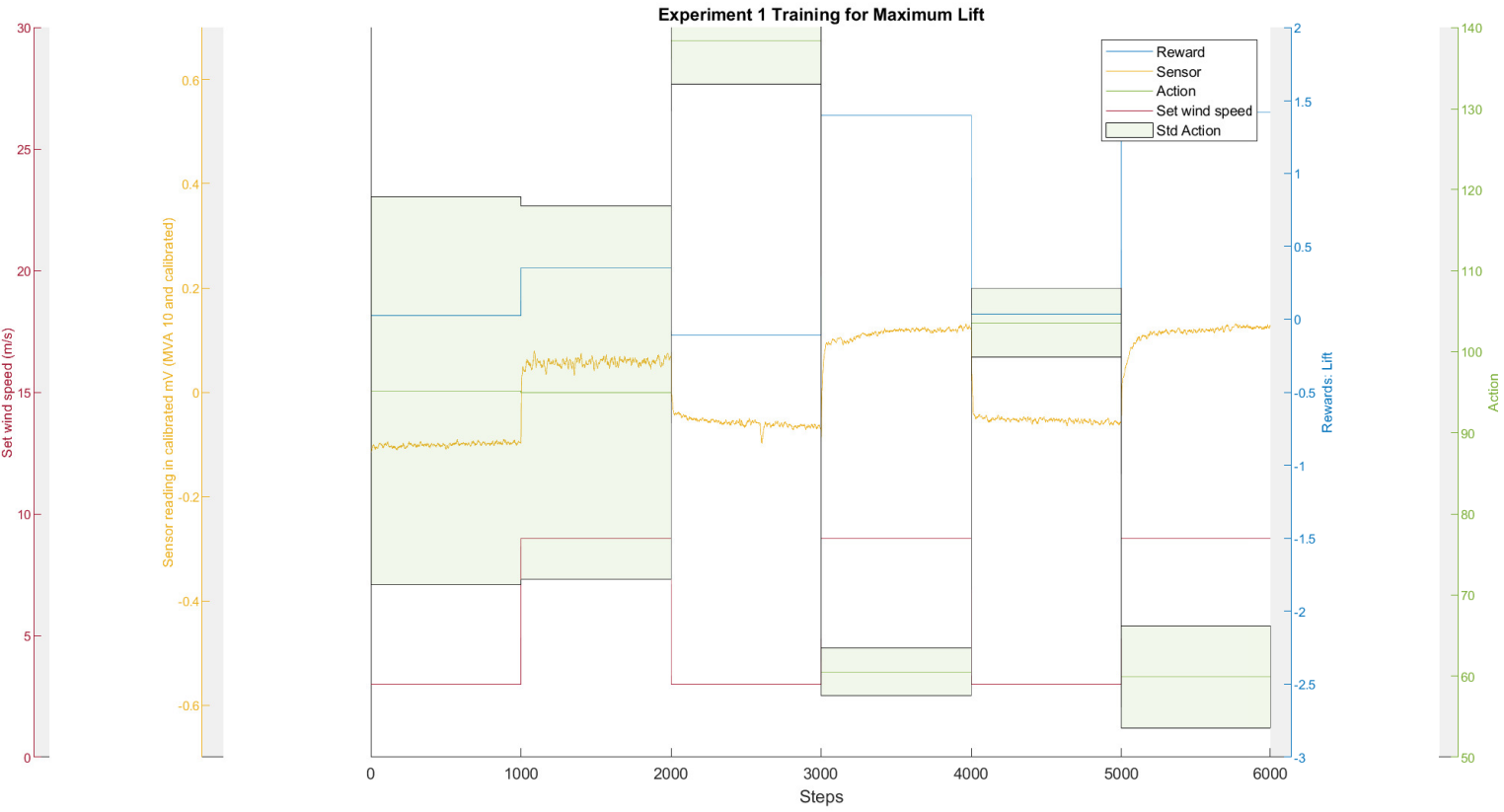


Figure 6.14: Training for maximum lift at 2 wind speeds

Exploration noise free test

In Figure 6.15 the results of the exploration noise free tests is shown. Testing lets the system choose its actions without exploration noise. In this plot the average lift value of each noise free test is displayed. The data points are displayed as x in the plot. The data points, for a specific wind speed, are connected via a trend line. The lift of the system at different wind speeds was tested after the training of episodes two and six. The wind speed tested were 3 ms^{-1}, 9 ms^{-1} and the unknown wind speed, chosen to be 7 ms^{-1}. In total six tests were performed.
It can be seen that the lift generated by the system differs for each wind speed but stays rela-

tively constant between the test for a given wind speed, however a small upwards trend can be seen.

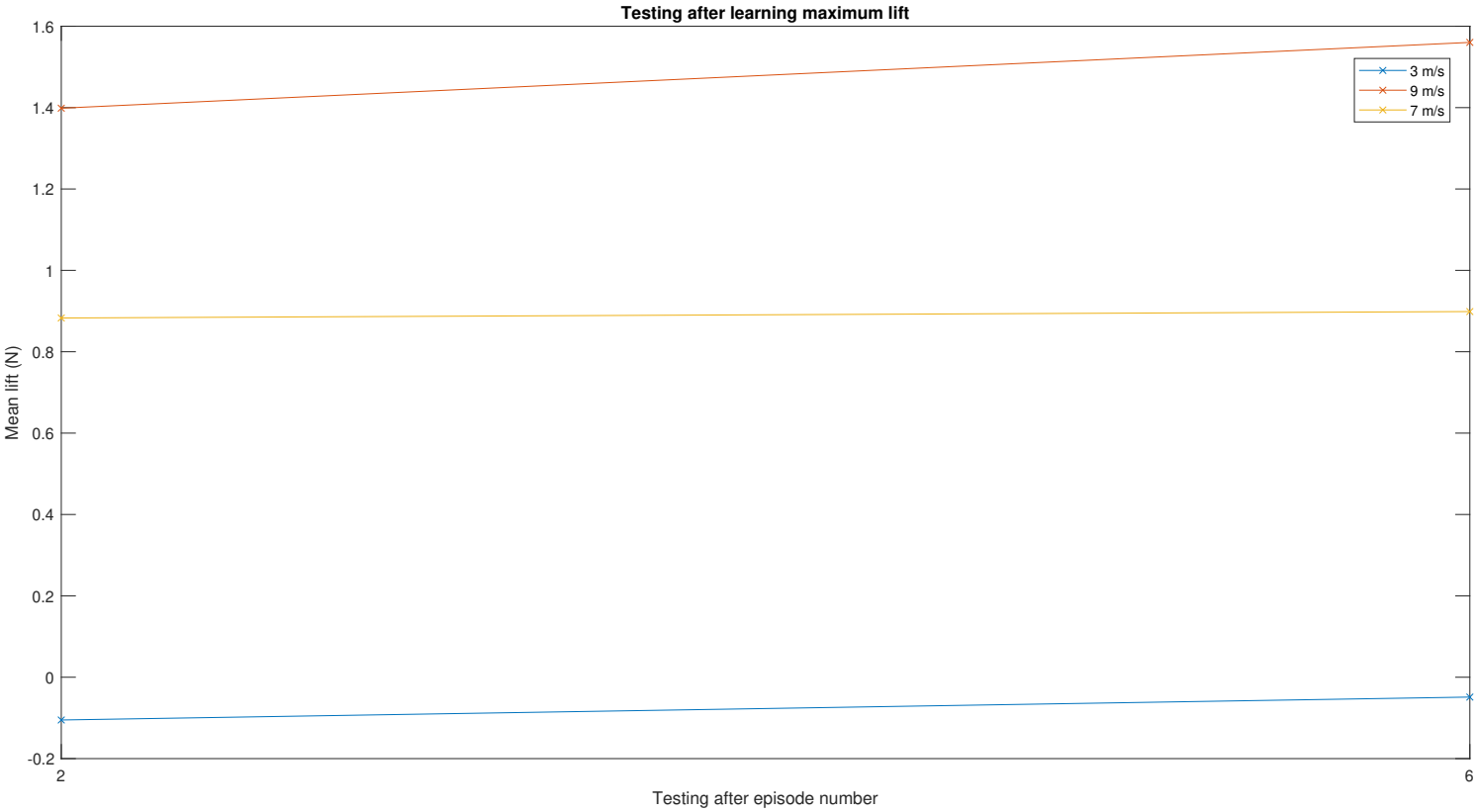


Figure 6.15: Testing of the system for lift

6.3.2 Experiment 2

In this section the results for experiment two are shown. Experiment two trained the RL algorithm at two wind speeds: $3\,ms^{-1}$ and $9\,ms^{-1}$. During the tests the system was also exposed to a wind speed of $7\,ms^{-1}$, this represent the unknown wind speed. Two training session were performed. One for optimizing normal force and the other for optimizing lift.

Plot of the training

Figure 6.16 and 6.17 both show a plot that contains the important training data for experiment 2. This plot has one x-axis and four y-axes. The axes for this plot are the same as the axes for the training plot of the previous experiment. The blue y-axis still present the reward, however the reward function has changed with respect to the last experiment. The reward function is now set to reach a target lift/normal force of 0.15 N. The reward function was set to normal force and lift separately, therefore the experiment was done twice, resulting in two plots.

The lines in these plots also represent the same variables as describe in the previous plot in Figure 6.14. However, an extra blue area can be seen in the plot. This represents the standard deviation of the reward.

When looking at the plot for normal force in Figure 6.16, it can be seen the the observa-

tions show a clear difference when the set wind speed changes. This is represented by the block shape of the yellow line. Additionally, it can be seen that the average action per episode, taken by the system, also changes throughout the training. In later episodes the average action taken remains relatively constant with regard to the wind speed. The same observation can be made for the average reward, shown by the blue line. It can be seen that the standard deviation of the reward is higher when the set wind speed is also higher.

In Figure 6.17, the lines represent the same measurement as in the previous figure, however the reward is now based on lift instead of normal force. Also for this experiment it can be seen that during the first 2 episodes the system was exploring and therefore the standard deviation surrounding both the green and blue lines is large. Throughout the learning the standard deviation becomes smaller and more constant. The standard deviation is never zero due to the exploration noise in the actions taken. Also in this plot, the block formation of the lines can be seen in the yellow observation line when the set wind speed changes. The values of the observation by the sensor, differs from the observations in the normal force experiment. This can be seen from the two levels the sensor reaches at the different set wind speeds. In Figure 6.16, the two levels are -0.4 and -0.2. For Figure 6.17 the levels are -0.2 and 0 at the same wind speeds. The system reaches the constant action sooner than in the normal force experiment, this is seen from the two plots. Furthermore, the average reward is lower when training on lift rather than training on normal force, shown by the lower blue lines in the plot.

