Optimizing the pulsed blowing parameters for active separation control with reinforcement learning

Technische Universität Berlin
Institute of Aeronautics and Atsronautics
Chair of Aerodynamics

Master Thesis

for obtaining the academic degree
Master of Science

submitted by

Alexandra Müller
[redacted]
Matriculation Number: [redacted]

on 10.02.2025

First Examiner:	Prof. Dr.-Ing. Julien Weiss
Second Examiner:	Dr.-Ing. Ben Steinfurth

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit eigenständig ohne Hilfe Dritter und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle Stellen die den benutzten Quellen und Hilfsmitteln unverändert oder sinngemäß entnommen sind, habe ich als solche kenntlich gemacht.

Sofern generative KI-Tools verwendet wurden, habe ich Produktnamen, Hersteller, die jeweils verwendete Softwareversion und die jeweiligen Einsatzzwecke (z.B. sprachliche Überprüfung und Verbesserung der Texte, systematische Recherche) benannt. Ich verantworte die Auswahl, die Übernahme und sämtliche Ergebnisse des von mir verwendeten KI-generierten Outputs vollumfänglich selbst.

Die Satzung zur Sicherung guter wissenschaftlicher Praxis an der TU Berlin vom 15. Februar 2023. `https://www.static.tu.berlin/fileadmin/www/10002457/K3-AMBl/Amtsblatt_2023/Amtliches_Mitteilungsblatt_Nr._16_vom_30.05.2023.pdf` habe ich zur Kenntnis genommen.

Ich erkläre weiterhin, dass ich die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Berlin, den 10.02.2025 Unterschrift: 

Declaration of Authorship

I hereby affirm that I have completed the present work independently, without the assistance of third parties, and solely using the listed sources and tools. I have appropriately identified all sections that are taken verbatim or in essence from the sources and tools used.

If generative AI tools were utilized, I have specified the product names, manufacturers, the respective software versions, and the purposes of their use (e.g., linguistic review and improvement of texts, systematic research). I take full responsibility for the selection, adoption, and all results of the AI-generated output I have used.

I have taken note of the *Satzung zur Sicherung guter wissenschaftlicher Praxis* at the TU Berlin dated February 15, 2023. `https://www.static.tu.berlin/fileadmin/www/10002457/K3-AMBl/Amtsblatt_2023/Amtliches_Mitteilungsblatt_Nr._16_vom_30.05.2023.pdf`.

Furthermore, I declare that I have not submitted this work, either in whole or in part, in the same or a similar form to any other examination authority.

Berlin, 10.02.2025 Signature: 

Technische Universität Berlin

Prof. Dr.-Ing. J. Weiss

FAKULTÄT V
VERKEHRS- UND
MASCHINENSYSTEME
Institut für Luft- und
Raumfahrt
Fachgebiet
Aerodynamik

MASTER-Arbeit

cand. M.Sc. Alexandra Müller,
Matrikelnummer [redacted]

Thema: Optimizing the pulsed blowing parameters for active separation control with reinforcement learning

The injection of pulsed jets into a boundary layer is well known to prevent or delay flow separation. The control authority of this method as well as its efficiency are governed by the operation parameters, especially the pulse duration and the time delay between successive pulses.

The objective pursued in this master thesis is to identify suitable parameters by setting up a reinforcement learning framework where the optimal conditions are found via an interaction between the algorithm and the wind tunnel experiment.

The following sub-tasks need to be completed:

- Literature review (active separation control, reinforcement learning)
- Implementation of reinforcement learning algorithm, including communication with control and measurement hardware
- Wind tunnel campaign
- Analysis of hyperparameters
- Oral presentation of results within *Luftfahrzeugbau-Kolloquium*

The master thesis will be supervised by Ben Steinfurth.

Berlin, 29.10.2024

Prof. Dr.-Ing. J. Weiss
head of Chair

Zusammenfassung

Strömungsablösung ist ein zentrales Phänomen in der Fluidmechanik und führt häufig zu negativen Effekten wie erhöhtem Widerstand oder Auftriebsverlust. Aktive Strömungskontrolle, insbesondere periodisch gepulstes Ausblasen, kann diese Ablösung reduzieren, indem sie der Grenzschicht zusätzlichen Impuls zuführt. In dieser Arbeit wird ein Deep Reinforcement Learning (DRL) Algorithmus, basierend auf Proximal Policy Optimization (PPO) verwendet, um die Parameter der aktiven Strömungskontrolle in einer voll turbulenten Diffusorströmung mit einer Reynolds-Zahl von $Re_\theta = 1000$ zu optimieren. Der Algorithmus steuert das Anregungssignal von fünf Pulsed Jet Aktuatoren am oberen Ende der Diffusorrampe, mit dem Ziel, die turbulente Ablöseblase zu verkleinern.

Sowohl modellbasierte als auch modellfreie Ansätze werden untersucht: Zunächst wird der Algorithmus mit interpolierten experimentellen Daten trainiert und anschließend direkt im Windkanal eingesetzt. Beide Methoden finden erfolgreiche Kontrollstrategien für einen festen Massenstromverbrauch ausgehend von einem festen Pulssignal am Anfang jeder Episode. Wie auch in anderen aktuellen Studien beschrieben, zeigt der Algorithmus, dass der Massenstrom am effizientesten genutzt wird, wenn das Anregungssignal eine geringe Pulszeit aufweist, also kurze Pulse im Verhältnis zur Periodendauer hat. Von den untersuchten Reward-Funktionen stellt sich die schrittweise Änderung der Strömungsumkehr pro Aktion als effizienteste heraus, da sie in weniger als 100 Episoden zu einer guten Strategie gelangt. Der Vergleich von Exploration und Exploitation zeigt, dass zu Beginn des Trainings Exploration notwendig ist, später hingegen ist Exploitation entscheidend für eine stabile Strategie.

Zusätzliche modellbasierte Trainings-Läufe mit zufälliger Initialisierung des Anregungssignals zeigen, dass der DRL-Algorithmus zwar erfolgreiche Kontrollstrategien entwickelt, aber eher einem bestimmten Gradienten vom Startpunkt zum Optimum folgt, anstatt die gesamte Trajektorie zu optimieren. Dies widerspricht dem eigentlichen Ziel. Als möglicher Lösungsansatz wird die Verwendung eines zusätzlichen Rewards pro Episode vorgeschlagen.

Diese Arbeit zeigt das Potenzial von DRL für die Echtzeitanpassung von Strömungskontrollstrategien in experimentellen Umgebungen auf und ebnet den Weg für zukünftige Anwendungen in der Luft- und Raumfahrt.

Abstract

Flow separation is a critical phenomenon in fluid mechanics, often leading to adverse effects such as increased drag, loss of lift, and aerodynamic inefficiencies. Active flow control techniques, particularly periodic pulsed blowing, offer an effective means to reduce separation by introducing additional momentum into the boundary layer. In this thesis, a Deep Reinforcement Learning (DRL) framework based on the Proximal Policy Optimization (PPO) algorithm is employed to optimize parameters for active flow control in a fully turbulent $Re_\theta = 1000$ diffuser flow. The algorithm varies the forcing signal of five Pulsed Jet Actuators located at the top of the diffuser ramp, with the primary objective of reducing the turbulent separation bubble within the diffuser.

Both model-based and model-free approaches are investigated: initially, the algorithm is trained on interpolated experimental data, and subsequently, it is deployed in direct interaction with the wind tunnel environment. Both methods develop successful active flow control strategies for a fixed mass flow consumption when starting from a fixed initial forcing signal. Fully consistent with recent studies, the algorithm suggests that the mass flow is used most efficiently when the actuation signal is characterized by a low duty cycle where the pulse duration is small compared to the pulsation period. Out of the investigated reward functions, the incremental reduction of flow reversal per action is proved to be the most sample efficient. In less than 100 episodes, a parameter combination is found that ensures a high control authority. The comparison between exploration and exploitation reveals that exploration is needed at the beginning of the training, whereas exploitation is required to follow a good strategy.

Additional model-based runs with a random initialization of the forcing signal reveal that although the DRL algorithm develops successful flow control strategies, it predominantly learns to follow an optimal gradient from the starting point to the optimum rather than optimizing the trajectory itself. This contradicts the overarching objective. As a solution, the introduction of an additional episode reward is suggested

The results of this thesis highlight the potential of DRL for real-time adaptation of flow control strategies in experimental settings, paving the way for future implementations in practical aerospace and engineering applications.

Contents

List of Figures

1. Introduction

Flow separation is one of the most undesirable phenomena in fluid mechanics due to its adverse effects, including increased drag and the potential for stall on an airfoil [5] or surge in a compressor [6]. Consequently, the reduction or control of flow separation remains a prominent research topic.

Separation control methods can be classified as either active or passive. Joshi et al. [7] provide a brief overview over both active and passive flow control techniques. Passive flow control devices aim to mix high momentum fluid to areas of low momentum fluid through changes in the geometry of the flow body. This can be achieved by either adding retrofits like turbulators or vortex generators to the body, by using rough or porous surfaces or by optimizing the existing design of the body. All passive methods are steady and don't require additional energy by definition. Active methods, on the contrary, aim to add or remove momentum to or from the boundary layer and thus require an additional energy source. There is a variety of techniques available. Among the oldest is steady suction and blowing, which dates back to the work of Prandtl in 1904 [8]. Prandtl successfully controlled the boundary layer around a cylinder using suction, thereby reducing flow separation. Another approach is periodic excitation, which offers enhanced control authority over the flow. Greenblatt and Wygnanski [9] provide a comprehensive overview of this method.

Periodic flow control problems are defined by several parameters. The momentum coefficient $C_\mu = (c_\mu, \langle c_\mu \rangle)$ represents the ratio of added momentum to free-stream momentum and therefore quantifies the relative momentum input from blowing. This coefficient can be divided into c_μ for the steady momentum and $\langle c_\mu \rangle$ for the oscillatory momentum. Another key parameter is the dimensionless frequency $F^+ = {}^{(f \cdot x)}/_{U_\infty}$, where f denotes the excitation frequency, x represents a reference length, and U_∞ the free-stream velocity. The efficiency of periodic excitation has been demonstrated by Seifert et al. [10], who showed that periodic blowing over a flap requires less momentum input compared to steady blowing.

Periodic excitation methods encompass a variety of approaches, including acoustic excitation [11], oscillating flaps [12, 13], and numerous variations of pulsed jets. In the present thesis, Pulsed Jet Actuators (PJAs) are utilized, similar to those employed by Steinfurth et al. [14] and Löffler et al. [15]. The periodic forcing of a PJA is defined by two timescales, the pulse duration t_p and the time delay between successive pulses t_off. The ratio between the pulse duration and the total period of the time signal is referred

to as the duty cycle $DC = {}^{t_\mathrm{p}}/_{(t_\mathrm{p}+t_\mathrm{off})}$.

Additionally to the previously mentioned parameters, flow control problems can also be dependent on other parameters, like the geometry. Identifying the optimal combination of parameters poses a substantial challenge, particularly when dealing with large parameter spaces. Several approaches exist for determining the most effective or optimal parameter combination. The simplest method involves conducting a full-factorial parameter variation, either experimentally or numerically, to locate the optimum. An example of this approach is provided by Steinfurth et al. [14]. However, this method becomes increasingly time-consuming as the dimensionality of the parameter space grows, making it necessary to use stochastic and Machine Learning (ML) methods.

A wide range of stochastic and ML-based techniques are available for active flow control optimization. Examples include the study on surrogate-based optimization by Löffler et al. [15], where the surrogate model is constructed using Gaussian process regression (or Kriging). Huang et al. [16] provide another approach with the optimization of the suction and blowing jet control on a NACA 0012 airfoil using a genetic algorithm with diversity control. Additionally, Montazer et al. [17] optimized a synthetic jet actuator using Response Surface Methodology, a combination of mathematical and statistical techniques that generates response surfaces for a set of design variables.

Recent advances in Deep Neural Networks (DNNs) have introduced new ML methods for optimizing active flow control parameters. Deep Reinforcement Learning (DRL) is particularly promising, as noted by Rabault et al. [18]. In this method an agent, which is typically represented by a DNN, selects an action that acts on an environment. In return, the environment provides a new state and reward to the agent, so that it can evaluate the effectiveness of the chosen action. Thus a DRL algorithm can learn empirical strategies through trial and error. DRL has already achieved success in diverse fields, including robotics [19], language processing [20], and games [21]. Its application to active flow control was first demonstrated by Rabault et al. [22]. They show that in a two-dimensional simulation, the wake of a cylinder flow can be stabilized and the drag reduced by training a DNN to vary the mass flow rate of two jets positioned on either side of the cylinder.

Subsequent studies have expanded upon this work. Ren et al. [23] investigated the same problem as Rabault et al. [22] but under weakly turbulent conditions. Fan et al. [24] identified DRL control strategies for turbulent cylinder flows using two fast-rotating smaller control cylinders located parallel to and downstream of the main cylinder. Tokarev et al. [25] achieved a drag reduction of $14\,\% - 16\,\%$ for a cylinder oscillating about its axis. Their study demonstrated that a DNN could develop a control strategy based on low-frequency harmonic oscillations to stabilize the Kármán vortex street at a low Reynolds number of $Re = 100$.

Other DRL applications for active flow control include the work of Shimomura et al. [26],

who used a Deep Q Network for closed-loop flow control to reduce flow separation over a NACA 0015 airfoil. Their approach achieved favorable results for an angle of attack of $\alpha = 15°$, where periodic activation of the actuator kept the flow attached for an extended period of time. Zheng et al. [27] developed an efficient active flow control strategy to suppress vortex-induced vibration of a cylinder at $Re = 100$ using a pair of jets positioned on the cylinder's poles. Their DRL framework reduced vibration amplitude by 82.7 %.

