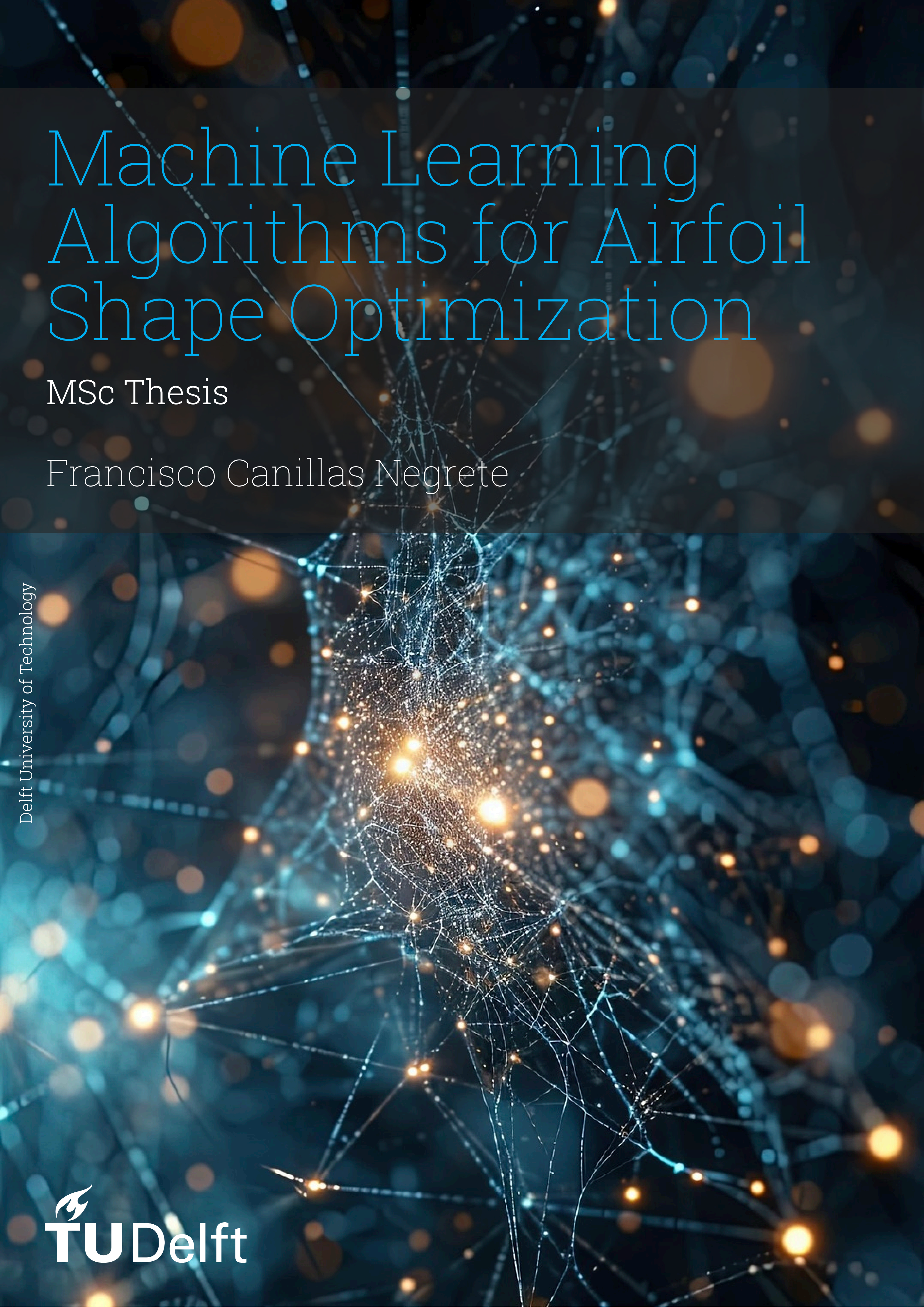


Machine Learning Algorithms for Airfoil Shape Optimization

MSc Thesis

Francisco Canillas Negrete

Delft University of Technology

TU Delft

Machine Learning Algorithms for Airfoil Shape Optimization

MSc Thesis

by

Francisco Canillas Negrete

Student number:	4910885	
Project duration:	November, 2024 – September, 2025	
Thesis committee:	Dr. N.A.K. Doan,	TU Delft, supervisor
	Dr. C. Varriale,	TU Delft, examiner
	Dr.ir. A.H. van Zuijlen,	TU Delft, chair

Faculty of Aerospace Engineering

TU Delft

Abstract

Airfoil shape optimization plays an important role in improving aerodynamic performance in aircraft and turbine applications. Recent advances in computational methods have allowed the integration of machine learning algorithms into aerodynamic design workflows. This MSc thesis investigates and compares two optimization algorithms, Factorization Machine Quantum Annealing (FMQA) and single-step Deep Reinforcement Learning (sDRL) in the context of airfoil shape optimization, including a comparison against a standard optimizer from SciPy (Differential Evolution).

A series of numerical experiments were conducted, interfacing these algorithms with a Lattice Boltzmann Method (LBM) solver across different aerodynamic objectives, flow regimes, and parameterization schemes, including classical approaches such as the Joukowski transformation and PARSEC, as well as a novel Variational Autoencoder (VAE)-based scheme. Benchmark cases, including an adapted version of Drela's optimization problem, were also carried out to further evaluate the algorithms. Performance was assessed mainly through the aerodynamic objective values obtained, the convergence behaviour, and the analysis of the optimized airfoil shapes.

Results show that both FMQA and sDRL successfully identified high performing airfoil shapes under different conditions. Convergence behaviour varied between the numerical experiments, with clearer and more consistent convergence trends observed when using the VAE-based parameterization. sDRL often achieved slightly better results in most of the aerodynamic objectives, including the maximization of the lift-to-drag ratio, while FMQA outperformed in specific numerical experiments. Compared with the standard optimizer, FMQA and sDRL produced competitive results: in some numerical experiments, the machine learning algorithms outperformed the SciPy optimizer (especially sDRL), while in others the standard optimizer achieved better performance. Testing the VAE-based parameterization confirmed that this scheme leads to improved aerodynamic performance across different optimization objectives compared to classical parameterization approaches.

These findings highlight FMQA and sDRL as flexible and competitive optimization strategies for aerodynamic design problems, outperforming the standard SciPy optimizer in some challenging scenarios and showing strong potential for application to practical, higher fidelity design scenarios.

Acknowledgements

With this document, I present the research conducted and the findings obtained as part of my MSc thesis work.

First, I would like to thank my supervisor, Anh Khoa Doan, for granting me the opportunity to carry out this research under his guidance and for providing support and advice throughout all stages of the thesis.

I am also very thankful to my family and friends for their encouragement and for motivating me to strive for the best version of myself throughout my studies.

Francisco Canillas Negrete
Delft, August 2025

Contents

List of Figures

List of Tables

Nomenclature

Symbols

Symbol	Definition
α	Angle of attack
a	Action in reinforcement learning
b	Bias in a neuron
c_i	Constraint function in optimization
c_s	Lattice speed of sound
C_D	Drag coefficient
C_L	Lift coefficient
C_M	Moment coeffcient
D	Drag force
f	Objective function in optimization
γ	Discount factor in a reward
K	Rank of the FM
L	Lift force
Ma	Mach number
$\nabla_{\mathbf{x}} J$	Gradient of the objective function J with respect to $\mathbf{x}$
ν	Kinematic viscosity
q_π	Action-value function following the policy π
r_t	Immediate reward received at the time step t
Re	Reynolds number
σ	Activation function
θ	Policy parameters
U	Inflow velocity
v_π	State-value function following the policy π
$\mathbf{v}_i$	FM model parameter for K-dimensional real vector
$\mathbf{w}$	Weight vector connecting neurons between two layers
w_i	FM model parameter for real scalar
$\mathbf{x}$	Vector of variables
Z_i	Latent paremeters

Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
CFD	Computational Fluid Dynamics
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
FMQA	Factorization Machine Quantum Annealing
FM	Factorization Machine
FSR	Function Smoothing Regularization
GP	Gaussian Process
LBM	Lattice Boltzmann Method
ML	Machine Learning
NN	Neural Network
PBO	Policy Based Optimization
PPO	Proximal Policy Optimization
QA	Quantum Annealing
QUBO	Quadratic Unconstrained Binary Optimization
RL	Reinforcement Learning
sDRL	Single-step Deep Reinforcement Learning
UIUC	University of Illinois at Urbana-Champaign
VAE	Variational Autoencoder

Chapter 1

Introduction

Airfoil shape optimization is a well established research area with many applications in fluid dynamics. In areas like aerodynamics, critical challenges like minimizing the drag or reducing aircraft fuel consumption are often addressed through shape optimization [1]. Shape optimization is crucial in aircraft and turbine design, significantly impacting both aerodynamic efficiency and energy consumption. The primary objective of the optimization process is to enhance the lift-to-drag ratio, i.e. the aerodynamic performance, by modifying the airfoil geometry.

Traditional optimization methods rely predominantly on computational fluid dynamics (CFD) simulations. However, these methods are computationally expensive and often struggle with complex design spaces. Recent advances in machine learning are helping to address these challenges with the aid of surrogate models. Machine learning offers a flexible and adaptable modeling framework that can be customized to tackle various challenges in fluid mechanics, including reduced-order modeling, experimental data processing, and, in this specific case, airfoil shape optimization [5]. Deep learning (DL), as the main algorithm of artificial intelligence (AI), has remarkable abilities to process high-dimensional and nonlinear data representations. By integrating the capabilities of DL with the decision-making capabilities of reinforcement learning (RL), deep reinforcement learning (DRL) offers an innovative approach to addressing complex system perception and decision-making problems [6]. This approach has been successfully applied to shape optimization in [1] and [7].

In contrast, machine learning approaches such as Factorization Machine Quantum Annealing (FMQA) algorithms have also emerged with potential in many research areas. Examples of these are found for metamaterials in [8] and function smoothing regularization in [9]. Applying it to shape optimization, the approach found in [10] has shown the applicability of this algorithm to aerodynamic topics such as designing airfoil geometries to improve their aerodynamic efficiency.

This research project will focus on developing an assessing recent machine learning algorithms for the problem of airfoil shape optimization. As it has previously been stated, Factorization Machine Quantum Annealing (FMQA) and Deep Reinforcement Learning have shown promises in tackling this problem. However, several open questions remain on their comparative performance and on which of these two algorithms is more suitable.

Therefore, the main research objective of this thesis is to explore and compare Factorization Machine Quantum Annealing (FMQA) and single-step Deep Reinforcement Learning (sDRL) algorithms in the context of airfoil shape optimization by setting tasks of increasing complexity, attempting to understand and explain their observed respective performances. In addition, a standard optimizer is employed as a baseline for comparison and validation of the results of the machine learning approaches.

The report is structured as follows. Chapter 2 presents the literature study, which forms the basis of this MSc thesis. This chapter introduces the main concepts and summarizes the key findings from the literature regarding the different methods considered. Chapter 3 outlines the research methodology, including a description of the methods, the computational setup, and the design of the numerical experiments. Chapter 4 includes the results obtained from the numerical experiments, followed by the consequent discussions. Finally, the main conclusions are provided in Chapter 5, and recommendations are presented in Chapter 6.

Chapter 2

Literature Study

The Literature Study has the goal of exploring and defining the fundamentals of these two machine learning algorithms, DRL and FMQA in the context of airfoil shape optimization. Firstly, a background section on the fundamentals of optimization and neural networks is covered. Then, DRL and FMQA will be compared in the context of airfoil shape optimization, highlighting their strengths and weaknesses. Additionally, a section on airfoil shape representation is also included to cover the different methodologies to represent the shapes for the future optimization problems of this research work. Then, the research objectives and research questions are presented as a bridge between the Literature Study and the actual MSc thesis work.

2.1 Background

2.1.1 Optimization Problem - Definition

Optimization is a relevant tool in decision science and in the analysis of any physical system. First, an objective should be identified to make use of this tool, that is, a measure of the performance of the system that is being studied. The objective depends on specific characteristics of the systems, called variables. In this case, the goal is to optimize the objective finding the right values of the variables. In many cases, these values are restricted or constrained by a set of equations and inequalities [11].

In mathematical terms, optimization is the minimization or maximization of a function subject to constraints on its variables. The key components of an optimization problem are:

- $\mathbf{x}$ is the vector of variables, also called parameters to be optimized.
- f is the objective function, a function of $\mathbf{x}$ that we want to maximize of minimize.
- c_i are the constraint functions, certain equations and inequalities that the vector $\mathbf{x}$ must satisfy.

Using this notation, the optimization problem can be written as

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \text{subject to} \quad \begin{array}{ll} c_i(\mathbf{x}) = 0, & i \in \mathcal{E}, \\ c_i(\mathbf{x}) \geq 0, & i \in \mathcal{I}. \end{array} \tag{2.1}$$

where $\mathcal{E}$ is the set of indices corresponding to equality constraints and $\mathcal{I}$ the set of indices corresponding to inequality constraints. In airfoil shape optimization, the objective function to be optimized is usually the lift-to-drag ratio. On the other hand, airfoil shape parameters would correspond to the design variables of the problem.

Optimization is a fundamental component of machine learning, as most machine learning algorithms formulate learning as an optimization problem where parameters of a given model are adjusted to minimize or maximize an objective function based on the provided data [12]. In the following, the focus will be on the shape optimization problem, a type of optimization problem where the objective function depends on the geometry of a domain.

2.1.2 Gradient-based vs Gradient-free methods

In optimization, two main classes of approaches have emerged to deal with shape optimization problems: gradient-based and gradient-free methods. Gradient-based methods are dependent on the gradient of the objective function J with respect to the design parameters $\mathbf{x}$, expressed as $\nabla_{\mathbf{x}} J$ [1]. These methods have lower computational costs than gradient-free methods, if the gradient is easy to compute. However, they have some drawbacks. Firstly, gradient-based approaches have the tendency to being trapped in local minima of the function rather than finding the global minimum point. Therefore, they are very sensitive to the initial guess. Secondly, the efficiency of the methods presents challenges when the objective function exhibits discontinuities or nonlinear behavior [13]. In this context, gradient-free methods often perform better than gradient-based, despite their potentially more complex implementation [1]. Besides that, these methods are better at finding the global optima. Within gradient-free methods, genetic algorithms show a good behavior in finding the global optima. However, they are more expensive computationally than gradient-based methods [13]. Particle-swarm optimization is also highlighted by its easy implementation and low memory cost [14]. Nevertheless, it is not as good as other methods in imposing constraints on the design parameters. Lastly, metropolis algorithms, such as simulated annealing, are well-known for having an optimal behavior escaping local minima. However, the results show a high dependency on the chosen meta-parameters of the algorithm [1]. Table 2.1 shows in summary the comparison between gradient-based and gradient-free methods.

Table 2.1: Gradient-based versus gradient-free methods [4].

	Gradient-based	**Gradient-free**
Advantages	Lower computational cost	Good at finding global optimum
	Widely used method in aerodynamics	Well suited for complex functions (non-linear)
Drawbacks	Convergence on local optima	High computational cost
	Sensitive to starting point	Limitation on number of design variables
	Requires continuous function	Low convergence speed when coupled with CFD
	Inability to use past optimum data	Inability to use past optimum data
	Poor efficiency for non-linear cost function	

Currently, there is an extensive literature supporting the use of neural networks alongside gradient-based and gradient-free methods for shape optimization. Fundamentals of neural networks are thus now introduced.

2.1.3 Neural networks

A neural network (NN) can be conceptualized as a collection of artificial neurons inspired by the structure of the human brain. These neurons are connected computational units that have the capability to be trained to approximate the mapping function between input and output spaces. Each connection provides the output of a neuron as the input to another. As can be seen in Figure 2.1, an input vector $\mathbf{x}$, associated with a set of weights $\mathbf{w}$, is provided to the neuron. The neuron then calculates a weighted sum of the inputs $\mathbf{w} \cdot \mathbf{x} + b$ to represent the importance of each input for the learning task. It adds a bias b to account for parts of the output that are independent of the input, as shown in Equation 2.2. The weight and the bias represent the degrees of freedom in the neuron (parameters to be adjusted to approximate the function).

$$z = b + \sum_k w_k \cdot x_k \tag{2.2}$$

This sum is processed through an activation function σ, which determines the degree to which the computed value influences the final output. The activation function is a hyperparameter (it is part of the choices made during the network design) [15].

$$a = \sigma(z) \tag{2.3}$$

Neural networks are typically organized into layers. In this scenario, the input receives the external data, the output layer produces the final result, and in between there are zero or more hidden layers that transform the data via weighted summations and activations. For a fully connected network, the neurons of one layer are exclusively connected to neurons in the preceding and following layers. One can see in Figure 2.2 an example of how a neural network looks like, connecting three inputs x_1, x_2 and x_3 to an unique output y through a series of interconnected layers.

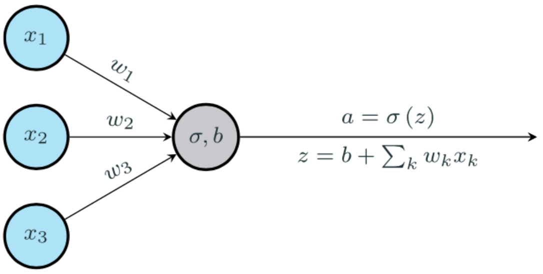


Figure 2.1: Artificial Neuron

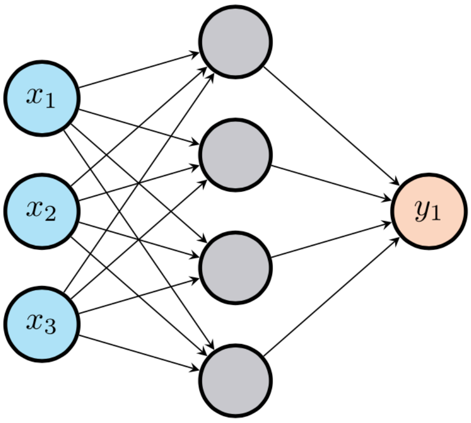


Figure 2.2: Neural Network

In order to design an efficient neural network, it is required to select appropriately nonlinear activation functions and optimize the weights and biases to minimize the value of a loss function that measures the network's prediction accuracy. The network architecture, the meta-parameters and the quality/size of the dataset are also factors to take into careful consideration for an efficient learning [3].

In the context of deep reinforcement learning, the agent (which will be discussed later in subsection 2.2.1) is a deep neural network (DNN). Here, the neural network learns to represent the relation between input (action) and output (reward) by adjusting the weights and biases. This adjustment is achieved through back-propagation, from the output layer to the input layer passing through the hidden layer. This process is known as training [7].

2.2 Algorithms in Airfoil Shape Optimization

Airfoil shape optimization is a crucial aspect of aerodynamic design. However, the complexity and non-linearity of fluid mechanics, combined with the high-dimensional design space, make optimizing airfoil shapes a particularly challenging task [4]. In this section, different algorithms to tackle the airfoil shape optimization problem are discussed.

In the late 1970s, one of the first papers on aerodynamic shape optimization was published by Hicks and Henne [16], where a conjugate gradient optimization algorithm was presented. Since then, aerodynamic shape optimization has become a very active area of research [17]. Classical airfoil shape optimization techniques have evolved from analytical methods to approaches based on computational methods. One classical method is conformal mapping, widely used in potential flow analysis. Using this technique, simple flow solutions can be transformed into complex airfoil shapes while preserving the main aerodynamic properties. However, conformal mapping has also its limitations. While it is effective for inviscid flows, it struggles in handling viscous effects, leading to the development of numerical methods [18].

An important improvement came with the introduction of adjoint methods, which use ideas from control theory (the study of how to guide systems toward better performance by adjusting inputs) and advanced mathematics to optimize shapes more efficiently [19]. In this context, the system being studied is the airflow around a shape such as an airfoil. Adjoint methods allow one to determine the impact of small shape changes on performance without needing to run a full simulation for every variation. This makes the design process much faster and more effective. These methods were introduced to address the issue of using a large number of design variables, which usually resulted in very high computational costs [20]. In [21] and [22], Pironneau made use of the adjoint-based gradient calculation in airfoil profile optimization by deriving the adjoint for the Stokes equations and for the incompressible Euler equations. Later, the adjoint method was also extended by Jameson [23] to the compressible Navier-Stokes equations with turbulent models, allowing to solve practical aerodynamic design problems. However, a key limitation of adjoint methods is their sensitivity to local minima, as they rely on gradient-based optimization and therefore there is no guarantee of finding the global optimum. Another important technique is numerical inverse design, an iterative process that adjusts the airfoil shape to achieve specific performance metrics. These methods have been applied to real-world problems, including business jet optimizations [18].

Another optimization technique to highlight is Bayesian Optimization, a data-driven technique that combines a surrogate model and an optimization algorithm, using updated prior knowledge [24]. In [25], Bayesian Optimization is applied to aerodynamic shape design of airfoils. This approach, also known as Efficient Global Optimization, is well-suited for optimization problems where the objective function is unknown or expensive to evaluate (one advantage of Bayesian Optimization is its applicability to general black-box functions [26]). In these scenarios, the objective function, such as aerodynamic drag or lift, is not given in an explicitly mathematical form or is costly to compute through simulations or experiments. To address this, Bayesian Optimization builds a probabilistic surrogate model that approximates the objective function based on previously evaluated data points. It consists of two essential aspects: a Bayesian regression model, usually a Gaussian process (GP), and an acquisition function to guide the search process.

Despite successful applications of Bayesian Optimization to black-box optimization problems, the approach is restricted to problems of moderate dimension. In particular, Bayesian Optimization struggles when the dimensionality exceeds 10-20 variables [27] [28]. This limitation emerges because the number of evaluations needed to cover the search space exponentially increases with the number of dimensions. As a result, scaling Bayesian Optimization to high-dimensional problems remains a major challenge.

Recent advances in machine learning have facilitated the development of new strategies for addressing the problem of airfoil shape optimization. In this section, Deep Reinforcement Learning and Factorization Machine Quantum Annealing are explored and discussed as promising machine learning approaches for shape optimization.

2.2.1 Deep Reinforcement Learning (DRL)

Theoretical background

The combination of deep learning (based on neural networks) with reinforcement learning (RL) is called deep reinforcement learning (DRL). Compared with simple RL, DRL enables the use of high-dimensional state spaces. Diving into reinforcement learning, it is essential to understand the different approaches in machine learning. The field of machine learning is currently split into three categories: supervised, unsupervised, and reinforcement learning. In supervised learning, learning is performed from a training set of labeled examples provided by a knowledgeable external supervisor with the goal of predicting the label of unlabeled data. Unsupervised learning is typically about finding the structure hidden in collections of unlabeled data [29]. In reinforcement learning (RL), an agent learns the optimal behavior to follow within an environment by trying to maximize a reward which is given based on trial-and-error interaction with its environment [7]. In comparison to supervised and unsupervised learning, reinforcement learning does not need large sets of user-provided data for its learning process. Instead, the procedure that follows is gaining experience by interacting with the environment and exploiting the obtained knowledge to improve its behavior [4].

In RL, the agent interacts with the environment through three signals of information: (i) an observation of the current state of the environment, (ii) an action performed by the agent on the environment, based on the previous observation, and (iii) a reward given to the agent after the mentioned action (see Figure 2.3). The goal of RL is to find an optimal decision policy that maximizes the accumulated reward [1]. The expected cumulative reward at timestep t is defined as: [29]

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t \tag{2.4}$$

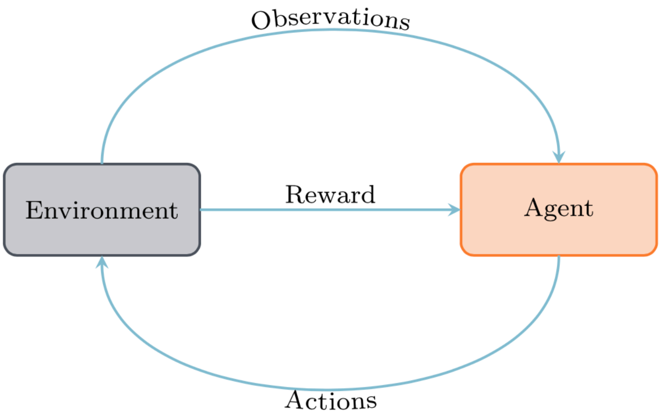


Figure 2.3: Schematic of a reinforcement learning framework [1]

where T is the total time, r_t is the immediate reward received at the time step t and $\gamma \in [0, 1]$ is a discount factor taking into account the impact of future rewards on the current expected reward.

RL methods are typically divided into two categories: model-free and model-based algorithms. Model-based algorithms rely on a model of the environment to perform their interactions. In contrast, model-free algorithms interact directly with the environment without requiring an explicit model. These algorithms are the most widely used in the deep reinforcement learning (DRL) community, due to their simplicity and ease of implementation. Model-free methods can be categorized into value-based methods and policy-based methods [15].

In RL, in order to maximize the cumulative reward, the agent must learn the optimal behavior for any given state, represented by the policy π. This policy assigns probabilities to each possible action based on the current state. The strategies used by the agent to achieve this optimal behavior can be categorized into two methods [29]:

1. Value-based methods. The agent does not directly learn a policy but instead estimates a value function. This value function helps to determine the best actions by selecting the one with the highest expected return. This approach leads to an optimal policy, where the agent always chooses actions that maximize future rewards [4]. There are two types of value functions:

   (a) The state-value function $v_\pi(s)$. It measures the expected return when starting in state s and following the policy π after. It is expressed as:

   $$v_\pi(s) = \mathbb{E}_\pi \left[R(\tau) \mid s \right] \tag{2.5}$$

   (b) The action-value function $q_\pi(s, a)$. It measures the expected return when starting in state s, taking action a and then following policy π:

   $$q_\pi(s, a) = \mathbb{E}_\pi \left[R(\tau) \mid s, a \right] \tag{2.6}$$

   The Bellman relates the value of a state to the value of its possible successor states. For the state-value function, this equation is written as:

   $$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.7}$$

The optimal state-value function, denoted as v_*, is:

$$v_* = \max_{\pi} v_{\pi}(s) \tag{2.8}$$

which satisfies the Bellman optimality equation:

$$v_* = \max_{a} \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{2.9}$$

If this final equation is solved, it will produce the optimal state-value function, which consequently leads to an optimal policy. These equations are the basis of reinforcement learning algorithms like Q-learning and Deep Q-Networks (DQN).