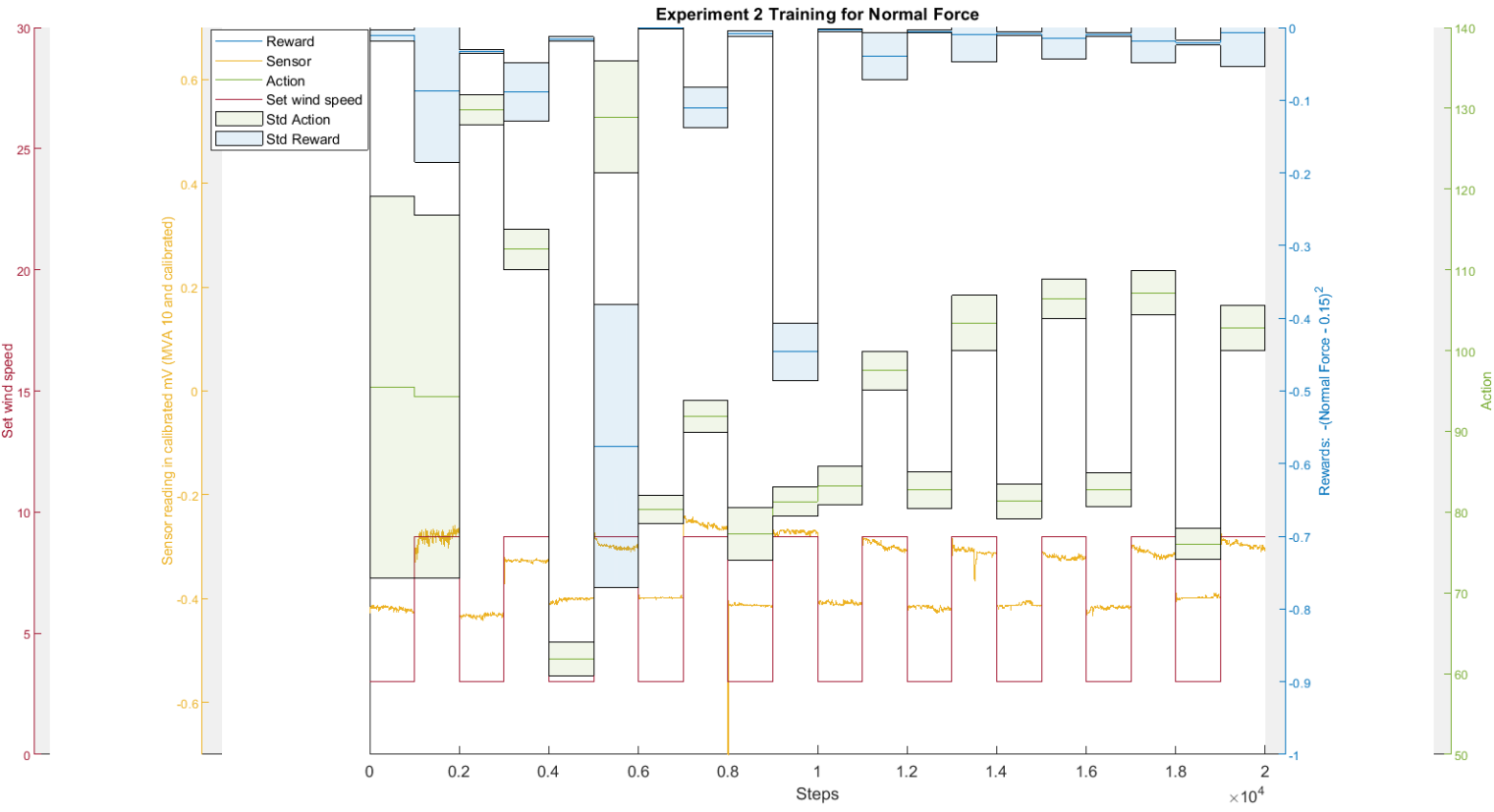


Figure 6.16: Training for Normal Force of 0.15, 2 wind speeds

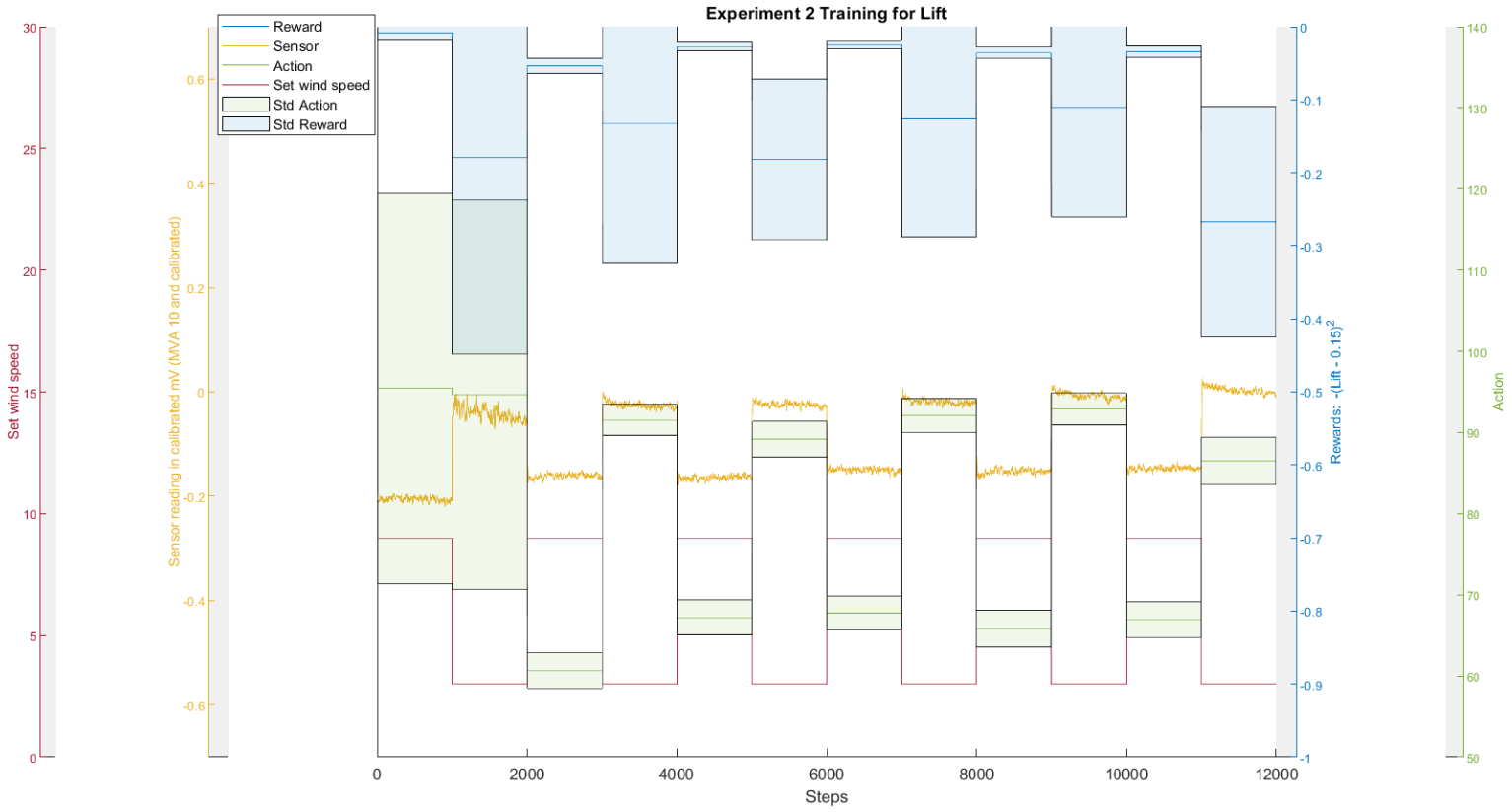


Figure 6.17: Training for Lift of 0.15, 2 wind speeds

Exploration noise free test

In this section the average lift of the exploration noise free tests for experiment two are shown. The tests were performed on the RL algorithm that trained on lift, this training can be seen in Figure 6.17. The plot which shows the results of testing the system for lift is shown in the Figure 6.18. The test were performed after episodes 2, 6 and 12, this is presented by the x-axis. In the plot the y-axis represent the average lift in N. The average lift was plotted for clarity as it did not change during the tests at the set wind speeds. The blue data represents the test performed at 3 ms^{-1}. The orange data represents the test performed at 9 ms^{-1}. The yellow data represents the test performed at 7 ms^{-1}, the so called unknown wind speed. The red line in the plot indicates the target lift value of 0.15 N. From the plot it can be seen that the lift lines, generated by the system, do not reach the red line which represents the target lift value in the test experiment. Only the yellow line, for the wind speed of 7 ms^{-1}, has a constant downward trajectory towards the target red line.

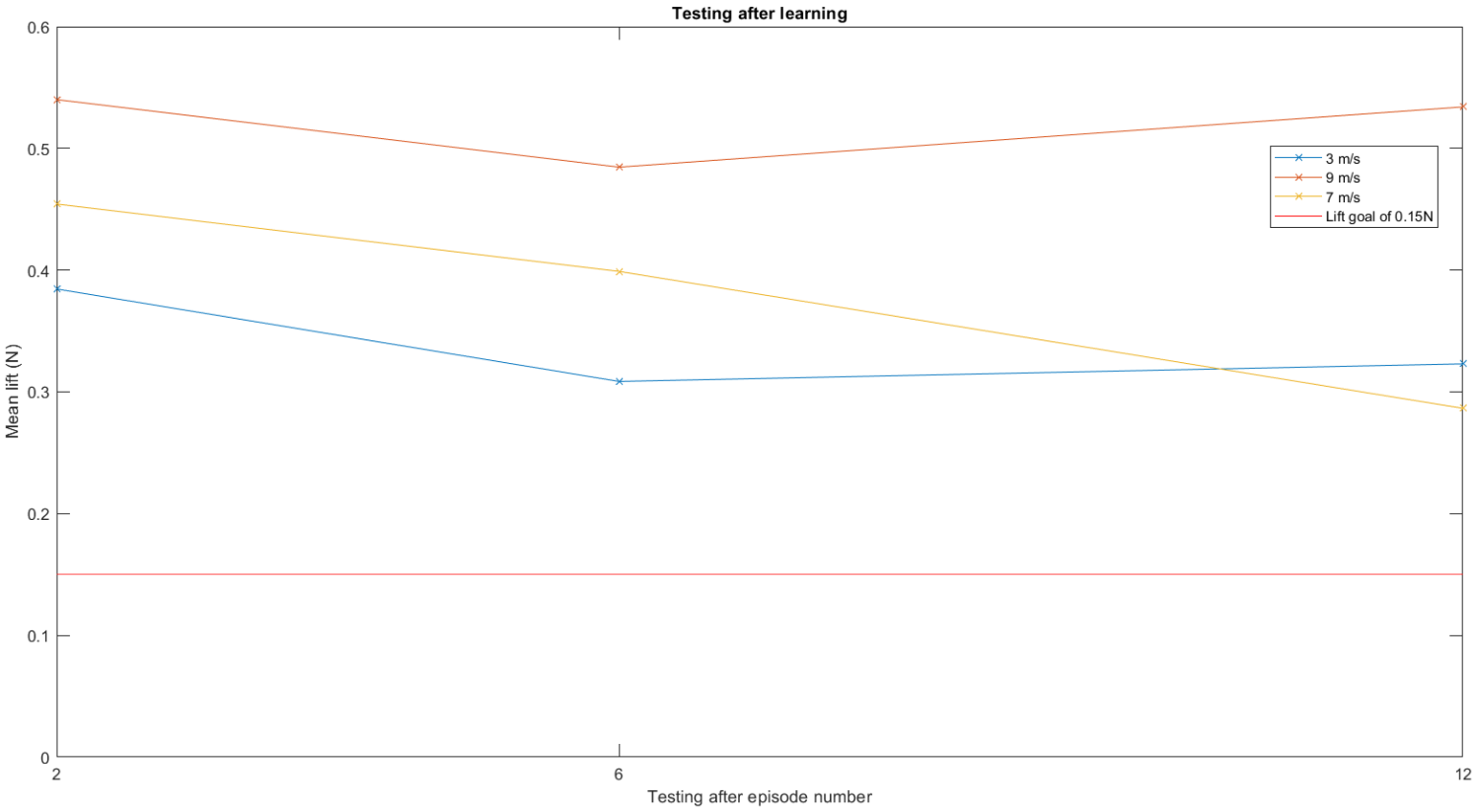


Figure 6.18: Testing of the system for lift

6.3.3 Experiment 3

In this section the results for experiment three are shown. Experiment three trained the RL algorithm at three wind speeds: 3 ms^{-1}, 9 ms^{-1} and 11 ms^{-1}. During the tests the system was also exposed to a wind speed of 7 ms^{-1}, this represent the unknown wind speed. Two training session were performed. One for optimizing normal force and the other for optimizing lift.