As shown, DRL provides a broad range of applications in optimizing active flow control parameters. This thesis focuses on optimizing the flow control parameters for a wind tunnel setup of a one-sided diffuser. This configuration was previously studied by Steinfurth et al. [14] and Löffler et al. [15]. Unlike many other studies, this setup involves a fully turbulent flow characterized by a Reynolds number of $Re_\theta = 1000$, based on a momentum thickness $\theta \approx 0.8\,\text{mm}$. The flow is actively controlled using five PJAs positioned at the top of the diffuser. DRL is employed to optimize the forcing parameters t_p and t_off, with the primary objective of reducing the turbulent separation bubble within the diffuser.

In DRL applications, various methods are available for policy updates. Viquerat et al. [28] provide an overview of the most commonly used DRL policies in fluid mechanics. In the present thesis, Proximal Policy Optimization (PPO) is selected for its sample efficiency, robustness, and ease of implementation. PPO was first introduced by Schulman et al. [29] and later used by Rabault et al. [22] in their pioneering work on applying DRL to active flow control.

In conclusion, this thesis focuses on optimizing the forcing parameters t_p and t_off of five PJAs to minimize the turbulent separation bubble within a one-sided diffuser. The novelty of this work lies in the application of a DRL algorithm utilizing PPO in conjunction with a fully turbulent flow, explored through both model-based and model-free approaches.

In DRL, an agent determines an action that interacts with an environment, which in return provides a new state and reward to the agent. In this thesis, the environment is the flow within the one-sided diffuser. The state is determined by wall-shear stress measurements along the diffuser, the reward is quantified by the suppression of reverse flow and the actions correspond to incremental changes of t_p and t_off. The theoretical foundation necessary to understand the DRL algorithm is presented in chapter 2.

To reduce wall time during training and to gain insight into the influence of periodic forcing parameters on the flow field, initial wind tunnel measurements are conducted for a set of predefined values of t_p and t_off. Thereafter, the DRL model is trained outside the wind tunnel environment by interpolating between the measurement data, thus making it a model-based approach. This research examines the impact of three distinct reward functions, as well as the role of exploration and exploitation, which are controlled through different standard deviations. This segment of the thesis is formatted as a paper and was presented at the DGLR STAB-Symposium in Regensburg (November 13 - November 14

2024). The paper, included in chapter 3, also provides a comprehensive description of the experimental setup and the application of the DRL on the studied problem.

The model-based training results are validated through additional wind tunnel experiments. Two DRL models are tested: one trained exclusively within the wind tunnel environment and another pre-trained on the interpolated data. The direct interaction with the wind tunnel environment in these experiments classifies both algorithms as model-free approaches. The results of these experiments and further investigations are described in chapter 4. The thesis concludes with a discussion of findings and future research directions, as outlined in chapter 5.

2. Methods

As explained in chapter 1, the goal of this thesis is to optimize the pulse parameters t_p and t_off in order to reduce the turbulent separation bubble on a one-sided diffuser. First, to better understand the influence of the pulse parameters on the flow field, samples are collected in a wind tunnel experiment within specific ranges of the pulse parameters. The Deep Reinforcement Learning (DRL) algorithm is then trained outside of the experimental setup by interpolating between the collected data points, thereby implementing a model-based approach. To validate the training process, additional wind tunnel experiments are conducted. These include training one model from scratch and analyzing the response of a pre-trained model. As the RL algorithm directly interacts with the wind tunnel environment, this approach is defined as model-free. The following chapter explains the experimental setup as well as the theoretical background of the RL algorithm.

2.1. Experimental Setup

The experimental environment consists of a one-sided diffuser designed to induce a turbulent separation bubble. Five Pulsed Jet Actuators (PJAs) are located at the top of the diffuser, with their forcing signal defined by the pulse parameters t_p and t_off. The influence of periodic blowing on the diffuser flow is assessed using eight wall-shear-stress sensors embedded in the diffuser wall.

A detailed description of the experimental setup used in both experiments can be found in chapter 3, section 2.1.

2.2. Reinforcement Learning

To optimize the forcing parameters t_p and t_off, a DRL algorithm utilizing Proximal Policy Optimization (PPO) [29] is employed. To gain a deeper understanding of the workings of this algorithm, it is essential to first examine Machine Learning methods and Neural Networks in general. Additionally, this chapter provides a more detailed explanation of DRL, with a particular focus on policy-based methods, to which the PPO algorithm belongs.

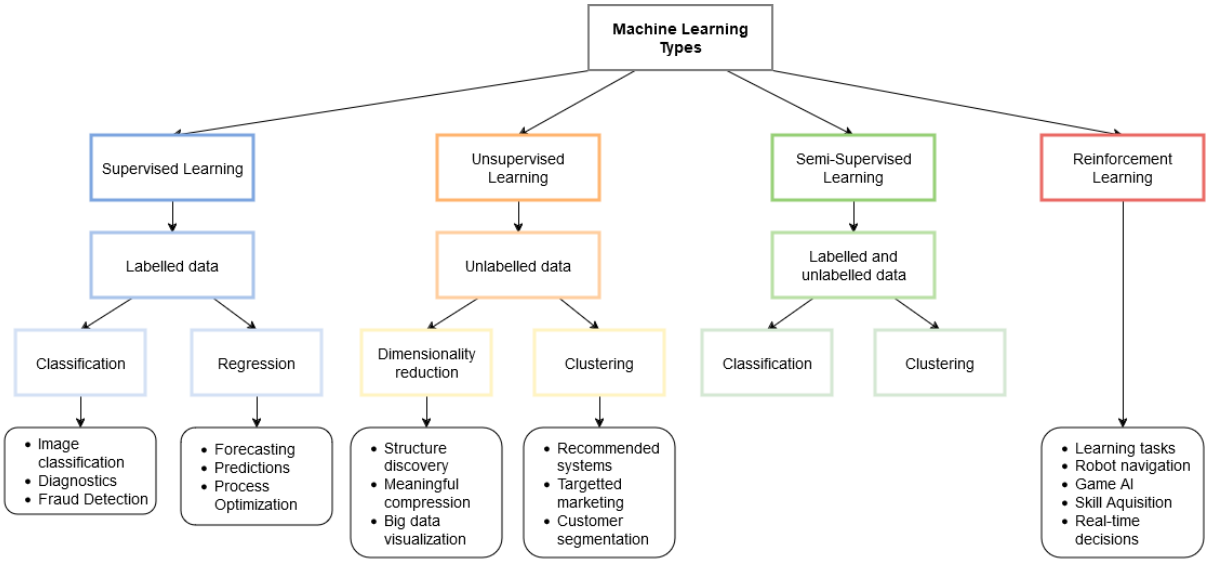


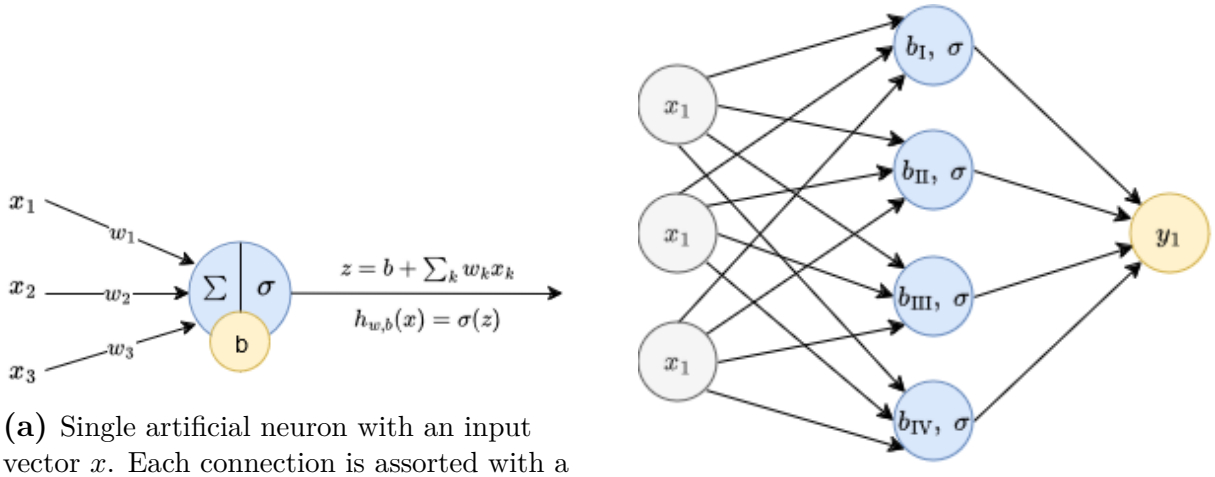
Figure 2.1.: Different types of machine learning methods with their area of application (modified from [1] and [2])

2.2.1. Introduction to Machine Learning and Artificial Neural Networks

Machine Learning (ML) refers to the ability of a system to learn and improve from experience automatically, without requiring explicit programming to address specific tasks [1]. ML is particularly useful in scenarios where no physics-based model exists for the variables of interest, when the available model algorithms are too complex for implementation or unknown. Instead of relying on physics-based models, ML employs algorithms that establish 'black-box' mappings between input and output variables. These 'black-box' models are trained on given datasets. However, their validity can only be assessed through statistical evaluation. The relevance and reliability of such models are highly dependent on the representativeness of the data used for training and on domain-specific knowledge of the problem [30].

There are four primary types of ML techniques: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning (RL) [1]. Figure 2.1 provides an overview over the different techniques together with their respective areas of application. The first three categories differ in the type of data used for training. Supervised Learning uses labeled data, Unsupervised Learning relies on unlabeled data and Semi-supervised Learning utilizes partially labeled data. In contrast, a RL model does not train on predefined datasets. Instead, it directly interacts with an environment, learning a specific strategy through trial-and-error.

Most ML methods use Artificial Neural Networks (ANNs) for their 'black-box' mapping between input and output variables. The basic unit of an ANN is a neuron. Each neuron



(a) Single artificial neuron with an input vector x. Each connection is assorted with a weight vector w. The output of the neuron is the activation function σ, which is applied to a function z that is corrected with a bias b.

(b) Example of a simple ANN with an input vector x, a hidden layer containing four neurons and an output layer containing one neuron.

Figure 2.2.: Representation of a single artificial neuron and a simple ANN (modified from [3] and [4])

is connected to an input vector x via weights w. The neuron computes a linear function of its inputs, represented as $z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$ with the bias b. It then applies a non-linear activation function σ to the computed value, generating an output $h_{w,b}(x) = \sigma(z)$. The schematic of a neuron is depicted in Figure 2.2a. The weights and biases serve as the degrees of freedom within the neuron, while the activation function σ represents a hyperparameter that must be determined during the design process [3], [4].

The simplest form of an ANN consists of one or more neurons organized in a single layer. In this configuration, each neuron is connected to every input, forming what is known as a fully connected layer. Each connection is characterized by a weight and every neuron has an assigned bias. Although the inputs are also structured in a layer, they do not possess biases, nor is an activation function applied to their outputs. This simple setup can be expanded by adding more layers of connected neurons. All layers between the input and output layer are called hidden layers. An example of a simple ANN is shown in Figure 2.2b. When a network possesses two or more hidden layers, it can be referred to as a Deep Neural Network (DNN) [3], [4].

During the learning process of an ANN, all weights and biases are iteratively adjusted to minimize the value of a preselected loss function. The loss function quantifies the quality of the network's predictions. This adjustment is performed using a stochastic gradient method. The gradients of the loss function are estimated with respect to the weights and biases using a back-propagation algorithm [3], [4].

2.2.2. Reinforcement Learning with Proximal Policy Optimization

Reinforcement Learning (RL) is a machine learning methodology that involves teaching an empirical strategy to an agent through trial and error. Based on the current state of an environment, the RL agent selects an action a. This action influences the environment, which subsequently returns a new observation to the agent. The observation consists of the environment's current state s and a reward r that evaluates the quality of the chosen action. One iteration of this process is referred to as a time step. After a predefined number of time steps, known as an episode, the RL agent is updated.

According to Garnier et al. [4], RL methods can be categorized based on their interaction with the environment and their optimization approach. In model-free methods the agent directly interacts with the environment, such as in a wind tunnel experiment. In contrast, model-based methods involve the agent interacting with a model of the environment, for example by using interpolated wind tunnel data.

Additionally, RL methods are divided into value-based and policy-based approaches, both of which aim to maximize their expected return. Value-based methods estimate the expected value of a state-action pair, which guides the agent to choose the best action in each state. A prominent example of value-based optimization is Q-learning, which relies on the learning of the state-action value function or Q-function to find an optimal policy. The Q-function is defined as:

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R(\tau)|s,a\right] \tag{2.1}$$

It denotes the expected discounted cumulative reward $R(\tau)$ when starting in state s and taking action a. It then follows the trajectory τ according to the policy π. In classical Q-learning, the Q-function is stored in a Q-table, which is a simple array representing the estimated value of the optimal Q-function $Q^*(s,a)$. Alternatively, a DNN can be used to generate a map that provides an estimate of the Q-value for each possible action to a given input state. This is called a Deep Q-Network (DQN). While DQNs work well on game environments with discrete action spaces, they struggle to perform well on continuous control tasks [4], [29].

A discrete action space consists of a specified number of actions. This applies to many computer games, like the Atari 2600 environment, since the actions are limited to certain control stick movements (e.g. up, down, left or right) [31]. Continuous control tasks are more complex, as they consist of an infinite number of actions. The active flow control problem investigated in this thesis is a continuous control task, as the DRL agent can choose any value for the pulse parameters t_{p} and t_{off} within a certain range. Duan et al. [32] provide more examples for continuous control problems, including locomotion and partially observable tasks.

Policy-based methods directly optimize a decision policy $\pi_\theta(a_t|s_t)$, where θ represents the policy's parameter vector, a_t denotes the action and s_t the state at a given time step t.