2. Policy-based methods. In these, the optimal policy is searched by directly optimizing the policy π by adjusting the policy parameters θ, rather than estimating a value function. There are advantages over value-based methods. First, it handles high-dimensional action spaces (it performs better in complex environments). It is also suited for both continuous and discrete action spaces (unlike value-based methods, which struggle with continuous action spaces). Lastly, these methods have smoother convergence properties [4]. The goal of policy optimization is to maximize the expected cumulative reward over a trajectory τ, which consists of a sequence of states and actions. The objective function of a policy π_θ is defined as:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \tag{2.10}$$

and seeks the optimal parameterization θ^* that maximizes $J(\theta)$:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \tag{2.11}$$

In order to optimize $J(\theta)$, policy gradient methods compute the gradient $\nabla_\theta J(\theta)$ and use gradient ascent to update the policy parameters. More information on how to calculate this gradient can be found in [15]. Policy gradient methods are trained based on episodes, a sequence of consecutive interactions of the agent with the environment. The temporally discounted sum of rewards that is observed during an episode acts as the quality assessment of the agent's current decision policy. The agent iteratively learns by repeating this process to choose more acceptable actions to maximize its rewards [1].

In deep reinforcement learning, the agent is represented by a deep neural network (DNN). The neural network learns to capture the relationship between input (action) and output (reward) data by continuously refining the weights and biases through back-propagation from the output layer back through the hidden layers to the input layer (process known as training).

Findings in Literature on DRL for shape optimization

[1] presents the first application of deep reinforcement learning (DRL) to shape optimization. In this paper, The DRL agent is based on Proximal Policy Optimization in combination with neural networks to generate 2D shape described by Bézier curves by interacting with a CFD-based environment in Fenics. This type of optimization falls within the category of policy gradient methods, which directly optimize a decision policy mapping states to actions, are well suited for continuous action spaces, and have been successfully applied to optimal control. Going back to the results of the paper, the research proves that the agent is able to generate wing-like optimal shapes within 3000 episodes, with the lift-to-drag ratio as the main aerodynamic objective.

In [4], a Markov decision process formulation of airfoil shape optimization is presented, also using Deep Reinforcement Learning as the main optimization algorithm. This approach is highlighted because it allows the agent to successively modify the thickness and camber of the airfoil at selected positions, using continuous values, in comparison with other research works where the action-space is more restricted. Another new aspect introduced in this research is that more than one aerodynamic objective is searched separately: the lift-to-drag ratio, the lift coefficient and drag coefficient. The flow solver environment is low-fidelity (Xfoil), with which the agent interacts and receives a positive reward when it maximizes (lift-to-drag ratio or lift coefficient) or minimizes (drag coefficient) the aerodynamic objective by modifying the airfoil shape. Again, the DRL agent is based on Proximal Policy Optimization (PPO). Results demonstrate the learning capabilities of DRL in all cases. DRL is also compared to a classical simplex method (where the optimizer is based on the Nelder-Mead simplex algorithm) and demonstrates a higher efficiency at exploring the design space (the L/D value obtained with the DRL approach is higher).

Other applications of DRL to shape optimization can be found in [7], where the authors explore the application of deep reinforcement learning techniques to the optimization of two- and three-dimensional shapes within CFD environments. The study presents an innovative single-step DRL framework for direct shape optimization. The main objective of the paper is to evaluate DRLs effectiveness in discovering optimal aerodynamic shapes without prior domain experience. What makes this paper different from [1] is the application of a single-step DRL (sDRL), a subset of DRL that can be related to the performance of a black-box optimization. Here, it is enough that the agent interacts only once per episode with the environment if the optimal behaviour to be learnt by the agent is independent of state. This means that the optimal action does not depend on a dynamically changing environment or previous states. It uses Policy Based Optimization (PBO), a single-step algorithm. Results showed the successful implementation of the DRL-CFD framework in many cases, obtaining shapes that perform just as well as a conventional airfoil in a scenario where the DRL had zero prior knowledge in aerodynamic concepts. These results highlight present the potential of this method for black-box shape optimization. In the following subsection, another innovative black-box optimization algorithm will be presented, Factorization Machine Quantum Annealing (FMQA).

2.2.2 Factorization Machine Quantum Annealing (FMQA)

Theoretical Background

Factorization Machine Quantum Annealing (FMQA) is a type of black-box optimization method that combines quantum annealing (QA) with factorization machine (FM), a machine learning method [9]. As previously covered, the main interest is to find a set of variables $\mathbf{x}$ that minimizes/maximizes the objective function $f(\mathbf{x})$. In FMQA, $\mathbf{x}$ is the variable vector of size N and is represented in binary form, taking the value 0 or 1 [30]. The reason why the design variables should be in binary form is because the quantum annealer requires the variables to be encoded in this form [8].

$$\begin{aligned}&\text{Minimize } f(\mathbf{x})\\&\text{subject to } \mathbf{x} \in [0,1]^N\end{aligned}$$

In this case, if the information about the objective function (gradient, convexity, etc.) is explicitly given, an efficient optimization can be performed. In contrast, when the objective function $f(\mathbf{x})$ corresponds to the output of a simulation or experiment, the function cannot be explicitly described. Mathematical optimization applied to a non explicit objective function is called black-box optimization.

As mentioned before, Factorization Machine Quantum Annealing uses quantum annealing and factorization machine in its optimization process. As shown in Figure 2.4, the optimization loop looks for the input $\mathbf{x}$ that minimizes the objective function, as well as approximates the black-box function by a second-order polynomial.

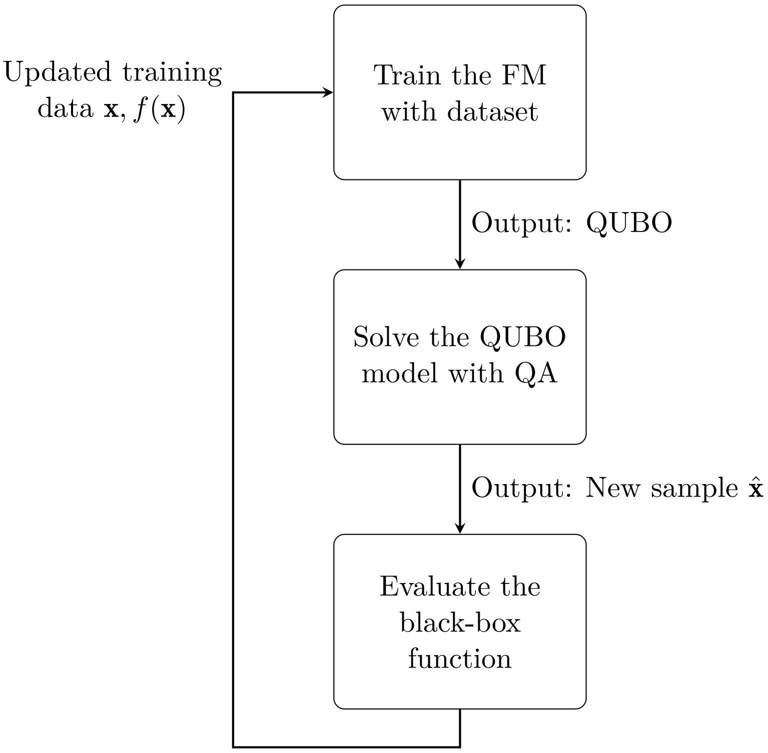

Updated training
data $\mathbf{x}$, $f(\mathbf{x})$
Train the FM with dataset
Output: QUBO
Solve the QUBO model with QA
Output: New sample $\hat{\mathbf{x}}$
Evaluate the black-box function

Figure 2.4: FMQA flow

First, the Factorization Machine processes the design variables $\mathbf{x}$ in binary form and learns from an unknown objective function expressed in QUBO-style (QUBO stands for Quadratic Unconstrained Binary Optimization), which is a second order polynomial that approximates the black-box function. In other words, it trains the machine-learning model. Then, the optimization solver, based on the QUBO model, identifies the next evaluation point $\hat{\mathbf{x}}$ that minimizes the quadratic polynomial. Next, the black-box function is evaluated with the new $\mathbf{x}$. If the polynomial approximation obtained by machine learning is reasonably good, the expected outcome of computing the new value $\mathbf{x}$ in the black-box will be low. If this is not the case, one can expect a better polynomial approximation in the next training cycle by adding the new computed data to the training data (training data is updated) and performing machine learning again [9] [30]. This optimization loop is shown in Figure 2.4. Here one can identify how the Factorization Machine and the Quantum Annealer have different roles in the optimization routine.

As stated earlier, the process of computing the quadratic polynomial that approximates the black-box function is performed by a machine learning model called the Factorization Machine. The FM models the QUBO-styled polynomial as shown in Equation 2.12:

$$f_{FM}(\mathbf{x}|\mathbf{w},\mathbf{v}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j. \tag{2.12}$$

where w_0 and w_i are FM model parameters for real scalar and $\mathbf{v}_i$ is the FM model parameter for K-dimensional real vector, where K is referred to as the rank of the FM. Equation 2.12 is in QUBO form except for w_0.

The objective of training the FM model is to minimize a loss function L, which measures the difference between the predicted value $f_{FM}(\mathbf{x})$ and the real outcome $f(\mathbf{x})$ of the black-box function, shown in Equation 2.13

$$L(w_0, w_i, \mathbf{v}_i) = \sum_{m} \left(f_{FM}(x_i^m) - f(x_i^m)\right)^2. \tag{2.13}$$

As the loss function L is differentiable with respect to the parameters w_0, $\{w_i\}$, and $\{\mathbf{v}_i\}$, the optimal values of these parameters can be learned using general gradient descent techniques. Then, after training f_{FM}, the Ising Machine, in this case, a Quantum Annealer (a specialized type of quantum computer designed to solve combinatorial optimization problems) obtains a new evaluation point $\hat{\mathbf{x}}$ using the QUBO model. After this process, the new result $f(\hat{\mathbf{x}})$ is obtained from the black-box function and added to the data set. In summary, FMQA is a black-box optimization algorithm designed to find the optimal solution to an optimization problem by iteratively repeating N times these processes, which have been stated earlier (also shown in Figure 2.4):

1. Training f_{FM} with FM using the (most recent) training data.
2. Estimate the new evaluation point $\hat{\mathbf{x}}$ that minimizes the acquisition function using the Ising Machine.
3. Evaluate the objective function $f(\mathbf{x})$ with $\hat{\mathbf{x}}$ to obtain $f(\hat{\mathbf{x}})$.
4. Add ($\hat{\mathbf{x}}$, $f(\hat{\mathbf{x}})$) to the training data.

Findings in Literature on FMQA for optimization

Looking into literature, the first paper to present FMQA as a suitable algorithm for black-box optimization is discussed in [8], where the authors present how FMQA can be incorporated into automated materials discovery. The algorithm consists of three parts: regression analysis for the target property by factorization machine, selection of candidate metamaterial based on the regression outcome and simulation of the metamaterial property. As can be seen, this cycle can be related to the general flow shown in Figure 2.4.

The design variables of the black-box optimization are in binary form and represent the structure of materials. Relating this to the airfoil shape optimization problem, the design variables involved in airfoil representation (chord length, camber, angle of attack, etc.) are usually represented as continuous variables. Therefore, other ways to handle this type of design variables need to be addressed.

In [31], the use of FMQA is expanded to black-box optimization with integer variables, integrating different encoding methods (binary encoding, one-hot encoding and domain wall encoding) that allow the user to encode continuous design variables. However, representing continuous variables with binary variables can lead, in most cases, to optimization inefficiencies. That is why in the study available in [9] the use of a Function Smoothing Regularization (FSR) to improve FMQA when dealing with continuous variable optimization problems is addressed. The main challenge is to minimize the noise introduced by the encoding of the continuous variables. In this paper, they present a one-hot representation to handle these variables. This representation is commonly used for categorical data but can also be applied to represent continuous variables in an optimization context. In one-hot encoding, the continuous variable is represented as an array of binary variables as:

$$y(c) \to [x_1, \cdots x_i \cdots x_N]_{\text{one-hot}}\,,$$
$$x_i = \begin{cases} 1 & i = c \\ 0 & i \neq c, \end{cases}$$

where $y(c)$ is a value of the continuous variable y, and $[...]_{\text{one-hot}}$ represents the one-hot vector. This vector, used to represent the continuous value $y(c)$, takes a value of 1 in the element at the position $i = c$ while all other elements remain 0. This ensures that exactly one binary variable is active at a time.

In conclusion, the papers discussed, [9] and [31], might solve one of the challenges presented by FMQA in terms of variable representation, which may be useful for the case of airfoil shape optimization. However, the literature that addresses shape optimization is still limited for FMQA. Fixstars Amplify presents a tutorial [10] where FMQA is applied for airfoil shape optimization. This research work will try to expand the application of this algorithm to airfoil shape optimization by testing this tutorial.

2.2.3 Airfoil Design Represention

In this subsection, the focus will be on airfoil design representation. The method of defining an airfoil in the optimization process is a key topic of interest in this research project. First, a brief explanation of PARSEC airfoils is provided, as this is among the most widely used methods for characterizing airfoils in the field of aerodynamics. Then, the two shape generation methods used in the optimization processes of the reference papers in FMQA and DRL are discussed. Finally, a novel airfoil parameterization approach that integrates Machine Learning techniques is introduced.

PARSEC airfoil

The PARSEC airfoil method is presented here. This method stands out for its own intuitiveness, where its eleven parameters describe the main geometrical characteristics of the airfoil. These specifications makes the PARSEC method suitable for aerodynamic optimization [32]. Table 2.2 and Figure 2.5 show the meaning of each parameter and its representation in an airfoil.

Table 2.2: PARSEC Airfoil Parameters

PARSEC Parameter	Geometry Parameter	Description
p_1	r_{le}	Leading edge radius
p_2	x_{up}	Upper crest horizontal position
p_3	z_{up}	Upper crest vertical position
p_4	$z_{xx,up}$	Upper curvature at crest position
p_5	x_{lo}	Lower crest horizontal position
p_6	z_{lo}	Lower crest vertical position
p_7	$z_{xx,lo}$	Lower curvature at crest position
p_8	z_{te}	Trailing edge vertical position
p_9	Δz_{te}	Trailing edge thickness
p_{10}	α_{te}	Trailing edge direction
p_{11}	β_{te}	Trailing edge wedge angle

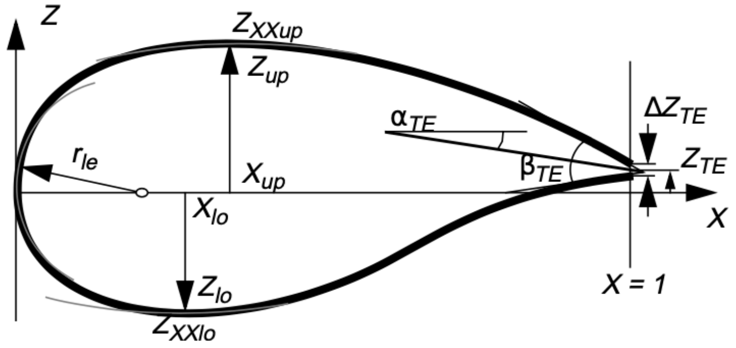


Figure 2.5: PARSEC parameters on an airfoil

The vertical airfoil coordinates are given by:

$$z(x) = \sum_{n=1}^{6} a_n x^{n-\frac{1}{2}} \tag{2.14}$$

where the coefficients a_n for the upper side of the airfoil can be found solving the following system of equations, in which the PARSEC parameters are included in Equation 2.15:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ p_2^{\frac{1}{2}} & p_2^{\frac{3}{2}} & p_2^{\frac{5}{2}} & p_2^{\frac{7}{2}} & p_2^{\frac{9}{2}} & p_2^{\frac{11}{2}} \\ \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & \frac{7}{2} & \frac{9}{2} & \frac{11}{2} \\ \frac{1}{2}p_2^{-\frac{1}{2}} & \frac{3}{2}p_2^{\frac{1}{2}} & \frac{5}{2}p_2^{\frac{3}{2}} & \frac{7}{2}p_2^{\frac{5}{2}} & \frac{9}{2}p_2^{\frac{7}{2}} & \frac{11}{2}p_2^{\frac{9}{2}} \\ -\frac{1}{4}p_2^{-\frac{3}{2}} & \frac{3}{4}p_2^{-\frac{1}{2}} & \frac{15}{4}p_2^{\frac{1}{2}} & \frac{35}{4}p_2^{\frac{3}{2}} & \frac{63}{4}p_2^{\frac{5}{2}} & \frac{99}{4}p_2^{\frac{7}{2}} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} p_8 + \frac{p_9}{2} \\ p_3 \\ \tan(p_{10} - \frac{p_{11}}{2}) \\ 0 \\ p_4 \\ \sqrt{2p_1} \end{pmatrix} \tag{2.15}$$

The coefficients of the lower side are also obtained by means of the system of equations:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ p_5^{\frac{1}{2}} & p_5^{\frac{3}{2}} & p_5^{\frac{5}{2}} & p_5^{\frac{7}{2}} & p_5^{\frac{9}{2}} & p_5^{\frac{11}{2}} \\ \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & \frac{7}{2} & \frac{9}{2} & \frac{11}{2} \\ \frac{1}{2}p_5^{-\frac{1}{2}} & \frac{3}{2}p_5^{\frac{1}{2}} & \frac{5}{2}p_5^{\frac{3}{2}} & \frac{7}{2}p_5^{\frac{5}{2}} & \frac{9}{2}p_5^{\frac{7}{2}} & \frac{11}{2}p_5^{\frac{9}{2}} \\ -\frac{1}{4}p_5^{-\frac{3}{2}} & \frac{3}{4}p_5^{-\frac{1}{2}} & \frac{15}{4}p_5^{\frac{1}{2}} & \frac{35}{4}p_5^{\frac{3}{2}} & \frac{63}{4}p_5^{\frac{5}{2}} & \frac{99}{4}p_5^{\frac{7}{2}} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} p_8 - \frac{p_9}{2} \\ p_6 \\ \tan(p_{10} + \frac{p_{11}}{2}) \\ 0 \\ p_7 \\ -\sqrt{2p_1} \end{pmatrix} \tag{2.16}$$

Joukowski Airfoil Generator

The program used as the basis for this research to implement FMQA is presented in [10]. In this approach, the airfoil is constructed using the Joukowski transform. The Joukowski transform is a mathematical technique, that can be used in airfoil design, that can map simple geometric shapes into more complex airfoil profiles. This transformation is useful in the design and analysis of airfoils for applications such as wind turbines and aircraft. The Joukowski transform is:

$$z(x, y) = \zeta(\xi, \eta) + \frac{c^2}{\zeta(\xi, \eta)} \tag{2.17}$$

This equation represents a mapping between two complex planes, where ζ is transformed into z ($\zeta \rightarrow z$), as shown in Figure 2.6. The airfoil is modeled as a circle in the ζ-plane, centered at ξ_0, η_0 and passing through the point $(c, 0)$. This circle is then rotated by an angle α in the z plane. The parameters ξ_0, η_0 and α correspond to the wing thickness, warping, and angle of attack, respectively.

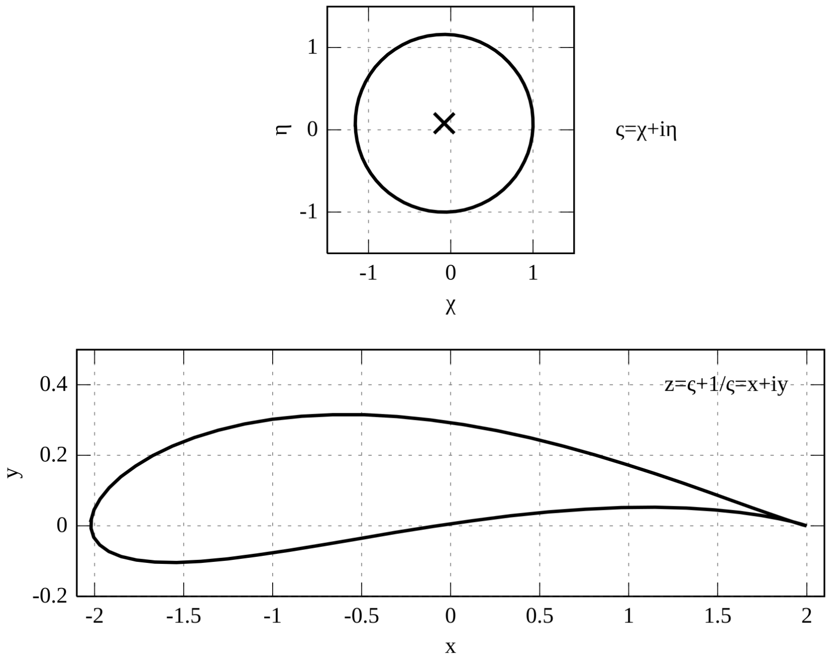


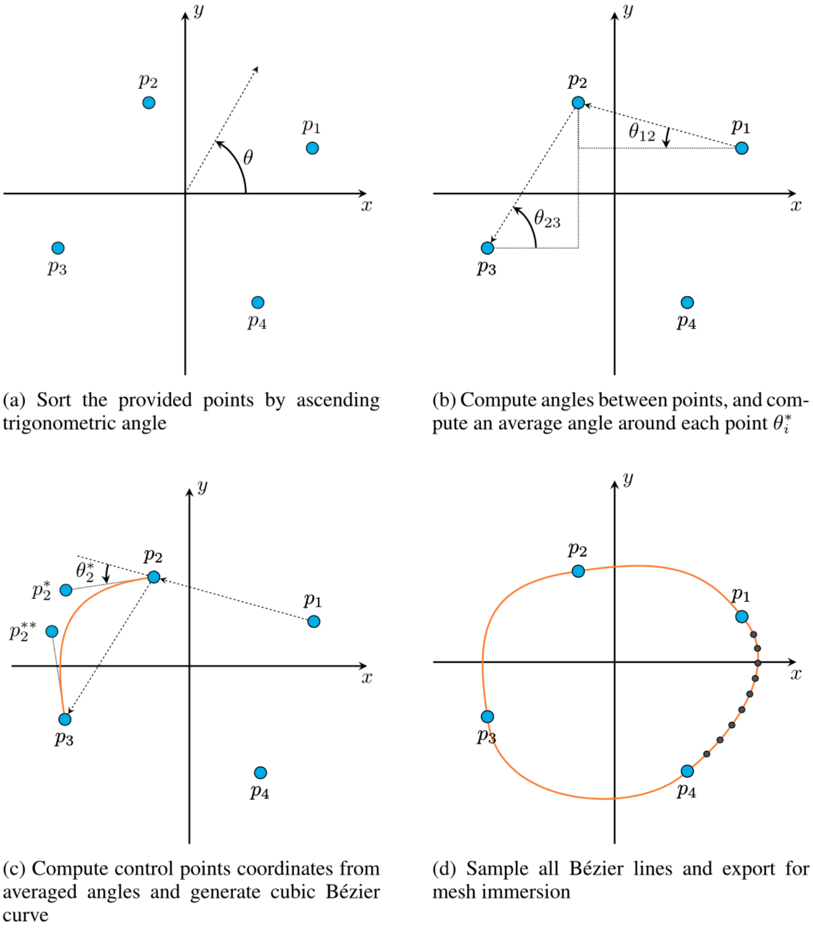
Figure 2.6: Joukowski transform

Shape generation using Bézier curves

One of the papers of reference for DRL, [1], uses Bézier curves to generate the airfoil shape in the optimization process. In this section, the process of generating shapes from a set of n_s points provided by the agent will be covered. The process would be the following: once the network provides the points, an ascending trigonometric angle sort is performed and the system computes the angles between the points. Then, an average angle is computed around each point:

$$\theta_i^* = \alpha \cdot \theta_{i-1,i} + (1-\alpha) \cdot \theta_{i,i+1} \tag{2.18}$$

where α takes a value between $[0, 1]$. This parameter α allows to adjust the sharpness of the curve, with the maximum smoothness being $\alpha = 0.5$. After computing the average angle, one can join the pair of points using a cubic Bézier curve. This curve is defined by four points: p_i and p_{i+1} are the points where the curve passes through and are given by the agent, and p_i^* and p_i^{**}, which are control points that define the tangent of the curve at p_i and p_{i+1}. As shown in Figure 2.7c, the tangents at p_i and p_{i+1} are controlled, respectively, by θ_i^* and θ_{i+1}^* [1].



(a) Sort the provided points by ascending trigonometric angle

(b) Compute angles between points, and compute an average angle around each point θ_i^*

(c) Compute control points coordinates from averaged angles and generate cubic Bézier curve

(d) Sample all Bézier lines and export for mesh immersion

Figure 2.7: Shape generation using cubic Bézier curves [1]

ML-based airfoil parameterization

The paper by Kilian Swannet [2] introduces a novel approach for airfoil design parameterization that utilizes variational autoencoders (VAEs), a class of neural networks known for their proficiency in reducing dimensionality. These generative machine learning models learn a low-dimensional latent representation of input data, representing the key patterns in a probabilistic manner. The encoder maps each input to a distribution, usually Gaussian, from which latent vectors are sampled. A regularization term, based on Kullback-Leibler divergence, ensures that the latent space remains smooth and continuous. This latent space supports the encoding of complex geometries and the generation of new airfoil shapes through sampling.