Plot of the training

The training of the system for the optimal normal force and optimal lift can be seen in the Figures 6.19 and 6.20, respectively. In these figures the plots have the same axes as described in the section of experiment two (Section 6.3.2). The lines and areas also represent the same variables.

From the normal force optimization experiment in Figure 6.19 it can be seen that the system explores during the first three episodes. This becomes obvious from the large green area representing a large standard deviation in the average action. It can be further seen that the sensor observation varies with the changing set wind speed in the block shape of the yellow line. The blue line, representing the average reward per step at the given episode, increases and becomes more constant as the number of episodes rise. The standard deviation of the average reward becomes smaller as the reward reaches zero. At approximately time step 16000 a downward spike in the yellow line can be observed. Both the blue and green lines have a larger standard deviation than in the episodes before, most likely due to this spike.

In Figure 6.20 the training of the RL algorithm optimized for lift is shown. Similarly to the normal force training plot the first three episodes are exploration episodes, seen by the large

standard deviation surrounding the blue and green lines. These lines represent the average reward per step in a given episode and the average action in an episode. Unlike the normal force training plot the reward does not become increasingly higher nor constant. The average action has no clear trajectory.

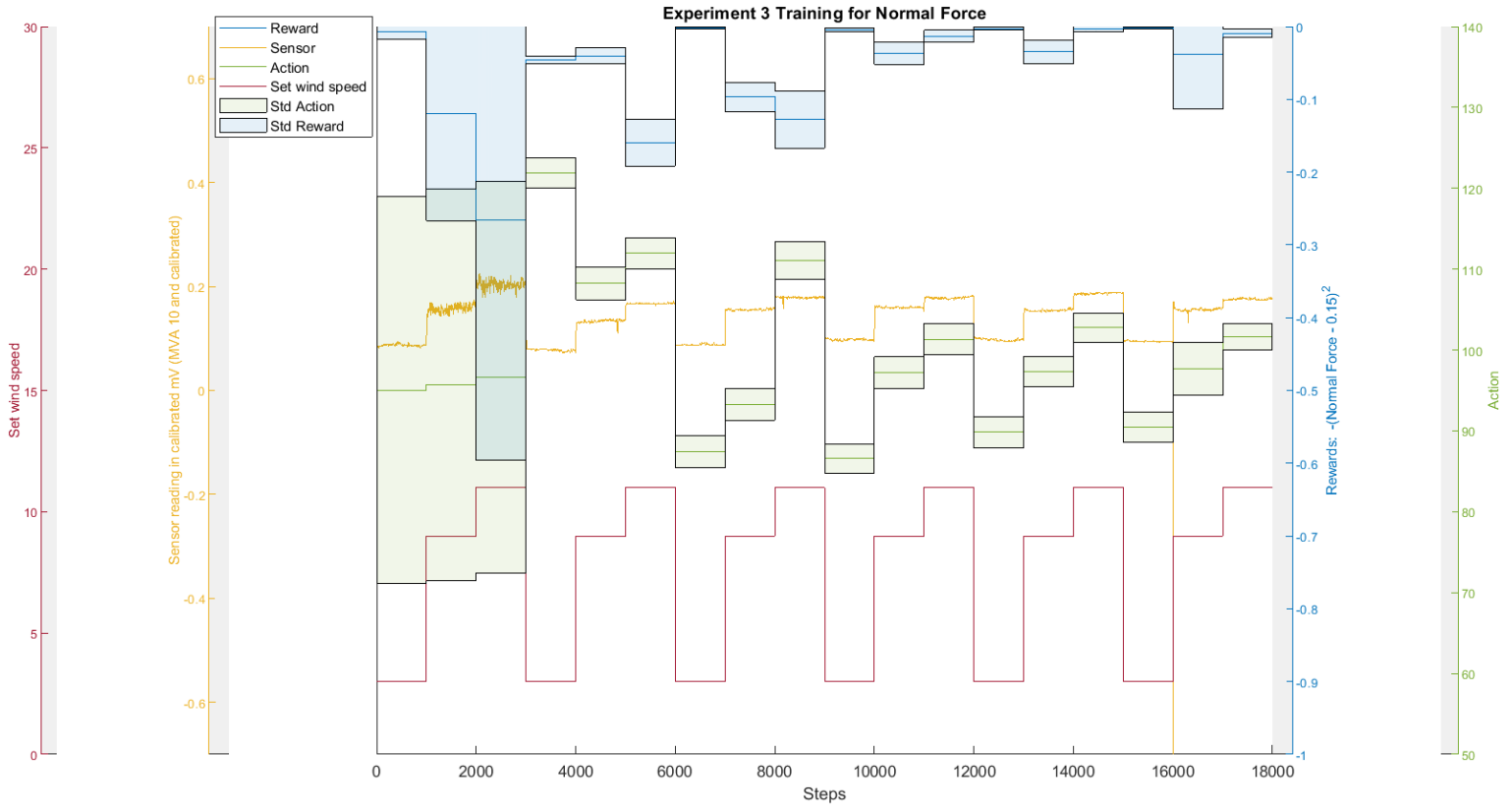


Figure 6.19: Training for Normal Force of 0.15, 3 wind speeds

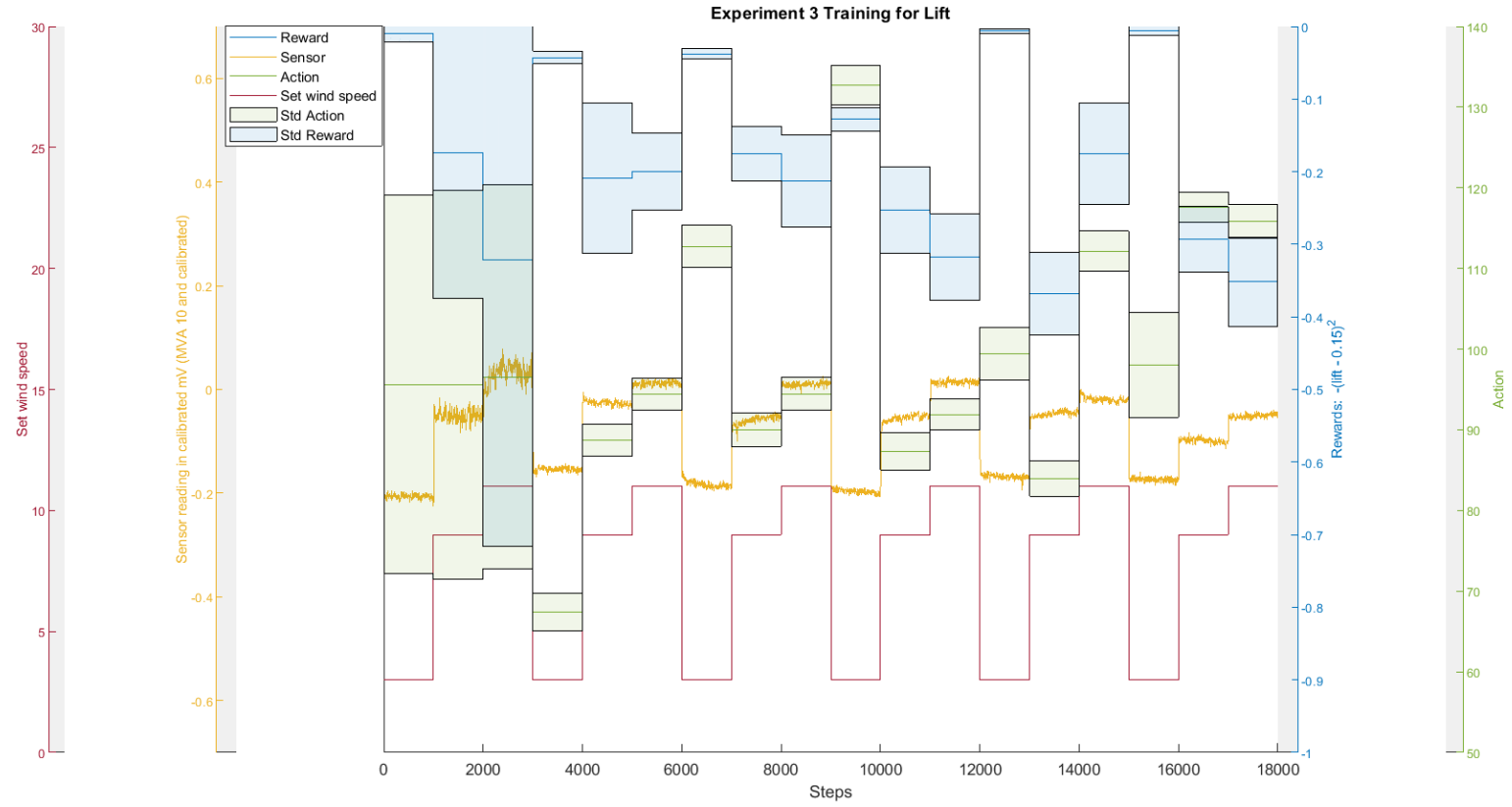


Figure 6.20: Training for Lift of 0.15, 3 wind speeds

Exploration noise free test

In this section the average lift of the exploration noise free tests for experiment three are shown. The tests were performed on the RL algorithm that trained on lift, the training can be seen in Figure 6.20. The plot which shows the results of testing the system for lift, is shown in the Figure 6.21. The test were performed after episodes 2, 6, 12 and 18 presented by the x-axis. In the plot the y-axis represent the average lift in N over the duration of the test. Only the average lift was plotted, for clarity purposes, as it did not change during the tests at set wind speeds. The blue data represents the test performed at $3\,\mathrm{ms}^{-1}$. The orange data represents the test performed at $9\,\mathrm{ms}^{-1}$. The purple line represents the $11\,\mathrm{ms}^{-1}$. The yellow data represents the test performed at $7\,\mathrm{ms}^{-1}$, the so called unknown wind speed. The red line in the plot indicates the target lift value of 0.15 N.

After episode 2, the blue and yellow lines start on a downward trajectory towards the red target lift line. The blue line, representing the lift at $3\,\mathrm{ms}^{-1}$, intersects the red target lift line. After episode 12 the yellow, purple and orange lines move towards the red target lift line. The blue line slightly deviates from the red target lift line.

Eventually, only the blue line briefly reaches the target lift for all testing trajectories. When comparing all final lift tests, performed after episode 18, it shows that the yellow line representing the so called unknown wind speed ends furthest away from the target red line, while both the purple and orange lines drastically drop towards the red line.