When the agent is represented by a DNN, it is referred to as Deep Reinforcement Learning (DRL) and the parameter vector θ is defined by the weights and biases of the DNN. While a value function can be used to assist in learning the policy parameters, it is not required for action selection [33]. Policy-based methods have three main advantages over value-based methods. First, they tend to exhibit better convergence properties, although they may be prone to being trapped in local minima. Second, they are well-suited for high-dimensional action spaces. Third, they enable the learning of stochastic policies [4].

To optimize the policy, an estimator $\hat{g}$ is computed and passed to a stochastic gradient ascent algorithm, which finds the optimal parametrization θ^* that maximizes $\hat{g}$. The estimator is defined as:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.2}$$

Here, $\mathbb{E}$ represents the expectation, while $\hat{A}_t$ is an estimator of the advantage function at time step t. The advantage function $A_t(a_t|s_t)$ represents the improvement in the expected cumulative reward when taking action a_t in state s_t, compared to the average of all possible actions in state s_t. In practice, the advantage function is not directly available and must be estimated using a separately learned value function [4], [29].

When the policy π_θ is represented by a DNN, the evaluation of $\hat{g}$ can then be efficiently performed using a back-propagation algorithm, provided that the gradient of the loss function corresponds to the policy gradient. The loss function is expressed as:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.3}$$

The so called vanilla deep policy gradient algorithm performs multiple steps of optimization on this loss L^{PG} using the same trajectory. However, this often leads to destructively large policy updates. To mitigate this issue and achieve better results, the loss can be adjusted in various ways [4], [29].

The Actor-Critic method improves learning performance by simultaneously utilizing two networks: the actor and the critic network. The concept of utilizing an actor that takes a random action and a critic that evaluates this action was first introduced to RL by Barto et al. [34]. In policy-based DRL methods, the actor represents the policy $\pi_\theta(a_t|s_t)$, which determines the actions taken by the agent. The critic predicts the value function, which is used to evaluate the quality of the actions selected by the actor. The value V predicted by the critic is incorporated into the gradient estimator $\hat{g}$ through the advantage function. As previously mentioned, the advantage function is not directly available but can be estimated using the cumulative reward R_t obtained by the actor after performing action a_t, along with the value predicted by the critic for the subsequent state s_{t+1}. This estimation is expressed by the following equation, with γ being a discount factor [4].

$$\hat{A}_t(a_t|s_t) \sim R_t(a_t|s_t) + \gamma V(s_{t+1}) - V(s_t) \tag{2.4}$$

In Trust Region Policy Optimization (TRPO) [35], which can also be categorized as an Actor-Critic method, the policy is updated by maximizing a surrogate objective function L^{TRPO}:

$$L^{TRPO}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right] \tag{2.5}$$

The term $r_t(\theta)$, known as the probability ratio, is defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{2.6}$$

The objective function L^{TRPO} compares the performance of the current policy π_θ with that of the previous policy $\pi_{\theta_{\text{old}}}$. TRPO further constrains the size of policy updates by employing second-order natural gradient optimization. This ensures that parameter updates occur within a trust region defined by a fixed maximum divergence between the distributions of the old and new policies. While this approach enhances stability, it results in a relatively complex algorithm [4].

The Proximal Policy Optimization (PPO) algorithm was first introduced by Schulman et. al. [29] and aims to simplify the constraint on policy updates. This is done by clipping the probability ratio $r_t(\theta)$, so that it remains within a specified interval $[1-\varepsilon,\ 1+\varepsilon]$, where ε is an additional hyperparameter introduced to control the extent of the clipping. Schulman et al. [29] recommend a value of $\varepsilon = 0.2$ for this purpose. The clipped objective of the PPO algorithm is defined as the actor loss $L_t^{actor}(\theta)$, given by:

$$L_t^{actor}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta),\ 1-\varepsilon,\ 1+\varepsilon)\hat{A}_t \right) \right] \tag{2.7}$$

To further enhance the performance of the PPO algorithm, additional loss terms are incorporated into the objective function. These include a critic loss $L_t^{critic}(\theta)$ and an entropy loss $L^{entropy}(\pi_\theta|s_t)$. The critic loss is defined as:

$$L_t^{critic}(\theta) = \left(V(s_t) - V^{\text{targ}} \right)^2 \tag{2.8}$$

where $V(s_t)$ is the value learned by the critic network and V^{targ} is derived from V and the advantage function. The entropy loss $L^{entropy}(\pi_\theta|s_t)$ is computed from the stochastic entropy of the probability distribution π_θ and ensures sufficient exploration. By combining these components, the overall objective function, which is maximized during each iteration, is expressed as:

$$L_t^{PPO}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{actor}(\theta) - c_1 L_t^{critic}(\theta) + c_2 L^{entropy}(\pi_\theta|s_t) \right] \tag{2.9}$$

Here, c_1 and c_2 are coefficients weighing the influence of the different loss terms. For this thesis, the coefficients are set to the values proposed by Schulman et al. [29], being $c_1 = 0.5$ and $c_2 = 0.001$.

The implementation of the PPO algorithm to optimize the active flow control parameters t_{p} and t_{off} is discussed in detail in chapter 3, section 2.2.

3. Scientific publication

The following chapter contains a paper presented at the DGLR STAB-Symposium held in Regensburg (November 13 - November 14 2024). It is currently under consideration for publication in the conference proceedings.

The paper explains in detail the experimental setup mentioned in section 2.1, as well as the implementation of the Proximal Policy Optimization (PPO) algorithm in order to optimize the forcing signal of a pulsed blowing system. The corresponding Python code can be found on *GitHub* [36]. The paper investigates three different reward functions and their influence on the performance of the algorithm.

Optimizing pulsed blowing parameters for active separation control in a one-sided diffuser using reinforcement learning

Alexandra Müller[1], Tobias Schesny[2], Ben Steinfurth[3], and Julien Weiss[4]

Chair of Aerodynamics, Institute of Aeronautics and Astronautics,
Technische Universität Berlin, Marchstr. 12-14, 10587 Berlin
[1] alexandra.mueller@campus.tu-berlin.de
[2] schesny@campus.tu-berlin.de
[3] ben.steinfurth@tu-berlin.de
[4] julien.weiss@tu-berlin.de

Abstract. Reinforcement learning is employed to optimize the periodic forcing signal of a pulsed blowing system that controls flow separation in a fully-turbulent $Re_\theta = 1000$ diffuser flow. Based on the state of the wind tunnel experiment that is determined with wall shear-stress measurements, Proximal Policy Optimization is used to iteratively adjust the forcing signal. Out of the reward functions investigated in this study, the incremental reduction of flow reversal per action is shown to be the most sample efficient. Less than 100 episodes are required to find the parameter combination that ensures the highest control authority for a fixed mass flow consumption. Fully consistent with recent studies, the algorithm suggests that the mass flow is used most efficiently when the actuation signal is characterized by a low duty cycle where the pulse duration is small compared to the pulsation period. The results presented in this paper promote the application of reinforcement learning for optimization tasks based on turbulent, experimental data.

Keywords: Active Flow Control, Reinforcement Learning, Proximal Policy Optimization

1 Introduction

Flow separation is often accompanied by an increase in drag and other unwanted behavior like stall on an airplane or surge in a compressor. This study focuses on active separation control, where an external energy source is required to manipulate the flow. Compared to classical boundary layer control by means of steady suction or blowing, a higher control authority is achieved with oscillatory blowing. Greenblatt and Wygnanski [1] provide an overview of different means of active flow control by periodic excitation. To quantify the relative momentum input associated with blowing in flow control applications, the dimensionless momentum coefficient $C_\mu = (c_\mu, \langle c_\mu \rangle)$ is defined as the ratio of added momentum over free-stream momentum. The momentum coefficient can be divided into

c_μ for the steady and $\langle c_\mu \rangle$ for the oscillatory momentum. Another parameter to describe means of active flow control is the dimensionless frequency $F^+ = {}^{(f \cdot x)}/_{U_\infty}$ with the excitation frequency f, a reference length x and the free-stream velocity U_∞. In their experiment, Seifert et al. [2] show that by periodically blowing over the flap of an airfoil with $F^+ = 2$, a smaller momentum input is needed for similar effect on lift and drag compared to the steady-blowing approach. This demonstrates the efficiency of the periodic excitation as well as the importance of the time scales defining the forcing signal. In the present study, the active flow control is performed by Pulsed Jet Actuators (PJAs), similar to those used by Steinfurth et al. [3] and Löffler et al. [4]. The periodic forcing of a PJA is defined by two timescales, the pulse duration t_p and the time delay between successive pulses t_off.

Recent advances in Machine Learning and Artificial Neural Networks (ANN) present new methods of optimizing active flow control parameters. As pointed out by Rabault et al. [5], deep reinforcement learning (DRL) is a promising strategy for flow control optimization. This method is based on teaching empirical strategies to an ANN through trial and error. DRL already achieved improvements in the fields of robotics [6], language processing [7] and games [8]. It was first applied to an active flow control problem by Rabault et al. [9]. The article shows that in a two-dimensional simulation, an ANN is able to learn an active flow control strategy by varying the mass flow rate of two jets, one on each side of a cylinder, thereby stabilizing the wake and reducing drag. Additional studies to control the flow around a cylinder were conducted by Ren et al. [10] who investigated the same problem as Rabault et al. [9] but in weakly turbulent conditions. Fan et al. [11] discovered control strategies for a turbulent cylinder flow with two fast-rotating smaller control cylinders using DRL in experimental and simulation environments whereas Tokarev et al. [12] studied the application of DRL to control the laminar flow over an oscillating cylinder. Other DRL applications include flow separation control over an airfoil by Shimomura et al. [13] or suppressing vortex-induced vibration by Zheng et al. [14].

The present paper aims to apply DRL to optimize the active flow control parameters in the one-sided diffuser that was previously studied by Steinfurth et al. [3] and Löffler et al. [4]. As opposed to the majority of previous studies, we consider experimental data of a fully-turbulent flow with a Reynolds number of $Re_\theta = 1000$ based on a momentum thickness $\theta \approx 0.8\,\mathrm{mm}$. The objective is to optimize the periodic forcing parameters t_p and t_off of a PJA in order to reduce the turbulent separation bubble on the diffuser. The state is returned with wall shear-stress measurements along the diffuser. Actions consist in incremental changes of t_p and t_off, and the suppression of reverse-flow is rewarded. During each episode, state-action-reward triplets are gathered for fifteen combinations of t_p and t_off. The main focus point of this study is the influence of different reward functions. Furthermore, the importance of explorative training is demonstrated. The experimental setup and DRL algorithm are explained in section 2. The results will be presented in section 3, followed by a conclusion in section 4.

2 Methodology

The following section provides a detailed overview of the experimental setup and DRL algorithm used for this study.

2.1 Experimental Setup

The environment examined in this paper is a one-sided diffuser with a deflection angle of $\alpha = 20\,^\circ$ and a ramp length of $L = 337\,\mathrm{mm}$. The test section is 600 mm wide and has a flat ceiling with a distance of 400 mm to the bottom of the diffuser opening section. The diffuser is situated in a closed-loop wind tunnel with a nominal velocity of $U_{\mathrm{ref}} = 20\,\mathrm{m/s}$ and a Reynolds number of $Re_\theta = 1000$, based on a momentum thickness $\theta \approx 0.8\,\mathrm{mm}$. For active flow control, five PJAs with a 2 mm spacing are embedded at the top of the diffuser ramp at an emission angle of $\varphi = 30\,^\circ$. Each PJA is connected to a magnetic valve that is controlled by a square wave signal. The square wave signal is composed of the pulse duration t_{p} and the time delay between successive pulses t_{off} (see Figure 1). The magnetic valves have a control uncertainty of $\pm 15\,\%$ with regard to the pulse duration t_{p}. The PJAs are connected to a compressed air supply with a fixed volume flow of $\dot{V} = 60\,\mathrm{l/min}$, which translates to a jet velocity $u_{\mathrm{jet}} = U_{\mathrm{ref}}$ when $t_{\mathrm{p}} = t_{\mathrm{off}}$. By finding the optimal actuation parameters for a fixed mass flow rate, the active flow control efficiency is maximized. The volume flow is measured by a flow sensor with a measuring range of $\dot{V} = 2$ - $200\,\mathrm{l/min}$ and an uncertainty of $\pm 3.3\,\%$. The control parameter of the volume flow is the pressure provided by a pressure control valve. A schematic of the complete measurement setup is depicted in Figure 2.