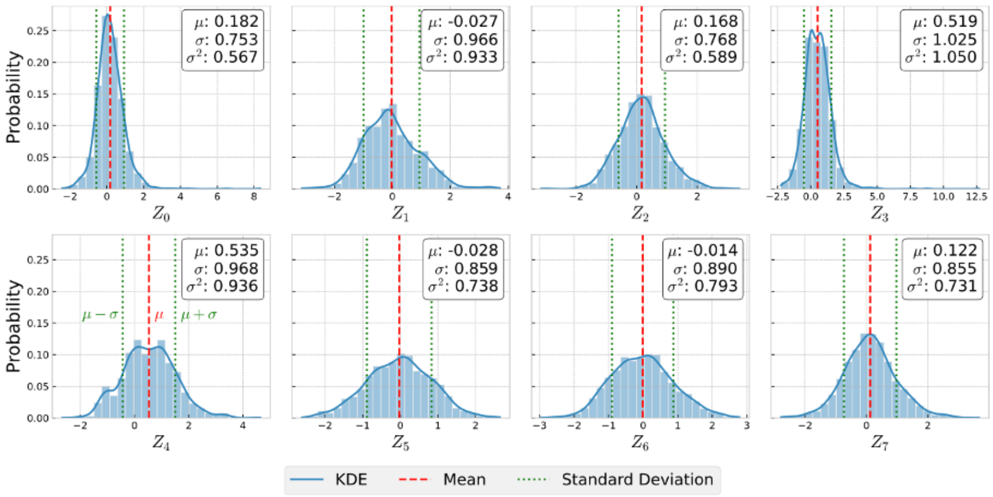


Figure 2.8: Distribution of the latent parameters [2]

The VAE network is trained on a dataset that contains 1619 airfoil data points extracted from the UIUC airfoil database [33], allowing the model to capture a wide range of geometric variations. By passing the dataset through the encoder network of the autoencoder, latent parameters Z_i are sampled from the probability distributions inferred by the encoder. The resulting inferred latent distributions, which contain the probability distribution of every latent dimension are shown in Figure 2.8.

Moreover, to investigate the influence of each latent parameter on specific airfoil characteristics, a series of geometric properties are computed from the generated airfoils. These include features such as maximum thickness, maximum camber, and other relevant shape properties. The visualization in Figure 2.9 provides insights into how each latent parameter influences these specific airfoil properties. For example, Z_3 appears to control the maximum thickness of the airfoil, while Z_4 mainly influences its maximum camber. Other latent parameters such as Z_5 and Z_6 do not appear to have captured any airfoil features.

2.3 Research Questions

This MSc thesis will primarily focus on a comparative study of two types of machine learning algorithms applied to the problem of airfoil shape optimization. As discussed previously, Factorization Machine Quantum Annealing (FMQA) has shown potential as an optimization method in many areas like metamaterials [8]. However, the documentation available regarding its application to shape optimization is still limited. In this context, Fixstars Amplify demo [10] provides a tutorial where a black-box airfoil optimization problem is addressed. Based on this tutorial, this research work will investigate the potential of FMQA in this optimization environment, in an attempt to expand the

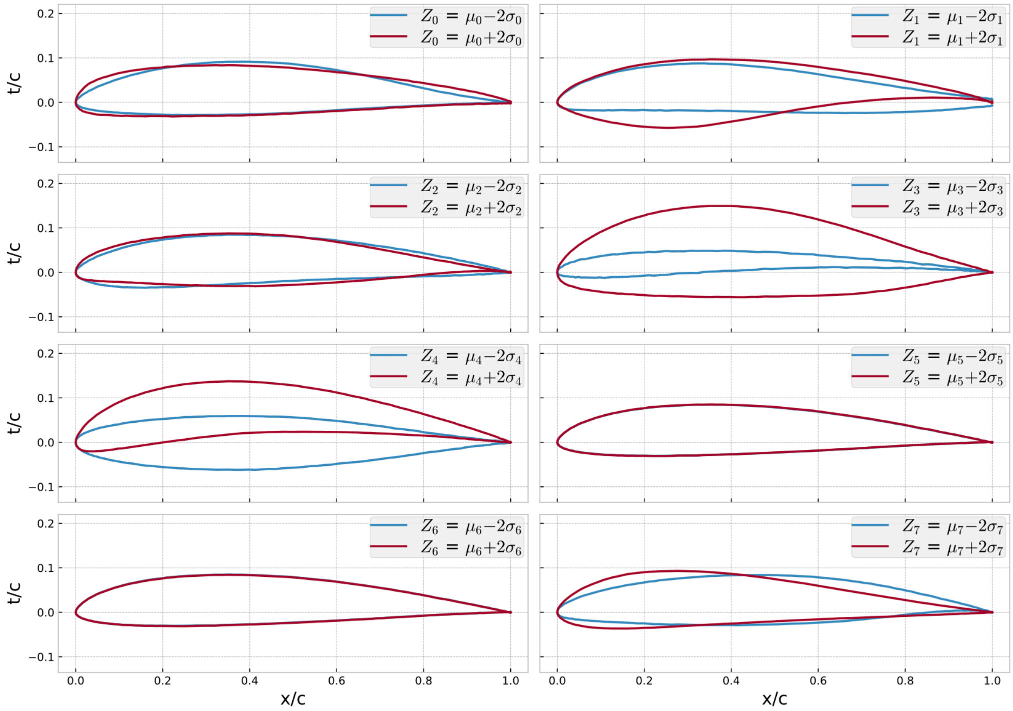


Figure 2.9: Effect of the latent parameters on the generated shape. Each parameter is set to $Z_i = \mu_i \pm 2\sigma_i$ while the rest are kept at their mean values [2]

applicability of this algorithm.

On the other hand, reinforcement learning techniques has demonstrated a larger potential on the topic of shape optimization. More specifically, Deep Reinforcement Learning has been succesfully applied to this topic in [1]. However, in this research work, the focus will be set on single-step Deep Reinforcement Learning, since its performance (the agent interact only once per episode with the environment) can be related to black-box optimization, the type of optimization found in the Fixstars Amplify tutorial [10]. Furthermore, this thesis will integrate the sDRL approach into the Fixstars Amplify framework to explore and compare these two algorithms.

2.3.1 Research Questions

In order to achieve the research objective, one can formulate several research questions that will explore the possibilities of implementing FMQA and sDRL to the airfoil shape optimization problem.

- How do Factorization Machine Quantum Annealing (FMQA) and single-step Deep Reinforcement Learning (sDRL) compare in terms of performance and efficiency for airfoil shape optimization, relative to a standard optimization method?
  - What are the advantages and limitations of FMQA compared to sDRL in dealing with complex aerodynamic design spaces?
  - What are the computational trade-offs between FMQA and sDRL in terms of convergence speed and stability?

- How do FMQA and DRL perform when optimizing airfoil shapes across different Reynolds number regimes?
- How does the choice of airfoil parameterization (Joukowski transformation, PARSEC) influence the optimization performance of FMQA and DRL?
- To what extent does VAE-based airfoil parameterization improve aerodynamic optimization results compared to classical parameterization techniques?

Chapter 3

Methodology

This chapter outlines the methodology used to address the airfoil shape optimization problem, with the aim of answering the research questions identified in Chapter 2. Based on the literature review, two machine learning algorithms (sDRL and FMQA) were selected for comparison. The chapter begins with the description of the theoretical foundation and implementation of the methods. Then, it details the computational setup, including the simulation model used for simulations, parameterization schemes, and integration of the optimization components. Finally, the experimental design is presented, introducing the test scenarios that will be used to evaluate and compare the performance of the two algorithms.

3.1 Description of the methods

This section presents the description of the the optimization methods used to address the airfoil shape optimization problem explored in this research. Two machine learning algorithms are considered: Policy-Based Optimization (PBO), a single-step Deep Reinforcement Learning (sDRL) method, and Factorization Machine Quantum Annealing (FMQA). The PBO method, introduced in [34], serves as the basis for implementing sDRL in this study. FMQA is explored as an alternative black-box optimization strategy, building on the theoretical background outlined in Chapter 2. In addition, the standard optimizer used as a comparison against the machine learning models is presented.

The selection of Policy-Based Optimization (single-step Deep Reinforcement Learning) and Factorization Machine Quantum Annealing (FMQA) is based on the findings of the literature review in Chapter 2. Both methods are well-suited for black-box optimization, where the objective function must be evaluated through simulations. Single-step DRL (sDRL) has demonstrated strong performance in previous airfoil shape optimization studies, while FMQA represents a novel approach in this domain with promising potential. By comparing these two methods, this research work aims to evaluate their strengths in terms of efficiency and flexibility across varying design and flow conditions.

3.1.1 Policy-based Optimization (PBO)

In this subsection, the Policy-based Optimization (PBO) approach is described, which is used in the computational setup to implement a form of single-step deep reinforcement learning.

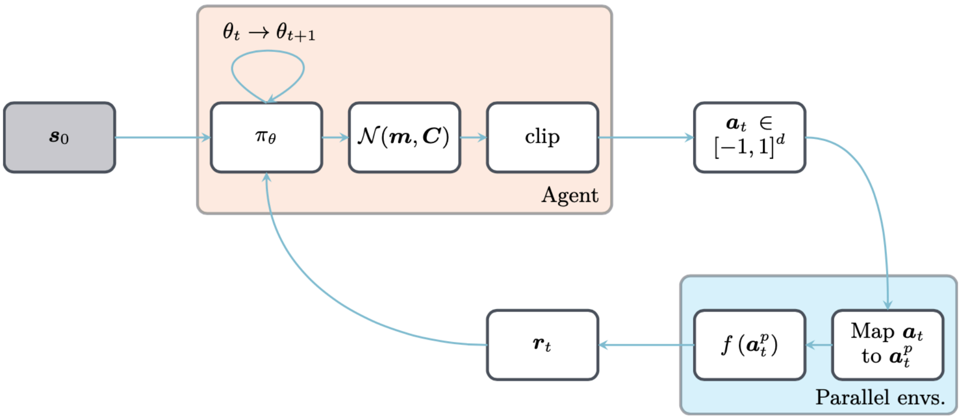

$\theta_t \rightarrow \theta_{t+1}$
$\boldsymbol{s}_0$
π_θ
$\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$
clip
Agent
$\boldsymbol{a}_t \in [-1,1]^d$
$\boldsymbol{r}_t$
$f(\boldsymbol{a}_t^p)$
Map $\boldsymbol{a}_t$ to $\boldsymbol{a}_t^p$
Parallel envs.

Figure 3.1: Action loop for the PBO model [3]

Policy-based optimization (PBO) is a degenerate policy gradient reinforcement learning (RL) algorithm that, as introduced in Chapter 2, suggests that it is enough to perform single-step episodes when the policy to be learned is state independent, that is, $\pi_\theta(s,a) \equiv \pi_\theta(a)$. The core idea of this algorithm is the following: while other policy gradient algorithms aim to find the optimal parameters θ^* such that following the policy π_{θ^*} maximizes the discounted cumulative reward over an entire episode, PBO instead looks for the optimal parameters θ^* such that $a^* = \pi_{\theta^*}(s_0)$ maximizes the instantaneous reward. Here, s_0 is a fixed input (typically a constant vector) that is consistently fed to the agent. The goal is for the optimal policy to eventually represent the best transformation from s_0 to a^*. The agent begins with a random policy defined by its initial parameters θ_0, after what it is given only a single opportunity per episode at finding the optimal. This can be shown in Figure 3.1, where the agent samples a population of actions from the current policy. Then it is incentivized to update its parameters so that future action samples yield higher rewards. A direct consequence of this setup is that PBO relies on policy networks smaller than those typically used in standard deep reinforcement learning (DRL) frameworks. Here, the agent does not need to learn a complex state-action relation, but only a simple transformation from a fixed input state to a given action [3].

In practice, PBO samples actions from a probability density function. Specifically, a d-dimensional multivariate normal distribution $\mathcal{N}(m, C)$ is used, where m is the mean vector and C is the full covariance matrix. As illustrated in Figure 3.2, three separate neural networks are employed to predict the necessary mean, standard deviation, and correlation. The output layers of these networks use hyperbolic tangent and sigmoid activation functions to constrain all values with appropriate ranges.

Once generated, the sampled actions are clipped to the interval $[-1,1]^d$ to ensure that all action components are within the valid range. Clipping is a simple truncation operation where any values that exceed the bounds are set to the nearest limit. This approach is preferred over using soft-limiting transfer functions such as hyperbolic tangent or soft clipping, which in certain cases yield slow convergence and numerical instabilities. The clipped actions are then mapped to their relevant physical ranges a^p as shown in Figure 3.1. Finally, stochastic gradient ascent on the policy parameters is performed using the Adam algorithm [35], based on a modified loss function:

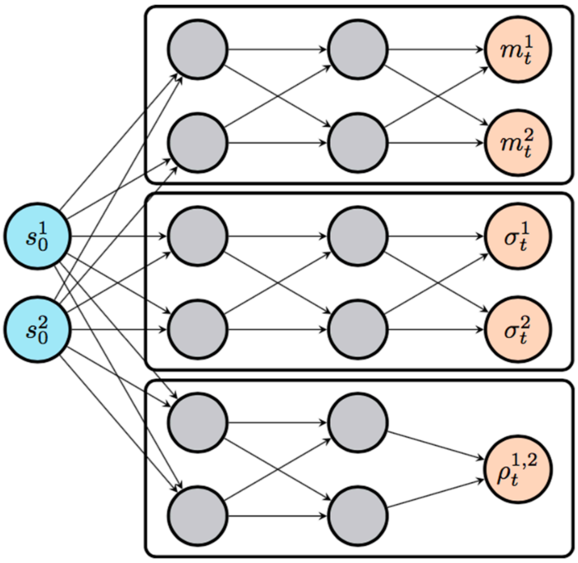


Figure 3.2: Policy networks used in PBO to map states to policy [3]

$$L(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\log \pi_\theta(a)\, \hat{R}(a)\right], \quad \text{with} \quad \hat{R}(a) = \max\left(\frac{r(a)-\mu_r}{\sigma_r}, 0\right). \tag{3.1}$$

where $\hat{R}(a)$ is the clipped normalized reward, μ_r is the reward average over the current generation, and σ_r the standard deviation. Regarding the networks architecture and PBO meta-parameters used in the simulations, all benchmarks cases discussed in Chapter 4 follow the same scheme listed in Table 3.1.

Table 3.1: Detail of the networks architecture and PBO meta-parameters. λ_r is the learning rate, n_g is the number of generations used for learning, n_e is the number of epochs, and n_b is the number of mini-batches

	m	σ	ρ
λ_r	5×10^{-3}	5×10^{-3}	1×10^{-3}
n_g	1	8	16
n_e	128	16	16
n_b	1	4	8
Arch.	[2, 2, 2]	[2, 2, 2]	[2, 2, 2]

In the description of the PBO approach found in [3], a series of simple minimization problems are outlined to exploit the optimization capabilities of the PBO algorithm. Based on these optimization environments, new scenarios based on optimization problems for airfoil shape optimization will be developed and further explained in section 3.2, where the objective function to be optimized in the main scenario will be the lift-to-drag ratio, as previously mentioned.

In order to train the PBO (sDRL) algorithm across the different airfoil optimization environments, a certain training configuration was followed. Each scenario is executed over

50 generations, with 6 individuals evaluated per generation at the same time. As a result, each experiment consists of 300 evaluations, and is repeated three times to account for variability between runs and to provide a more robust assessment of the algorithm's performance. Each individual represents a unique set of airfoil design variables (proposed by the policy) which are evaluated in the LBM solver to obtain the corresponding aerodynamic objective, the lift-to-drag ratio in most cases.

3.1.2 Factorization Machine Quantum Annealing (FMQA)

As discussed in Chapter 2, the tutorial provided in [10] forms the basis of the method developed to address airfoil shape optimization via Factorization Machine Quantum Annealing (FMQA). Similarly to the previous method, the tutorial presents a black-box optimization approach, based on factorization machine and simulated annealing in this case. The purpose of this would be to optimize an airfoil profile with a limited number of evaluations. The goal of the optimization process is to maximize aerodynamic performance, expressed as the ratio between the lift force L and drag force D acting on an airfoil situated in a flow field:

$$r_{LD} = L/D \tag{3.2}$$

Regarding how the airfoil shape will be constructed, there are several methods which were discussed in Chapter 2. More specifically, this tutorial applies the Joukowski transformation to generate the airfoil representation. By default, FMQA performs a minimization. Therefore, the negative value of the lift-to-drag ratio r_{LD} is considered the objective function, to maximize the value.

Moreover, airfoil shape optimization involves non-binary decision variables, in particular, it uses integer-valued input variables. However, as discussed in Chapter 2, FMQA requires the variables to be represented in binary form. Therefore, it is necessary to convert the input variables into binary decision variables. As introduced in [31], a method for performing this transformation is one-hot encoding. For this reason, the baseline program in the Fixstars Amplify tutorial employs one-hot encoding for this purpose.

As mentioned above, this FMQA tutorial uses the Joukowski transform as a method to construct the airfoil. In Chapter 2, the design parameters ξ_0, η_0 and α were defined as the wing thickness, warping, and angle of attack, respectively. These parameters used for the mapping are real numbers. However, this tutorial applies a discrete optimization, where each parameter is selected from a finite number of candidates. This operation is called integer indexing. The tutorial includes a function that allows to check the correspondence between the integer indices used when generating the airfoil and the design parameter values (ξ_0, η_0, α).

FMQA is interfaced with a fluid flow simulator based on the Lattice Boltzmann Method (LBM), due to its computational cost and ease of setting boundary conditions. The choice of LBM over a traditional CFD solver is made because of the high computational cost of traditional CFD simulations. These models provide fast and reasonably accurate predictions of aerodynamic performance. This allows optimization algorithms, such as FMQA, to explore the design space more efficiently without calling a traditional CFD simulation, reducing the overall computational expense.

Under the default conditions set in the Fixstars Amplify tutorial, one simulation of the flow around the airfoil takes approximately 20 seconds of computation time. Once the simulation is complete, the visualization displays the flow velocity vectors (in black) and the pressure force acting on the airfoil (in red), which represents the combined lift and drag forces, as shown in Figure 3.3. The surrounding environment is colored to indicate the vorticity distribution.

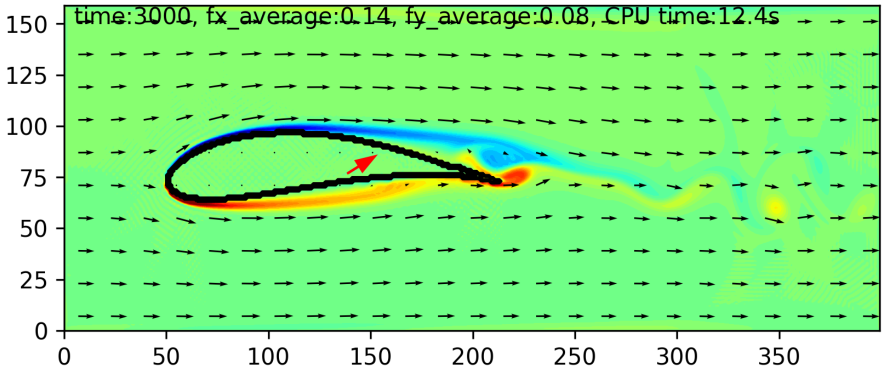


Figure 3.3: Example of the simulation of an airfoil in the Lattice Boltzmann Method solver

The FMQA framework used in the Fixstars tutorial employs a second-order factorization machine, implemented via the `TorchFM` class, to approximate the objective function. This model is trained using supervised regression, where the input consists of binary-encoded design variables, and the target is the lift-to-drag ratio obtained from the LBM simulation. Training is carried out using the Adam optimizer, with the mean squared error as the loss function. The number of training epochs is fixed, and the model is trained again from scratch at each optimization step using all samples previously evaluated.

Once the model is trained, the next step would be to determine the binary input vector that minimizes the predicted objective value. As discussed in Section 2.2.2, this is done by transforming the model into a Quadratic Unconstrained Binary Optimization (QUBO) problem, which is then passed to the Amplify AE solver provided by Fixstars. Then, the solver gives a new candidate vector that is expected to minimize the objective (in this case, minimize $f(\mathbf{x}) = -L/D$). If the proposed solution has already been evaluated in a previous iteration, it is discarded or slightly changed to ensure the diversity of candidate solutions.

The candidate vector is then evaluated by constructing the corresponding airfoil shape and simulating it using the LBM solver. The resulting objective value is added to the dataset used to train the factorization machine. This process is repeated over multiple cycles, allowing the factorization machine to progressively improve its approximation of the objective function. In each cycle, the FMQA algorithm explores regions of the design space that are predicted to provide high performance and also less-sampled areas, guided by the predictions of the factorization machine.

At the start of the optimization process, a number of initial samples N_0 are generated by randomly sampling binary design vectors. These design variables are evaluated using the LBM solver to generate the training dataset for the factorization machine. Once initialized, the main FMQA loop starts and is executed N cycles. In each cycle, the factorization machine is trained again using all available data, by solving the corresponding QUBO problem, and the resulting airfoil is evaluated using the LBM solver. The total number of simulations performed during the complete optimization is therefore $N_0 + N$. The detailed pseudocode for this FMQA optimization loop can be found in the original Fixstars Amplify tutorial [10].

3.1.3 SciPy Optimization: Differential Evolution

Optimization experiments are also performed using the `differential_evolution` function from the `scipy.optimize` library in Python, providing a baseline comparison of the machine learning algorithms with a standard optimizer. Differential Evolution is a stochastic global optimization method that operates on a population of solutions. During each iteration, new trial candidates are produced by mutating existing solutions through combinations with others in the population. [36]. This population-based approach increases the probability of avoiding local minima, making it particularly suitable for airfoil shape optimization problems.

In the context of this work, Differential Evolution was used to optimize the airfoil objective functions under the same parameterization schemes and flow conditions as FMQA and sDRL. The solver was configured with the following settings [37]:

```
res = differential_evolution(f, bounds, maxiter=10, popsize=2, tol=0.1)
```

- Objective function (f): the aerodynamic objective function defined for each optimization case (lift-to-drag maximization, drag minimization, etc.)
- Bounds: the design variables limits that correspond to the selected parameterization scheme (Joukowski, PARSEC or VAE).
- maxiter = 10: the maximum number of generations through which the entire population is evolved. The maximum number of function evaluations is: (`maxiter` + 1) × `popsize` × `number of parameters`.
- popsize = 2: a multiplier that determines the total population size. Since the default population is (`popsize` × `number of parameters`), this configuration results in a relatively small population, lowering computational costs.
- tol = 0.1. the relative tolerance for convergence. Optimization stops when the population has converged within 10% relative tolerance.

These relatively conservative settings (low population size and small number of iterations) were selected so that the computational cost was comparable with the machine learning models while allowing the standard otpimizer to explore the design space. Given that the number of design variables ranges from 9 to 12 (depending on whether PARSEC or VAE-based parameterization is used), the total number of function evaluations would be approximately 200-250, which is comparable to the number of evaluations used for FMQA and sDRL.

3.2 Computational Setup

The computational setup includes developing a program in Python to compare the algorithms in airfoil optimization problems. In order to do that, the code available in Fixstars Amplify [10] and discussed previously will be used as a reference. It presents a black-box optimization of an airfoil geometry with fluid flow simulation, where FMQA is implemented using the Fixstars Amplify platform. To create the airfoil geometry, the airfoil generator class `Joukowski.WingGenerator` is applied, which is based on the Joukowski transform. On the other hand, `lbm.Solver` models the fluid dynamics of the problem, which uses a two-dimensional Lattice Boltzmann Method (LBM) as a solver. Lastly, it is important to highlight the libraries `PyTorch` and `Amplify`, used to support the FMQA optimization process. These will be adopted in the implementation of the new code when applying FMQA.

On the other hand, the code available in GitHub [34], based on the paper [7], presents a case of black-box optimization method based on single-step deep reinforcement learning. As previously mentioned, this implementation employs Policy-Based Optimization (PBO) as the reinforcement learning algorithm. Part of the computational setup involves interfacing the sDRL code with components from the FMQA tutorial, specifically using the Lattice Boltzmann Method (LBM) solver from FMQA. This integration requires modifications to both the FMQA and sDRL codes to ensure compatibility in data representation and objective function evaluation, which will be detailed in this section.

As mentioned in other sections, FMQA requires the design variables to be represented in binary form. In the Fixstars Amplify tutorial [10], the design variables are chosen from a discrete array and then converted into binary form using one-hot encoding to represent the discrete choices in a format compatible with quantum annealing. In contrast, the sDRL algorithm performs optimization using continuous design variables. Therefore, parts of the original code must be adjusted to support continuous variables instead of discrete ones. Specifically, the airfoil generator class `Joukowski.WingGenerator` should be modified so that, instead of generating an array of discrete values between the minimum and maximum limits and selecting a value by index, the continuous variable itself is passed directly to the `generate_wing` function. This step significantly simplifies the implementation for sDRL, as it eliminates the need to encode the design variables in binary form for compatibility with the optimizer. Therefore, FMQA's use of discrete variables is an inherent limitation of the algorithm.

For most of the experimental campaign, the fluid flow simulator based on the Lattice Boltzmann Method (`lbm`) will remain unchanged for both the FMQA and sDRL programs. However, some modifications to accommodate the VAE-based parameterization and other benchmark cases will be incorporated into the solver.

3.2.1 Objective Function

As defined in subsection 2.1.1, specifying an objective function is a necessary step when performing optimization. This objective function should reflect the performance of the system under study, in this case, the aerodynamic performance in airfoil shape optimization. To obtain a computed value of aerodynamic performance, an airfoil must first be constructed and then simulated under specified flow conditions. Therefore, referring to Figure 3.4, the objective function should involve steps such as constructing the airfoil shape, running the simulation using the fluid flow simulator model, and computing the objective value based on the simulation results.

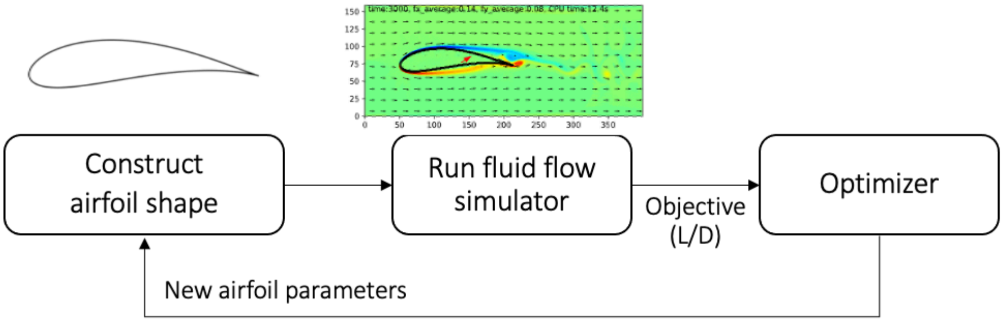


Figure 3.4: Optimization loop

Constructing the Airfoil Shape

In this part of the code, the airfoil shape is constructed based on the design parameters $\mathbf{x}$ (binary-coded variables in the case of FMQA, and continuous variables for sDRL). For FMQA, the binary-coded variables must be decoded into array indices that correspond to discrete values of the airfoil parameters. This decoding step is omitted in sDRL, as the program directly operates with continuous (real) values.