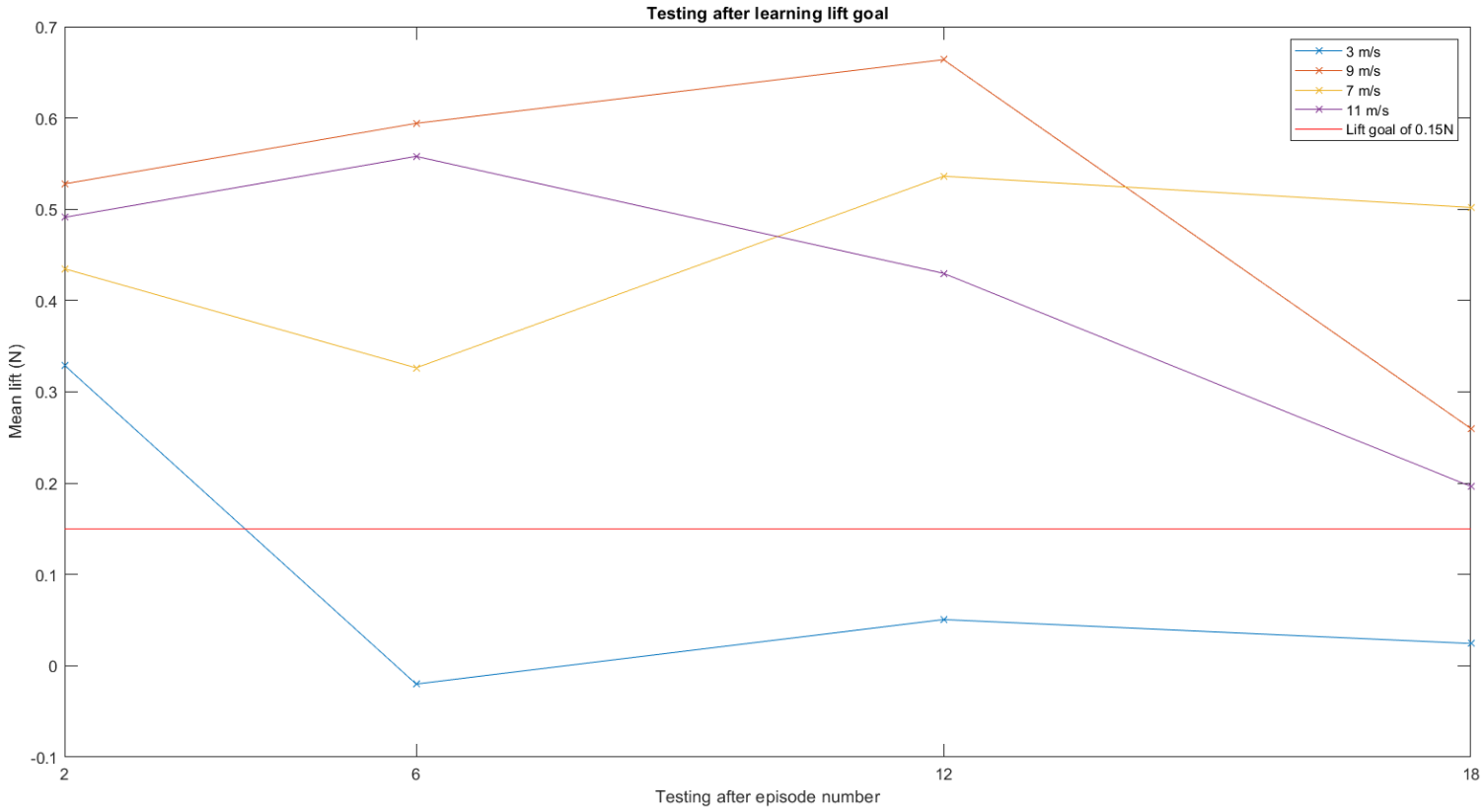


Figure 6.21: Testing of the system for lift

6.3.4 Sensitivity analysis

During the project multiple training runs were performed. Many of these runs showed that the system is sensitivity to unknown factors. In the first section a sudden measurement change is shown, the second section shows a dynamic wind speed test.

MVA drop

Below a graph is shown which showcase the effects of a small change in the system, in this case a power drop to the Wheatstone bridge, and how this effect affects the results. In this run of experiment two on normal force, after episode 19, the voltage supplied to the Wheatstone bridge suddenly dropped

The sudden drop can be seen in Figure 6.22 which shows the outputs of the resistor side and the sensor side of the Wheatstone bridge. The resistor side should be constant. The y-axis represents the normalized MVA observations measured in the Wheatstone bridge by the DFRobot. The x-axis represents the total steps. After time step $2 * 10^4$, which is the last step of episode 19, the normalized MVA for V2 drops from the constant nearly zero, to a sudden -0.25. The measurement over V1 in the Wheatstone bridge also showed that at step $2 * 10^4$ the normalized MVA drops from the steady fluctuation between -0.4 and -0.3, to a fluctuation between -0.6 and -0.5.

These two plotted lines show that the setup is sensitive to effects other than wind speed. The drop in MVA measurements in V2 and V1 rendered the results unusable and experiment two was run again.

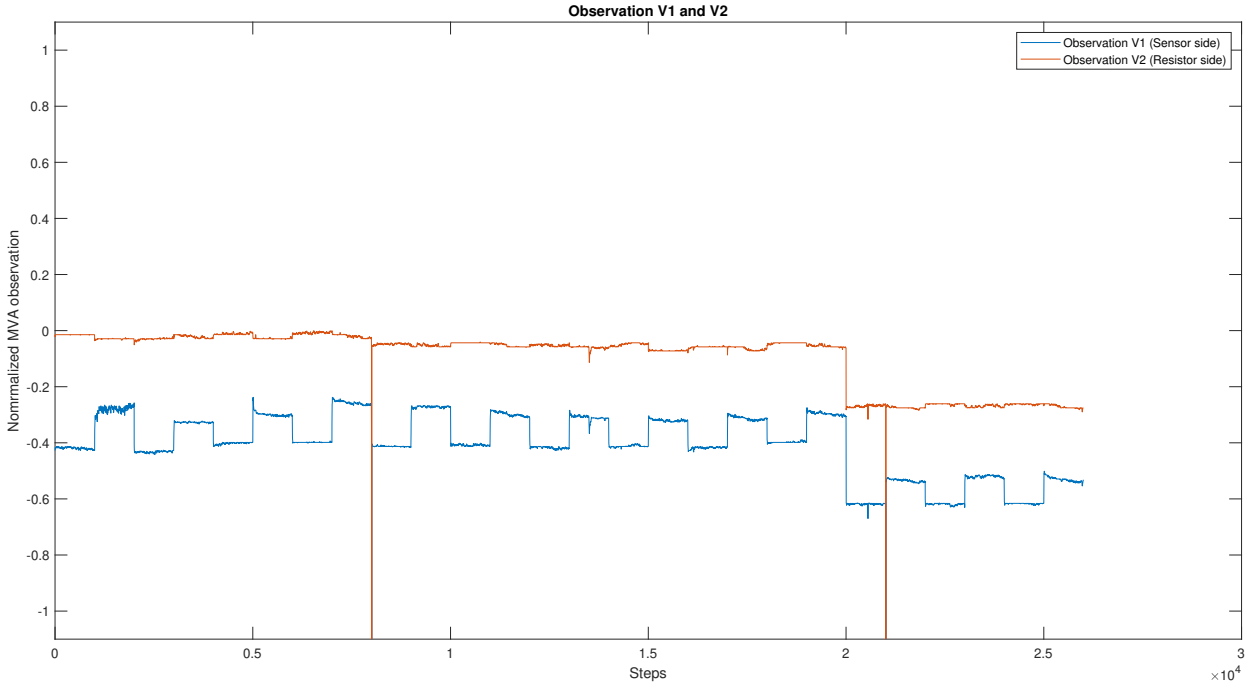


Figure 6.22: Observation of V1 and V2 with an unexpected drop

Dynamic wind speed

In all of the results above the wind speed was set for a whole episode. During the testing of the setup and RL algorithm a dynamic wind speed was tried. However, due to these changing wind speeds, the lift data could not be recorded. This was attributed to wind tunnel limitations. The plot in Figure 6.23 shows how the system behaves in a dynamic environment.

In Figure 6.23 the outputs of the normalized MVA observations at the sensor side, V1, and the resistor side, V2, of the Wheatstone bridge are plotted in blue and red, respectively. These outputs correspond to the left y-axis. The action taken by the system is plotted in yellow and corresponds to the right y-axis. The x-axis presents the time steps with steps taken at a 10 Hz interval. The wind speed started at 11 ms^{-1}, moved to 3 ms^{-1}, followed by 9 ms^{-1}, then wind speed zero and finally back up to 9 ms^{-1}. The dynamic wind speed turning points can be followed by the peaks and valleys in the V1 MVA in Figure 6.23. From the action plot in the graph it is seen that the action changes as the V1 MVA also changes. With the exception of the first wind speed, the actions taken increase as the wind speed goes down and decrease when the wind speed goes up. This showcases the reaction of the system to these dynamic wind speeds performed in one test.

6.4 Conclusion

This chapter has shown the results of the experiment that were performed. The key sections in the multiple graphs were also highlighted. It can be concluded that each experiment served a purpose and produced differing results. The baselines were established and traditional aerodynamic analyses gives insights into the forces upon the airfoil. The analysis of the found results from the simulations and experiments are given in the next chapter.

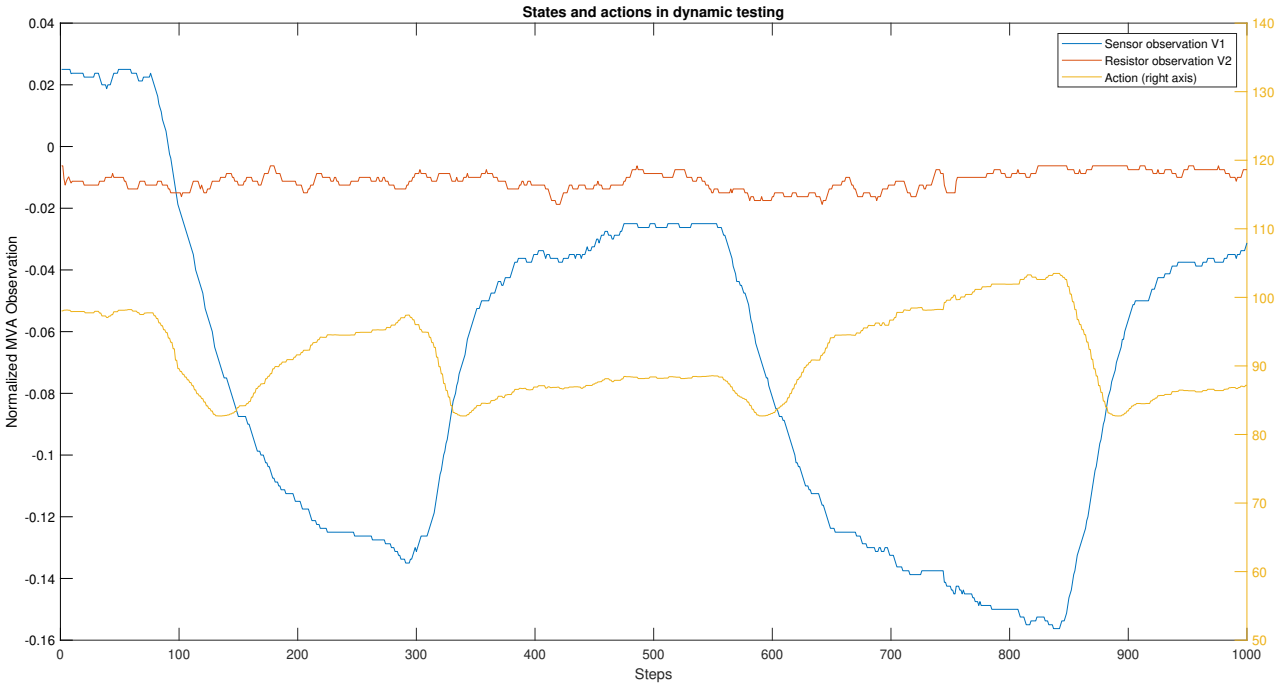


Figure 6.23: Observation and actions of the dynamic test

7 Discussion

In this chapter the results are discussed in more detail. The conclusion is found and the interpretations of the results are given. Furthermore, the drawbacks of this project and future research are discussed.