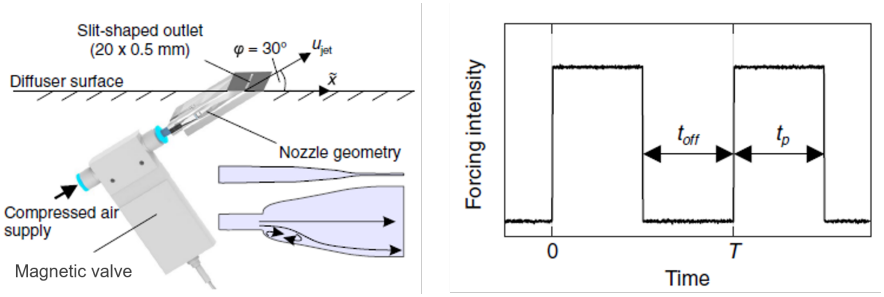


Fig. 1. PJA for active separation control. Left: illustration of the actuator. Right: control signal [3]

To measure the effectiveness of the active flow control, eight calorimetric wall-shear-stress sensors are embedded in the center line of the diffuser. With this setup, only the two-dimensional influence of the PJAs on the diffuser flow is observed. The used wall-shear-stress sensors were first introduced by Weiss et al. [15,16]. They consist of an electrically heated microbeam suspended over a small cavity. Two additional lateral detector beams act as resistance thermometers by measuring the thermal wake of the heated beam. The sensors are sensitive to the

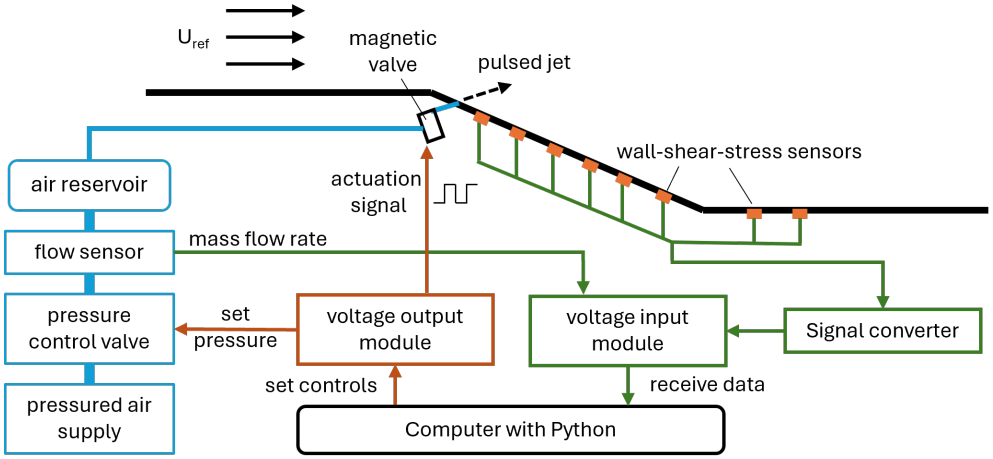


Fig. 2. Schematic depiction of the experimental setup

flow direction and can be calibrated in a dedicated facility to further obtain the wall-shear-stress values. In this study, only the voltage output is required as it already provides information about the flow direction. At each sensor location, the forward flow fraction γ is determined as the percentage of positive values of the signal relative to the total length of the signal. The signals are obtained at a frequency $f_s = 500\,\mathrm{Hz}$ and a measurement time $T = 10\,\mathrm{s}$. As a scalar parameter for the PJA effectiveness, the normalized integral of the forward-flow fraction Γ is defined as follows with Δx being the distance between the first and the last sensor.

$$\Gamma = \frac{1}{\Delta x} \int \gamma \, \mathrm{d}x \tag{1}$$

To reduce the wall time required for training, but also to better understand the flow field, measurement data are gathered for a set of predefined values for t_p and t_off. The DRL algorithm is then trained outside of the wind tunnel experiment by interpolating between the data points and thus making it a model-based approach. To construct this model of the system response, samples are taken in the range t_p, $t_\mathrm{off} = 1$ - $25\,\mathrm{ms}$ with an increment of $\Delta t = 2\,\mathrm{ms}$ as well as $t_\mathrm{p} = 1$ - $7\,\mathrm{ms}$ and $t_\mathrm{off} = 3$ - $19\,\mathrm{ms}$ with an increment of $\Delta t = 1\,\mathrm{ms}$. Figure 3 shows the integrated forward flow fraction Γ over the entire parameter space. The parameter space contains one global optimum at $t_\mathrm{p} = 1\,\mathrm{ms}$ and $t_\mathrm{off} = 9\,\mathrm{ms}$ with a maximum integrated forward flow fraction $\Gamma = 0.98$. This is consistent with results by Steinfurth et al. [3] and Löffler et al. [4], where small t_p also proved to be effective. Furthermore, Figure 3 depicts the distribution of the forward flow fraction γ of a flow field with and without actuation. The latter suggest reverse-flow at the bottom of the ramp ($x = 250$ - $400\,\mathrm{mm}$), which indicates the presence of a turbulent separation bubble. Actuation increases the forward flow fraction γ in the whole flow field until almost all measurement positions indicate a γ close to 1, meaning a forward-facing flow in the wall-region, for the optimal actuation case.

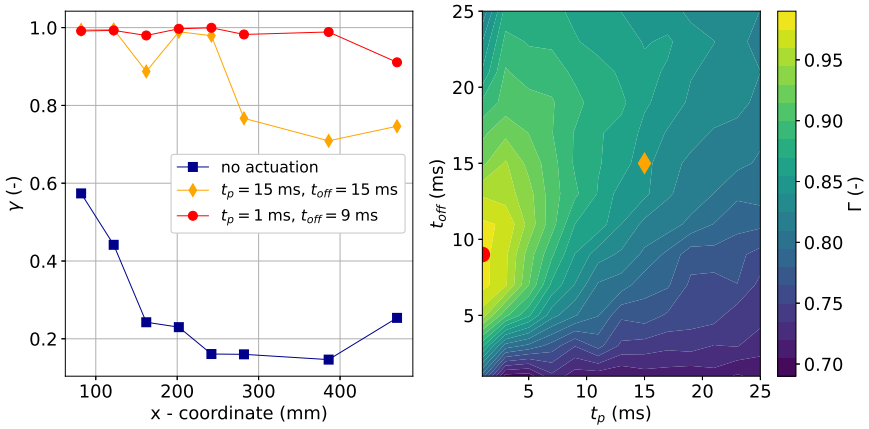


Fig. 3. Measurement results. Left: distribution of the forward flow fraction γ for different actuation parameters. Right: integrated forward flow fraction Γ over entire parameter space

2.2 Reinforcement Learning

DRL is a machine learning methodology based on teaching empirical strategies to an ANN through trial and error. It can be simplified as a cyclic process. The DRL agent, represented by one or more ANNs, performs an action based on a specific policy. The policy represents a probability distribution $\pi(a_t|s_t)$ that maps a state of the environment s_t to an action a_t. The continuous action space spans incremental changes of the actuation parameters $a_t = [\Delta t_\mathrm{p},\ \Delta t_\mathrm{off}]$. The actuation parameters at a time step t are defined as:

$$t_\mathrm{p}^t = t_\mathrm{p}^{t-1} + \Delta t_\mathrm{p}^t \qquad \text{and} \qquad t_\mathrm{off}^t = t_\mathrm{off}^{t-1} + \Delta t_\mathrm{off}^t \tag{2}$$

The performed action has an effect on an environment, in this case the previously described one-sided diffuser. Observation of the environment provides a new state and reward to the DRL agent. The state of the environment is represented by an eight-element vector that includes the forward-flow fractions γ along the diffuser test section. The reward function will be described later in this section. Based on the information about the state, the DRL agent generates a new action and the process is repeated. One loop of the process is called a time step and a sequence of time steps is called an episode. The DRL policy is updated at the end of each episode. A schematic of the process is shown in Figure 4.

There are several algorithms that may be used as the DRL policy update. Viquerat et al. [17] provide a brief overview over the most commonly used DRL policies in the context of fluid mechanics. For this study, it was decided to use the Proximal Policy Optimization (PPO) due to its sample efficiency, robustness and easy application. PPO was first introduced by Schulman et al. [18] and belongs to the Policy Gradient methods. Policy Gradient methods compute an estimator of the policy gradient and pass it to a stochastic gradient ascent algorithm [18]. The approach in the present paper is model-based since the environment is

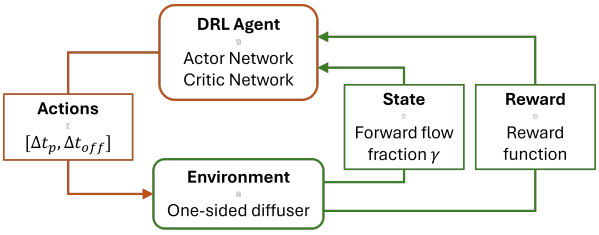


Fig. 4. Schematic of one time step in Deep Reinforcement Learning

modeled from experimental wind tunnel data. The algorithm is implemented in Python using the machine learning package PyTorch. The code is based on PPO implementations by [19] and [20].

The PPO algorithm requires two networks. The actor network provides an action based on an observation. It consists of an input layer with eight neurons, representing the state of the environment (i.e., the forward-flow fraction distribution), two hidden layers with 256 neurons, followed by an output layer with two neurons corresponding to the actions defined by Δt_p and Δt_off. To ensure a continuous action space the actor network returns a Gaussian Normal distribution with a mean value μ between ± 1 ms and a pre-defined standard deviation *std* for each element in the action space. The final action is then derived as a random sample from this distribution. The standard deviation can therefore be used as a dial to enforce exploration and exploitation as shown in section 3. The critic network learns to predict the value function which is used to evaluate the action chosen by the actor network. It has the same structure as the actor network, except for the output layer, which has only one neuron for the critic value. Both networks are updated with a learning rate $lr = 3 \cdot 10^{-4}$.

The main focus of this study is to investigate the effect of three different reward functions r on the optimization performance. The function $r = \Gamma$ returns the integral forward flow fraction Γ of the current time step, whereas $r = \Delta\Gamma$ returns the difference in Γ compared to the previous time step. The third reward is defined for each time step t as the sum of integrated forward flow fraction Γ of all previous time steps divided by the number of the current time step t:

$$r = \Gamma_\mathrm{sum} = \frac{\sum_{i=0}^{t} \Gamma_i}{t} \tag{3}$$

The three reward functions are investigated for different standard deviations of the action space $std = [0.4\,\mathrm{ms},\ 0.2\,\mathrm{ms},\ 0.001\,\mathrm{ms}]$. The standard deviation remains constant throughout the learning process. Each episode starts with the same initial actuation parameters $t_\mathrm{p} = t_\mathrm{off} = 15\,\mathrm{ms}$. The learning processes are run for 1,500 episodes with 15 time steps per episode.

3 Results

Figure 5 shows the cumulative reward over the first 500 episodes for all reward functions and standard deviations. The PPO algorithm aims to maximize the

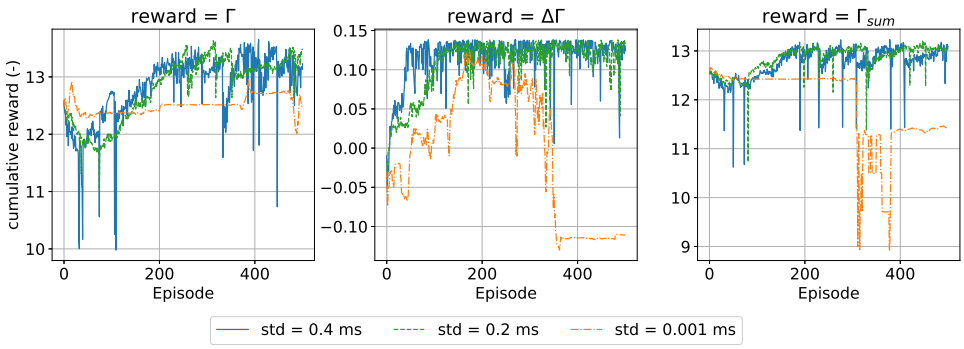


Fig. 5. Cumulative reward per episode for different reward functions

cumulative reward, which is defined as the sum of all rewards per time step in one episode. Thus, low rewards in single time steps are permitted, as long as the cumulative reward of the episode is high. As explained previously, the standard deviation governs the width of the Normal Distribution from which the actions are derived, thereby enforcing either exploration for high standard deviations or exploitation for low standard deviations. In Figure 5, the blue and green curves depicting the standard deviations $std = 0.4\,\text{ms}$ and $std = 0.2\,\text{ms}$ increase for all three reward functions. This indicates that the PPO agent develops an effective strategy for the flow control optimization. The signal for the higher standard deviation $std = 0.4\,\text{ms}$ is noisier and increases slightly faster for $r = \Gamma$ and $r = \Delta\Gamma$. The orange curves depicting $std = 0.001\,\text{ms}$ are smoother, but differ from the curves for the other standard deviations in all cases. The cumulative reward increases and decreases for $r = \Delta\Gamma$ while not increasing at all for $r = \Gamma$ and $r = \Gamma_{\text{sum}}$. It is also generally lower compared to the other standard deviations. The low standard deviation $std = 0.001\,\text{ms}$ equates to exploitation, meaning that the agent will always choose the "best" action without any random excursions. An untrained algorithm first needs a lot of information about the parameter space to learn an effective strategy. Without exploration the chances that the algorithm "accidentally" pursues the correct direction is much lower.

The influence of the standard deviation is also highlighted in Figure 6 as an example for the $r = \Delta\Gamma$ function. It depicts the progression of the actuation parameters t_{p} and t_{off} during specific episodes throughout the learning process. For $std = 0.4\,\text{ms}$ and $std = 0.2\,\text{ms}$, the algorithm seems to pursue a random direction in the first episode, because of the noise added to the choice of the action with a non-negligible standard deviation. After a few hundred episodes, the algorithm repeatably finds the region of high forward flow fraction and therefore high reward. For $std = 0.001\,\text{ms}$, the algorithm only follows a single direction in each episode. Although the actions are approaching the region of high reward in the episodes 200 and 300, the algorithm does not seem to learn an effective policy. A future approach, that is also pursued in other studies, could be to first run the training to a high standard deviation for a small number of episodes for fast learning and then switch to a lower *std* for the rest of the learning process to reduce the noise and randomness. This study aims to optimize the flow control

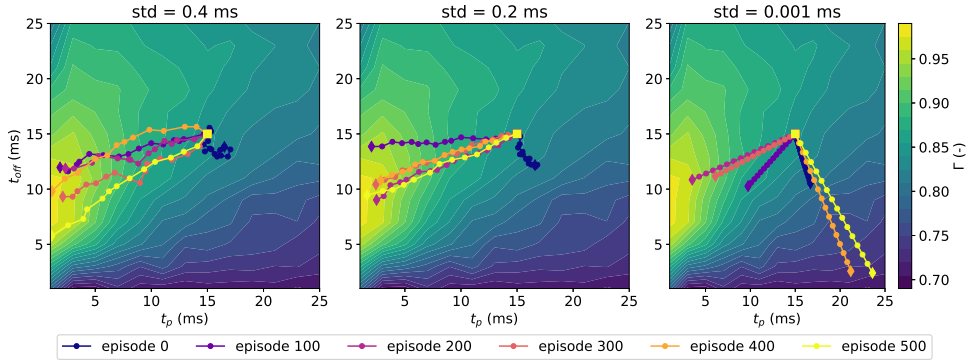


Fig. 6. Progression of the action space $[t_\mathrm{p},\ t_\mathrm{off}]$ during specific episodes for different standard deviations *std*. The displayed reward function is $r = \Delta\Gamma$. The initial time step of each episode is depicted as □ and the final time step as ♢.

parameters by maximizing the forward flow fraction. To see if the PPO agents have reached this goal, Figure 7 depicts Γ at the last time step of each episode. The distributions are very similar to that of the cumulative reward with Γ increasing for $std = 0.4\,\mathrm{ms}$ and $std = 0.2\,\mathrm{ms}$. Starting from an initial $\Gamma = 0.84$, the algorithms for $r = \Gamma$ and $r = \Gamma_\mathrm{sum}$ reach values around $\Gamma = 0.9$ after 200 episodes. The algorithm for $r = \Delta\Gamma$ achieves values around $\Gamma = 0.96$ after 50 episodes for $std = 0.4\,\mathrm{ms}$ and 150 episodes for $std = 0.2\,\mathrm{ms}$. Similar to the cumulative reward, the curves for the integral forward flow fraction decrease or stay constant for a standard deviation of $std = 0.001\,\mathrm{ms}$.