There are different methods to construct an airfoil, as introduced in subsection 2.2.3. The base program, based on the Fixstars Amplify tutorial [10], utilizes the Joukowski transform representation, as described in subsection 3.1.2. In order to evaluate alternative parameterization methods, the computational setup will be adapted to incorporate more complex airfoil representations, which may lead to improved optimization results. This is the case for PARSEC parameterization and VAE-based parameterization.

To optimize using the PARSEC parameterization, a new module `PARSEC` was implemented. This module replicates the structure of the original `Joukowski` module used in the FMQA tutorial, and includes the same core classes and functions, such as `generate_wing` and `draw`, to maintain compatibility with the existing FMQA and sDRL implementations. The `PARSEC` module internally solves the system of equation described in subsection 2.2.3 to generate upper and lower surface coordinates based on 11 geometric design parameters. In addition, the angle of attack was incorporated as a twelfth parameter, allowing the generated airfoil shape to be rotated during the optimization process.

In addition to classical parameterizations (such as PARSEC, Joukowski), a VAE-based method was implemented to generate airfoil shapes from a small set of latent parameters. As already introduced in subsection 2.2.3, the VAE (variational autoencoder) is trained on a dataset that contains 1619 airfoil data points, and learns to represent these shapes in a low-dimensional vector. During optimization, either FMQA or sDRL proposes a latent vector, which is then passed to the decoder of the VAE to reconstruct the corresponding airfoil geometry.

To maintain consistency with the existing structure, a `VAE` module was created, following the same interface as the `Joukowski` and `PARSEC` modules. This includes a `generate_wing` function that enables integration with both optimization algorithms. As will be detailed in Chapter 4, the latent values are constrained within a safe range (between -1.5 and 1.5) to ensure the airfoil shape remains realistic and out of numerical noise.

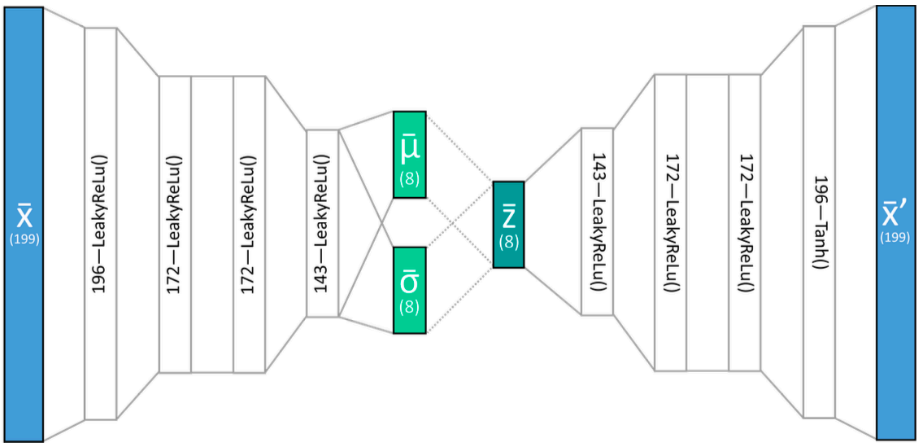


Figure 3.5: VAE architecture [2]

The VAE architecture used in this research work is shown in Figure 3.5. It consists of a fully connected neural network, with an encoder that maps the input airfoil coordinates to a low-dimensional latent vector, and a decoder that reconstructs the airfoil shape. Specifically, the architecture contains four hidden layers with 196, 172, 172, and 143 neurons, respectively. This structure is mirrored in both the encoder and decoder networks to ensure symmetry and stable reconstruction performance. Once trained, only the decoder is used during optimization to generate airfoil shapes from latent vectors proposed by FMQA and sDRL.

In summary, to understand better how these different parameterizations are used in this research work, Table 3.2 presents a comparison between the three implemented methods. Joukowski, PARSEC and VAE. Each approach encodes the airfoil geometry differently and introduces trade-offs in terms of complexity, flexibility and integration with the optimization algorithms.

Table 3.2: Comparison of airfoil parameterization methods.

Parameterization	Type	# Parameters	Interpretability	Flexibility
Joukowski	Analytical	2 (+AoA)	High	Low
PARSEC	Geometric	11 (+AoA)	High	Medium
VAE-based	Data-driven	8 (+AoA)	Low	High

Joukowski and PARSEC are classical parameterization techniques rooted in physics, offering well-defined geometric interpretations. On the one hand, the Joukowski transformation offers a simple and analytically defined airfoil shape controlled by only two design parameters (ξ_0, η_0). On the other hand, the PARSEC method allows for greater geometric control through 11 parameters, including the leading-edge radius, trailing-edge angle, and the location of maximum thickness.

In contrast, the VAE-based parameterization learns to represent airfoil shapes directly from data, rather than relying on predefined geometric rules or equations. This results in a more flexible and expressive representation. However, as discussed in Section 2.2.3, most parameters in the VAE's latent space do not correspond to interpretable geometric features (only two appear to influence the maximum thickness and camber of the airfoil). Consequently, although the latent space is less intuitive than the classical methods, it provides a smoother structure that can facilitate more efficient exploration during the optimization process.

Running the Simulation Model

After the airfoil geometry is generated via a selected parameterization method, the simulation is performed. A Lattice Boltzmann Method (LBM) solver is provided through the `lbm.Solver` class, which simulates incompressible, two-dimensional laminar flow around the airfoil. The solver takes the airfoil coordinates as input and runs a simulation of the flow field under predefined boundary conditions. These conditions are described in Table 3.3, which will later be modified to incorporate other Reynolds numbers.

Table 3.3: Simulation settings used in the Lattice Boltzmann Method solver.

Parameter	**Value**
Grid resolution	400×160
Time Steps	3000
Inlet velocity	0.1 (lattice units)
Kinematic viscosity	0.02 (lattice units)
Relaxation time (τ)	$3 * vis + 0.5 = 0.56$
Boundary conditions	No-slip walls at top and bottom

Computing the Objective Value

Once the LBM simulation is complete, the results derived from the pressure and velocity fields are used to compute the aerodynamic forces acting on the airfoil shape. Specifically, the lift and drag forces are extracted from the surface pressure distribution, and these are used to calculate the lift-to-drag ratio. This ratio serves as the primary objective in most of the test cases performed in this research work. Additional performance metrics, involving combinations of the lift and drag coefficients, are also considered and discussed in Section 4.6.

How this objective is formulated within the optimization process depends on the nature of each algorithm. Since FMQA is a minimization algorithm, the objective is expressed as the negative of the lift-to-drag ratio. In contrast, in sDRL the agent is trained to maximize the expected cumulative reward, so the lift-to-drag ratio is used directly as the reward signal.

3.2.2 Integration of the Optimization Environment

The integration of the optimization environment differs significantly between the FMQA and sDRL frameworks. These differences lie mainly in the architectural design of each approach and in how the interaction between the optimizer, parameterization method, and fluid flow simulator is managed. In the FMQA framework, the environment is implemented such that the optimizer and objective function are defined within the same script. The parameterization method and the LBM solver are implemented as separate modules, and they are directly called from the main program. This configuration ensures that all components are located in a single place, facilitating their identification and accessibility. However, it also means that any modification to the optimizer or the objective function would modify the whole script.

In contrast, the sDRL implementation uses a separate environment defined independently from the optimizer in order to handle the objective function. This environment includes the airfoil construction, simulation, and reward calculation. As in the FMQA setup, the parameterization method and the LBM solver are implemented externally and are called within the environment when evaluating a candidate design. In this configuration, the agent interacts with this environment by taking actions (providing different parameter values) and receiving feedback in the form of a reward, which in most cases corresponds to the lift-to-drag ratio. This separation between the environment and the learning algorithm makes it easier to adapt the setup to different scenarios, such as changing the parameterization method or modifying simulation parameters like the Reynolds number.

These differences reflect the design approaches typically used in each method: FMQA is structured around a single evaluation loop, while sDRL relies on interaction between agent and external environment.

3.3 Design of Numerical Experiments

The experiment phase will begin by evaluating the performance of FMQA and sDRL using simple airfoil shapes generated via Joukowski parameterization. The primary objective in this initial stage is to maximize the lift-to-drag ratio under controlled conditions. In later phases, the focus will shift towards more complex aerodynamic objectives.

Once the initial tests have been performed, the study will transition to more complex geometric representations, such as the PARSEC parameterization, which offer great flexibility and accuracy in describing airfoil shapes. In addition to these conventional methods, this project will incorporate a novel parameterization approach developed by K. Swannet that utilizes variational autoencoders (VAEs), a class of neural networks known for their proficiency in reducing dimensionality [2]. Based on the available literature, this will be the first study interfacing this VAE-based parameterization with optimization algorithms such as FMQA and sDRL.

To evaluate these parameterizations in a computationally efficient way, the experiments will be conducted using a simulation model based on a Lattice Boltzmann Method (LBM) solver, as previously mentioned. The solver simulates incompressible flow at a Reynolds number of approximately 800 by default (considering the constructed airfoil as the simulated shape), which represents moderate laminar flow conditions. To test across varying Reynolds numbers, the viscosity is modified while keeping the inflow velocity low to ensure that the Mach number remains below 0.1, mimicking incompressible flow behavior.

Another key parameter to vary is the angle of attack, to see how the algorithms adapt the airfoil shape in response to a changing aerodynamic environment (caused by varying the angle of attack). This variation allows the study to observe whether FMQA and sDRL adjust design strategies effectively under different flow conditions, such as increased lift or early flow separation. Other metrics such as the convergence and a comparison of the objective for the same number of iterations will be tested for both FMQA and sDRL, in an attempt to decide which advantages one has over the other.

Therefore, to rigorously compare the performance of Factorization Machine Quantum Annealing (FMQA) and single-step Deep Reinforcement Learning (sDRL) in aerodynamic shape optimization, a structured set of experiments was designed based on the logic detailed above. The goal of these experiments is to assess not only the optimization quality, but also the adaptability of each algorithm across different shape representations and flow conditions.

Five main experiment types were conducted. In all cases, the results of FMQA and sDRL are compared against a standard optimizer (`differential_evolution` from `scipy.optimize`), unless otherwise specified.

1. **Baseline Comparison**
   FMQA and sDRL are tested under the same conditions using Joukowski parameterization to assess their baseline optimization behaviour.

2. **Parameterization Sensitivity**
   The algorithms are evaluated using three shape parameterization schemes: Joukowski, PARSEC, and VAE-based. This assessment enables the analysis of how each design representation impacts performance.

3. **Reynolds Number Variation**
   The Reynolds number is increased from its default value of 800 to 5000, in order to study its effect on aerodynamic performance. To achieve this increase, the kinematic viscosity is varied while keeping the inflow velocity constant.

4. **Parametric Study: Angle of Attack**
   In other test scenarios, the angle of attack was treated as a design parameter. In this experiment, however, it is fixed at specific values to assess its impact on aerodynamic performance and to determine which angle yields the most favorable optimization results for both FMQA and sDRL algorithms. For simplicity, only the machine learning algorithms are considered here, omitting the standard optimizer.

5. **Benchmark Cases**
   Additional aerodynamic objectives are explored to evaluate the capabilities of FMQA and sDRL as optimization algorithms. First, an aerodynamic objective considering a target lift coefficient and the drag coefficient is analysed. Then, another aerodynamic objective is replicated from literature considering the drag coefficients at different lift coefficient values.

Chapter 4

Results and Discussion

This chapter presents the results of airfoil optimization experiments conducted using single-step Deep Reinforcement Learning (sDRL) and Factorization Machine Quantum Annealing (FMQA). The objective is to compare both algorithms in terms of performance, efficiency and robustness across a range of different airfoil parameterizations (Joukowski, PARSEC, and VAE-based), flow conditions (varying the Reynolds number) and optimization objectives (maximizing the lift-to-drag ratio, targeting a specific lift coefficient, etc). Performance is mainly evaluated through the obtained aerodynamic objective values, convergence behaviour and analysis of the optimized airfoil shapes. Moreover, the performance of both algorithms is compared against a standard optimizer.

4.1 Baseline Comparison

This section presents a baseline comparison between FMQA and sDRL for the airfoil shape optimization problem under the same default conditions. The objective is to assess the performance of both algorithms when applied to a common aerodynamic target using a fixed parameterization scheme. In this case, the Joukowski transformation is used to generate the airfoil shape, and the optimization objective is to maximize the lift-to-drag ratio (L/D) under the default simulation settings described in Table 3.3. Based on the inlet velocity, kinematic viscosity, and a reference airfoil shape, the corresponding Reynolds number for this configuration is approximately 800.

The design space for this experiment is defined by the three parameters of the Joukowski transformation: ξ_0, η_0, and α. These variables control key geometric characteristics of the airfoil, such as the wing thickness, camber, and angle of attack, respectively. To ensure realistic and aerodynamically feasible designs, the parameters are constrained within the ranges $\xi_0 \in [1.0, 10.0]$, $\eta_0 \in [0.0, 10.0]$, and $\alpha \in [0.0°, 40.0°]$. A summary of these parameter bounds is provided in Table 4.1. These limits were applied consistently across all experiments conducted with FMQA and sDRL, ensuring that both algorithms operated within the same feasible design region.

Table 4.1: Limits of the design parameters in Joukowski transformation

Parameter	Minimum Value	Maximum Value
ξ_0	1.0	10.0
η_0	0.0	10.0
α (°)	0.0	40.0

FMQA is run with $N_0 = 50$ initial randomly generated training samples and a total of $N = 100$ objective function evaluations, corresponding to 50 optimization cycles. In contrast, sDRL is executed over 50 generations with 6 individuals per generation, resulting in a total of 300 evaluations. To allow a fair comparison between the two algorithms, the sDRL results are initially truncated to the first 100 evaluations, as shown in Figure 4.3. Performance is evaluated on the basis of convergence behavior, final objective function values, and the resulting optimized airfoil shapes.

Figure 4.1 shows the optimization history of the FMQA run under the defined baseline conditions. The initial training samples are shown in blue, while the evaluations from the optimization cycles are shown in red. The history reveals that the minimum value of the objective function, defined as $f(\mathbf{x}) = -L/D$, reaches approximately -4.0. The progression of the evaluations during the optimization phase appears stochastic, with little evidence of a consistent convergence trend. Although the results of the optimization cycle show an improvement over the initial training data, FMQA does not exhibit a clearly directed optimization trajectory. Nevertheless, it successfully identified an optimal solution early in the optimization phase.

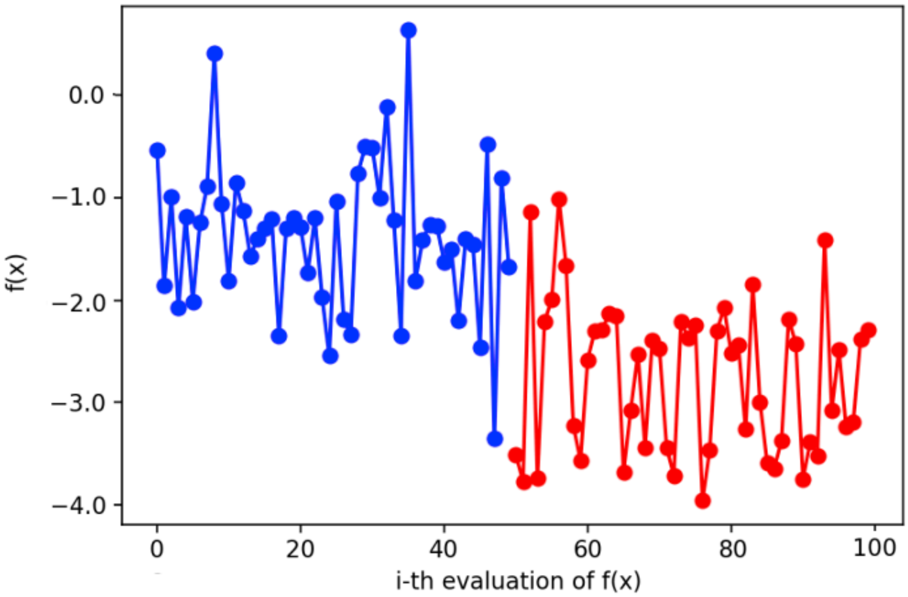


Figure 4.1: Optimization history of FMQA using the Joukowski transformation ($N_0 = 50$, $N = 100$).

Once the optimization process is completed, FMQA returns the set of design parameters that minimize the objective function. The resulting airfoil shape, constructed using these parameters, is shown in Figure 4.2. The airfoil exhibits low camber (appearing nearly symmetric), reduced thickness, and an orientation of approximately 10 degrees. Therefore, this thin, symmetric geometry is associated with an improved lift-to-drag (L/D) ratio under the given flow conditions.

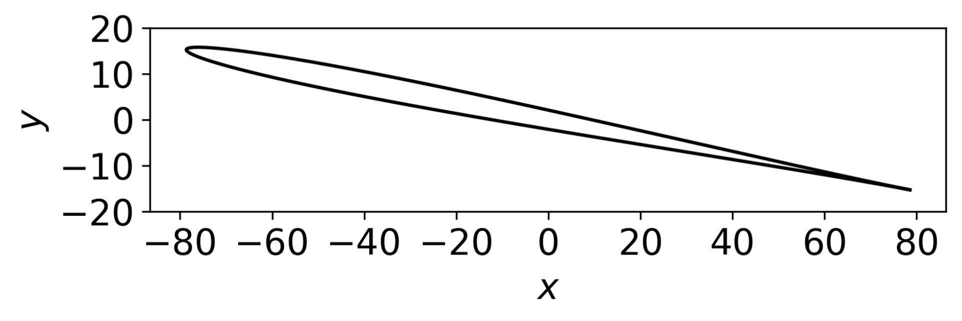


Figure 4.2: Airfoil obtained with FMQA using the Joukowski transformation ($N_0 = 50$, $N = 100$).

These characteristics suggest that the optimizer tends to favor shapes that reduce drag, which is typical in flows with low Reynolds numbers. Under these conditions, flow separation and higher viscous losses may occur in airfoils with too much camber or thickness. The resulting shape makes sense for this flow regime, since keeping the airfoil thin and nearly symmetric can help reduce drag without making the flow separate too easily.

Moving on to the sDRL experiment, it is important to note that the algorithm was run for a total of 300 evaluations. However, to enable a fair comparison with FMQA, only the first 100 evaluations are shown in Figure 4.3. Unlike FMQA, sDRL performs training and optimization simultaneously, which means there is no clear distinction between an initial training phase and the subsequent optimization steps.

The optimization history displays a behaviour similar to that observed with FMQA: the minimum objective function value again reaches approximately −4.0, and the progression of evaluations appears stochastic, as seen in Figure 4.3. Once more, the optimal solution is identified early in the optimization process (around the 40th evaluation).

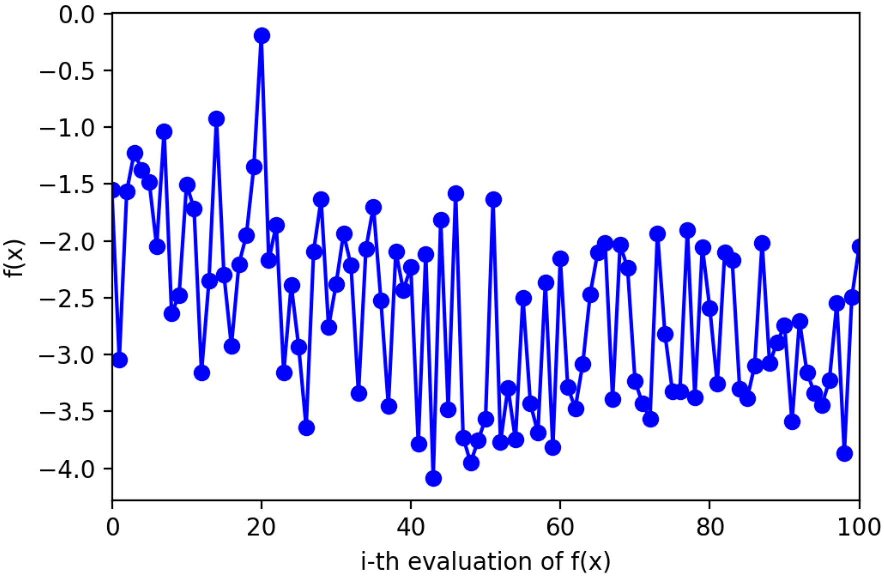


Figure 4.3: sDRL optimization history (first 100 evaluations) for baseline comparison with FMQA

On the other hand, Figure 4.4 shows the optimization history of sDRL plotted in terms of generations. As mentioned previously, each generation consists of 6 individuals, corresponding to 6 objective function evaluations. For each generation, both the average objective value and the best value among the individuals are computed. In the plot, the average value per generation is shown in red, while the best evaluation per generation is shown in blue.

The best value curve indicates that the optimal solution was reached within the first 10 generations, with a minimum objective value of approximately $f(\mathbf{x}) = -4.0$. This corresponds to the minimum found in Figure 4.3. However, the average values do not converge to this optimum over the 50 generations. Although a downward trend is observed, it is not sufficient to be considered converged. This behavior confirms the stochastic nature of the optimization process already observed in the previous figure.

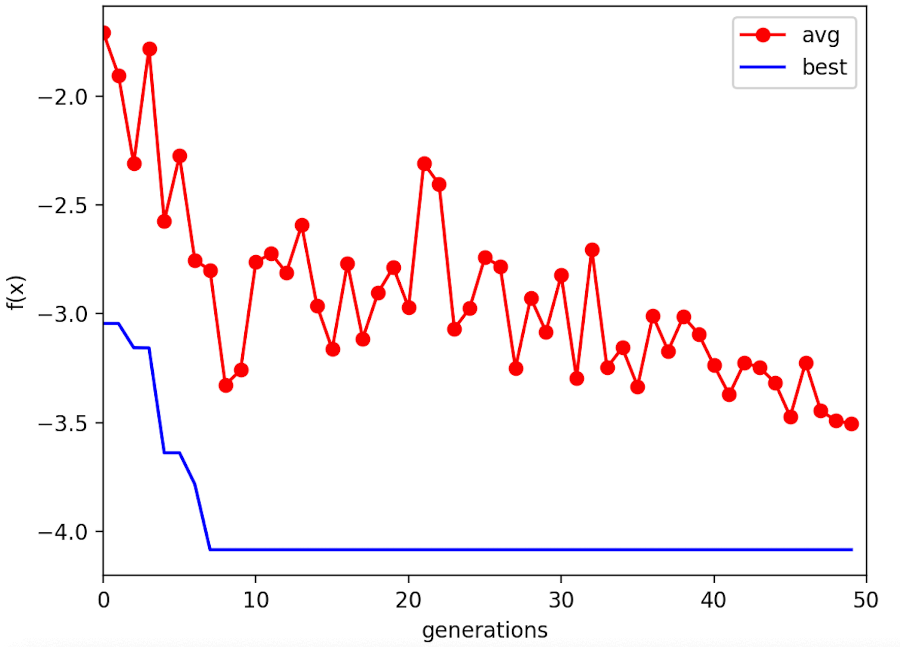


Figure 4.4: Optimization history of sDRL using the Joukowski transformation.

Regarding the optimized airfoil shape, the result obtained by sDRL is shown in Figure 4.5, plotted alongside the initial airfoil used by the algorithm. In the sDRL experiments, optimization begins from a fixed initial airfoil shape derived from the policys first action, which enables a direct comparison between the initial and optimized designs. In contrast, FMQA explores the design space through random sampling of discrete parameter combinations and does not rely on a predefined starting point. For this reason, a similar initial-to-final shape comparison is not applicable in the FMQA case.

Returning to Figure 4.5, it is evident how the airfoil shape evolved throughout the sDRL generations. The most notable change is a reduction in overall thickness. Similar to the optimized shape obtained by FMQA, the airfoil exhibits very low camber ($\eta_0 = 0.764$), being almost a symmetric airfoil. Additionally, the orientation of the airfoil has shifted slightly toward a less inclined profile compared to the initial design, with the final angle of attack being approximately 10 degrees.

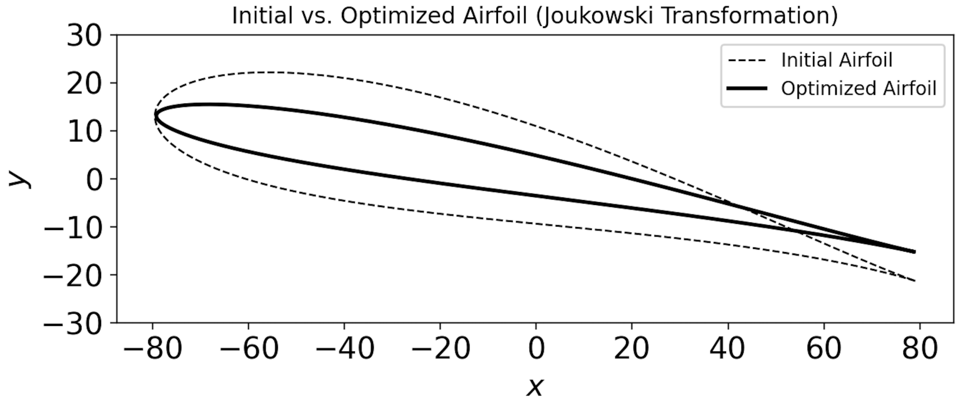


Figure 4.5: Airfoil obtained with sDRL using the Joukowski transformation (initial vs. optimized)

To sum up, FMQA and sDRL performed similarly in this initial baseline comparison. Both algorithms identified an optimum of approximately $f(\mathbf{x}) = -L/D = -4.0$ reached around the 40th evaluation (corresponding to the 40th optimization cycle in the case of FMQA). Regarding the airfoil shapes, the resulting geometries exhibit comparable characteristics, with the airfoil optimized by FMQA being slightly thinner than the one obtained through sDRL.