7.1 Answering the RQ

In the introduction the research question was given, which is as follows:

1. "How can a 3D printed flow sensor in combination with RL be used to actively control the flight control surfaces of an airfoil in a real-world setup such that the airfoil reaches specific predetermined lift values?"

This project performed three experiments with an airfoil combined with a 3D printed flow sensor and flaps controlled by an RL learning algorithm in an EWT to answer the research question and the sub research questions. The two sub research questions were formulated to assist in answering the main research question.

1. To what degree can the 3D printed flow sensor detect a change in flow over the airfoil due to a change in wind speed or flight control surfaces?

2. To what degree can the RL algorithm distinguish different readings from the 3D printed flow sensor?

Sub research question one can be answered with the baseline experiments. Here, the 3D printed flow sensor showcased the possibility of detecting the different wind speeds and the effects of the flaps on the airflow over the airfoil. Hysteresis was however observed in the sensor readings. Therefore, the answer to sub research question one is that a 3D printed flow sensor can detect a change in flow over the airfoil, however the accuracy can be called into question due to hysteresis.

The second sub research question can be answered with experiment two and three. With the readings of the 3D printed flow sensor the RL algorithm showed that it is capable of changing it's chosen action when the readings also changed. In the figures the action of the RL algorithm can be followed in response to the environmental inputs it receives from the 3D printed flow sensor. Therefore, the answer to sub research question two is that the RL algorithm can distinguish different readings, as it responds differently with different received observations.

Finally, the research question can be answered with experiments two and three as well. Namely, it was found that the RL algorithm can actuate the flaps in such a way that the predetermined normal force target is met during training. However, the algorithm has difficulties with finding the optimal actions for a predetermined lift target at known and unknown wind speeds. Therefore, a 3D printed sensor in combination with an RL algorithm has potential to be used to control the flight control surfaces of an airfoil to optimize the lift values. However, at this stage and setup of this project, the results show that it does not work as well as expected.

7.2 Looking at the results

In the following section a more detailed observation of the results is given. First the baseline values are discussed, followed by experiments one through three, finally a short discussion of the sensitivity analysis follows.

7.2.1 Baselines values

First discussed are the baseline values. The reward baselines are based on the forces exerted on to the airfoil, which are measured by the stinger in the EWT. The observation baselines are measured by the DFRobot at V1 and V2 of the Wheatstone bridge.

Reward baselines

The baseline value results are presented in Figure 6.1 for normal force, 6.2 for axial force, 6.3 for lift force and 6.4 for drag force, as measured by the stinger and calculated from the stinger measurements at different wind speeds. The reward for the RL algorithm is based on the forces measured by the EWT stinger. The effects of the flap position can clearly be seen through the block step like shape of the coloured lines, which correspond with the changing of the force when the position of the flap changes. At this stinger angle the lift force is mostly made up out of normal force, and the drag force is mostly made up out of the axial force. Therefore, the graphs for the respective forces are similar. These base line values are relevant as they show the expected force at a given flap position and wind speed. Furthermore, they showcase the range of force upon the airfoil. This was used to determine a target lift in experiments two and three. The maximum lift can be observed at flap position 50 and the maximum drag can be observed at flap position 150, regardless of wind speed. It can also be seen that at flap position 100 the lift is the same for all wind speeds. From this it can be assumed that at this flap position the lift coefficient C_L, seen in Equation 2.4, is close to 0. This is further supported by the low lift values found for all wind speeds at this point.

Looking at the top right corner of Figure 6.1 the green line representing $11\,\mathrm{ms}^{-1}$ and the purple line representing $9\,\mathrm{ms}^{-1}$ are moving upwards. At time step 3500 at the beginning of the block step movement for each of the coloured lines, a small corner overlaps. This is due to the manual starting of the measurement of the EWT. The start times differ in between experiments, this is why a horizontal shift can be observed.

Observation baselines

In Figure 6.5 the output of the Wheatstone bridge, measured by the DFRobot at point V1, is shown in mV. This figure showcases that the resolution of the DFRobot is still lacking, as the values measured for a given wind speed and flap position oscillates between two integer values. Higher resolution measurement systems would find a value in between these larger DFRobot measurements. Therefore, Figure 6.6 shows the moving average of the DFRrobot measurement of the Wheatstone bridge sensor side, V1. This was done to increase the resolution of the measurement. The graph shows that the 3D printed flow sensor can measure the effects of the flap position on the airflow around the airfoil. Furthermore, it can also distinguish between different wind speeds. This shows that the RL algorithm has the potential to use these measurements to accurately actuate the flaps to match a set normal force or lift. Figure 6.7 shows the output of the Wheatstone bridge measured by the DFRobot at point V2 in mV. The measurements are nearly constant for each wind speed and flap positions. Therefore, using only this measurement would have been unusable for the RL algorithm. Knowledge from this graph is used to see if external factors influenced the measurements.

Hysteresis

Dijkshoorn et al. (2021) found that the 3D printed flow sensor showed hysteresis. This was also tested for a flow sensor in an airfoil set up, the results are shown in Figure 6.8 and 6.9. From the Figure 6.9 it can be seen that the sensor exhibits hysteresis. This was an extra challenge for the RL algorithm. However, to exclude the possibility of the airflow inducing the hysteresis observed in the sensor plot, the hysteresis of the lift is also checked, because the sensor observation is directly influenced by the airflow over the airfoil. In Figure 6.8 it can be observed that

the lift shows no hysteresis. The lift is also directly influenced by the airflow over the airfoil. Therefore the airflow is not the cause of the hysteresis seen in the sensor plot.

7.2.2 Analyses forces

In Figure 6.10 it can be seen that for all wind speeds hysteresis effect is present in the sensor. For the same lift value at a given wind speed different sensor values are observed. This hysteresis effect can increase the difficulty for the RL algorithm to find the desired action.

Figure 6.11 shows the lift versus drag at given wind speeds during the baseline experiments. This plot shows the aerodynamic properties of the tested airfoil. This plot visualizes the relationship between lift and drag, this can show the positions which are optimal for particular actions. For example, landing would require low lift and high drag, therefore a flap position which corresponds to the right most points of the curve, is desired for this action. This graph can also be used to compare air foil designs. If this test would have been performed with the airfoil without the sensor, a comparison can be made between the graphs to see the effects the sensor might have on the aerodynamic properties of the airfoil. From the figure it looks like the $9\,ms^{-1}$ data is a slight outlier. It is not clear if this is due to a measurement issue or if this is the actual behavior of the airfoil in the wind tunnel.

7.2.3 Simulations

From Figure 6.12 it can be seen that the RL algorithm can find the optimal flap position of 50. This is already found after two episodes of exploration. The results of the other simulation, seen in Figure 6.13, show that the RL algorithm can also work with finding the optimal flap position for reaching a certain lift in the simulation. It took four episodes to find the two flap positions for the different observation values that were used as input. The average flap positions over the other episodes did fluctuate slightly but are still very close to the optimal, as seen by the average reward. It can also be seen that the system does not always improve with more training, because at episode 12 the average action taken was suddenly higher than the previous actions for that observation level. The actions for this observation level did however return to more optimal values in the next episode. From this it is conclude that the RL algorithm can perform in the simulated environment. This also gives an indication that the RL algorithm has the possibility to perform in a real experimental setup. However, the simulated environment is only a simplification of the real experiments and the real word experiments will come with substantially more challenges for the RL algorithm than the simulated environment.

7.2.4 Experiment 1

Training

Figure 6.14 shows the actions taken at different wind speeds, observation of V1 and the given rewards. The actions taken, represented by the green line, are not as expected. The expectation was that the action would reach a flap position of 50, as this was found to be the flap position that provides maximum lift regardless of wind speed, as discussed previously. Especially the first action taken after the two exploration episodes, was the exact opposite of the desired action. After this the action followed a downwards trajectory for the $3\,ms^{-1}$ wind speed episodes. The actions taken for the $9\,ms^{-1}$ wind speed, after the exploration episodes, were close to the optimal action (at approximately a flap position of 60). However, it did not improve in the following episode.

Exploration noise free test

To test the system the noise free tests were performed without exploration noise. These tests were performed to assess how the system behaves for known and unknown wind speeds.The

lifts from the exploration noise free test, at each tested wind speed, is lower than the maximum lift values found in the baseline values for that wind speed. Particularly with the $3\,ms^{-1}$ wind speed tests the lift did not even reach positive values. This indicates that the system has trouble finding the optimal flap position for maximum lift. However, as the system was only trained for six episodes, this conclusion is drawn on a small training set without giving the RL algorithm a lot of opportunities to learn and therefore improve. The RL algorithm did however find the optimal flap position in less episodes in the simulated environment.

7.2.5 Experiment 2

In this section the results of experiment two are discussed. The RL algorithm is trained at two wind speeds and then exposed to an unknown wind speed in the noise free tests.

Training normal force and lift

In Figure 6.16 and Figure 6.17 the training of the RL algorithm at wind speed $3\,ms^{-1}$ and $9\,ms^{-1}$ is shown. In the first two episodes the algorithm explores the environment. Figure 6.16 shows that the system, trained on a normal force goal, is still changing the actions it takes in regards to a set wind speed until episode 10. Thereafter a more stable action and reward is observed. During these actions it can be seen that the reward is close to zero, indicating that the system reaches a lift value near the set lift goal of 0.15 N for different wind speeds. This is exactly what the system was trained to do. While the performance does not reach the levels of the system in the simulated environment, it does show that with a few extra episodes it approaches the performance of the previous mentioned system. At episode 18 it can be seen that the system starts to perform comparatively worse for the episode at $3\,ms^{-1}$.

The system is also trained on lift, as shown in Figure 6.17. After the exploration episodes the system has a low reward caused by its inaccurate actions to reach the desired lift value. Particularly for episodes at wind speeds of $3\,ms^{-1}$ the system has a low standard deviation in the reward which remains constant. This low standard deviation, especially compared to episodes with $9\,ms^{-1}$ wind speeds, can be attributed to the quadratic nature of the lift. The lift changes less at lower wind speeds. When comparing this graph to the graph for normal force, it becomes clear that the system has more trouble with reaching the optimal lift versus reaching the optimal normal force. This possibly could be attributed to variations in the manually performed steps or to the more complex function behind lift, as it is a combination of normal and axial force. It also can be conclude that the RL algorithm has more difficulty in the real lift experiments than in the simulated environment.