Comparing the three different reward functions reveals that the fastest learning with the highest achieved integral forward flow fraction Γ can be accomplished with a reward function $r = \Delta\Gamma$. This can be explained as the parameter space only contains one optimum and the DRL algorithm quickly learns to follow the steepest gradient.

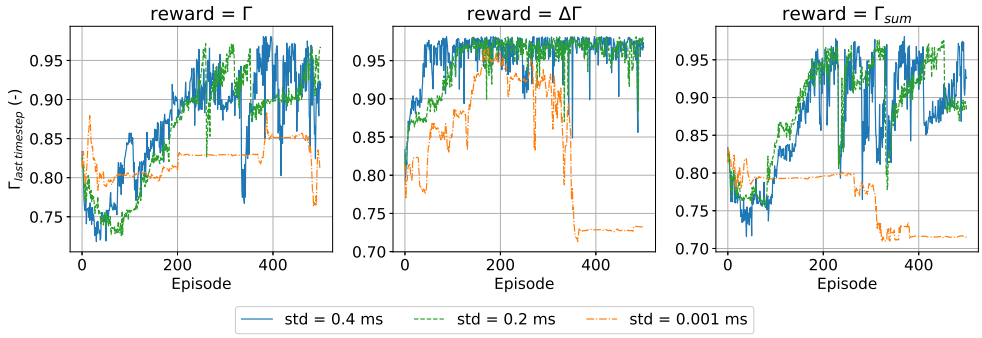


Fig. 7. Γ at the last time step of each episode for different reward functions

4 Conclusion

This paper demonstrates the optimization of active flow control parameters in a fully-turbulent one-sided diffuser flow based on experimental data using DRL

with the PPO algorithm. The forcing parameters t_p and t_off of five PJAs embedded at the top of the diffuser ramp are adjusted while an array of wall-shear-stress sensors delivers the distribution of the forward flow fraction, that defines the state of the diffuser flow. An investigation of the parameter space reveals an optimum in Γ at a low pulse time t_p, which is equivalent to a low duty cycle $DC = {t_\mathrm{p}}/{(t_\mathrm{p}+t_\mathrm{off})}$. These results are consistent with the studies by Steinfurth et al. [3] and Löffler et al. [4]. The former conducted a conventional parameter study and the latter used surrogate modeling to optimize the active flow control parameters.

The DRL is performed for three different reward functions at three different standard deviations. For higher standard deviations, all reward functions yield reasonable optimization strategies when starting the episode from fixed initial values for t_p and t_off. Lacking the needed exploration at the beginning of the training, the algorithms trained with a small standard deviation do not perform well in comparison. Out of the three reward functions $r = \Delta\Gamma$ proves to be the most efficient. This can be explained by the parameter space containing only one optimum. In fact, other techniques may be viewed as more suitable for the task at hand. Nonetheless, applying DRL to this case does not only increase the knowledge on this specific topic, but also prepares for the next investigation approaches.

So far, each learning episode starts at the same initial t_p and t_off values. Thinking about possible active flow control applications in a real, non-testbed environment, a DRL algorithm might be subject to variable input parameters. The next step is to examine how well the algorithm performs when the actuation parameters are randomly initiated at the beginning of each episode.

In summary, this study shows that training a DRL algorithm to perform flow control based on experimental data is possible while also pointing out topics for further investigation and improvement.

References

1. D. Greenblatt and I. J. Wygnanski, “The control of flow separation by periodic excitation,” *Progress in Aerospace Sciences*, vol. 36, no. 7, pp. 487–545, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042100000087
2. A. Seifert, A. Darabi, and I. Wyganski, “Delay of airfoil stall by periodic excitation,” *Journal of Aircraft*, vol. 33, no. 4, pp. 691–698, 1996. [Online]. Available: https://doi.org/10.2514/3.47003
3. B. Steinfurth and J. Weiss, “Efficiency enhancement in active separation control through optimizing the duty cycle of pulsed jets,” *AIAA Journal*, vol. 60, no. 12, pp. 6566–6580, 2022. [Online]. Available: https://doi.org/10.2514/1.J061667
4. S. Löffler, B. Steinfurth, and J. Weiss, *Surrogate-Based Exploration of Active Separation Control Parameters: An Experimental Study*, 2023. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2023-0076
5. J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu, “Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape

optimization," *Journal of Hydrodynamics*, vol. 32, pp. 234–246, 2020. [Online]. Available: https://doi.org/10.1007/s42241-020-0028-y
6. L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," 2017. [Online]. Available: https://arxiv.org/abs/1710.06542
7. D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," 2017. [Online]. Available: https://arxiv.org/abs/1607.07086
8. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602
9. J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control," *Journal of Fluid Mechanics*, vol. 865, p. 281–302, 2019.
10. F. Ren, J. Rabault, and H. Tang, "Applying deep reinforcement learning to active flow control in weakly turbulent conditions," *Physics of Fluids*, vol. 33, no. 3, p. 037121, 03 2021. [Online]. Available: https://doi.org/10.1063/5.0037371
11. D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Reinforcement learning for bluff body active flow control in experiments and simulations," *Proceedings of the National Academy of Sciences*, vol. 117, no. 42, pp. 26 091–26 098, 2020. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2004939117
12. M. Tokarev, E. Palkin, and R. Mullyadzhanov, "Deep reinforcement learning control of cylinder flow using rotary oscillations at low reynolds number," *Energies*, vol. 13, no. 22, 2020. [Online]. Available: https://doi.org/10.3390/en13225920
13. S. Shimomura, S. Sekimoto, A. Oyama, K. Fujii, and H. Nishida, "Closed-loop flow separation control using the Deep Q network over airfoil," *AIAA Journal*, vol. 58, no. 10, pp. 4260–4270, 2020. [Online]. Available: https://doi.org/10.2514/1.J059447
14. C. Zheng, T. Ji, F. Xie, X. Zhang, H. Zheng, and Y. Zheng, "From active learning to deep reinforcement learning: Intelligent active flow control in suppressing vortex-induced vibration," *Physics of Fluids*, vol. 33, no. 6, p. 063607, 06 2021. [Online]. Available: https://doi.org/10.1063/5.0052524
15. J. Weiss, Q. Schwaab, Y. Boucetta, A. Giani, C. Guigue, P. Combette, and B. Charlot, "Simulation and testing of a MEMS calorimetric shear-stress sensor," *Sensors and Actuators A: Physical*, vol. 253, pp. 210–217, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424716309116
16. J. Weiss and A. Giani, "Calorimetric wall-shear-stress microsensors for low-speed aerodynamics," *Experiments in Fluids*, vol. 65, no. 63, 2024. [Online]. Available: https://doi.org/10.1007/s00348-024-03803-2
17. J. Viquerat, P. Meliga, A. Larcher, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics: An update," *Physics of Fluids*, vol. 34, no. 11, p. 111301, 11 2022. [Online]. Available: https://doi.org/10.1063/5.0128446
18. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1707.06347
19. henanmemeda, "Rl-adventure-2," 2018. [Online]. Available: https://github.com/henanmemeda/RL-Adventure-2/blob/master/3.ppo.ipynb
20. P. Tabor, "Youtube-code-repository," 2020. [Online]. Available: https://github.com/philtabor/Youtube-Code-Repository/tree/master/ReinforcementLearning/PolicyGradient/PPO/torch

4. Additional results

As previously explained, experimental data regarding the influence of the pulse parameters t_p and t_off on the flow field was gathered. This data is then used to train the PPO algorithm in a model-based approach. The paper presents results for three reward functions, $r = \Gamma$, $r = \Delta\Gamma$, and $r = \Gamma_\mathrm{sum}$, evaluated at three different standard deviations $std = [0.4\,\mathrm{ms},\ 0.2\,\mathrm{ms},\ 0.001\,\mathrm{ms}]$. Among these, the reward function $r = \Delta\Gamma$ demonstrated the highest sample efficiency, identifying an optimal combination of pulse parameters at low t_p in fewer than 100 episodes at a standard deviation $std = 0.4\,\mathrm{ms}$. To assess whether the algorithm would perform effectively in a real-world environment, an additional series of wind tunnel measurements was conducted. In these experiments, the algorithm directly interacts with the wind tunnel environment, adopting a model-free approach. For these measurements, the same setup as described in chapter 3, section 2.1, is employed. Furthermore, the DRL algorithm is executed in a deterministic mode, to test the application of the learned active control strategy. Additional model-based training runs are conducted to investigate the behavior of a random initialization of the pulse parameters at the start of each episode.

4.1. Experimental validation

In the first experiment, the PPO algorithm is trained from the beginning. The main difficulty during the model-free training is the long response time for each time step, due to the control time for the volume flow and the measurement time for the wall-shear-stress sensors. Thus the number of episodes is limited by the available time in the wind tunnel. As a consequence, the combination $r = \Delta\Gamma$ with $std = 0.4\,\mathrm{ms}$ is selected for this experiment, as it proved to be the most sample-efficient during the model-based training. The algorithm is trained for 50 episodes with 15 time steps per episode. The results are presented in the Figures 4.1 and 4.2. It is observed that the cumulative reward increases over the training process. However, due to the high standard deviation, significant noise is present in the reward curve. Despite this, the forcing parameters converge towards a low pulse duration t_p and a medium time delay t_off, which aligns with the expected behavior seen in the model-based training and previous studies by Steinfurth et al. [14] and Löffler et al. [15]. The only parameter deviating from expectations is the integral forward flow fraction Γ at the final time step. During the first 20 episodes, Γ increases slightly from $\Gamma = 0.7$ to $\Gamma = 0.8$, after which it drops back to $\Gamma = 0.7$ and remains nearly constant for the remaining episodes. Examination of Figure 4.2 reveals that, during episodes 36 and 45, regions of $\Gamma > 0.7$ are reached, contradicting the results presented in Figure 4.1. This

suggests the presence of a measurement error.

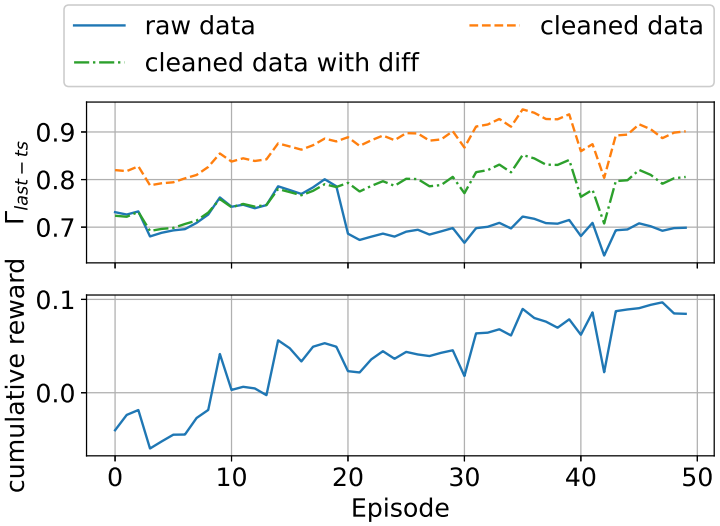


Figure 4.1.: Result of the model-free wind tunnel training: Integral forward flow fraction Γ at the last time step and the cumulative reward plotted over the number of episodes.

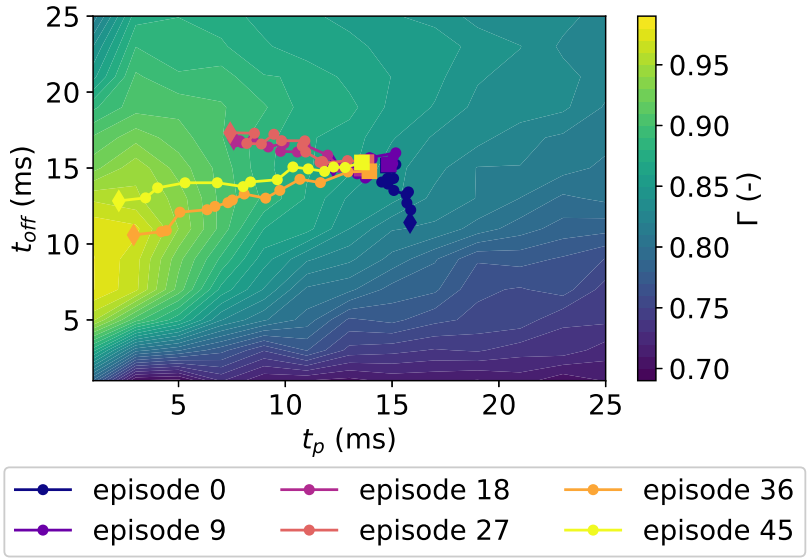


Figure 4.2.: Result of the model-free wind tunnel training: Progression of the action space $[t_\mathrm{p},\ t_\mathrm{off}]$ during specific episodes. The initial time step of each episode is depicted as $\square$ and the final time step as $\diamondsuit$.