Besides the comparison between both machine learning algorithms, an additional optimization using the `differential_evolution` algorithm from SciPy is performed to validate the results. Figure 4.6 shows the resulting airfoil, which is slightly thicker and more cambered compared to the previously optimized geometries.

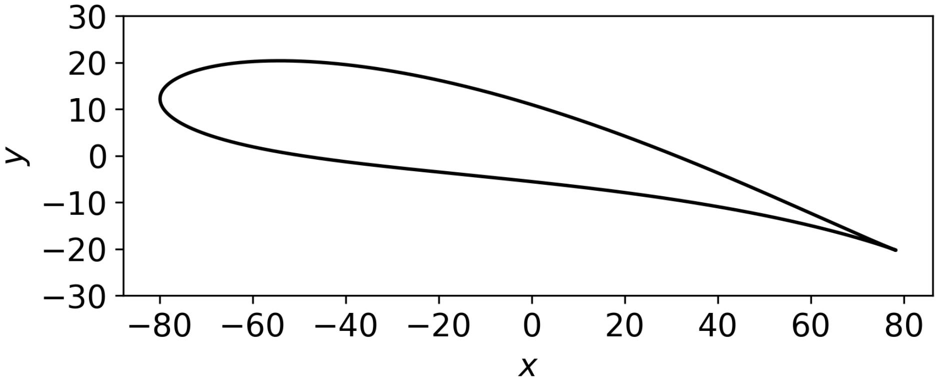


Figure 4.6: Airfoil obtained with SciPy using the Joukowski transformation

Table 4.2 summarizes the optimized design parameters along with their respective lift-to-drag ratios for FMQA, sDRL, and the SciPy optimizer. Both machine learning algorithms achieve higher L/D values compared to standard optimization, with sDRL achieving the highest overall. These results will be compared in Section 4.2 to see whether using the PARSEC parameterization gives similar outcomes or leads to better results in both algorithms.

Table 4.2: Optimized values of the design parameters in the Joukowski transformation and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
ξ_0	1.0	1.954	3.593
η_0	0.0	0.764	2.888
α (°)	11.0	10.2	11.74
L/D	3.959	4.085	3.392

4.2 Parameterization Sensitivity: PARSEC

This section explores the effect of using a different parameterization scheme, PARSEC, on the performance of the algorithms in the context of airfoil shape optimization. While the previous experiments were based on the Joukowski transformation, PARSEC provides a more flexible and descriptive way to define an airfoil shape using a set of geometric parameters, such as leading edge radius, trailing edge angle, and curvature control points.

The aim of this experiment is to evaluate how each algorithm performs when the parameterization scheme is changed, and whether the use of PARSEC parameterization can lead to improved optimization results. The algorithms are applied under the same simulation conditions as in the baseline comparison, with the only modification being the parameterization type. As before, the results are compared in terms of convergence behaviour, optimized objective values and the characteristics of the resulting airfoil shapes.

The design space in this case is defined by 11 geometric parameters, including the angle of attack. For a detailed description of these parameters, refer to Section 2.2.3. As before, to ensure aerodynamically feasible airfoil designs, the parameters are constrained within defined ranges, as shown in Table 4.3. The angle of attack is also included as a design variable, with bounds set to $\alpha \in [0.0°, 40.0°]$. Both FMQA and sDRL used these parameter limits during the optimization process.

Table 4.3: Limits of the PARSEC Airfoil Parameters in the FMQA and sDRL code.

Parameter	**Minimum Value**	**Maximum Value**
r_{le}	0.005	0.00938
x_{up}	0.36	0.45
z_{up}	0.045	0.057
$z_{xx,up}$	-0.555	-0.26
x_{lo}	0.3	0.56
z_{lo}	-0.058	-0.04
$z_{xx,lo}$	0.28	1.1
z_{te}	-0.02	-0.009
Δz_{te}	0.005	0.0082
α_{te} (°)	-7.44	-4.58
β_{te} (°)	5.72	16.6

In this experiment, FMQA is run with $N_0 = 30$ initial randomly generated training samples and a total of $N = 60$ objective function evaluations. This configuration was selected instead of the previously used $N_0 = 50$, $N = 100$ setup, as it led to better optimization results in preliminary testing. sDRL, on the other hand, is again executed over 50 generations with 6 individuals per generation, resulting in a total of 300 evaluations.

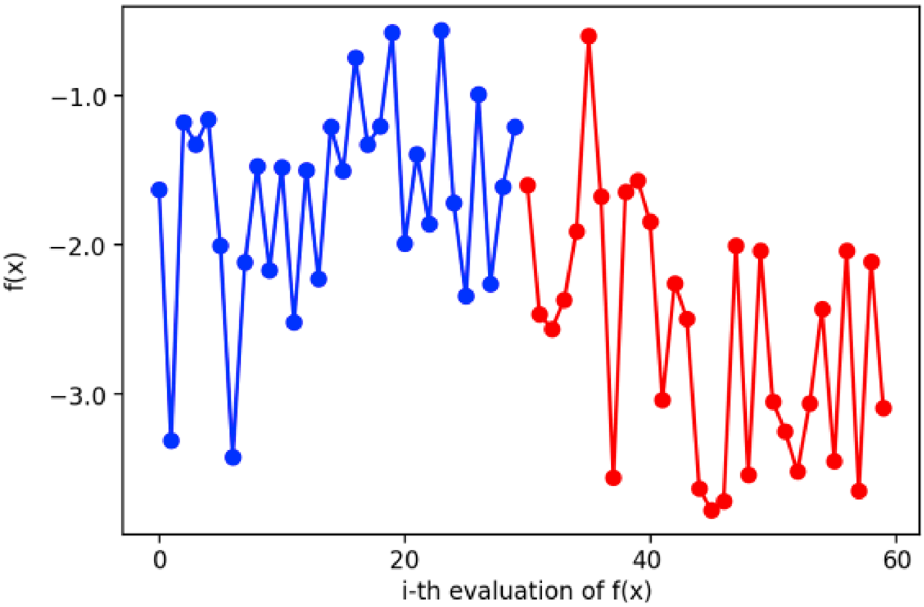


Figure 4.7: Optimization history of FMQA using PARSEC parameterization ($N_0 = 30$, $N = 60$)

Figure 4.7 shows the optimization history of the FMQA run using the PARSEC parameterization. The objective function, defined as $f(\mathbf{x}) = -L/D$, reaches a minimum value of approximately -3.5 to -4. The evaluations during the optimization cycles show random variation, without a clear trend towards convergence. Still, they generally produce better values than the initial training data. Interestingly, several of the best performing candidates are discovered early in the process, indicating that FMQA is able to identify promising solutions within the first few optimization cycles.

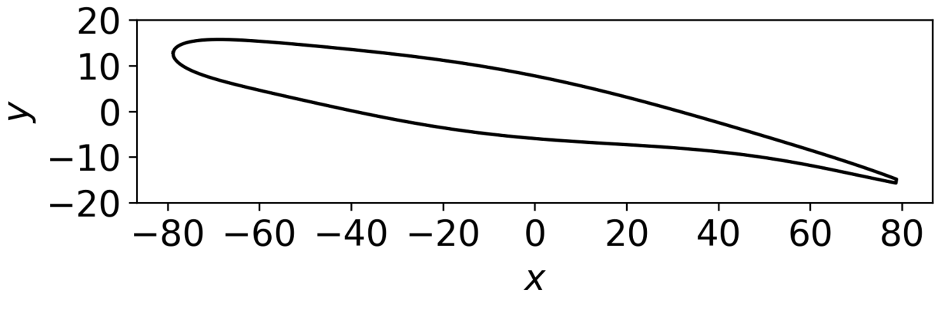


Figure 4.8: Airfoil obtained with FMQA using PARSEC parameterization ($N_0 = 30$, $N = 60$).

The FMQA algorithm returns an optimal candidate that minimizes the objective function once the optimization cycles are completed. The resulting airfoil, plotted in Figure 4.8, shows a more complex shape compared to the one obtained using the Joukowski transformation. The thickness appears relatively constant up to the mid-chord, after which it gradually decreases towards the trailing edge. The orientation is similar to the previously computed airfoil, with an angle of attack of around 10 degrees. It can be considered a relatively thin airfoil; however, compared to the airfoil shown in Figure 4.2, the thinness of the PARSEC shape is barely noticeable.

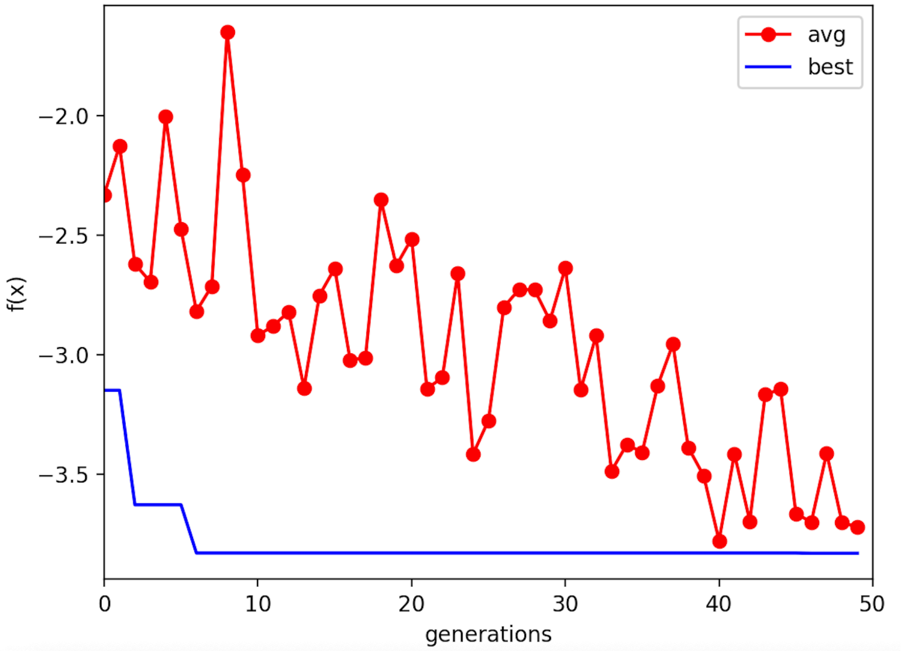


Figure 4.9: Optimization history of sDRL using PARSEC parameterization.

Regarding the sDRL case, Figure 4.9 shows the evolution of the objective function throughout the 50 generations of the optimization process. The best value curve indicates that a minimum is reached after approximately 5 generations, with a value between −3.5 and −4. This suggests that the optimum was found early in the optimization process. In contrast to previous experiments, the average value curve shows a clear downward trend over the generations, gradually approaching the minimum during the final stages. This may indicate partial convergence, although the curve does not completely flatten. This behaviour marks a difference in the optimization process, as earlier results (particularly those using the Joukowski transform) showed stochastic trends with no clear sign of convergence.

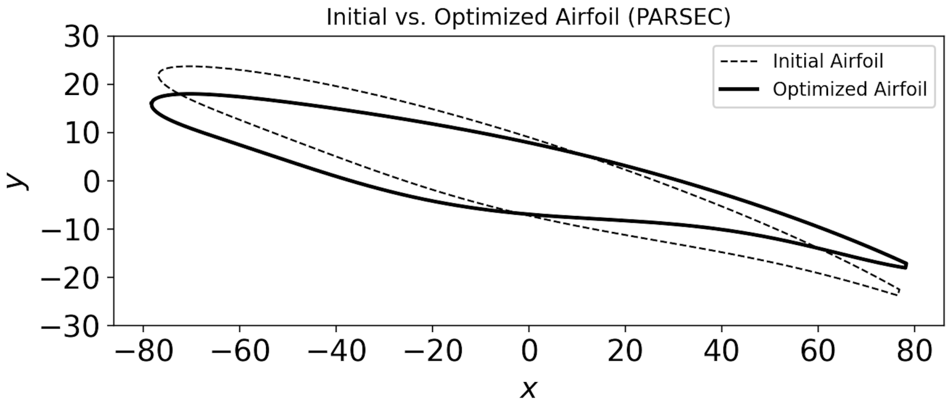


Figure 4.10: Airfoil obtained with sDRL using PARSEC parameterization (initial vs. optimized).

The resulting airfoil obtained from the sDRL optimization using PARSEC parameterization is shown in Figure 4.10, compared to the initial best individual identified during the optimization process. Comparing both shapes, there is a noticeable but slight change in orientation from the initial to the optimized airfoil. Overall, the geometry does not largely change, with the optimized airfoil appearing slightly thinner near the trailing edge. The shape closely resembles the one obtained from the FMQA optimization, with only minor geometric variations. Once again, while these airfoils may appear slightly thicker than those obtained using the Joukowski transformation, they still fall within the range of thin airfoil geometries.

Overall, both FMQA and sDRL produced similar results in terms of the optimized lift-to-drag ratio and the resulting airfoil shape. The optimal objective value was found to be between -3.5 and -4 and the final airfoil geometries are comparable in their main characteristics. In terms of convergence, sDRL demonstrated a more stable trend, with the average values approaching the optimum.

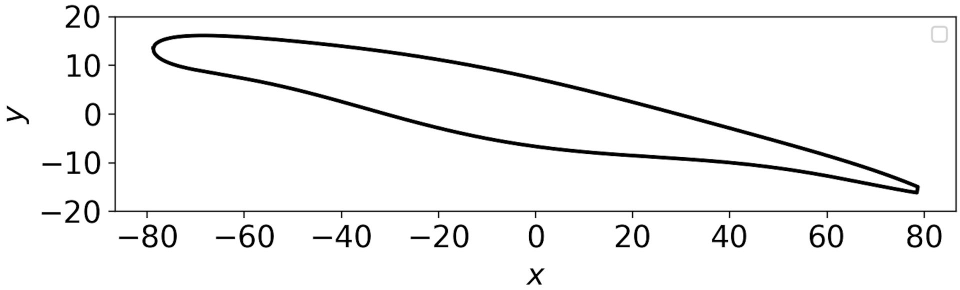


Figure 4.11: Airfoil obtained with SciPy using PARSEC parameterization

Similarly to the previous experiment, an extra optimization was performed using the `differential_evolution` algorithm from SciPy in order to assess whether the machine learning algorithms improve the aerodynamic objective. The resulting optimized geometry is shown in Figure 4.11, exhibiting a slightly thinner leading edge compared to the previous airfoil shapes.

Table 4.4: Optimized values of the design parameters in PARSEC parameterization and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
r_{le}	0.009	0.006	0.0077
x_{up}	0.41	0.44	0.38
z_{up}	0.047	0.045	0.048
$z_{xx,up}$	-0.47	-0.32	-0.44
x_{lo}	0.37	0.38	0.46
z_{lo}	-0.044	-0.053	-0.042
$z_{xx,lo}$	0.57	0.96	0.87
z_{te}	-0.019	-0.016	-0.012
Δz_{te}	0.005	0.005	0.0076
α_{te} (°)	-6.96	-5.63	-6.00
β_{te} (°)	9.89	15.73	13.79
α (°)	9.00	11.21	9.78
L/D	3.783	3.828	3.869

Table 4.4 shows the optimized design variables together with the optimized lift-to-drag ratios, for both the machine learning algorithms and the standard optimizer. Unlike the baseline comparison, the SciPy optimizer appears to slightly outperform FMQA and sDRL in terms of the aerodynamic objective.

Compared to the baseline comparison using the Joukowski transformation, the optimal objective values obtained with the PARSEC parameterization were slightly lower (except for the SciPy optimizer), as shown in Table 4.5. However, this difference is not large. sDRL appears to produce the best results under both parameterization schemes when compared to FMQA. The resulting airfoil shapes using PARSEC parameterization can also be classified as thin airfoils, although they appear slightly thicker than those obtained using the Joukowski parameterization. These findings suggest that while the selection of parameterization slightly affects the optimization results, the algorithms are capable of producing high performing airfoil designs under different shape definitions.

Table 4.5: Optimum of $f(\mathbf{x})$ for the different parameterization schemes

Parameterization	**FMQA** (L/D)	**sDRL** (L/D)	**SciPy** (L/D)
Joukowski	3.959	4.085	3.392
PARSEC	3.783	3.828	3.869

4.3 Reynolds Number Variation

This section investigates the effect of varying the Reynolds number on the performance of FMQA and sDRL in the context of airfoil shape optimization. Although the previous experiments were conducted at a default Reynolds number of approximately 800, changing this parameter allows assessing how each algorithm adapts to different flow regimes. To solve the optimization problem at different Reynolds numbers, it is necessary to modify one of the variables that defines it. The Reynolds number is defined as:

$$Re = \frac{UL}{\nu} \tag{4.1}$$

where U is the inflow velocity, L the characteristic length and ν the kinematic viscosity. These variables were assigned specific values in the default configuration of the LBM solver: the airfoil shape had a characteristic length of $L = 160$ lattice units, the inflow velocity was set to $U = 0.1$ lattice units, and the kinematic viscosity was $\nu = 0.02$ lattice units.

To increase the Reynolds number, either the inflow velocity or the kinematic viscosity must be modified. However, increasing the inflow velocity can degrade the accuracy of the Lattice Boltzmann Method (LBM) solver. This is because the solver assumes a low Mach number for accurate simulation of near-incompressible flows, defined by:

$$Ma = \frac{U}{c_s} \ll 1 \tag{4.2}$$

where c_s is the lattice speed of sound. In order to maintain a low Mach number, it is preferable to keep the inflow velocity at or below $U = 0.1$.

For these reasons, the kinematic viscosity was chosen as the variable to adjust. Since it is inversely proportional to the Reynolds number, it was calculated directly by rearranging the formula $\nu = \frac{UL}{Re}$. In this research work, the Reynolds number is increased to values between $Re = 1000$ and $Re = 5000$. However, setting the Reynolds number too high may lead to numerical instability in the LBM solver, resulting in overflow errors or non-physical values such as NaNs.

Another important consideration is the orientation of the airfoil, specifically maintaining the angle of attack in the range $\alpha \in [0.0°, 10.0°]$. At higher angles, especially when combined with high Reynolds numbers, the flow tends to separate from the airfoil surface, generating strong vortices and nonlinearities that the solver may not be able to capture accurately. The LBM solver is best suited for smooth, laminar flows, and becomes unstable in the presence of large gradients or separated flow regions. In this experiment, the PARSEC parameterization is used due to its greater geometric flexibility, which allows for more detailed airfoil shapes that may be better suited for flow conditions at higher Reynolds numbers. In particular, a Reynolds number of 5000 is considered, representing the upper limit tested within the stability constraints of the LBM solver.

In this case, the FMQA algorithm is run with $N_0 = 150$ and $N = 300$, since this configuration resulted in a better optimum. Increasing the number of evaluations allows the algorithm to explore the design space more thoroughly, which can lead to improved performance. Additionally, using 300 objective function evaluations matches the total used by sDRL over 50 generations, allowing for a fair comparison between both methods. Figure 4.12 shows the optimization history of the FMQA run at a Reynolds number of 5000 using the PARSEC parameterization, with the configuration described in the previous paragraph. The objective function reaches a value of approximately $f(\mathbf{x}) = -7.0$ early in the optimization process. However, the evaluations during the optimization cycles exhibit stochastic behaviour, with limited improvement compared to the initial training data. As a result, no clear convergence trend is observed. Still, the FMQA algorithm manages to identify high-performing candidates early in the process.

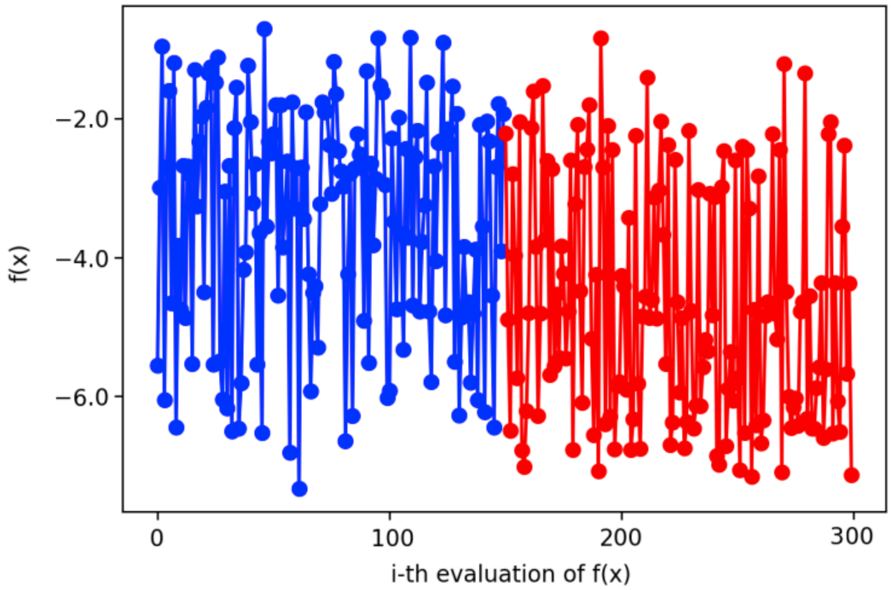


Figure 4.12: Optimization history of FMQA using PARSEC parameterization and $Re = 5000$ ($N_0 = 150$, $N = 300$)

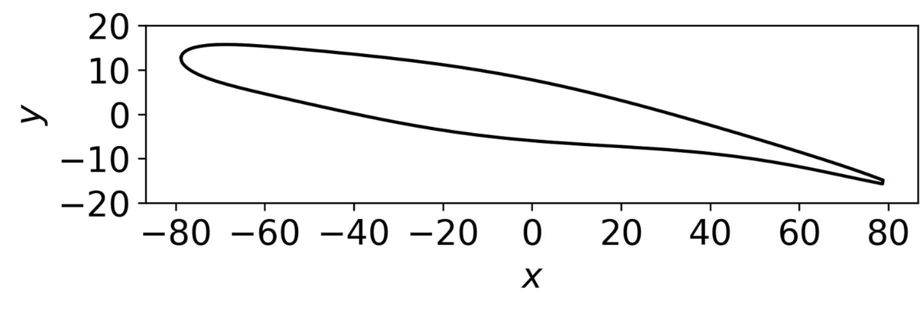


Figure 4.13: Airfoil obtained with FMQA using PARSEC parameterization and $Re = 5000$ ($N_0 = 150$,$N = 300$).

Regarding the optimized airfoil obtained from this optimization process, the resulting shape is shown in Figure 4.13. The mean camber line is noticeably curved, indicating a positive camber, which is typically associated with increased lift at low to moderate angles of attack. The leading edge is rounded and smooth, facilitating flow attachment and reducing the risk of separation. Towards the trailing edge, the airfoil tapers into a thin, sharp profile, which is desirable to minimize wake and drag. The overall geometry resembles the optimized airfoil obtained in the previous section (shown in Figure 4.8), suggesting that a similar shape may remain effective under different Reynolds number conditions.

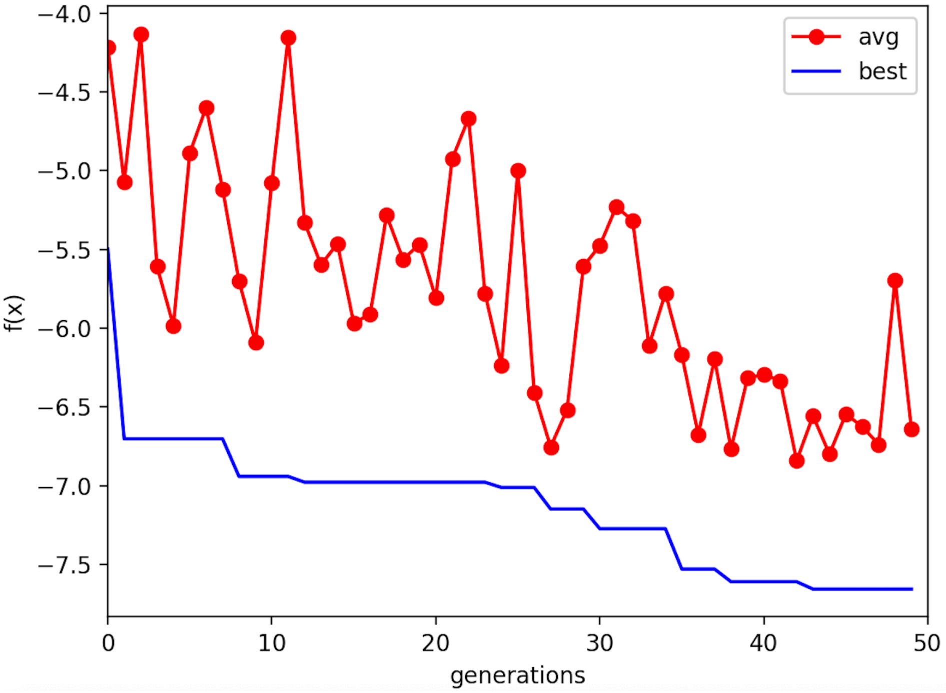


Figure 4.14: Optimization history of sDRL using PARSEC parameterization and $Re = 5000$.

Moving on to the sDRL experiment at a Reynolds number of 5000, Figure 4.14 shows the corresponding optimization history, represented by the best performing individual and the average objective value plotted per each generation. The best value curve indicates that the algorithm reaches approximately $f(\mathbf{x}) = -7.0$ between the 10th and 20th generations, improving to around $f(\mathbf{x}) = -7.5$ after the 40th generation. This result is slightly better than the optimum obtained with FMQA. Meanwhile, the average value curve does not fully converge to the optimum but shows a gradual downward trend, improving from around $f(\mathbf{x}) = -4.0$ to approximately $f(\mathbf{x}) = -6.0$ over the optimization process.