Exploration noise free test

The noise free tests were performed to asses the system's ability to reach the desired lift value without exploration noise. The system does not reach the set lift for any of the tested wind speeds in any of the tests performed. This is shown by the coloured trend lines, representing the tests performed for known and unknown wind speed, not reaching the red target lift line in the plot of Figure 6.18. Furthermore, no clear trends can be seen for the known speeds which can indicate that more learning will not lead to a better result. Opposite to the expectation, the systems lift at the unknown wind speed is marginally closer to the set lift. It can be concluded from this graph, and the previous two training graphs, that the system has difficulties reaching a set lift value when trained on only two wind speeds. This is the case for the known and unknown wind speeds.

7.2.6 Experiment 3

In this section the results of experiment three are discussed. In this experiment the RL algorithm is trained on three wind speeds and then exposed to an unknown wind speed in the noise free tests.

Training normal force and lift

Figure 6.19 and Figure 6.20 shows the training data of the RL algorithm trained at three different wind speeds. Exploration was done in the first three episodes. From the plot in Figure 6.19 it can be seen that the actions gradually become constant for each set wind speed and the reward becomes increasingly smaller. However, the rewards for the $9\,ms^{-1}$ wind speed does not converge towards zero, unlike the rewards for the other wind speed episodes do. It is apparent that even for the wind speed of $11\,ms^{-1}$ the standard deviation of the reward becomes smaller in the later episodes of the training. This can possibly be attributed to both the parabolic reward function and that the average reward is close to zero, which is the maximum possible reward for the system.

In Figure 6.20 the reward and actions for each wind speed fluctuates more than in the normal force graph seen in Figure 6.19. Again, this can possibly be attributed to variations in the manually performed steps or the more complicated function behind lift. The $3\,ms^{-1}$ training comes close to the maximum reward of zero. Both wind speed $9\,ms^{-1}$ and $11\,ms^{-1}$ training episodes do not have a trajectory towards maximizing the reward. From this it can be concluded that the system has difficulties finding the set lift value for higher wind speeds. This can possibly be due to the observations of the sensor at the set wind speeds of $9\,ms^{-1}$ and $11\,ms^{-1}$ which values are close together. Certain action and wind speed combinations, at different wind speeds, receive almost the same observations. Therefore it can be possibly more difficult for the agent to distinguish between the two different wind speeds.

Exploration noise free test

In Experiment 3 the noise free tests are performed to asses how the system behaves without exploration noise at different wind speeds. The results are seen in Figure 6.21. Similarly to experiment two, none of the tests performed at different wind speeds reached the set target lift value. After the initial fluctuation, the $3\,ms^{-1}$ wind speed test does not approach the set lift value. Surprisingly, the $9\,ms^{-1}$ and $11\,ms^{-1}$ wind speeds tests come closest to the set lift value of 0.15 N, in the final test performed after episode 18. During the training the rewards for both these wind speeds did not reach the maximum possible reward, indicating that the system did not reach the desired lift for these wind speeds. The actions taken in the last test episodes are not similar to the actions taken during the last training episodes. This could indicate that more learning has the possibility to be beneficial to the system.

While the system does not match the lift for the known and unknown wind speeds, after being trained on three wind speeds, it performs better on the known wind speeds than when only trained on two wind speeds. Therefore, it can be concluded that training on more wind speeds leads to a slightly better performance of the RL algorithm in the last exploration noise free tests. However, in the case of training with three wind speeds the system is trained for 18 episodes as compared to only 12 episodes for two wind speeds. While both systems were exposed to each wind speed for six episodes the system trained for the three wind speeds was trained longer, due to fact that it was exposed to an extra wind speed. Therefore, the conclusion made earlier cannot be solely attributed to the use of more wind speeds whilst training, because the total amount of training for the RL algorithms differed.

7.2.7 Sensitivity analysis

This section discusses the results of the training runs which show the sensitivity of the system. First the drop in the measurements of the Wheatstone bridge is discussed and then the dynamic wind speed is addressed.

MVA drop

In Figure 6.22 it can be seen that there are multiple spikes in the observation. This indicates an unknown fluctuation possibly caused by a measurement error. The drop in the normalized MVA voltage over V1 and V2 can also be seen in Figure 6.22. This drop could be caused by a power supply issue to the Wheatstone bridge. The power supply issue could be caused by the Arduino or the laptops fluctuating power supply to the USB output. This goes to show that the system and setup is sensitive to external factors. Consequently, many training episodes and tests were performed multiple times.

Dynamic wind speed

A test was performed to assess the response of the setup and the system whilst exposed to dynamic wind speeds. Due to limitations of the EWT no lift data could be recorded while performing this dynamic wind speed test. From Figure 6.23 it becomes clear that the sensor observation also changes with the dynamic wind speed, and that the resistor observation remained constant as expected.

Also from Figure 6.23 it becomes clear that the RL algorithm can respond to the fluctuating observations caused by the dynamic wind speed during the test. There is potential to train the system with dynamic wind speeds, if a set up which allows for data acquisition during a dynamic test is available. Hysteresis can be present in the aerodynamics when using dynamic wind speeds. This would have to be characterized before training a system on dynamic wind speeds.

7.3 Drawbacks and limitations

As with any research, this project has its drawbacks and limitations. In this section these will be highlighted and discussed in more detail.

7.3.1 Simulations

The simulation environment, used to test the RL algorithm, has several limitations. The set observation values do not change with the change in flap position. This happens in real experiments as seen from Figure 6.6, which was not taken into account in the simulated environment. The added noise to the set observation values might slightly compensate this issue while also simulating that the wind speed provided in the real experiments are not completely constant.

Another issue with the set value is that the hysteresis effect, seen in the sensor, was not taken into account. Furthermore, the linear change in the sensor readout in an episode, seen in Figure 6.16 and Figure 6.14, were not taken into account.

The delay of the system and the averaging of the reward over one second were compensated by giving every 10 data points the same reward. This is based on the average lift of the previous 10 data points. This compensation method does not entirely cover the limitations of the real world setup.

7.3.2 TD3

The project uses a TD3 algorithm for its RL algorithm. While implementing this algorithm some challenges were encountered due to the setup that was used in this project. While the

Arduino sends a constant stream of data to the laptop with the RL algorithm it is not always received correctly. It has been observed that parts of a measurement were lost. This can include the complete V1 measurement or a single value of the V1 measurement. Sometimes the V2 measurements were also lost. This issue was combated by using the previously received measurement as a substitute. This solution does work, however it is not ideal. In particular, when multiple sequential measurements are completely lost or partially lost. In cases where there are fluctuations or a trend the substitute measurement will not always show these differences.

A major limitation for the drawn conclusion of this project is the low amount of training and testing runs performed in the experiments. This can partially be attributed to the manual work necessary for this setup, which is discussed below. Another cause is the time constraints, as the design and the setup phases of this project were extensive. The low amount of runs leads to less refined results and fewer robust conclusions. However, the low amount of episodes per run was enough for the simulation tests. This could indicate that the limitations of the experimental setup and its proposed solutions can possibly have a big affect on the training of the RL algorithm.

7.3.3 Timing limitations

The set up required many adaptions in regards to the use of time. These adaptions are a drawback of this project. The main reason time is important, is to synchronize the measurements from the EWT to the rewards of the RL algorithm. However, the EWT PC required manual setting of its time, therefore the synchronization of the data points could have been shifted. In the current setup the synchronization is performed via time stamps.

As the EWT records data by the second, and not in milliseconds, the reward is averaged over a whole second. This has the consequence that some of the rewards received by the RL algorithm at the start of a new second contains information about future actions. The reason this is possible, is not time travel, but because the rewards are synchronized after the training. The synchronization happened afterwards, because the rewards needed to be transferred via a USB flash drive.

Another issue with the timing in this project was a structural unknown delay in the EWT executable while recording data. By estimating the delay the auto clicker time step was set to 0.07 seconds.This ensured that the data recording happened closer, but not exactly, to the desired frequency of 10 Hz. This solution worked for this project, but it is not recommended or ideal. A wind tunnel that allows for direct data read outs does not need an auto clicker to provided consistent data, hence solving all of the timing limitations of this project.

7.3.4 Manual work

The connection and synchronization between the EWT PC and the laptop with the RL algorithm was sub optimal. This meant certain steps had to be performed or started manually. Furthermore, there was also some manual work involved in normalization, as the sensors is pushed by hand.

Due to the fact that EWT PC is old, the battery which powers the internal clock is depleted. As a result, the EWT PC forgets the time and date if not plugged in. Therefore, for every measurement session the date and time had to be manually set. While date and minute setting is not an issue, seconds are more difficult to accurately set. Any manual step, including this one, is prone to human error and is therefore a limitation.

The EWT PC could not send data to the laptop with the RL algorithm. Therefore, the re-

ward data had to be transferred with an USB flash drive and manually entered into the correct folder. The lack of connection between the two computers resulted in adding another manual step to this set up, increasing the chance of error and time necessary per session.

To ensure that data on the EWT is recorded while the RL algorithm is performing actions, the auto clicker had to be started before the RL algorithm starts. Turning the auto clicker on was also done by hand. Even though a grace period is given by the RL algorithm human error can always cause a late start or a non start.

As the starting time of the auto clicker is manually determined, the data from the baseline values per wind speed have different amount of data points for the first flap position. This could lead to a horizontal shift as seen in the top right corner in Figure 6.3. This could make comparison between the different wind speeds more complex.

The normalization step of the sensor is performed at the start of each training run. The normalization of the sensor is performed by pushing the sensor as far down in the wind direction by hand. From the results it can be seen that the normalized sensor read out varies greatly between sessions. This indicates that the normalization step is not a robust process and is prone to human variation. For that reason a trained system in one session is not usable in another session. Within one session the drawbacks of the normalization process done by hand are limited.

After the airfoil is placed on the stinger of the EWT the stinger output is tared to zero. This step is also performed by hand. If this is not performed, the measurements include the weight of the airfoil and can therefore not be used in the RL algorithm. Once again, each manual step adds an extra layer of complexity creating a process prone to error.

7.3.5 Setup

This project uses a setup with many connecting parts, as seen in Figure 4.1. As mentioned above, the EWT PC is old and therefore many of the limitations can be attributed to the EWT setup.

The EWT control to keep the wind speed at the set value is not as constant as anticipated. A small variation around a set wind speed can be seen of approximately $0.15\,\mathrm{ms}^{-1}$. While this does not compromise the results, this creates an extra level of complexity for the RL algorithm. This fluctuation in wind speed can lead to a fluctuation in sensor observation. Therefore this can be considered a drawback of the setup with the EWT.

Another drawback is when starting the speed control for the EWT, the EWT can sometimes randomly turn the fan on to its maximum speed. The speed control enables a wind speed to be set to a desired value. This results in all the connective wiring to the servo motors to be disconnected abruptly. Due to this random action of the EWT initiation steps have to be re-performed including reconnecting the servo motors.

The manual taring of the stinger after the airfoil was attached showed fluctuations. Resulting in that the airfoil weight was considered different and therefore the zero tare value fluctuates between training runs. This makes comparisons based on optimal action less accurate as this can possibly differ between training sessions.