Further analysis of the local forward flow fraction γ at each sensor location shows that, starting from episode 20, the value of γ at the 7th position consistently drops to approximately 0 (see Figure 4.3). This finding correlates with the timeline of the experiment,

as the first 20 episodes were conducted on a different day than the following 30 episodes. To address this issue, the integral forward flow fraction Γ is re-calculated for each time step, excluding the data from the 7th sensor (see the orange curve in Figure 4.1). However, since the dataset is sparse, omitting a sensor introduces inaccuracies in the integral calculation, resulting in higher Γ values compared to the raw data. To correct this, Γ is adjusted by subtracting the mean difference between the raw and re-calculated values during the first 20 episodes (green curve in Figure 4.1). The corrected curve shows that Γ in the last time step is slowly increasing and then fluctuating around $\Gamma = 0.8$. This behavior is consistent with the behavior of tp and t_{off} displayed in Figure 4.2.

The experiment demonstrates the ability of a DRL algorithm to develop an efficient active control strategy in an experimental environment. The measurement and calculation error in the integral forward flow fraction does not affect the behavior of the PPO algorithm. The error mechanism remains consistent across all time steps, and the optimal Γ continues to correspond to the same combination of t_{p} and t_{off}.

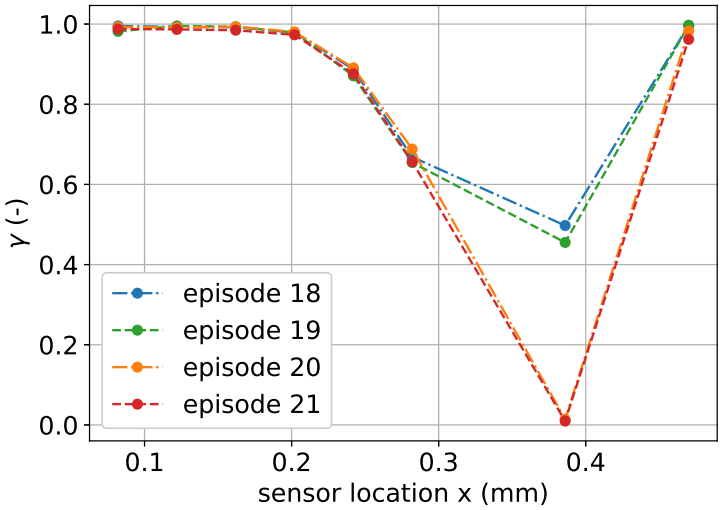


Figure 4.3.: Measured forward flow fraction γ at the last time step of the episodes 18 - 21. Starting from episode 20 a measurement error can be observed at the 7th sensor location.

In the second experiment, the behavior of a pre-trained model is examined. The PPO algorithm is initially trained for 500 episodes outside of the wind tunnel using a model-based approach. The model-based training employs the reward function $r = \Delta\Gamma$ and a standard deviation $std = 0.4\,\text{ms}$. This combination is again selected for its high sample efficiency. Following this, the model undergoes further training in the wind tunnel using a model-free approach for an additional 20 episodes, with a reduced standard deviation of $std = 0.2\,\text{ms}$. The lower standard deviation is chosen to reduce exploration and enhance exploitation. The results for the second experiment are presented in the Figures 4.4 and 4.5. The values for the integral forward flow fraction Γ are adjusted to account for the same measurement

error observed in the first experiment. Similar to the findings described in chapter 3, section 3, the cumulative reward and the integral forward flow fraction increase rapidly during the model-based training and then stabilize, fluctuating around constant values. During the experiment, both the reward and Γ exhibit slight decreases compared to the pre-trained model. However, as shown in Figure 4.5, the algorithm adheres to the previously learned policy during the experiment. The observed decrease in Γ and cumulative reward can be attributed to the measurement error in the integral forward flow fraction Γ.

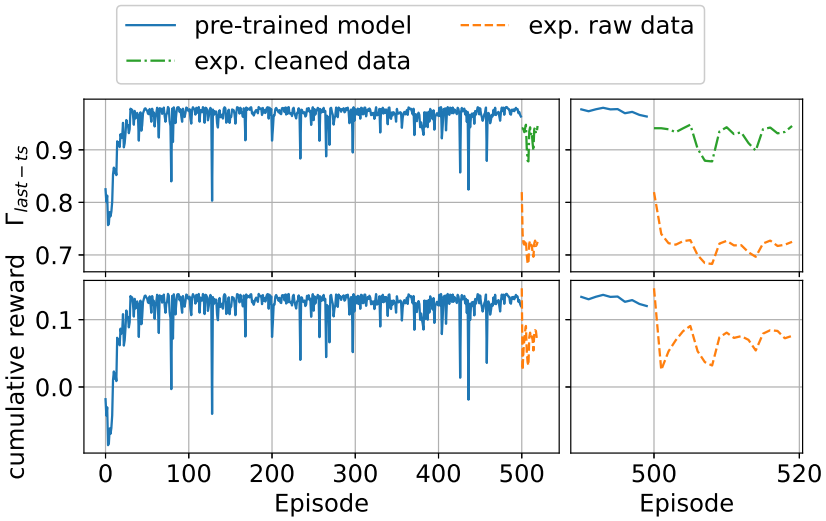


Figure 4.4.: Results of the wind tunnel experiment with a pre-trained model: Integral forward flow fraction Γ at the last time step and the cumulative reward plotted over all 520 episodes (left) and the last 30 episodes (right).

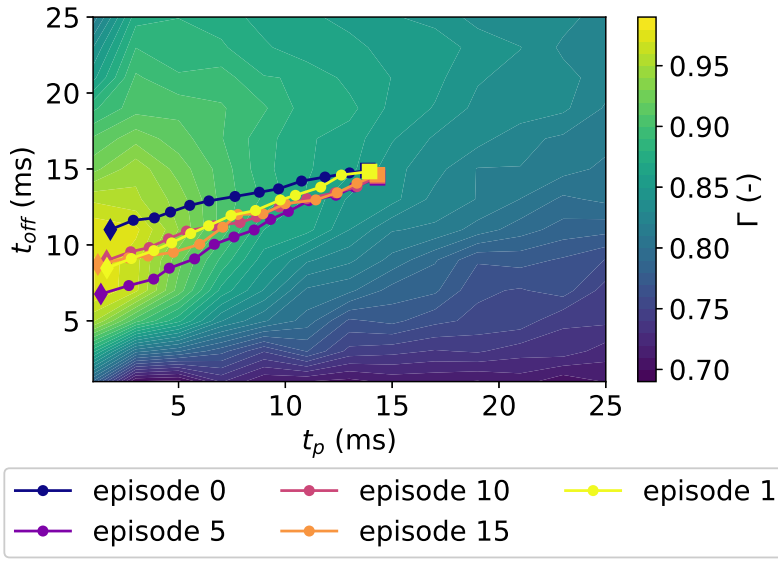


Figure 4.5.: Results of the wind tunnel experiment with a pre-trained model: Progression of the action space [t_{p}, t_{off}] during the episodes trained in the wind tunnel.

This experiment demonstrates that a model, pre-trained on the computer, can be applied in a model-free wind tunnel environment. Here, the model closely follows the previously learned strategy. This also shows that the interpolation of experimental data to calculate state and reward during the model-based training is a good approximation of the real wind tunnel environment.

4.2. Deterministic mode

In this section, a pre-trained PPO algorithm is executed in a deterministic mode to evaluate the application of the learned active control strategy. During this execution, no policy updates are performed, and the action is not sampled as a random variable from a Gaussian Normal distribution. Instead, the actor network directly returns the mean value μ to the DRL algorithm. An exemplary run of 100 episodes is conducted using a pre-trained model with the reward function $r = \Delta\Gamma$ and a standard deviation of $std = 0.2\,\mathrm{ms}$, as this combination has demonstrated the development of an effective control strategy during model-based training. The progression of the action space throughout this run is depicted in Figure 4.6. In deterministic mode, all episodes follow an identical trajectory, consistently reaching the same t_p and t_off values at each time step. Furthermore, the algorithm employs the maximum possible step size of $\Delta t_\mathrm{p} \approx -1\,\mathrm{ms}$ to reach the region of high integral forward flow fraction Γ in the minimal number of steps.

In conclusion, the results obtained in deterministic mode align with the expectations. When the DRL algorithm is not in training mode but instead applies the learned strategy, each episode strictly adheres to the policy developed during training.

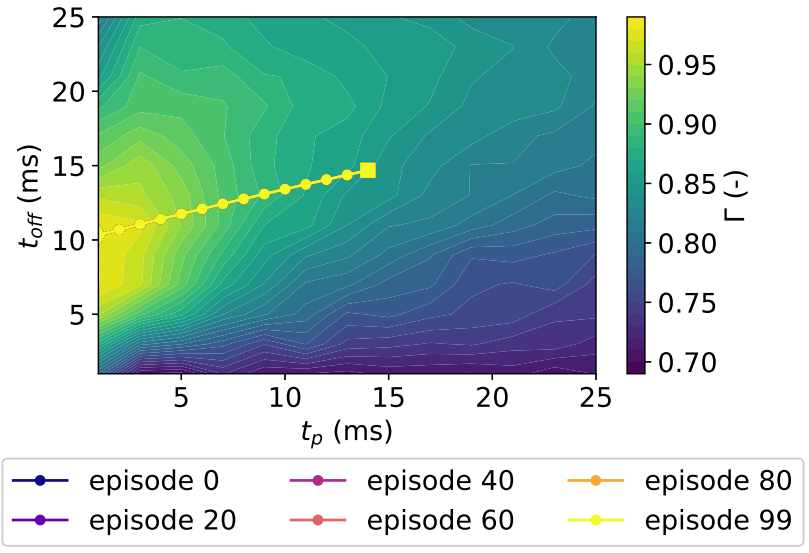


Figure 4.6.: Progression of the action space [t_p, t_off] during the application of the DRL algorithm with $r = \Delta\Gamma$ and $std = 0.2\,\mathrm{ms}$ in deterministic mode.

4.3. Random initialization

So far each learning episode starts at the same initial values for the pulse parameters t_p and t_off. Thinking about possible active flow control applications in a real, non-testbed environment, a DRL algorithm might be subject to variable input parameters. Therefore this section examines the algorithms performance with random initialization of the actuation parameters at the beginning of each episode.

Figure 4.7 shows an example model-based run for the reward function $r = \Delta\Gamma$ at a standard deviation $std = 0.2\,\mathrm{ms}$. Because the learning is expected to be much slower, the algorithm is trained for 10,000 episodes with 50 time steps per episode. The distribution of Γ at the last time step and the cumulative reward show that the algorithm manages to reach regions of high reward, but not consistently. The whole learning process is subject to a high degree of randomness.

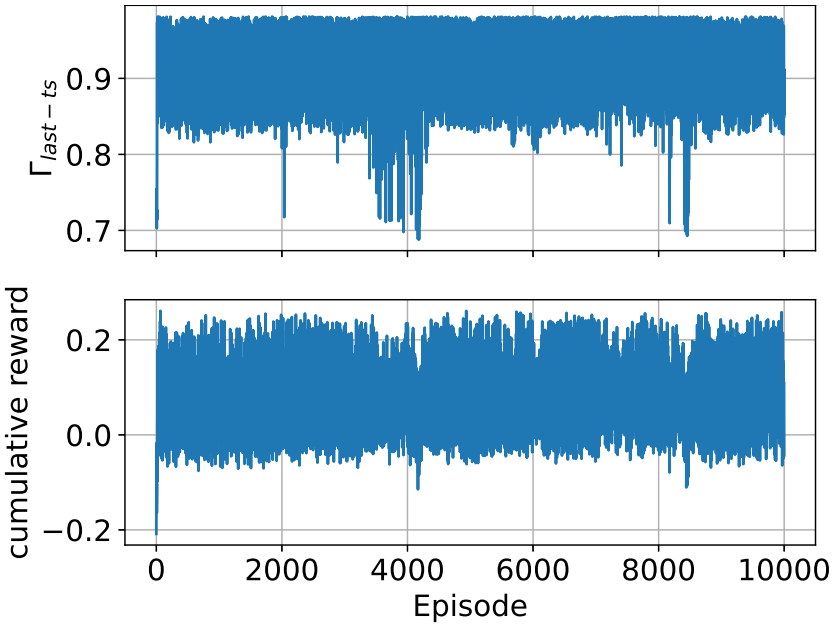


Figure 4.7.: Result of the model-based training with random initialization, $r = \Delta\Gamma$ and $std = 0.2\,\mathrm{ms}$: Integral forward flow fraction Γ at the last time step and the cumulative reward plotted over the number of episodes.

To gain a deeper understanding of the observed behavior, Figure 4.8 illustrates the progression of the action space over different sets of episodes. The algorithm appears to follow a similar trajectory through the parameter space for hundreds of consecutive episodes, regardless of the initial values for t_p and t_off. This behavior contradicts the intended function of the DRL algorithm. Ideally, the algorithm should determine the shortest path to the optimum from any point in the parameter space. Instead, it learns the gradient of previously successful episodes and applies this gradient to following episodes.