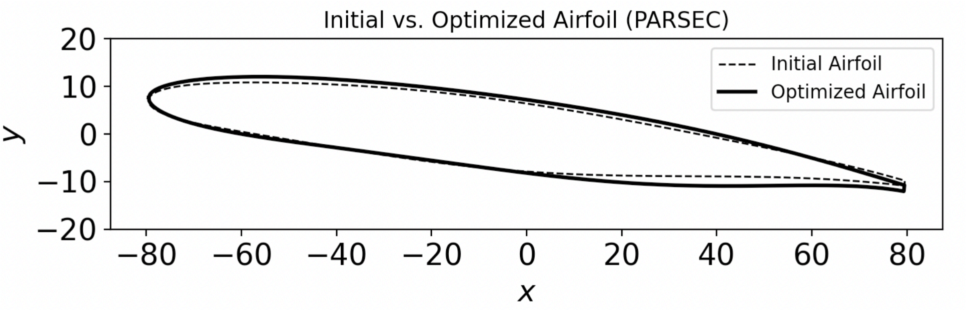


Figure 4.15: Airfoil obtained with sDRL using PARSEC parameterization and $Re = 5000$ (initial vs. optimized).

In Figure 4.15, a comparison between the initial and optimized airfoil shapes from the sDRL experiment is shown. Visually, the optimized shape exhibits only minor differences, including a slightly higher angle of attack and a more pronounced curvature on the upper surface. The thickness and trailing edge appear quite similar to the initial design. Compared to the FMQA optimized shape, this geometry appears slightly thicker near the trailing edge, with a more uniform thickness distribution along the chord.

To sum up, FMQA and sDRL produced both similar and distinct results. In terms of convergence, both algorithms exhibited stochastic behaviour over the optimization process. However, sDRL showed a slight downward trend in the average values, indicating a more consistent progression towards the optimum. Regarding other metrics, sDRL improved the optimized objective value by approximately 6.97% compared to FMQA, as shown in Table 4.6. Both resulting airfoils can again be classified as thin, with the sDRL optimized shape appearing slightly thicker near the trailing edge. For reference, Figure 4.16 shows the optimized geometry obtained with the SciPy, which closely resembles the one obtained with FMQA.

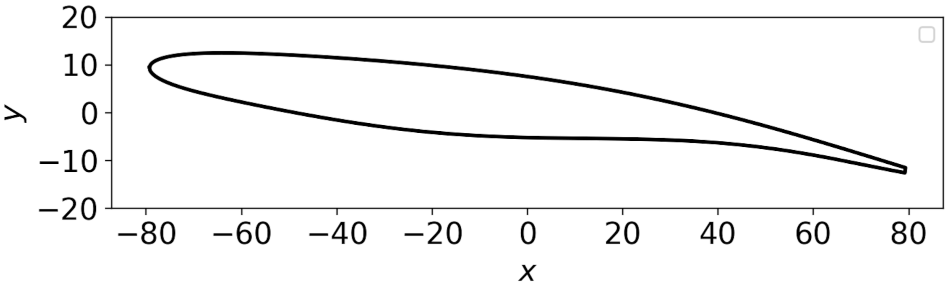


Figure 4.16: Airfoil obtained with SciPy using PARSEC parameterization and $Re = 5000$.

Table 4.6 shows the optimized design variables and the optimized aerodynamic objectives for the algorithms using PARSEC parameterization with $Re = 5000$. In this case, the objective value obtained with the SciPy optimizer is lower than those achieved by the machine learning algorithms. As discussed before, sDRL provides the highest lift-to-drag ratio.

Table 4.6: Optimized values of the design parameters in PARSEC parameterization and the corresponding objective function values ($Re = 5000$).

	FMQA	**sDRL**	**SciPy**
r_{le}	0.008	0.0081	0.0067
x_{up}	0.40	0.37	0.44
z_{up}	0.057	0.053	0.047
$z_{xx,up}$	-0.36	-0.28	-0.34
x_{lo}	0.45	0.55	0.34
z_{lo}	-0.043	-0.045	-0.042
$z_{xx,lo}$	0.59	0.37	0.61
z_{te}	-0.010	-0.011	-0.018
Δz_{te}	0.008	0.008	0.0067
α_{te} (°)	-5.69	-7.29	-6.89
β_{te} (°)	9.25	7.13	6.83
α (°)	8.00	6.23	6.72
L/D	7.157	7.656	7.010

Table 4.7 shows the comparison between the results obtained in Section 4.2 at a Reynolds number of 800 and those achieved at a Reynolds number of 5000. The results indicate that increasing the Reynolds number leads to improved aerodynamic performance, reflected by higher lift-to-drag ratios. Since the kinematic viscosity was reduced to achieve a higher Reynolds number, viscous effects became less dominant in the flow simulation. This reduction in viscous effects results in lower drag, thereby increasing the lift-to-drag ratio. The optimized values at $Re = 5000$ are nearly double those at $Re = 800$, with sDRL achieving the best performance between the two machine learning algorithms.

Table 4.7: Optimum of $f(\mathbf{x})$ for the different Reynolds number using PARSEC parameterization

Reynolds	**FMQA** (L/D)	**sDRL** (L/D)	**SciPy** (L/D)
800	3.783	3.838	3.869
5000	7.157	7.656	7.010

4.4 Parameterization Sensitivity: VAE-based

In this section, the impact of using a novel parameterization scheme is investigated for both FMQA and sDRL. The selected scheme is a VAE-based parameterization, a machine learning approach proposed by K. Swannet [2], which was introduced in Section 2.2.3. The architecture of the VAE was described in detail in Subsection 3.2.1. The goal is to compare the results with those obtained in the previously studied cases and assess whether the VAE-based parameterization improves aerodynamic optimization performance compared to classical representation techniques.

The design space in this case is defined by the latent variables of the VAE, which are used to construct the airfoil shape. To ensure the generation of realistic and aerodynamically feasible geometries, these latent variables are constrained within the range $[-1.5, 1.5]$, as summarized in Table 4.8. In this experiment a Reynolds number of 5000 is selected, since the previous section demonstrated that higher Reynolds numbers lead to a noticeable improvement in the lift-to-drag ratio. The angle of attack is also kept below 10 degrees to prevent flow separation and other instabilities that may arise at higher Reynolds numbers.

Table 4.8: Limits of the latent variables in VAE-based parameterization

Latent Variables	Minimum Value	Maximum Value
z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7	-1.5	1.5

After testing different configurations, FMQA performed best with $N_0 = 50$ and $N = 100$ evaluations in terms of aerodynamic results. Figure 4.17 shows the optimization history of the FMQA algorithm using the VAE-based parameterization. Overall, the optimization displays stochastic behaviour, with only the last 20 evaluations showing a slight convergence, oscillating between values $f(\mathbf{x}) = -6$ to $f(\mathbf{x}) = -8$. The best candidate is identified around the middle of the optimization process, with a value exceeding than $f(\mathbf{x}) = -8$.

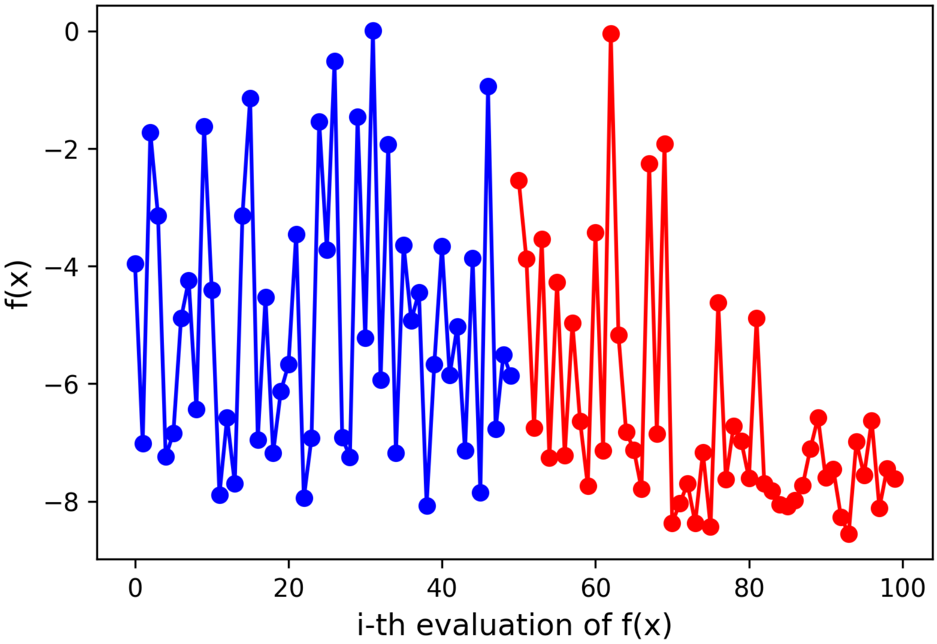


Figure 4.17: Optimization history of FMQA using VAE-based parameterization and $Re = 5000$ ($N_0 = 50$, $N = 100$)

The best latent variables from FMQA optimization with VAE-based parameterization were used to generate the airfoil shown in Figure 4.18. Generally speaking, the resulting shape appears relatively thin overall, with the maximum thickness located approximately at the mid-chord. It also counts with slight positive camber (looking almost symmetric) and a smooth, rounded leading edge that supports attached flow. The airfoil also tapers gradually towards the back, forming a thin trailing edge that helps to reduce pressure drag and supports smoother wake behaviour. Overall, this airfoil seems to be well suited for low drag. Compared to the airfoils obtained in previous sections, this design is almost as thin as those generated using the Joukowski transformation and noticeably thinner than those optimized using the PARSEC parameterization.

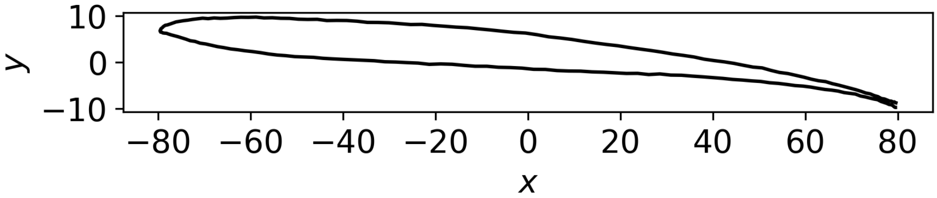


Figure 4.18: Airfoil obtained with FMQA using VAE-based parameterization and $Re = 5000$ ($N_0 = 50$, $N = 100$).

On the other hand, Figure 4.19 shows the optimization history of sDRL using VAE-based parameterization. The best value curve indicates that the optimum is reached after approximately the 10th generation (corresponding to 60 objective function evaluations), with a value lower than $f(\mathbf{x}) = -8.5$. Additionally, the average value curve shows a convergence trend towards the optimum, gradually flattening at a value slightly above the minimum.

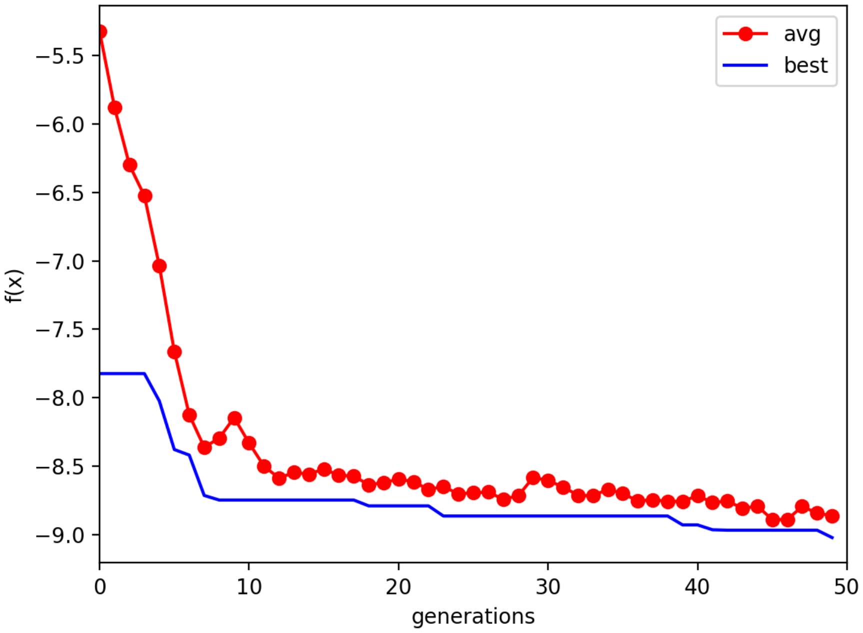


Figure 4.19: Optimization history of sDRL using VAE-based parameterization and $Re = 5000$.

Throughout the optimization process, the airfoil shape evolved towards a thinner configuration, as shown in Figure 4.20. The leading edge remains mostly unchanged, while the rest of the chord exhibits a reduction in thickness. In addition, the optimized shape shows a slightly reduced camber, and the trailing edge is sharper and more tapered. A small irregularity can be observed on the upper surface near mid-chord. This irregularity is likely a consequence of the VAE-based parameterization rather than an intentional aerodynamic feature. In general, the optimization process resulted in a shape typically associated with improved aerodynamic efficiency. Compared to the shape obtained with FMQA, the sDRL optimized shape maintains similar thinness.

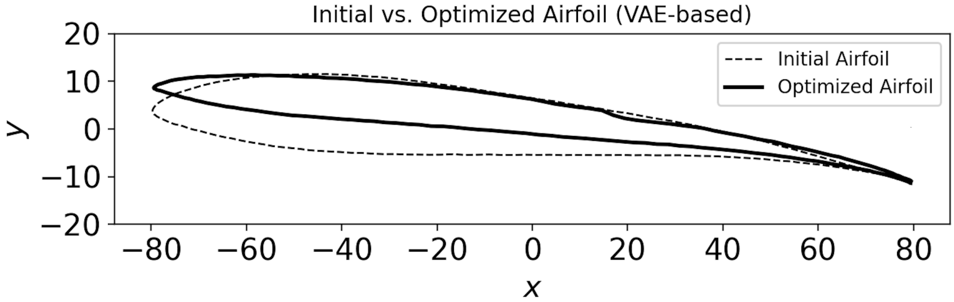


Figure 4.20: Airfoil obtained with sDRL using VAE-based parameterization and $Re = 5000$ (initial vs. optimized).

In summary, sDRL exhibited a clear convergence trend, whereas FMQA showed more stochastic behaviour during the optimization process. Both optimized airfoils exhibited geometric features associated with good aerodynamic performance. For reference, the geometry obtained with the SciPy optimizer is shown in Figure 4.21, achieving a shape similar to that of the sDRL case, including a geometric irregularity around the midchord.

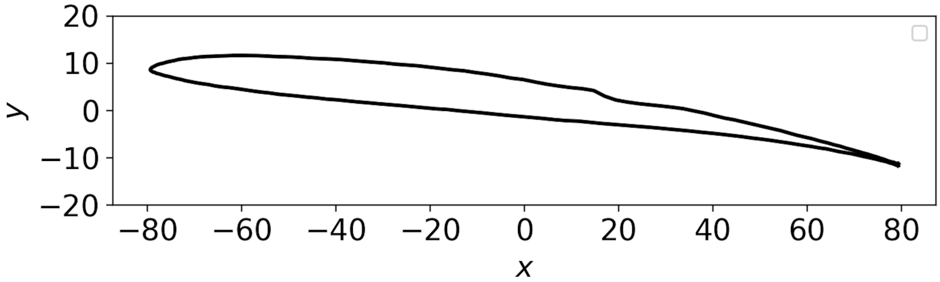


Figure 4.21: Airfoil obtained with SciPy using VAE-based parameterization and $Re = 5000$.

Table 4.9 show the optimized variables together with the optimized objective values for FMQA, sDRL and the SciPy optimizer. Among the three optimization processes, sDRL achieves the highest lift-to-drag ratio, followed by SciPy and then FMQA. Therefore, in this case, only one of the machine learning algorithms outperforms the standard optimizer.

Table 4.9: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
z_0	-1.05	-1.15	-1.48
z_1	1.42	1.20	0.83
z_2	1.05	1.50	1.50
z_3	-1.50	-1.50	-1.38
z_4	0.22	0.28	0.28
z_5	-0.45	-0.11	-0.85
z_6	0.15	1.13	-0.92
z_7	-0.60	-0.35	-0.52
α (°)	5.75	7.08	7.24
L/D	8.556	9.022	8.649

In order to determine whether VAE-based airfoil parameterization improves aerodynamic optimization results compared to classical techniques, it is compared with the results obtained with PARSEC parameterization at $Re = 5000$, shown in Table 4.10. The results indicate that the algorithms performed better when using the VAE-based parameterization, with higher lift-to-drag ratios. Although the VAE occasionally produces small geometric irregularities, it allows the discovery of airfoil shapes with better aerodynamic performance. Therefore, under the simulated conditions, the comparison supports that VAE-based parameterization leads to improved aerodynamic optimization outcomes.

Table 4.10: Optimum of $f(\mathbf{x})$ for the different parameterization schemes for $Re = 5000$

Parameterization	**FMQA** (L/D)	**sDRL** (L/D)	**SciPy** (L/D)
PARSEC	7.157	7.656	7.010
VAE-based	8.556	9.022	8.649

4.5 Parametric Study: Angle of Attack

In this section, a parametric study is conducted to investigate the effect of the angle of attack on aerodynamic performance. In the previous sections, the angle of attack was included as one of the design variables during the optimization process. However, the impact of this parameter on the lift-to-drag ratio was not explicitly analyzed. By isolating this parameter and varying it while performing shape optimization, this study aims to better understand how changes in the angle of attack influence the aerodynamic behaviour.

This study is carried out under the same flow simulation conditions used in previous experiments, specifically a Reynolds number of 5000, and the type of airfoil representation would be VAE-based parameterization. These settings were chosen because of their demonstrated ability to produce high performing airfoils in earlier sections.

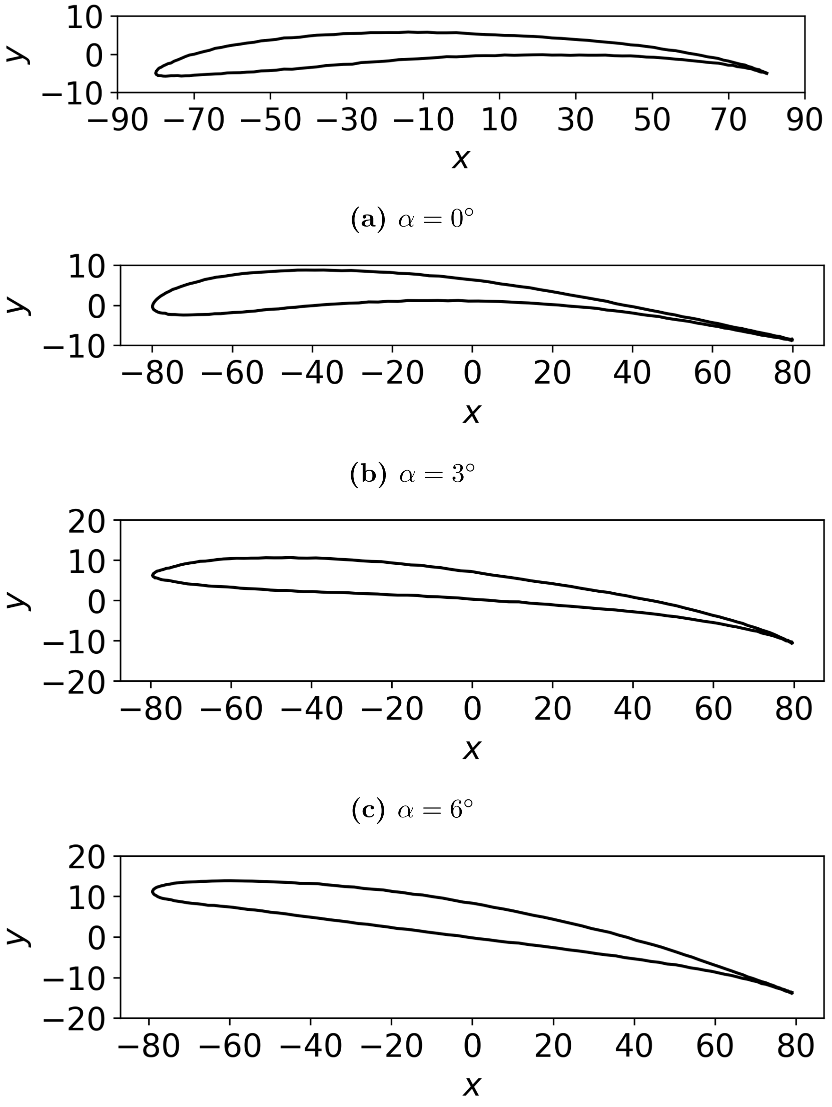
Five discrete angles of attack were selected to have an accurate representation from $\alpha = 0°$ to $\alpha = 12°$, with increments of 3° (that is, $\alpha = 0°$, 3°, 6°, 9°, and 12°). For each fixed angle of attack, the optimization process was carried out independently to identify the optimal airfoil shape that corresponds to that specific orientation. The assessment includes a comparison of the resulting airfoil geometries and the associated lift-to-drag ratios obtained from each optimization run, for both FMQA and sDRL. For simplicity, this study only considers machine learning algorithms, omitting the SciPy comparison. Furthermore, a detailed analysis of the optimization history is excluded here in order to emphasize the aerodynamic results (see Appendix A for details).

Figure 4.22 illustrates the optimized airfoil shapes obtained using FMQA at five different angles of attack ($\alpha = 0°$, 3°, 6°, 9°, 12°) under a Reynolds number of 5000 using VAE-based parameterization. Across the range of angles, all the airfoils share a generally thin and streamlined geometry with smooth leading edges and sharp trailing edges.

Interestingly, optimized airfoils at $\alpha = 0°$ and $\alpha = 3°$ exhibit higher camber than those optimized at higher angles of attack. This result suggests that, at lower angles of attack, the optimizer compensates for the lack of incidence by increasing the camber to enhance lift generation. At higher angles of attack, the shape can rely more on the geometric orientation to generate lift, allowing for less-cambered profiles.

Although higher camber increases lift at low angles, it may also lead to increased drag or reduced aerodynamic efficiency at higher incidence. Therefore, the optimizer appears to reduce camber at higher angles to maintain a good lift-to-drag ratio and avoid the penalties associated with excessive curvature. The comparison of the lift-to-drag ratios obtained from the optimization at different angles will be discussed at the end of the section.

Regarding other aerodynamic features, as the angle of attack increases, the optimized airfoils tend to show minor changes in thickness distribution and variations in the trailing edge orientation. Overall, the FMQA optimization combined with the VAE-based parameterization demonstrates the ability to adapt the airfoil shape to different aerodynamic regimes, maintaining smooth and aerodynamically efficient geometries across a range of angles of attack.



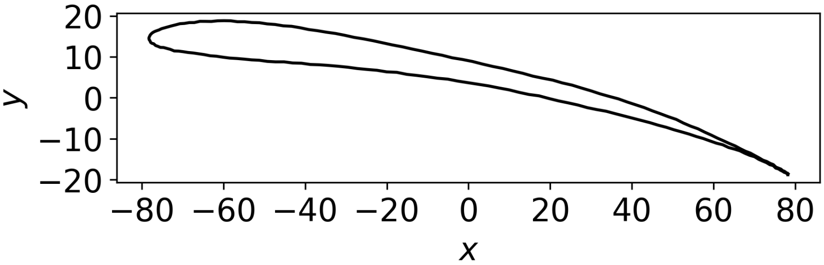
(a) $\alpha = 0°$

(b) $\alpha = 3°$

(c) $\alpha = 6°$

(d) $\alpha = 9°$

Figure 4.22: Parametric study of the angle of attack with VAE-based parameterization. FMQA airfoils



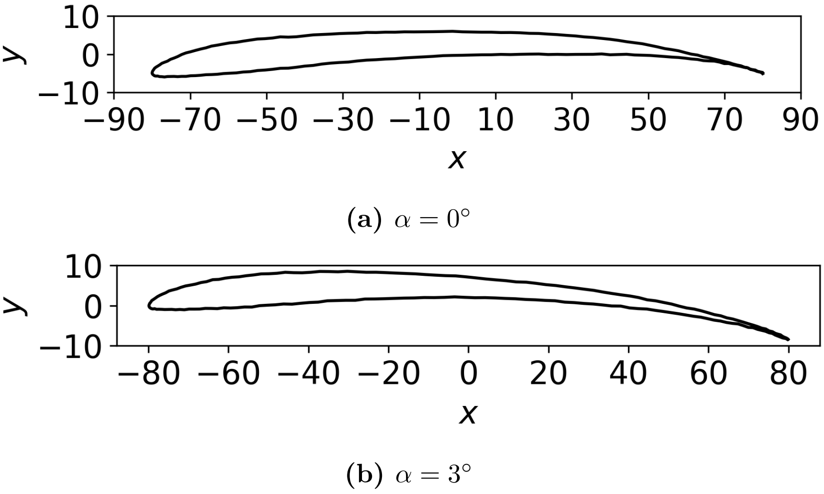
(e) $\alpha = 12°$

Figure 4.22: Parametric study of the angle of attack with VAE-based parameterization. FMQA airfoils.

On the other hand, Figure 4.23 shows the optimized airfoil shapes obtained using sDRL at the same five angles of attack and under the same conditions as those used for the FMQA based optimization. Similarly to the previous case, all resulting shapes fall into the category of thin airfoils with smooth leading edges. In particular, the airfoils at $\alpha = 9°$ and $\alpha = 12°$ appear slightly thicker at the leading edge compared to those optimized at lower angles of attack.

The same camber trend observed at lower angles of attack in the FMQA optimized airfoils is also identified in the sDRL results, with airfoils optimized at lower angles showing greater camber than those optimized at higher angles of attack. Again, this suggests that a higher camber may be a favorable design feature for improving lift generation at low angles of attack.