A Wheatstone bridge is usually measured by taking the difference between the voltages in point V1 and V2. In this case the voltage of V1 and V2 are measured and no differential mea-

surement is taken. This means that the resolution of the system is limited, as the DFRrobot cannot measure above a certain voltage.

Another drawback of the DFRobot is the low resolution of its measurements. This can be seen from Figure 6.5, as the measured mV values always jump between integer values. While a higher measurement resolution can provide a decimal value. To combat this low resolution a MVA was taken of ten measurements. This can come with some drawbacks for the system, as the effects of the action taken are diluted in the observations sent to the RL algorithm. Nonetheless the RL algorithm can deal with this in the simulated environment. This environment does however not capture all the complexities seen in the real environment.

As mentioned in the sensitivity analysis, and its corresponding discussion, the power supply to the Wheatstone bridge is a sensitive point in the set up. The Arduino does not always supply the required constant voltage. This can be attributed to the Arduino receiving unsteady power from the laptop or in the case that other components, which are also connected to the Arduino, require large amounts of power in a small time interval. This is also the reason that the servo motors are not powered by the Arduino, but are connected to the EWT PC via a micro controller. Even with this precaution the sensitivity analysis shows that the Wheatstone bridge remains a sensitive component.

This project uses a 3D printed flow sensor. After the numerous amount of experiments, the used flow sensor showed slight signs of plastic deformation. The long term effect of this deformation, known as creep, have not been measured in an experiment as one experimental run is too short to notice this effect. Therefore, it is not known how the RL algorithm handles these long term effects.

7.4 Future research

In this section the future research and possible additions to this research are discussed. Furthermore the contributions to the Portwings project are highlighted.

7.4.1 Combating limitations in the future

Many of the limitations can be attributed to the EWT and its connectivity to the laptop with the RL algorithm. This can possibly be solved by upgrading and updating the EWT to make it more compatible for data exportation, external automatic control and internal control. This might be possible by using LabVIEW (NI, 2021b) and NI MAX (NI, 2021a). The current EWT is from 2007 and runs on windows XP with a depleted clock battery, replacing the battery is recommended.

An alternative option would be to build an own wind tunnel and setup. This could also offer customization options, such as a dynamically changing angle of attack and millisecond observations.

Alternatively, the chosen RL algorithm could be changed. In experiment three certain action and wind speed combinations receive almost the same observations, making it possibly more difficult for the RL algorithm to distinguish between the two different wind speeds. An alternative RL algorithm could be more suited to deal with this concern. A recommended RL algorithm would be the Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient algorithm seen in Meng et al. (2021).

7.4.2 Future possibilities and integration

Once the short comings of the current setup have been addressed possible options can include adding multiple 3D printed sensors or changing the current flow sensor. Increasing the current amount of strain gauges from two to four, can be a possible change to the current 3D printed flow sensor. Four strain gauges would allow the sensor to be measured via a full Wheatstone bridge, which increases the sensitivity compared to a half bridge. Adding the extra strain gauges would likely change the size of the sensor, the effects of this will have to be characterized before a four strain gauge sensor can be used. An example of a 3D printed flow sensor with four strain gauges is shown in Appendix A.

The RL algorithm has a simple reward function though more complex reward functions are possible. These reward functions could be used to let the RL algorithm find optimal actions for different tasks. These optimal actions can allow the performance of more difficult tasks. These tasks could include periodic movements such as flapping, take-off, landing and optimizing efficiency. Especially now, in an world where environmental innovations are increasingly more important, energy efficient flight measured by 3D printed sensor and optimized by a RL algorithm, would be useful. 3D printed sensor have a bright future and can be implemented in many different settings (Califano et al., 2021).

Besides the more general future possibilities, this project was done for the PortWings project. The PortWings project aims to create a flying and flapping robotic bird. This project can be used for experimental validation of the ability of an RL algorithm in combination with a 3D printed flow sensor to control flaps to optimize lift. This is the first step towards flow control and flow analysis using 3D printed sensors.

7.5 Conclusion

In this chapter the results from the simulations, experiments and sensitivity analyses are discussed. Furthermore, the drawbacks and limitations, and future research is explained. It can be concluded that using these results the research question and sub research questions posted in the introduction can be answered. It is established in this chapter that combining the RL algorithm with a 3D printed flow sensor in an real world experimental set up requires more research before it can be successful.

8 Conclusion

This project aimed to answer the research question:

1. "How can a 3D printed flow sensor in combination with RL be used to actively control the flight control surfaces of an airfoil in a real-world setup such that the airfoil reaches specific predetermined lift values?"

The sub research questions are as followed.

1. To what degree can the 3D printed flow sensor detect a change in flow over the airfoil due to a change in wind speed or flight control surfaces?

2. To what degree can the RL algorithm distinguish different readings from the 3D printed flow sensor?

The research questions are answered in the discussion chapter, Chapter 7. It can be concluded from this previous chapter that the RL algorithm can actively control the chosen flight control surface of the airfoil in a real world setup based on inputs from a 3D printed flow sensor. However, it did not always reach the predetermined lift values. Furthermore, many limitations were addressed. Overall it can be concluded that with the current set up, the research has too many limitations to form a concrete scientific advice. The conclusion related to each design pillar is discussed in more detail below.

The design pillars: the 3D printed flow sensor, the airfoil, the RL algorithm and wind tunnel, each have a more elaborate conclusion below. The sensor exhibits some drawbacks, such as hysteresis, however it can clearly observe differences in wind speeds. The airfoil with flaps has shown to affect the aerodynamic properties and therefore can be used for the RL algorithm. The RL algorithm has the ability to find the optimal action for a given reward function, however is severely limited by the setup. This can be concluded from the simulations as the RL algorithm could reach its goal in the simplified setting. Finally, the wind tunnel setup was found to cause many limitations and is a major drawback in this project. Many of the experiments which did not reach the desired goals were affected by the wind tunnel limitations in some form. The main improvement point is the setup.

The active control leaves room for improvements as the results showed a lack of accuracy. The lack of accuracy was constant throughout the experiments and it can be concluded that it is due to fundamental flaws in the system. Especially in the case when optimizing the action for a set lift value when trained both on two and three wind speeds. Furthermore, the chosen positions for the flaps when faced with an unknown wind speed were less than optimal. The current setup used has many limitations, which reduces the robustness of the results and conclusions. Therefore from this project it can be concluded there is still a long way to go before the use of 3D printed sensors combined with RL can eventually be implemented in control processes and research. Its a promising technique, however not market ready.

A Appendix Sensors

A.1 Short circuited sensor

In Figure A.1 a 3D printed flow sensor is shown. This 3D printed flow sensor was printed without any edges of non-conductive material in the flexible middle section of the 3D printed flow sensor. In this sensor the strain gauges short circuited.

Figure A.1: Short circuited 3D printed flow sensor

A.2 Flow sensor with four strain gauges

In Figure A.2 a 3D printed flow sensor is shown that contains four strain gauges. This sensor allows for measurement to be taken using a full Wheatstone bridge.

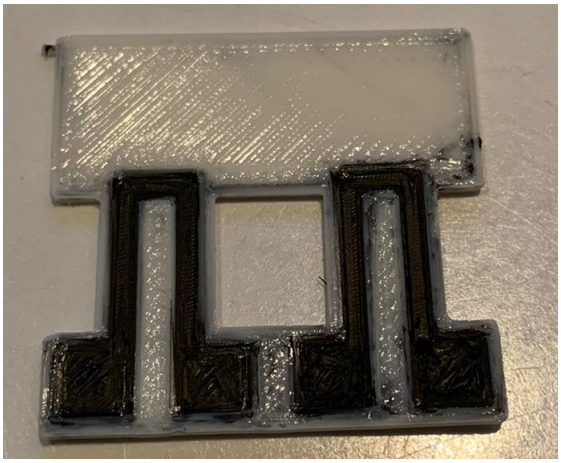


Figure A.2: Sensor with four strain gauges

Bibliography

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng (2015), TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org.
https://www.tensorflow.org/

Aerolab (2021), Educational wind tunnel.
https://www.aerolab.com/aerolab-products/educational-wind-tunnel-ewt/

AirfoilTools (2021), NACA 2412.
http://airfoiltools.com/airfoil/details?airfoil=naca2412-il

Anderson, J. (2016), *Fundamentals of Aerodynamics*, McGraw-Hill series in aeronautical and aerospace engineering, McGraw-Hill Education, ISBN 9781259251344.
https://books.google.nl/books?id=_7VLjwEACAAJ

Arduino (2021), Arduino mega 2560.
https://store.arduino.cc/products/arduino-mega-2560-rev3

ArduinoGetStarted (2021), Arduino-Servo Motor.
https://arduinogetstarted.com/tutorials/arduino-servo-motor

Atmel (Seen 2021), AVR121: Enhancing ADC resolution by oversampling.
http://ww1.microchip.com/downloads/en/appnotes/doc8003.pdf

Baldi, P. and K. Hornik (1989), Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima, **vol. 2**, no.1, p. 53–58, ISSN 0893-6080, doi:10.1016/0893-6080(89)90014-2.
https://doi.org/10.1016/0893-6080(89)90014-2

Bhatt, S. (2021), 5 Things You Need to Know about Reinforcement Learning.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba (2016), OpenAI Gym, *CoRR*, **vol. abs/1606.01540**.
http://arxiv.org/abs/1606.01540

Brunton, S. L., B. R. Noack and P. Koumoutsakos (2020), Machine Learning for Fluid Mechanics, in *Annual Review of Fluid Mechanics*, volume 52, Annual Reviews.
https://www.annualreviews.org/doi/pdf/10.1146/annurev-fluid-010719-060214

Bushkovskyi, O. (2021), 4 Types of Machine Learning Algorithms.
https://theappsolutions.com/blog/development/machine-learning-algorithm-types/

Califano, F., R. Rashad, A. Dijkshoorn, L. G. Koerkamp, R. Sneep, A. Brugnoli and S. Stramigioli (2021), Decoding and realising flapping flight with port-Hamiltonian system theory, *Annual Reviews in Control*, **vol. 51**, pp. 37–46, ISSN 1367-5788, doi:https://doi.org/10.1016/j.arcontrol.2021.03.009.
https://www.sciencedirect.com/science/article/pii/S1367578821000171

Cessna Aircraft Company (2021), Cessna 172.
https://web.archive.org/web/20110511100227/http:

//textron.vo.llnwd.net/o25/CES/cessna_aircraft_docs/single_engine/skyhawk/skyhawk_s%26d.pdf

Chris Liechti (2021), pySerial.
https://pythonhosted.org/pyserial/

Colin Cutler (2021), How Does Lowering Flaps Affect an Airplane's Angle of Attack (AOA)?
https://www.boldmethod.com/blog/2013/10/how-does-lowering-flaps-affect-angle-of-attack/

Cui, J. (2021), 3D printed piezo-resistive flow sensors for use on wings.
http://essay.utwente.nl/85947/

David Halliday, Robert Resnick, J. M. (1988), Fundamentals of Physics (3rd edition), p. 378.