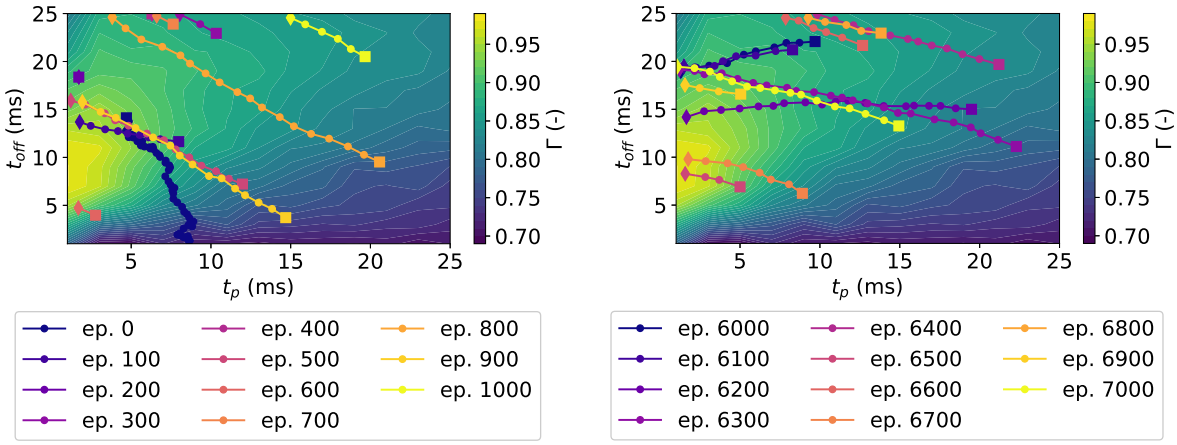


Figure 4.8.: Result of the model-based training with random initialization, $r = \Delta\Gamma$ and $std = 0.2\,\text{ms}$: Progression of the action space $[t_\text{p},\ t_\text{off}]$ for two sets of consecutive episodes.

A potential approach to altering this behavior is modifying the reward function. The reward function $r = \Gamma_{sum}$ may be more effective for random initialization, as it accounts for the mean value of Γ accumulated over all preceding time steps. However, the distribution of Γ and cumulative reward across all episodes (see Figure 4.9) indicates that $r = \Gamma_{sum}$ does not outperform $r = \Delta\Gamma$. Although a sequence of episodes (3400–4000) exhibits favorable results, a substantial range of episodes (500–1600) produces poor outcomes. Furthermore, beyond 4000 episodes, the results vary greatly, suggesting that the algorithm fails to establish an effective strategy. When analyzing sets of consecutive episodes, as depicted in Figure 4.10, the same behavioral pattern observed for $r = \Delta\Gamma$ emerges. The algorithm follows a specific gradient through the parameter space rather than attempting to determine the path to the optimal integral forward flow fraction.

This finding demonstrates that, while the DRL algorithm successfully develops flow control strategies for three different reward functions in both model-based and model-free training, it predominantly learns to follow an optimal gradient from the starting point to the optimum rather than optimizing the trajectory itself. The overarching objective remains for the DRL algorithm to identify the shortest route to the optimum, independent of the initial combination of forcing parameters t_p and t_off. Another potential solution could involve the implementation of a composite reward function that incorporates a final reward for the entire episode in addition to the rewards per time step. This concept is further discussed in the conclusions.

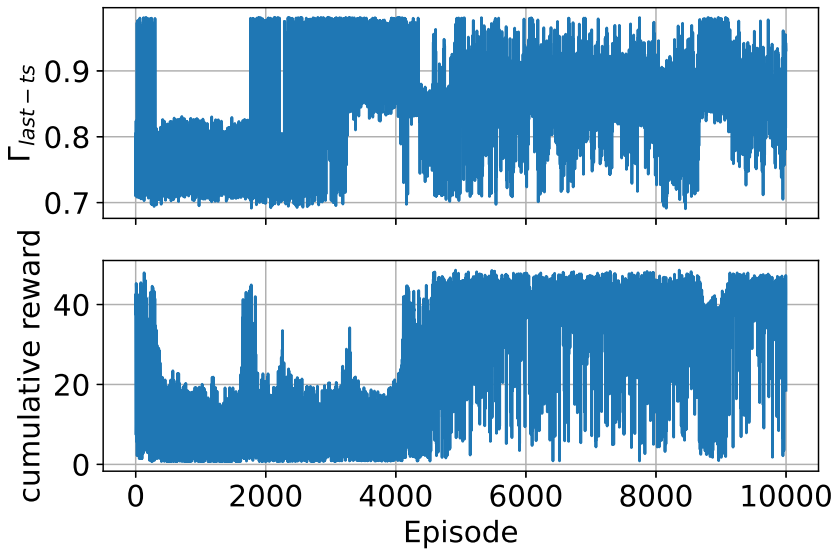


Figure 4.9.: Result of the model-based training with random initialization, $r = \Gamma_{\text{sum}}$ and $std = 0.2\,\text{ms}$: Integral forward flow fraction Γ at the last time step and the cumulative reward plotted over the number of episodes.

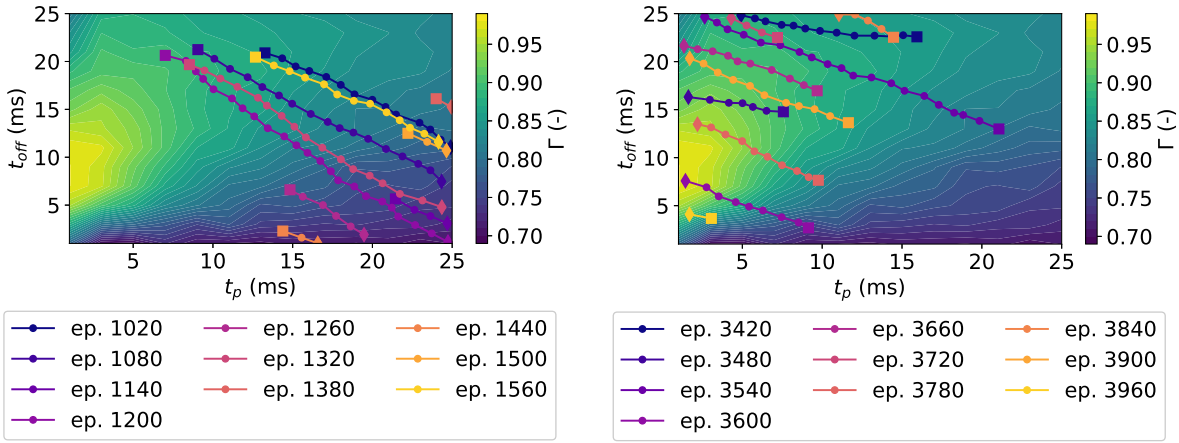


Figure 4.10.: Result of the model-based training with random initialization, $r = \Gamma_{\text{sum}}$ and $std = 0.2\,\text{ms}$: Progression of the action space $[t_{\text{p}},\ t_{\text{off}}]$ for two sets of consecutive episodes.

5. Conclusion

In this thesis, a Deep Reinforcement Learning (DRL) framework based on the Proximal Policy Optimization (PPO) algorithm [29] is successfully implemented to optimize parameters for active flow control in a fully turbulent one-sided diffuser flow. The algorithm varies the forcing signal of five Pulsed Jet Actuators (PJAs) located at the top of the diffuser ramp, with the primary objective of reducing the turbulent separation bubble within the diffuser. The forcing signal is composed of the pulse time t_p and the time between successive pulses t_off. The action, that the DRL agent passes to the environment, corresponds to incremental changes of t_p and t_off. The state of the environment, more specifically the diffuser flow, is defined by the distribution of the forward flow fraction γ, which is measured using an array of eight wall-shear-stress sensors. This thesis examines three different reward functions, that are all derived from the integral forward flow fraction Γ. The DRL algorithm is implemented in Python using the machine learning package PyTorch. The corresponding code is based on PPO implementations by [37] and [38] and can be accessed via *GitHub* [36].

A primary experimental investigation of the parameter space reveals an optimum in the integral forward flow fraction Γ at a low pulse time t_p, corresponding to a low duty cycle DC. This finding aligns with results reported by Steinfurth et al. [14] and Löffler et al. [15], both of whom optimized the flow control parameters for the same one-sided diffuser flow investigated in the present thesis. While Steinfurth et al. employed a conventional parameter study, Löffler et al. utilized surrogate modeling to achieve their results.

Model-based DRL is performed for three different reward functions at three different standard deviations for action selection representing either exploration or exploitation. During model-based training, state and reward are derived from interpolation of the measured data from the preliminary parameter study. All episodes start with fixed initial values of t_p and t_off. The two higher standard deviations $std = 0.2\,\mathrm{ms}$ and $0.4\,\mathrm{ms}$, corresponding to exploration, yield effective optimization strategies across all reward functions. The main disadvantage of exploration is that the DRL algorithm may randomly deviate from a successful strategy. In contrast, the small standard deviation $std = 0.001\,\mathrm{ms}$, corresponding to exploitation, closely follows the policy. However, it performs poorly due to insufficient exploration in the early stages of training. To combine the advantages of both methods, it is suggested to start the training with a high standard deviation to explore the parameter space until a good strategy is found, and then switch to a small standard deviation to exploit the found strategy.

The observed behavior aligns with findings in other Reinforcement Learning (RL) applications that encounter the conflict between exploration and exploitation. A well-known example illustrating this conflict is the multi-armed bandit problem, a sequential decision-making process. A simplified version of this problem considers a gambler who, at each time step, must choose between N slot machines, commonly referred to as 'armed bandits'. The reward associated with each machine follows a stochastic distribution, from which a random sample is drawn upon selection. The objective is to identify the machine yielding the highest reward through a series of iterative choices [39]. Since the gambler lacks prior knowledge of the reward distribution for each action (i.e., the selection of a slot machine), an estimate must be formed based on previous observations. There are two fundamental strategies to solve this problem: exploration and exploitation. In the case of pure exploration, the gambler selects a slot machine at random at each time step. While this approach ensures comprehensive data collection across all machines, it fails to consistently reach regions of high rewards, as the decision process remains random [33]. This behavior can be related to the PPO results characterized by a high standard deviation of $std = 0.4\,\text{ms}$: although the algorithm achieves good control strategies in a relatively short time, there are significant variations in the performance. At the other extreme, pure exploitation selects the machine with the highest estimated reward at every time step. However, since this estimation is based on past observations, the strategy remains susceptible to incomplete or misleading information. Consequently, the selected machine may not necessarily be the one that offers the highest actual reward [33]. A similar phenomenon appears to have occurred in the PPO algorithm with the lowest investigated standard deviation of $std = 0.001\,\text{ms}$. The algorithm converged to the estimated optimal strategy but failed to sufficiently explore the parameter space, leading to suboptimal performance. As observed in the PPO algorithm examined in this thesis, the most effective approach for solving the multi-armed bandit problem involves a combination of both exploration and exploitation. One of the simplest implementations of this principle is the ε-greedy strategy, where a random machine is selected with probability ε. During the remaining $1 - \varepsilon$ fraction of time steps the machine with the highest estimated reward is chosen [33]. More advanced techniques include Thompson Sampling (Bayesian) [40] and the Upper-Confidence-Bound Algorithm [41].

Among the investigated reward functions, $r = \Delta\Gamma$ is found to be the most efficient, achieving values around $\Gamma = 0.96$ after only 50 episodes with a standard deviation of $std = 0.4\,\text{ms}$. This efficiency can be attributed to the simplicity of the parameter space, which contains a single global optimum without local maxima or minima. The algorithm is able to follow the positive gradient directly to the optimum with ease. In fact, other techniques may be viewed as more suitable for the task at hand. Nonetheless, applying DRL to this case does not only increase the knowledge on this specific topic, but also prepares for the next investigation approaches.

To validate the previous results and assess whether the DRL algorithm performs efficiently in a real-world environment, additional wind tunnel experiments are conducted. In the

first of these model-free approaches, a DRL algorithm is trained from scratch in the wind tunnel. For this, the sample-efficient combination of the reward function $r = \Delta\Gamma$ with a standard deviation of $std = 0.4\,\text{ms}$ is trained over 50 episodes. During this experiment, a measurement error is identified, leading to partially incorrect values for the integral forward flow fraction Γ. Nevertheless, this error does not influence the DRL algorithm, which develops a successful optimization strategy.

In the second experiment, the behavior of a pre-trained model is examined. The DRL algorithm first undergoes model-based training for 500 episodes using the same reward function $r = \Delta\Gamma$ and standard deviation $std = 0.4\,\text{ms}$. The pre-trained model is deployed in the wind tunnel for an additional 20 episodes and with a lower standard deviation of $std = 0.2\,\text{ms}$, to enhance exploitation. In the wind tunnel, the model closely follows the previously learned strategy. This result also demonstrates that the interpolation of experimental data during the model-based training is a good approximation of the real wind tunnel environment.

To evaluate the application of the learned active control strategy, a pre-trained DRL algorithm is executed in a deterministic mode. In deterministic mode, no policy updates are performed and the action is directly provided by the actor network, instead of being sampled from a Normal distribution. As expected, the algorithm strictly adheres to the previously learned policy.

Up to this point, each learning episode begins with the same initial values for t_p and t_off. Considering potential applications of active flow control in real-world, non-testbed environments, it is likely that a DRL algorithm will encounter varying input parameters. Therefore, the algorithm's performance with randomly initialized actuation parameters at the start of each episode is evaluated for the two reward functions $r = \Delta\Gamma$ and $r = \Gamma_\text{sum}$. The results reveal that for consecutive episodes, the algorithm follows a similar trajectory through the parameter space, regardless of the initial values for t_p and t_off. This behavior deviates from the intended purpose of the DRL algorithm. In an ideal scenario, the algorithm should identify the most direct path to the optimum, regardless of its starting position within the parameter space. However, rather than independently determining the optimal trajectory, it instead captures the gradient of previously successful episodes and replicates it in subsequent episodes.