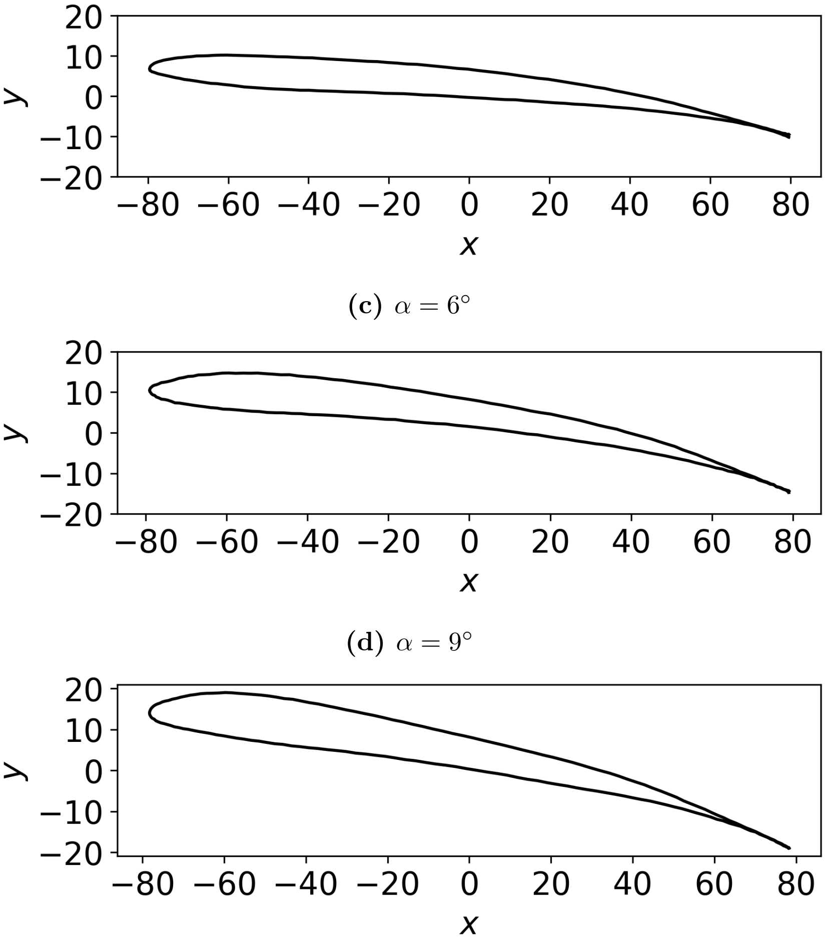
In general, both algorithms produced very similar optimized airfoils, with only minor variations in thickness distribution along the chord, all characteristic of thin airfoil shapes.



(a) $\alpha = 0°$

(b) $\alpha = 3°$

Figure 4.23: Parametric study of the angle of attack with VAE-based parameterization. sDRL airfoils.



(c) $\alpha = 6°$

(d) $\alpha = 9°$

(e) $\alpha = 12°$

Figure 4.23: Parametric study of the angle of attack with VAE-based parameterization. sDRL airfoils.

Regarding the obtained lift-to-drag ratios at each angle of attack, Table 4.11 presents a comparison between the FMQA and sDRL algorithms. The results indicate that both algorithms produce very similar L/D values across all angles, demonstrating comparable optimization performance under the specified conditions. Both algorithms achieve their highest lift-to-drag ratio at $\alpha = 6°$, with sDRL showing slightly superior performance. For the rest of the angles, this algorithm also slightly outperforms FMQA by a small margin. Despite these minor differences, the results suggest that both algorithms are well-suited for this task.

Table 4.11: Optimum of $f(\mathbf{x})$ at the different angles of attack for VAE parameterization and Re $= 5000$

$\alpha(°)$	**FMQA** ($r_{LD} = L/D$)	**sDRL** ($r_{LD} = L/D$)
0.00	5.312	5.3389
3.00	8.0258	8.5650
6.00	8.7351	8.9042
9.00	8.2277	8.5329
12.00	7.0682	7.0208

4.6 Benchmark Cases

In this section, other benchmark cases beyond the lift-to-drag ratio optimization are assessed to further evaluate the performance of the algorithms. Specifically, a new aerodynamic optimization problem is formulated, where the objective function includes a target lift coefficient while simultaneously trying to minimize the drag coefficient of the airfoil shape. Then, Drela's optimization problem [38] is introduced, with some modifications to align it with the type of simulations supported by the Lattice Boltzmann Method (LBM) solver.

4.6.1 New Aerodynamic Objective: Target Lift Coefficient

In this subsection, a new aerodynamic optimization problem is designed to evaluate performance under multiple criteria. The objective function, defined in Equation 4.3, incorporates both a target lift coefficient and the drag coefficient of the airfoil. The goal is to minimize the resulting cost function, balancing lift generation with drag reduction.

$$\text{Minimize } f(\mathbf{x}) \equiv \beta \cdot (C_{L_{\text{target}}} - C_L)^2 + C_D^2 \tag{4.3}$$

where $\beta = 10$ and $C_{L_{\text{target}}} = 0.8$. The design variables include both the airfoil shape parameters and the angle of attack. The simulations are performed at a Reynolds number of 5000. The algorithms will be evaluated under two parameterization schemes: PARSEC and VAE-based. Consequently, this experiment also serves to assess whether VAE-based parameterization improves aerodynamic optimization results when evaluated under a different objective than the lift-to-drag ratio.

The analysis begins with a discussion of the convergence behavior and the optimized airfoil geometries obtained from each parameterization scheme and algorithm. At the end of the section, a comparison of the optimized objective values is presented to assess the overall performance of both FMQA and sDRL under each parameterization, including their comparison with the SciPy optimizer.

PARSEC parameterization

First, the performance of FMQA and sDRL will be evaluated using the PARSEC parameterization under the new aerodynamic objective. In this case, FMQA is run with $N_0 = 50$ and $N = 100$. Figure 4.24 presents the corresponding optimization history under the defined conditions. At first glance, the algorithm appears to improve the training data over the course of the optimization cycles, displaying an overall positive convergence trend, despite a few less favorable evaluations scattered throughout the process. In addition, the optimization history suggests that the optimum was identified early in the process. This optimum has a value close to zero, meaning that the FMQA algorithm successfully identified an airfoil shape that satisfies the target lift coefficient of $C_L = 0.8$ while also minimizing drag.

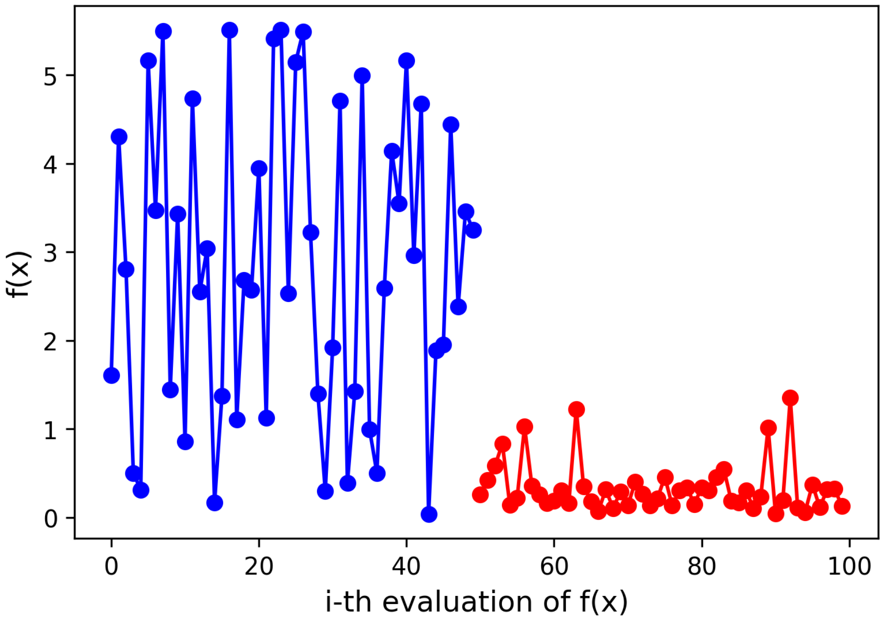


Figure 4.24: Optimization history of FMQA ($N_0 = 50$, $N = 100$) with PARSEC parameterization and $Re = 5000$ for the new objective.

The corresponding airfoil shape obtained with FMQA and PARSEC parameterization is shown in Figure 4.25. It presents a relatively thin airfoil (slightly thicker than other PARSEC derived airfoils) characterized by a smooth, rounded geometry. The thickness is evenly distributed along the chord, with a trailing edge that tapers gradually towards the end.

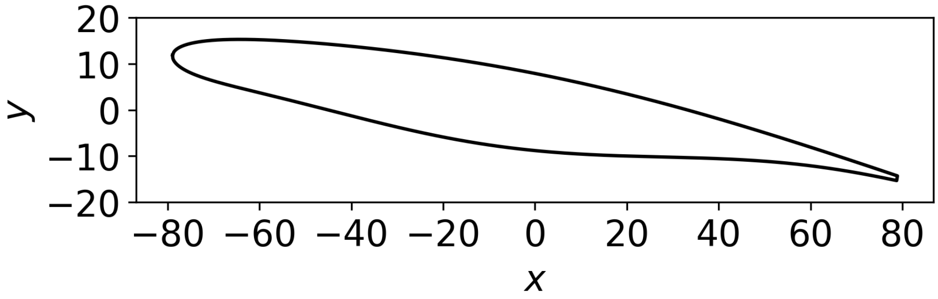


Figure 4.25: Airfoil obtained with FMQA ($N_0 = 50$, $N = 100$) with PARSEC parameterization and $Re = 5000$ for the new objective.

In addition, the camber of the airfoil shape is moderate with a maximum located around the mid-chord. The smooth curvature and the absence of sharp edges suggest that the shape is ideal for maintaining attached, laminar flow across most of the surface, which is desirable under the given conditions. All in all, it appears that airfoil is well-suited for the given task of achieving a certain lift coefficient while minimizing drag, something also shown by the optimization history.

Moving on to the sDRL optimization, Figure 4.26 shows the optimization history of sDRL with PARSEC parameterization and its progression over evaluations. The figure presents a clear improvement in the objective function, demonstrating the ability of the algorithm to explore and exploit the design space effectively. Within the first 20 generations, the average value curve seems to gradually approach the best value curve, showing a positive convergence trend. After this phase, the average value curve begins to flatten towards the optimum, suggesting that the search process has entered a phase of fine-tuning around high performing designs. Again, the fact that the optimum is close to zero demonstrates the ability of the algorithm to find a shape that reaches the target lift coefficient while minimizing drag.

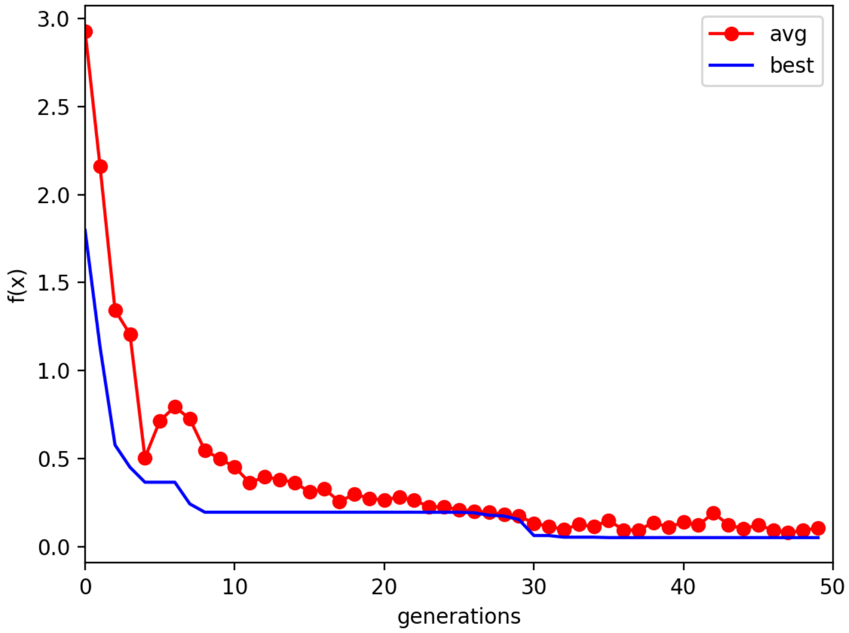


Figure 4.26: Optimization history of sDRL with PARSEC parameterization and $Re = 5000$ for the new objective

The airfoil obtained through the sDRL optimization with PARSEC parameterization is illustrated in Figure 4.27. Compared to the optimized airfoil obtained with FMQA, this shape appears slightly thinner and has a more pointed leading edge. Regarding other aerodynamic features such as the trailing edge, the sDRL airfoil seems sharper and thinner than the FMQA airfoil, which could help reduce pressure drag but may also make it more sensitive to flow separation or instability. Overall, the sDRL shape appears more aggressive in its design, whereas the FMQA algorithm presents a smoother geometry. The optimized objective function values will be discussed at the end of the subsection together with those obtained using VAE-based parameterization.

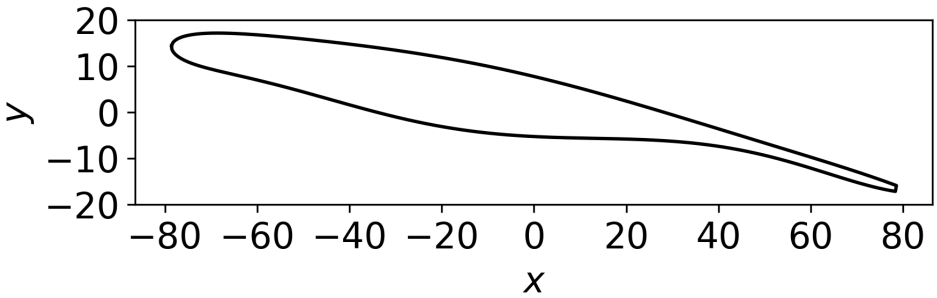


Figure 4.27: Airfoil obtained with sDRL with PARSEC parameterization and $Re = 5000$ for the new objective

Similarly to the previous experiments, an extra comparison with an standard optimizer (`differential_evolution` from SciPy) is performed to validate the results. Figure 4.28 illustrates the geometry obtained with the SciPy optimizer. The resulting shape is a thin airfoil, closely resembling the characteristics of the one achieved with FMQA. At the end of the subsection, the optimized objective value obtained with the standard optimizer will be discussed for both parameterization schemes.

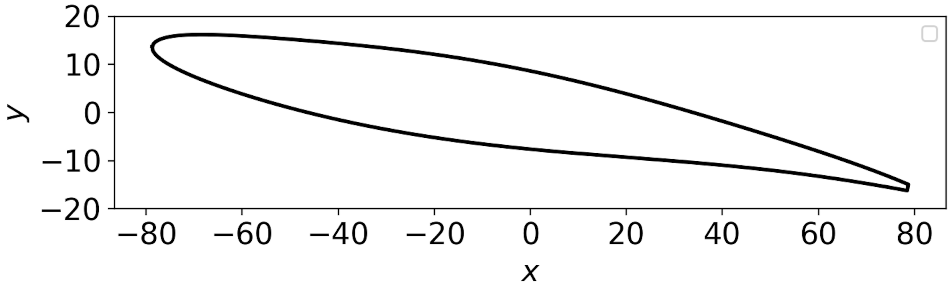


Figure 4.28: Airfoil obtained with SciPy with PARSEC parameterization and $Re = 5000$ for the new objective

VAE-based parameterization

After evaluating PARSEC parameterization, the use of VAE-based parameterization is also analyzed for both algorithms under this new objective. FMQA is run again with $N_0 = 50$ and $N = 100$. Figure 4.29 shows the optimization history of this algorithm with VAE-based parameterization and $Re = 5000$. The convergence behavior is similar to that observed in the previous case, showing an overall positive trend. Although some less favorable evaluations appear throughout the optimization cycles, most of the predictions are around the optimum. This optimum appears again to be close to zero, demonstrating the ability of the FMQA algorithm to reach the target lift coefficient while minimizing drag.

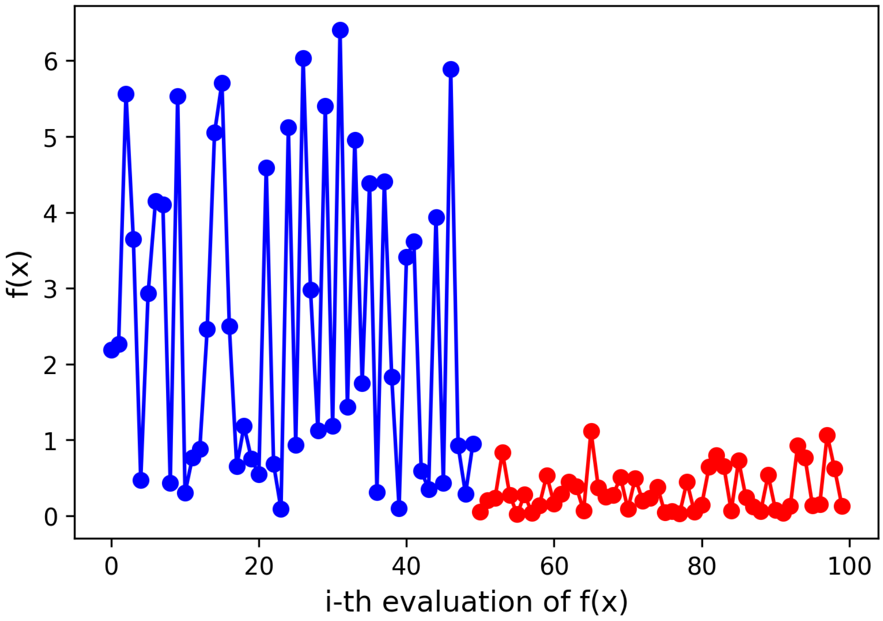


Figure 4.29: Optimization history of FMQA ($N_0 = 50$, $N = 100$) with VAE-based parameterization and $Re = 5000$ for the new objective

Figure 4.30 shows the optimized airfoil from FMQA optimization with VAE-based parameterization. The resultant shape shows a cambered airfoil with a smooth and rounded leading edge. It appears noticeably thicker near the leading edge compared to other optimized airfoils, which may contribute to improved flow stability. The trailing edge is sharp and clean, as expected to reduce wake and pressure drag.

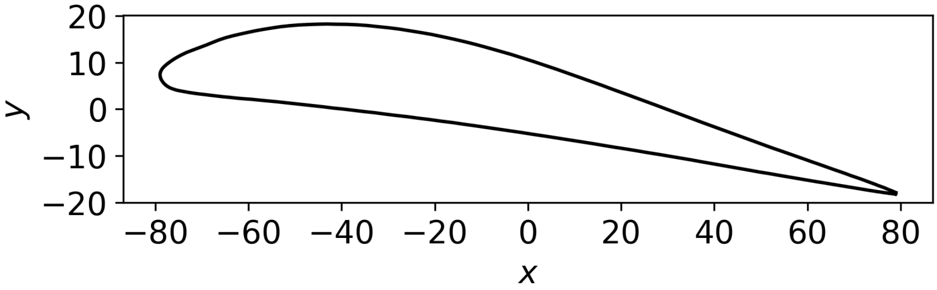


Figure 4.30: Airfoil obtained with FMQA ($N_0 = 50$, $N = 100$) with VAE-based parameterization and $Re = 5000$ for the new objective

On the other hand, Figure 4.31 shows the optimization history of sDRL with VAE-based parameterization for this new objective. The figure shows a clear downward trend in the objective function across generations. Within the first 10-20 generations, the average value curve converges toward the best value curve, indicating a positive convergence trend in the algorithm. Near optimal values are achieved around generation 10. The final objective values approach zero, demonstrating the algorithms capability to satisfy the target lift coefficient while simultaneously minimizing the drag coefficient.

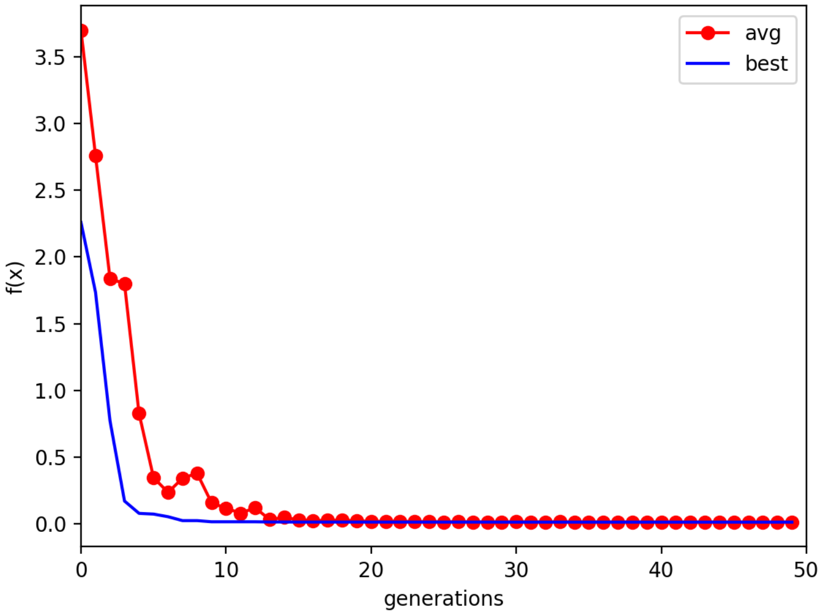


Figure 4.31: Optimization history of sDRL with VAE-based parameterization and $Re = 5000$ for the new objective

The optimized shape obtained from the previous optimization process is shown in Figure 4.32. It corresponds to a cambered airfoil with a smooth and rounded leading edge. Additionally, the airfoil shape appears to be thinner than the optimized airfoil obtained with FQMA, especially near the leading edge. The trailing edge in this case appears slightly sharper and thinner in comparison, a feature typically associated with reduced wake and pressure drag.

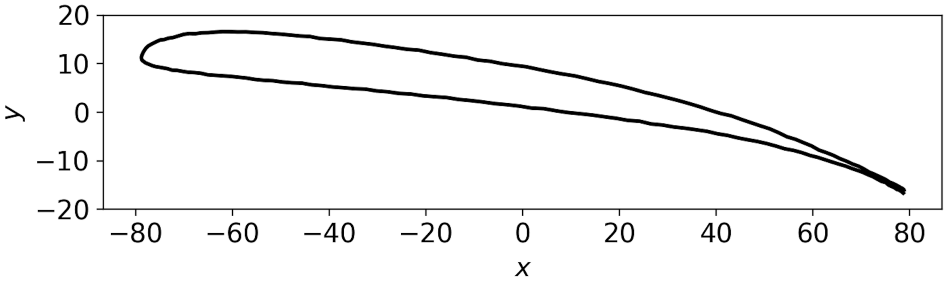


Figure 4.32: Airfoil obtained with sDRL with VAE-based parameterization and Re = 5000 for the new objective

For comparison with the standard optimizer, Figure 4.33 shows the geometry achieved with `differential_evolution` algorithm from SciPy. The airfoil resembles the FMQA optimized geometry in camber and overall shape but exhibits a thinner thickness distribution. It is nearly as thin as the sDRL optimized geometry, although with a slightly thicker leading edge. The optimized objective value will be discussed below together with the machine learning results.

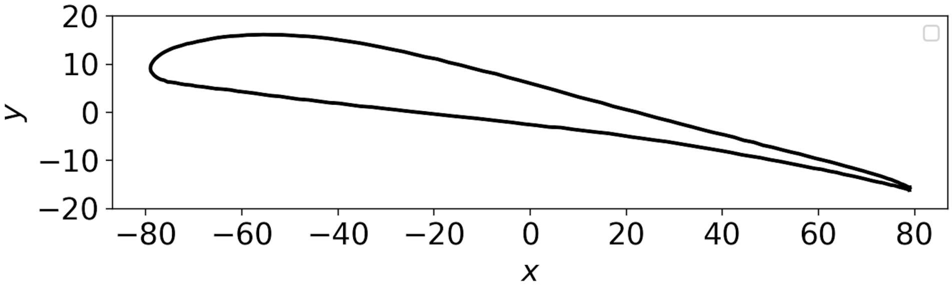


Figure 4.33: Airfoil obtained with SciPy with VAE-based parameterization and Re = 5000 for the new objective

The optimized objective values for the new aerodynamic objective are presented in Table 4.12, categorized by parameterization method and optimization algorithm. Broadly speaking, both FMQA and sDRL successfully met the new objective across parameterization methods: achieving the target lift coefficient while minimizing the drag coefficient. However, upon closer examination, the VAE-based parameterization appears to have achieved lower objective values compared to the PARSEC parameterization for both algorithms, indicating a more effective minimization of the objective. Therefore, this numerical experiment corroborates the idea that using VAE-based parameterization results in improved aerodynamic optimization performance.

Table 4.12: Optimum of $f(\mathbf{x})$ for the different parameterization schemes for $Re = 5000$

Parameterization	**FMQA ($f(\mathbf{x})$)**	**sDRL ($f(\mathbf{x})$)**	**SciPy ($f(\mathbf{x})$)**
PARSEC	0.04215	0.05053	0.03449
VAE-based	0.02243	0.01207	0.01262

Regarding algorithmic performance, FMQA achieved better results when using the PARSEC parameterization, whereas sDRL obtained a lower objective value with the VAE-based parameterization. This suggests that each algorithm performed best within a different parameterization scheme. Compared to the standard optimizer, SciPy outperformed both FMQA and sDRL under PARSEC parameterization. With the VAE-based parameterization, the SciPy results fall between those of FMQA and sDRL, with sDRL achieving the best overall performance. For details on the optimized design parameters, see Appendix B.