DFRobot (2021a), Gravity: I2C ADS1115 16-Bit ADC Module.
https://www.dfrobot.com/product-1730.html

DFRobot (2021b), Gravity I2C ADS1115 16-Bit ADC Module Arduino.
$https://wiki.dfrobot.com/Gravity__I2C_ADS1115_16-Bit_ADC_Module_Arduino_%26_Raspberry_Pi_Compatible__SKU__DFR0553$

Diabase (2021), Diabase h-series multi-material 3d printer.
https://www.diabasemachines.com/

Dijkshoorn, A., J. Cui, S. Stramigioli and G. Krijnen (2021), First results of a Soft, 3D-Printed, Resistive Cantilever Flow Sensor, in *2021 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, IEEE, United States, doi:10.1109/FLEPS51544.2021.9469814.
https://2021.ieee-fleps.org/

Dijkshoorn, A., P. Werkman, M. Welleweerd, G. Wolterink, B. Eijking, J. Delamare, R. Sanders and G. J. M. Krijnen (2018), Embedded sensing: integrating sensors in 3-D printed structures, **vol. 7**, no.1, pp. 169–181, doi:10.5194/jsss-7-169-2018.
https://jsss.copernicus.org/articles/7/169/2018/

Eclipson (2021), MODEL T.
https://www.eclipson-airplanes.com/modelt

Fan, D., L. Yang, Z. Wang, M. S. Triantafyllou and G. E. Karniadakis (2020), Reinforcement learning for bluff body active flow control in experiments and simulations, **vol. 117**, no.42, pp. 26091–26098, ISSN 0027-8424, doi:10.1073/pnas.2004939117.
https://www.pnas.org/content/117/42/26091

Farnell (2021), PCI 6221.
https://nl.farnell.com/en-NL/ni/779066-01/multifunction-i-o-device-16bit/dp/3621347

Fujimoto, S., H. van Hoof and D. Meger (2018), Addressing Function Approximation Error in Actor-Critic Methods.

Gautam Reddy, Jerome Wong-Ng, A. C. T. J. S. M. V. (2018), Glider soaring via reinforcement learning in the field, **vol. 562**, no.1, pp. 236–239.
https://www.nature.com/articles/s41586-018-0533-0

Gautam Reddy, Antonio Celani, T. J. S. M. V. (2016), Learning to soar in turbulent environments, **vol. 113**, no.33, pp. E4877–E4884, doi:10.1073/pnas.1606075113.
https://www.pnas.org/content/113/33/E4877

Gazzola, M., A. A. Tchieu, D. Alexeev, A. de Brauer and P. Koumoutsakos (2016), Learning to school in the presence of hydrodynamic interactions, *Journal of Fluid Mechanics*, **vol. 789**, p. 726–749, ISSN 1469-7645, doi:10.1017/jfm.2015.686.
http://dx.doi.org/10.1017/jfm.2015.686

goldensoft (2021), GS Auto Clicker.
https://goldensoft.org/

Gudmundsson, S. (2014), General Aviation Aircraft Design Applied Methods and Procedures.

Guido Novati, Siddhartha Verma, D. A. D. R. W. M. v. R. P. K. (2017), Synchronisation through learning for two self-propelled swimmers, **vol. 12**, no.3, p. 036001, doi:10.1088/1748-3190/aa6311.
https://pubmed.ncbi.nlm.nih.gov/28355166/

HBM (2021), The Wheatstone Bridge Circuit.
https://www.hbm.com/en/7163/wheatstone-bridge-circuit/

javatpoint (2021), Difference between Artificial intelligence and Machine learning.

Kim, H., M. Jordan, S. Sastry and A. Ng (2004), Autonomous Helicopter Flight via Reinforcement Learning, in *Advances in Neural Information Processing Systems*, volume 16, Eds. S. Thrun, L. Saul and B. Schölkopf, MIT Press.
https://proceedings.neurips.cc/paper/2003/file/b427426b8acd2c2e53827970f2c2f526-Paper.pdf

Kim, M., D.-K. Han, J.-H. Park and J.-S. Kim (2020), Motion Planning of Robot Manipulators for a Smoother Path Using a Twin Delayed Deep Deterministic Policy Gradient with Hindsight Experience Replay, *Applied Sciences*, **vol. 10**, p. 575, doi:10.3390/app10020575.

Kosmas, D. (2020), Model-Based hysteresis compensation and control with 3D printed lousy sensors.
http://essay.utwente.nl/84814/

Leigh, S. J., R. J. Bradley, C. P. Purssell, D. R. Billson and D. A. Hutchins (2012), A Simple, Low-Cost Conductive Composite Material for 3D Printing of Electronic Sensors, **vol. 7**, no.11, doi:https://doi.org/10.1371/journal.pone.0049365.
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0049365

Lillicrap, T., J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra (2015), Continuous control with deep reinforcement learning, *CoRR*.

Ma, P., Y. Tian, Z. Pan, B. Ren and D. Manocha (2018), Fluid Directed Rigid Body Control Using Deep Reinforcement Learning, **vol. 37**, no.4, ISSN 0730-0301, doi:10.1145/3197517.3201334.
https://doi.org/10.1145/3197517.3201334

Magar, K. T., G. W. Reich, C. Kondash, K. Slinker, A. M. Pankonien, J. W. Baur and B. Smyers (2016), Aerodynamic parameters from distributed heterogeneous CNT hair sensors with a feedforward neural network, **vol. 11**, no.6, p. 066006, doi:10.1088/1748-3190/11/6/066006.
https://doi.org/10.1088/1748-3190/11/6/066006

Mark, A., Y.-H. Xu and B. T. Dickinson (2019), Review of Microscale Flow-Sensor-Enabled Mechanosensing in Small Unmanned Aerial Vehicles, *Journal of Aircraft*.

Meng, L., R. Gorbet and D. Kulic (2021), Memory-based Deep Reinforcement Learning for POMDP, *CoRR*, **vol. abs/2102.12344**.
https://arxiv.org/abs/2102.12344

Microsoft (2021), Microsoft Visual Studio.
https://visualstudio.microsoft.com/

Milano, M. and P. Koumoutsakos (2002), Neural Network Modeling for Near Wall Turbulent Flow, **vol. 182**, no.1, pp. 1–26, ISSN 0021-9991, doi:https://doi.org/10.1006/jcph.2002.7146.
https://www.sciencedirect.com/science/article/pii/S0021999102971469

Minsky, M. and S. A. Papert (1969), Perceptrons An Introduction to Computational Geometry, in *MA: MIT Press*, Cambridge.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller (2013), Playing Atari with Deep Reinforcement Learning, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.
http://arxiv.org/abs/1312.5602

NASA (2021), The lift Equation.
https://www.grc.nasa.gov/www/k-12/airplane/lifteq.html

National Instruments (2021), SCXI module.
https://www.ni.com/nl-nl/support/model.scxi-1000.html

NI (2021a), Installing NI Measurement Automation Explorer (MAX).

NI (2021b), LabVIEW.
359436

NinjaTek (2021), NinjaFlex 3D printer filament.
https://ninjatek.com/shop/ninjaflex/#tech-specs

NXP (2021), FRDM-K64F: Freedom Development Platform for Kinetis® K64, K63, and K24 MCUs.
https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F

OpenAI (2021), Twin Delayed DDPG.
$https://spinningup.openai.com/en/latest/algorithms/td3.html#background$

Ortiz, X., D. Rival and D. Wood (2015), Forces and Moments on Flat Plates of Small Aspect Ratio with Application to PV Wind Loads and Small Wind Turbine Blades, *Energies*, **vol. 8**, pp. 2438–2453, doi:10.3390/en8042438.

Palmiga Innovation (2021), Material info for PI-ETPU 95-250 Carbon Black.
https://rubber3dprinting.com/pi-etpu-95-250-carbon-black/

PortWings (2021), The PortWings project.
www.portwings.eu

Prakken, E. (2019), 3D Printed flow sensor.
http://essay.utwente.nl/77727/

Prusa 3D (2021), Prusa i3 MK3S.
https://www.prusa3d.nl/product/original-prusa-mk3/

Roi Harel, Olivier Duriez, O. S. J. F. N. H. W. M. G. W. B. F. S. O. H. R. N. (2016), Decision-making by a soaring bird: time, energy and risk considerations at different spatio-temporal scales, Phil. Trans. R. Soc. B3712015039720150397, doi:https://doi.org/10.1098/rstb.2015.0397.
https://royalsocietypublishing.org/doi/10.1098/rstb.2015.0397

Rosenblatt, F. (1958), The perceptron: a probabilistic model for information storage and organization in the brain, in *Psychol*, pp. 65:386–408.

Rumelhart, D., G. Hinton and R. Williams (1986), Learning representations by back-propagating errors, **vol. 323**, no.1, p. 533–536, doi:https://doi.org/10.1038/323533a0.
https://www.nature.com/articles/323533a0

Schouten, M., B. Prakken, R. Sanders and G. Krijnen (2019), Linearisation of a 3D printed flexible tactile sensor based on piezoresistive sensing, in *2019 IEEE SENSORS*, pp. 1–4, doi:10.1109/SENSORS43011.2019.8956652.

Schouten, M., G. Wolterink, A. Dijkshoorn, D. Kosmas, S. Stramigioli and G. Krijnen (2021), A Review of Extrusion-Based 3D Printing for the Fabrication of Electro- and Biomechanical Sensors, **vol. 21**, no.11, pp. 12900–12912, doi:10.1109/JSEN.2020.3042436.

Seo, D., Y. Kim and S. Kwon (2014), Micro Shear-Stress Sensor for Separation Detection During Flight of Unmanned Aerial Vehicles Using a Strain Gauge, **vol. 14**, no.4, pp. 1012–1019, doi:10.1109/JSEN.2013.2292338.

Shademan, M. and A. Naghib-Lahouti (2020), Effects of aspect ratio and inclination angle on aerodynamic loads of a flat plate, **vol. 14**, no.2, doi:https://doi.org/10.1186/s42774-020-00038-7.

Shemelya, C., L. Banuelos-Chacon, A. Melendez, C. Kief, D. Espalin, R. Wicker, G. Krijnen and E. MacDonald (2015), Multi-functional 3D printed and embedded sensors for satellite qualification structures, in *2015 IEEE SENSORS*, pp. 1–4, doi:10.1109/ICSENS.2015.7370541.

Smith, S. W. (1985), *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, ISBN 978-0966017632.
https://www.dspguide.com/

Sutton, R. S. and A. G. Barto (2018), *Reinforcement learning: An introduction*, MIT press.

Tesauro, G. (1992), Practical Issues in Temporal Difference Learning, **vol. 8**, no.3–4, p. 257–277, ISSN 0885-6125, doi:10.1007/BF00992697.
https://doi.org/10.1007/BF00992697

TowerPro (2021), SG90 Micro Servo 9g.
https://www.towerpro.com.tw/product/sg90-7/

Van Rossum, G. and F. L. Drake (2009), *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, ISBN 1441412697.

Verma, S., G. Novati and P. Koumoutsakos (2018), Efficient collective swimming by harnessing vortices through deep reinforcement learning.

Wikipedia contributors (2021a), Airfoil — Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Airfoil&oldid=1058708130, [Online; accessed 17-December-2021].

Wikipedia contributors (2021b), Lift (force) — Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Lift_(force)&oldid=1058873752, [Online; accessed 17-December-2021].

Xu, Y., X. Wu, X. Guo, B. Kong, M. Zhang, X. Qian, S. Mi and W. Sun (2017), The Boom in 3D-Printed Sensor Technology, **vol. 17**, no.5, ISSN 1424-8220, doi:10.3390/s17051166.
https://www.mdpi.com/1424-8220/17/5/1166