A potential approach to improve this behavior is the introduction of a composite reward function that considers a final reward for the whole episode additional to the rewards per time step. In the present thesis, only a single reward per time step r_timestep has been considered. At the end of each episode, an additional final episode reward r_episode could be incorporated into the reward structure. The total reward r would then be defined as follows:

$$r = f_\text{timestep} \cdot r_\text{timestep} + f_\text{episode} \cdot r_\text{episode} \tag{5.1}$$

The weighting factors f_timestep and f_episode allow for adjusting the relative influence of the

time step and episode rewards. Setting $f_{\text{timestep}} = 1$ and $f_{\text{episode}} = 0$ implies that only the time step reward contributes to the policy update, while the opposite choice fully excludes r_{timestep} in favor of the episode reward. Potential definitions for the episode reward include the integral forward flow fraction at the final time step,

$$r_{\text{episode}} = \Gamma_{\text{last-ts}} \tag{5.2}$$

or the median integral forward flow fraction across all n time steps t within the episode:

$$r_{\text{episode}} = \frac{\sum_{t=0}^{n} \Gamma_t}{n} \tag{5.3}$$

In conclusion, the DRL algorithm investigated in this thesis successfully develops flow control strategies for three different reward functions in both model-based and model-free training, when starting each episode from fixed initial pulse parameters. Example training runs with random initialization reveal, that the algorithm predominantly learns to follow an optimal gradient from the starting point to the optimum rather than optimizing the trajectory itself, thus contradicting the overarching objective. As a solution, the introduction of an additional episode reward is suggested.

Bibliography

[1] Sarker, Iqbal H.: Machine Learning: Algorithms, Real-World Applications and Research Directions. In: *SN Computer Science* 2 (2021), 160. http://dx.doi.org/10.1007/s42979-021-00592-x. – DOI 10.1007/s42979–021–00592–x. – ISSN 2661–8907

[2] Khadka, Rajesh: *Machine Learning Types #2.* https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1. Version: 2017. – Accessed: 12/2024

[3] Géron, Aurélien: *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Second edition. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo : O'Reilly, 2019. – ISBN 9781492032649

[4] Garnier, Paul ; Viquerat, Jonathan ; Rabault, Jean ; Larcher, Aurélien ; Kuhnle, Alexander ; Hachem, Elie: A review on deep reinforcement learning for fluid mechanics. In: *Computers & Fluids* 225 (2021), 104973. http://dx.doi.org/https://doi.org/10.1016/j.compfluid.2021.104973. – DOI https://doi.org/10.1016/j.compfluid.2021.104973. – ISSN 0045–7930

[5] Dubs, Fritz: *Aerodynamik der reinen Unterschallströmung.* 6. Auflage. Birkhäuser, 1990 (Flugtechnische Reihe BV005542824 1). – ISBN 3764318724

[6] Cumpsty, N. A.: *Compressor aerodynamics.* Reprinted ed. w/new preface, introd., and updated bibliogr. Krieger Publishing Company, 2004. – ISBN 1575242478

[7] Joshi, Sanket N. ; Gujarathi, Yash S.: A review on active and passive flow control techniques. In: *International Journal on Recent Technologies in Mechanical and Electrical Engineering* 3 (2016), Nr. 4, S. 1–6

[8] Prandtl, Ludwig ; Betz, Albert: *Vier Abhandlungen zur Hydrodynmaik und Aerodynamik.* Andreas Dillmann, Universitätsverlag Göttingen, 2010 (Göttinger Klassiker der Strömungsmechanik)

[9] Greenblatt, David ; Wygnanski, Israel J.: The control of flow separation by periodic excitation. In: *Progress in Aerospace Sciences* 36 (2000), Nr. 7, 487-545. http://dx.doi.org/https://doi.org/10.1016/S0376-0421(00)00008-7. – DOI https://doi.org/10.1016/S0376–0421(00)00008–7. – ISSN 0376–0421

[10] Seifert, A. ; Darabi, A. ; Wyganski, I.: Delay of airfoil stall by periodic excitation. In: *Journal of Aircraft* 33 (1996), Nr. 4, 691-698. http://dx.doi.org/10.2514/3.47003. – DOI 10.2514/3.47003

[11] Collins, Frank G. ; Zelenevitz, James: Influence of sound upon separated flow over wings. In: *AIAA journal* 13 (1975), Nr. 3, S. 408–410

[12] Bremm, Martin ; Stephan, Ralph ; Hörnschemeyer, Ralf ; Stumpf, Eike: Dynamic Trailing-Edge Flap Movement as a Non-Conventional Aerodynamic Lift Mechanism. In: *32nd Congress of the International Council of the Aeronautical Sciences, Shanghai, China*, 2021

[13] Bremm, Martin ; Hörnschemeyer, Ralf ; Stumpf, Eike: EXPERIMENTS TO INVESTIGATE LIFT PRODUCTION MECHANISMS ON OSCILLATING TRAILING-EDGE FLAPS. In: *33nd Congress of the International Council of the Aeronautical Sciences, Stockholm, Sweden*, 2022

[14] Steinfurth, Ben ; Weiss, Julien: Efficiency Enhancement in Active Separation Control Through Optimizing the Duty Cycle of Pulsed Jets. In: *AIAA Journal* 60 (2022), Nr. 12, 6566-6580. http://dx.doi.org/10.2514/1.J061667. – DOI 10.2514/1.J061667

[15] In: Löffler, Stephan ; Steinfurth, Ben ; Weiss, Julien: *Surrogate-Based Exploration of Active Separation Control Parameters: An Experimental Study.* 2023

[16] Huang, Liang ; Huang, George ; LeBeau, Raymond ; Hauser, Thomas: Optimization of Airfoil Flow Control Using a Genetic Algorithm with Diversity Control. In: *Journal of Aircraft* 44 (2007), Nr. 4. http://dx.doi.org/10.2514/1.27020. – DOI 10.2514/1.27020

[17] Montazer, E ; Mirzaei, M ; Salami, E ; Ward, T A. ; Romli, F I. ; Kazi, S N.: Optimization of a synthetic jet actuator for flow control around an airfoil. In: *IOP Conference Series: Materials Science and Engineering* 152 (2016), Nr. 1. http://dx.doi.org/10.1088/1757-899X/152/1/012023. – DOI 10.1088/1757–899X/152/1/012023

[18] Rabault, Jean ; Ren, Feng ; Zhang, Wei ; Tang, Hui ; Xu, Hui: Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. In: *Journal of Hydrodynamics* 32 (2020), 234-246. http://dx.doi.org/https://doi.org/10.1007/s42241-020-0028-y. – DOI https://doi.org/10.1007/s42241–020–0028–y

[19] Pinto, Lerrel ; Andrychowicz, Marcin ; Welinder, Peter ; Zaremba, Wojciech ; Abbeel, Pieter: *Asymmetric Actor Critic for Image-Based Robot Learning.* https://arxiv.org/abs/1710.06542. Version: 2017

[20] Bahdanau, Dzmitry ; Brakel, Philemon ; Xu, Kelvin ; Goyal, Anirudh ; Lowe, Ryan ; Pineau, Joelle ; Courville, Aaron ; Bengio, Yoshua: *An Actor-Critic Algorithm for Sequence Prediction*. https://arxiv.org/abs/1607.07086. Version: 2017

[21] Mnih, Volodymyr ; Kavukcuoglu, Koray ; Silver, David ; Graves, Alex ; Antonoglou, Ioannis ; Wierstra, Daan ; Riedmiller, Martin: *Playing Atari with Deep Reinforcement Learning*. https://arxiv.org/abs/1312.5602. Version: 2013

[22] Rabault, Jean ; Kuchta, Miroslav ; Jensen, Atle ; Réglade, Ulysse ; Cerardi, Nicolas: Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. In: *Journal of Fluid Mechanics* 865 (2019), S. 281–302. http://dx.doi.org/10.1017/jfm.2019.62. – DOI 10.1017/jfm.2019.62

[23] Ren, Feng ; Rabault, Jean ; Tang, Hui: Applying deep reinforcement learning to active flow control in weakly turbulent conditions. In: *Physics of Fluids* 33 (2021), 03, Nr. 3, 037121. http://dx.doi.org/10.1063/5.0037371. – DOI 10.1063/5.0037371. – ISSN 1070–6631

[24] Fan, Dixia ; Yang, Liu ; Wang, Zhicheng ; Triantafyllou, Michael S. ; Karniadakis, George E.: Reinforcement learning for bluff body active flow control in experiments and simulations. In: *Proceedings of the National Academy of Sciences* 117 (2020), Nr. 42, 26091-26098. http://dx.doi.org/10.1073/pnas.2004939117. – DOI 10.1073/pnas.2004939117

[25] Tokarev, Mikhail ; Palkin, Egor ; Mullyadzhanov, Rustam: Deep Reinforcement Learning Control of Cylinder Flow Using Rotary Oscillations at Low Reynolds Number. In: *Energies* 13 (2020), Nr. 22. https://doi.org/10.3390/en13225920

[26] Shimomura, Satoshi ; Sekimoto, Satoshi ; Oyama, Akira ; Fujii, Kozo ; Nishida, Hiroyuki: Closed-Loop Flow Separation Control Using the Deep Q Network over Airfoil. In: *AIAA Journal* 58 (2020), Nr. 10, 4260-4270. http://dx.doi.org/10.2514/1.J059447. – DOI 10.2514/1.J059447

[27] Zheng, Changdong ; Ji, Tingwei ; Xie, Fangfang ; Zhang, Xinshuai ; Zheng, Hongyu ; Zheng, Yao: From active learning to deep reinforcement learning: Intelligent active flow control in suppressing vortex-induced vibration. In: *Physics of Fluids* 33 (2021), 06, Nr. 6, 063607. http://dx.doi.org/10.1063/5.0052524. – DOI 10.1063/5.0052524. – ISSN 1070–6631

[28] Viquerat, J. ; Meliga, P. ; Larcher, A. ; Hachem, E.: A review on deep reinforcement learning for fluid mechanics: An update. In: *Physics of Fluids* 34 (2022), 11, Nr. 11, 111301. http://dx.doi.org/10.1063/5.0128446. – DOI 10.1063/5.0128446. – ISSN 1070–6631

[29] Schulman, John ; Wolski, Filip ; Dhariwal, Prafulla ; Radford, Alec ; Klimov, Oleg: Proximal Policy Optimization Algorithms. (2017). http://dx.doi.org/arXiv:1707.06347v2. – DOI arXiv:1707.06347v2

[30] Simeone, Osvaldo: *Machine Learning for Engineers*. First edition. Cambridge University Press, 2022. http://dx.doi.org/10.1017/9781009072205. http://dx.doi.org/10.1017/9781009072205

[31] Bellemare, Marc G. ; Naddaf, Yavar ; Veness, Joel ; Bowling, Michael: The arcade learning environment: An evaluation platform for general agents. In: *Journal of Artificial Intelligence Research* 47 (2013), S. 253–279

[32] Duan, Yan ; Chen, Xi ; Houthooft, Rein ; Schulman, John ; Abbeel, Pieter: Benchmarking deep reinforcement learning for continuous control. In: *International conference on machine learning* PMLR, 2016, S. 1329–1338

[33] Sutton, Richard S. ; Barto, Andrew G.: *Reinforcement learning : an introduction*. Second edition. Cambridge, Massachusetts ; London, England : The MIT Press, 2018 (Adaptive computation and machine learning series). – ISBN 9780262039246

[34] Barto, Andrew G. ; Sutton, Richard S. ; Anderson, Charles W.: Neuronlike adaptive elements that can solve difficult learning control problems. In: *IEEE transactions on systems, man, and cybernetics* (1983), Nr. 5, S. 834–846

[35] Schulman, John ; Levine, Sergey ; Abbeel, Pieter ; Jordan, Michael ; Moritz, Philipp: Trust Region Policy Optimization. In: *Proceedings of the 32nd International Conference on Machine Learning* Bd. 37. Lille, France : PMLR, 07–09 Jul 2015 (Proceedings of Machine Learning Research), 1889–1897

[36] Müller, Alexandra: *AFC optimization with PPO*. https://github.com/alex-mueller259/afc_optimization_with_ppo. Version: 2024. – Accessed: 02/2025

[37] henanmemeda: *RL-Adventure-2*. https://github.com/henanmemeda/RL-Adventure-2/blob/master/3.ppo.ipynb. Version: 2018

[38] Tabor, Phil: *Youtube-Code-Repository*. https://github.com/philtabor/Youtube-Code-Repository/tree/master/ReinforcementLearning/PolicyGradient/PPO/torch. Version: 2020

[39] Bubeck, Sébastien ; Munos, Rémi ; Stoltz, Gilles: Pure Exploration in Multi-armed Bandits Problems. In: Gavaldà, Ricard (Hrsg.) ; Lugosi, Gábor (Hrsg.) ; Zeugmann, Thomas (Hrsg.) ; Zilles, Sandra (Hrsg.): *Algorithmic Learning Theory*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. – ISBN 978–3–642–04414–4, S. 23–37

[40] Scott, Steven L.: A modern Bayesian look at the multi-armed bandit. In: *Applied Stochastic Models in Business and Industry* 26 (2010), Nr. 6, 639-658. http://dx.doi.org/https://doi.org/10.1002/asmb.874. – DOI https://doi.org/10.1002/asmb.874

[41] Carpentier, Alexandra ; Lazaric, Alessandro ; Ghavamzadeh, Mohammad ; Munos, Rémi ; Auer, Peter: Upper-Confidence-Bound Algorithms for Active Learning in Multi- armed Bandits. In: Kivinen, Jyrki (Hrsg.) ; Szepesvári, Csaba (Hrsg.) ; Ukkonen, Esko (Hrsg.) ; Zeugmann, Thomas (Hrsg.): *Algorithmic Learning Theory*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–24412–4, S. 189–203