4.6.2 Drela's Optimization Problem

Other benchmark cases, such as Drela's optimization problem [38], are analyzed in this subsection. As can be seen below, a 6-point weighted optimization problem is defined, where the drag coefficient C_D is optimized at 6 different fixed lift coefficients C_L with the shape parameters S_i and angle of attack α as design variables. Some modifications were introduced to the original objective function defined by Drela. Since the LBM solver simulates laminar flow at low Reynolds numbers, achieving shapes with a lift coefficient of $C_L = 1.6$ was a difficult task. Therefore, instead of evaluating the drag coefficient at lift values $C_L = [0.8, ..., 1.6]$, the interval was adjusted to $C_L = [0.8, ..., 1.4]$ to better suit the limitations of the simulation setup.

$$\begin{aligned}\text{Minimize: } F(\{S_i\}, \alpha) \equiv\ & \frac{5}{45} C_D|_{C_L=0.8} + \frac{6}{45} C_D|_{C_L=1.0} + \frac{7}{45} C_D|_{C_L=1.1} \\ & + \frac{8}{45} C_D|_{C_L=1.2} + \frac{9}{45} C_D|_{C_L=1.3} + \frac{10}{45} C_D|_{C_L=1.4}\end{aligned}$$

$$\begin{aligned}\text{Subject to: } & C_M \geq -0.133 \qquad \left(\frac{t}{c}\right)_{0.33c} \geq 0.128 \qquad \left(\frac{t}{c}\right)_{0.90c} \geq 0.014 \\ & C_L = [0.8,\ 1.0,\ 1.1,\ 1.2,\ 1.3,\ 1.4]\end{aligned}$$

The optimization problem is subject to aerodynamic and geometric constraints on the pitching moment coefficient, the thickness-to-chord ratio at $x/c = 0.33$ and $x/c = 0.90$, and the lift coefficient. These constraints are enforced by adding penalties to the objective function whenever the design fails to meet the required conditions. Regarding other simulation conditions, a Reynolds number of 1000 was selected to remain within the stable and accurate operating range of the LBM solver. Additionally, VAE-based parameterization was selected as the parameterization scheme. To compute the drag coefficient values, a lift-drag polar is constructed by simulating the airfoil at different angles of attack. The drag coefficients are then interpolated or extrapolated to estimate C_D at the desired lift values C_L, and the optimization objective is computed for each candidate accordingly.

In this numerical experiment, the algorithms are compared in terms of performance and convergence behavior, the optimized objective value, and the resulting airfoil shapes, including a final comparison with the standard optimizer. FMQA is run this time $N_0 = 100$ and $N = 200$. Figure 4.34 shows the optimization history of this algorithm for Drela's optimization problem at the defined simulation conditions. The initial training data explores a wide range of the design space, including candidates that violate the constraints. In such cases, penalties are applied to the objective function, resulting in values exceeding 1. During the optimization cycles, most candidate designs satisfy the constraints, with only a few evaluations exceeding the limits. Overall, the optimization results show clear improvement over the initial training data, with most candidates from the optimization cycles achieving objective values below 1. This indicates a positive convergence trend in the algorithm, despite a small number of evaluations that violate the constraints.

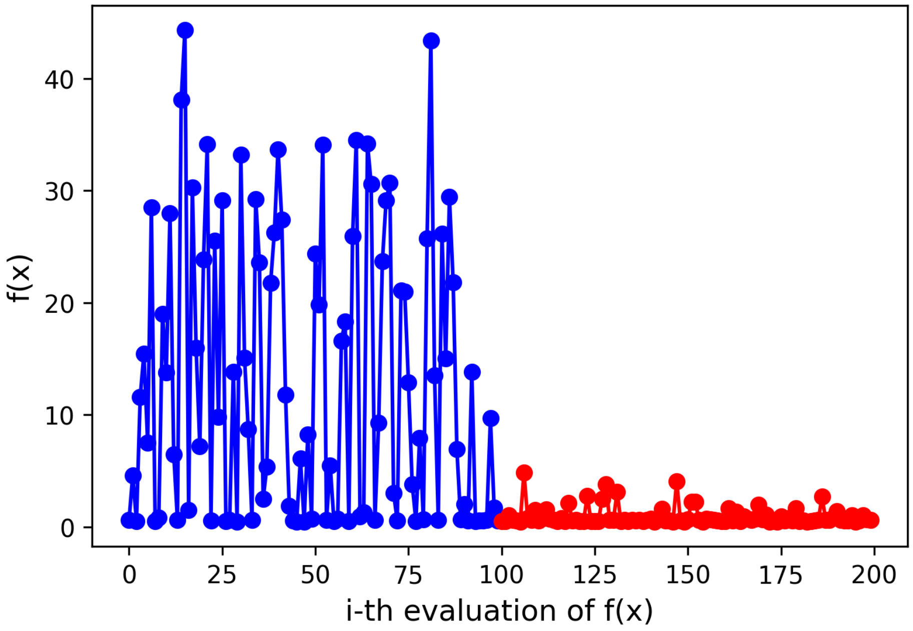


Figure 4.34: Optimization history of FMQA ($N_0 = 100$, $N = 200$) with VAE-based parameterization and $Re = 1000$ for Drela's optimization problem.

On the other hand, Figure 4.35 presents the optimized airfoil shape obtained by the FMQA algorithm for Drela's optimization problem. The geometry shows moderate thickness near the leading edge and a relatively thin trailing edge. The profile also shows high camber, with the upper surface being considerably more convex than the lower surface. Compared to other optimized airfoils obtained in this MSc thesis, this geometry has a more distinctive form, resembling the DAE-11 airfoil presented in [38], but with some geometric variations.

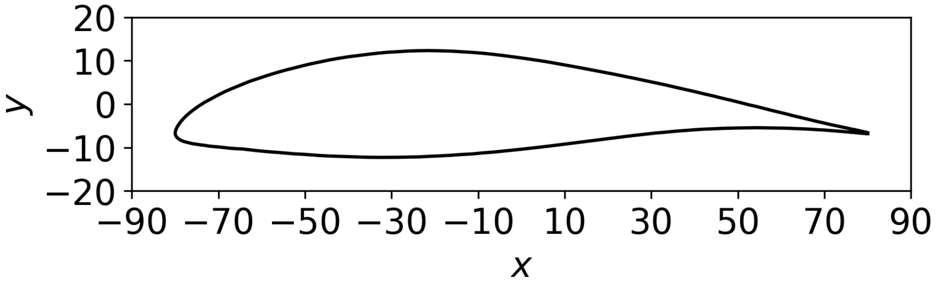


Figure 4.35: Airfoil obtained with FMQA ($N_0 = 100$, $N = 200$) with VAE-based parameterization and $Re = 1000$ for Drela's optimization problem.

Moving on to the optimization performed by sDRL, Figure 4.36 shows the optimization history of this algorithm under the defined conditions for Drela's optimization problem. Overall, the algorithm demonstrates good performance, with only the first generations containing individuals that slightly violate the constraints. After approximately the 20th generation, the average value curve converges to the best value curve, indicating a positive convergence trend. Compared to FMQA, sDRL identifies a greater number of candidates that satisfy the constraint ranges. As noted, convergence of the objective function is achieved after the 20th generation, corresponding to approximately 120 evaluations.

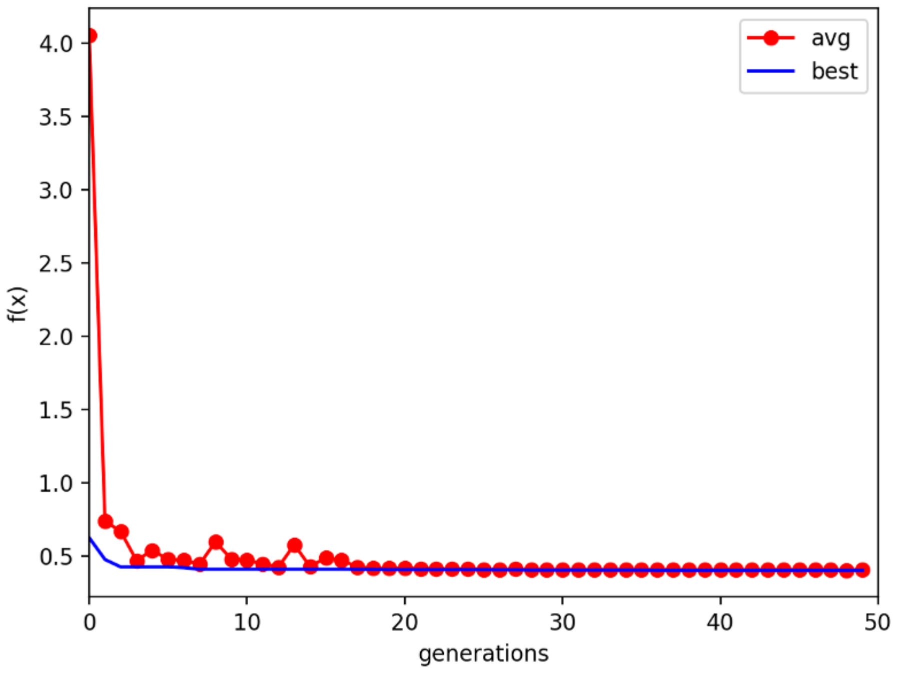


Figure 4.36: Optimization history of sDRL with VAE-based parameterization and $Re = 1000$ for Drela's optimization problem.

The resulting airfoil shape from the previous optimization process is shown in Figure 4.37. The geometry generated from the optimal parameters obtained by sDRL exhibits moderate thickness, with the maximum thickness located forward of the mid-chord. The airfoil also displays high camber and a sharp trailing edge. These geometric characteristics are similar to those of the FMQA optimized airfoil. However, in this case, the sDRL optimized airfoil shows smoother leading and trailing edges, as well as a more gradual thickness transition compared to the FMQA result.

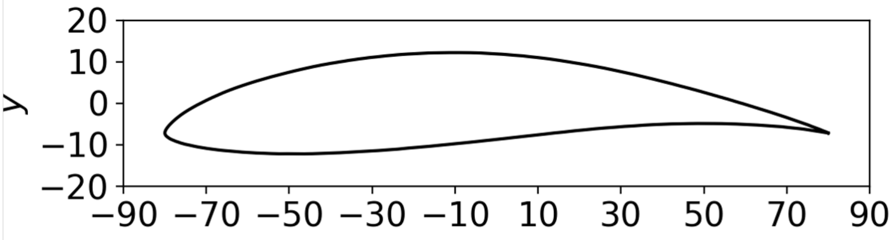


Figure 4.37: Airfoil obtained with sDRL with VAE-based parameterization and $Re = 1000$ for Drela's optimization problem.

Finally, the performance and results of FMQA and sDRL are compared with a standard optimizer. Figure 4.38 shows the optimized geometry obtained with the `differential_evolution` algorithm from SciPy. The resulting airfoil resembles the geometries achieved with FMQA and sDRL. In this case, the lower surface appears flatter than in the sDRL optimized airfoil, whereas the thickness distribution is slightly greater from mid-chord towards the trailing edge. The objective value achieved with SciPy optimization will be compared with the results of FMQA and sDRL at the end of the subsection.

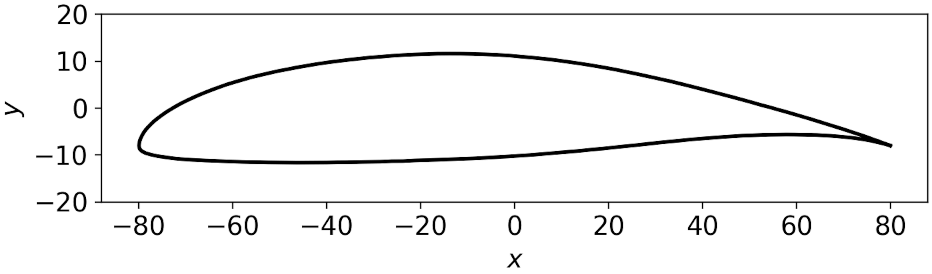


Figure 4.38: Airfoil obtained with SciPy with VAE-based parameterization and $Re = 1000$ for Drela's optimization problem.

To summarize, Table 4.13 presents the optimized objective values obtained by the machine learning algorithms and the standard optimizer for Drela's optimization problem. The algorithms successfully minimized the drag coefficient across the six different lift coefficients. However, the objective value is slightly lower in the case of sDRL. Specifically, the objective value obtained by sDRL is 8.85% lower than the one achieved by FMQA, indicating a slight improvement in optimization performance. Compared with the standard optimizer, sDRL also outperformed SciPy, whereas FMQA achieved a slightly higher objective value than SciPy, suggesting that the classical optimizer achieved a deeper minimization in this case.

Table 4.13: Optimum of $f(\mathbf{x})$ for Drela's optimization problem

Parameterization	**FMQA ($f(\mathbf{x})$)**	**sDRL ($f(\mathbf{x})$)**	**SciPy ($f(\mathbf{x})$)**
VAE-based	0.44426	0.40497	0.4156

Overall, sDRL outperformed FMQA in this optimization problem. Although FMQA demonstrated strong performance during the optimization process, with a clear positive convergence trend, the results obtained by sDRL were slightly better. The algorithm identified candidates within the constraints ranges and converged to a lower minimum objective value. Furthermore, the sDRL optimized airfoil showed a geometry more favorable for low drag performance, with smoother leading and trailing edges. Regarding the comparison with SciPy, the standard optimization provided a useful baseline, outperforming FMQA but not surpassing sDRL.

Chapter 5

Conclusions

The main objective of this MSc thesis was to explore and compare the FMQA and sDRL algorithms for airfoil shape optimization. For this purpose, a series of numerical experiments were conducted to assess their performance, convergence behaviour, optimized objective values, and the resulting airfoil geometries. This section presents the conclusions drawn from these experiments, highlighting the strengths and limitations of each algorithm in the tested scenarios through the different research questions.

The research was structured around four main research questions, presented below.

- How do Factorization Machine Quantum Annealing (FMQA) and single-step Deep Reinforcement Learning (sDRL) compare in terms of performance and efficiency for airfoil shape optimization, relative to a standard optimization method?

Performance and efficiency of FMQA and sDRL were assessed primarily by the aerodynamic objective values, their convergence behaviour, and the characteristics of the optimized airfoil shapes. Besides a comparison between both machine learning algorithms, FMQA and sDRL were compared against a standard optimizer from SciPy (differential evolution) to validate their capabilities.

Across the numerical experiments, sDRL generally outperformed FMQA in terms of optimized objective values, with only a few exceptions. In terms of convergence, both algorithms showed both stochastic behaviour and positive convergence trends, depending on the numerical experiment. FMQA often displayed stochastic fluctuations with less clear convergence trends in the first baseline comparisons, whereas sDRL showed a more consistent progression, particularly in cases with higher Reynolds numbers and advanced parameterization schemes. In the benchmark cases, both algorithms showed clear convergence trends.

The resulting airfoil shapes from both FMQA and sDRL displayed good aerodynamic properties, with only minor differences in thickness distribution and camber. The use of different airfoil parameterizations was reflected in the diversity of airfoil geometries.

When compared to the standard optimizer, both FMQA and sDRL produced competitive results. In some cases, SciPy outperformed the machine learning algorithms, while in others, especially with sDRL, the ML based approaches identified slightly better optimization candidates. This suggests that while classical optimizers remain effective for airfoil design, machine learning algorithms (particularly sDRL) offer the potential to deliver improved results in certain scenarios.

Advantages and Limitations

- FMQA demonstrated flexibility and was especially effective when coupled with advanced parameterizations such as the VAE-based scheme. However, its convergence behaviour was often more stochastic and less stable.

- sDRL generally showed clearer convergence trends and achieved slightly better results in most numerical experiments, although it did not always outperform FMQA in every case.

Computational trade-offs

Both FMQA and sDRL were similarly demanding in terms of computational cost, with each simulation typically requiring approximately 75 - 90 minutes. The main distinction was observed in their convergence behaviour: sDRL generally exhibited smoother convergence trends, whereas FMQA was more stochastic and less stable.

- How do FMQA and DRL perform when optimizing airfoil shapes across different Reynolds number regimes?

When the Reynolds number was increased from 800 to 5000, both FMQA and sDRL achieved significant improvements in aerodynamic performance, with lift-to-drag ratios nearly doubling. Both algorithms adapted to the new regime, but sDRL again demonstrated a clearer convergence trend and achieved a slightly higher optimized value (around 7% better than FMQA). The optimized airfoil shapes remained thin across both regimes. These results indicate that both algorithms are capable of handling changes in flow regime within the limits of the LBM solver. The increase in Reynolds number primarily improved aerodynamic performance rather than altering the overall optimization dynamics.

- How does the choice of airfoil parameterization (Joukowski transformation, PARSEC) influence the optimization performance of FMQA and DRL?

The parameterization scheme had a noticeable impact on some aspects of the optimization results. With the Joukowski transformation, both algorithms produced very thin airfoils and achieved similar objective values, with sDRL slightly outperforming FMQA. With PARSEC parameterization, the optimized shapes became thicker, the convergence improved (especially for sDRL), and both algorithms found high quality designs. However, these geometric differences did not translate into higher aerodynamic performance, as the objective values obtained with PARSEC were comparable to (or slightly lower than) those achieved with Joukowski.

Therefore, the choice of parameterization strongly influences the type of geometries explored during optimization, but does not necessarily guarantee improved aerodynamic results. While this was the case for the classical schemes explored here, the following research question shows that the VAE-based parameterization can lead to different outcomes.

- To what extent does VAE-based airfoil parameterization improve aerodynamic optimization results compared to classical parameterization techniques?

The VAE-based parameterization significantly enhanced optimization results compared to classical methods (Joukowski and PARSEC). Both FMQA and sDRL displayed clearer convergence behaviour under this scheme, and the resulting airfoil geometries exhibited aerodynamic features that led to higher lift-to-drag ratios, although minor geometric irregularities were observed in a few cases. Benchmark cases further reinforced the advantages of the VAE approach, as it produced lower objective values compared to PARSEC in multi-objective optimization scenarios. In general, VAE parameterization improved aerodynamic performance and demonstrated great potential as a powerful alternative to classical schemes in airfoil optimization.

With all the explored cases, this MSc thesis extends the application of FMQA and sDRL in the context of airfoil shape optimization, demonstrating their capabilities through a series of optimization problems covering different aerodynamic objectives, flow regimes, and parameterization schemes. The research objective was achieved by assessing and comparing their performance across multiple numerical airfoil shape optimization cases, using both classical and VAE-based parameterizations, and always relative to a standard optimizer baseline. This work highlights the potential of FMQA and sDRL as flexible and competitive optimization strategies for practical aerodynamic design problems, while showing that in certain scenarios they can even outperform classical optimizers (particularly in the case of sDRL). Overall, the thesis provides a foundation for their application to more complex and computationally demanding scenarios.

Chapter 6

Recommendations for Future Work

While the results of this study demonstrated the capabilities of FMQA and sDRL for airfoil shape optimization across a range of aerodynamic objectives, parameterization schemes, and flow regimes, several opportunities remain open for further research and development. This work also represents one of the first attempts to integrate VAE-based parameterization within optimization algorithms such as FMQA and sDRL, highlighting its potential and the need for further investigation into its applicability and performance in more complex aerodynamic design scenarios. Building on the results of this work, several directions can be explored to further advance the applications of FMQA and sDRL in aerodynamic shape optimization:

Integration with higher-fidelity solvers
This work made use of a low-fidelity solver based on the Lattice Boltzmann Method (LBM) due to its computational cost and ease of setting boundary conditions. Future research could extend the optimization problems to more complex higher-fidelity solvers to assess the performance of the algorithms under more realistic aerodynamic conditions, including the effects of turbulent flows.

3D aerodynamic shape optimization
The results presented in this MSc thesis were based on 2D airfoil shapes to evaluate the aerodynamic objectives of the optimization problems, chosen for their ease of application and simplicity. Future work could extend this approach to 3D geometries to better reflect practical aerodynamic design challenges.

Multi-objective optimization
This MSc thesis considered individual aerodynamic optimization objectives to evaluate the capabilities of FMQA and sDRL. Future research could adapt these algorithms to handle multiple objectives simultaneously, such as maximizing the lift-to-drag ratio while minimizing the pitching moment, to assess their performance in more complex and realistic design scenarios.

Extension of VAE-based parameterization to more complex problems
This work demonstrated that using VAE-based parameterization in the presented optimization problems led to improved aerodynamic performance. Future research could build on this approach by applying VAE-based parameterization to more complex aero-

dynamic optimization problems. Potential extensions include three-dimensional configurations, higher Reynolds number flows, and multi-objective design frameworks, enabling a broader and more comprehensive comparison with classical parameterization schemes.

Experimental validation

In order to complement the numerical findings, future work could involve validating the optimized airfoil shapes through wind tunnel experiments or high-fidelity CFD simulations. Such validation would help confirm the results obtained with FMQA and sDRL and assess their robustness beyond the simulated conditions used in this work.

Bibliography

[1] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, and E. Hachem, "Direct shape optimization through deep reinforcement learning," *Journal of Computational Physics*, vol. 428, 3 2021.

[2] K. Swannet, C. Varriale, and N. A. K. Doan, "Towards Universal Parameterization: Using Variational Autoencoders to Parameterize Airfoils," in *AIAA SciTech Forum and Exposition, 2024*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2024.

[3] J. Viquerat, R. Duvigneau, P. Meliga, A. Kuhnle, and E. Hachem, "Policy-based optimization: single-step policy gradient method seen as an evolution strategy," 4 2021. [Online]. Available: http://arxiv.org/abs/2104.06175

[4] T. P. Dussauge, W. J. Sung, O. J. Pinon Fischer, and D. N. Mavris, "A reinforcement learning approach to airfoil shape optimization," *Scientific Reports*, vol. 13, no. 1, 12 2023.

[5] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Annual Review of Fluid Mechanics. Machine Learning for Fluid Mechanics," 2019. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010719-060214

[6] Z. Liu, M. Zhang, D. Sun, L. Li, and G. Chen, "A deep reinforcement learning optimization framework for supercritical airfoil aerodynamic shape design," *Structural and Multidisciplinary Optimization*, vol. 67, no. 3, 3 2024.

[7] H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga, and E. Hachem, "Single-step deep reinforcement learning for two- and three-dimensional optimal shape design," *AIP Advances*, vol. 12, no. 8, 8 2022.

[8] K. Kitai, J. Guo, S. Ju, S. Tanaka, K. Tsuda, J. Shiomi, and R. Tamura, "Designing metamaterials with quantum annealing and factorization machines," *Physical Review Research*, vol. 2, no. 1, 3 2020.

[9] K. Endo and K. Z. Takahashi, "Function Smoothing Regularization for Precision Factorization Machine Annealing in Continuous Variable Optimization Problems," 7 2024. [Online]. Available: http://arxiv.org/abs/2407.04393

[10] "Black-Box Optimization of the Airfoil Geometry with Fluid Flow Simulation." [Online]. Available: https://amplify.fixstars.com/en/demo/fmqa_3_aerofoil

[11] J. Nocedal and S. J. Wright, "Numerical Optimization Second Edition," Tech. Rep.

[12] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A Survey of Optimization Methods from a Machine Learning Perspective," 6 2019. [Online]. Available: http://arxiv.org/abs/1906.06821

[13] S. N. Skinner and H. Zare-Behtash, "State-of-the-art in aerodynamic shape optimisation methods," pp. 933–962, 1 2018.

[14] Y. Y. Wang, B. Q. Zhang, and Y. C. Chen, "Robust airfoil optimization based on improved particle swarm optimization method," *Applied Mathematics and Mechanics (English Edition)*, vol. 32, no. 10, pp. 1245–1254, 10 2011.

[15] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics," *Computers and Fluids*, vol. 225, 7 2021.

[16] R. M. Hicks and P. A. Henne, "Wing Design by Numerical Optimization," *Journal of Aircraft*, vol. 15, no. 7, pp. 407–412, 1978.

[17] M. Secanell, A. Suleman, and P. Gamboa, "Design of a morphing airfoil using aerodynamic shape optimization," *AIAA Journal*, vol. 44, no. 7, pp. 1550–1562, 7 2006.

[18] R. Burkard, P. Deuflhard, A. Jameson, J.-L. Lions, and G. Strang, *Computational Mathematics Driven by Industrial Problems*, 1999.

[19] A. Jameson, "Aerodynamic Shape Optimization Using the Adjoint Method," Tech. Rep., 2003.

[20] Y. Yu, Z. Lyu, Z. Xu, and J. R. Martins, "On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization," *Aerospace Science and Technology*, vol. 75, pp. 183–199, 4 2018.

[21] O. Pironneau, "On optimum profiles in Stokes flow," Tech. Rep. 1, 1973.

[22] ——, "On optimum design in fluid mechanics," Tech. Rep. 1, 1974.

[23] A. Jameson, L. Martinelli, and N. A. Pierce, "Optimum Aerodynamic Design Using the Navier-Stokes Equations," Tech. Rep., 1998.

[24] D. Y. Xu, Y. Shen, W. Huang, Z. Y. Guo, H. Zhang, and D. F. Xu, "Parametric Modelling and Variable-Fidelity Bayesian Optimization of Aerodynamics for a Reusable Flight Vehicle," *Fluid Dynamics*, vol. 59, no. 6, pp. 2083–2095, 12 2024.

[25] R. L. Liu, Q. Zhao, X. J. He, X. Y. Yuan, W. T. Wu, and M. Y. Wu, "Airfoil optimization based on multi-objective bayesian," *Journal of Mechanical Science and Technology*, vol. 36, no. 11, pp. 5561–5573, 11 2022.

[26] R. R. Lam, M. Poloczek, P. I. Frazier, and K. E. Willcox, "Advances in bayesian optimization with applications in aerospace engineering," in *AIAA Non-Deterministic Approaches Conference, 2018.* American Institute of Aeronautics and Astronautics Inc, AIAA, 2018.

[27] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Freitas, "Bayesian Optimization in a Billion Dimensions via Random Embeddings," 1 2013. [Online]. Available: http://arxiv.org/abs/1301.1942

[28] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “Recent Advances in Bayesian Optimization,” *ACM Computing Surveys*, vol. 55, no. 13s, 12 2023.

[29] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction Second edition, in progress,” Tech. Rep.

[30] “Black-Box Optimization with Quantum Annealing and Ising Machines.” [Online]. Available: https://amplify.fixstars.com/en/demo/fmqa_0_algebra

[31] Y. Seki, R. Tamura, and S. Tanaka, “Black-box optimization for integer-variable problems using Ising machines and factorization machines,” 9 2022. [Online]. Available: http://arxiv.org/abs/2209.01016

[32] D. Huang, “Physics-Informed Neural Networks for Fluid Mechanics,” Tech. Rep., 2022.

[33] M. Selig, “UIUC Airfoil Data Site,” 1996. [Online]. Available: https://m-selig.ae.illinois.edu/ads/coord_database.html

[34] J. Viquerat and J. Rabault, “sDRL: PBO.” [Online]. Available: https://github.com/jviquerat/pbo

[35] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” 12 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] J. Qiang and C. Mitchell, “A Unified Differential Evolution Algorithm for Global Optimization,” Tech. Rep.

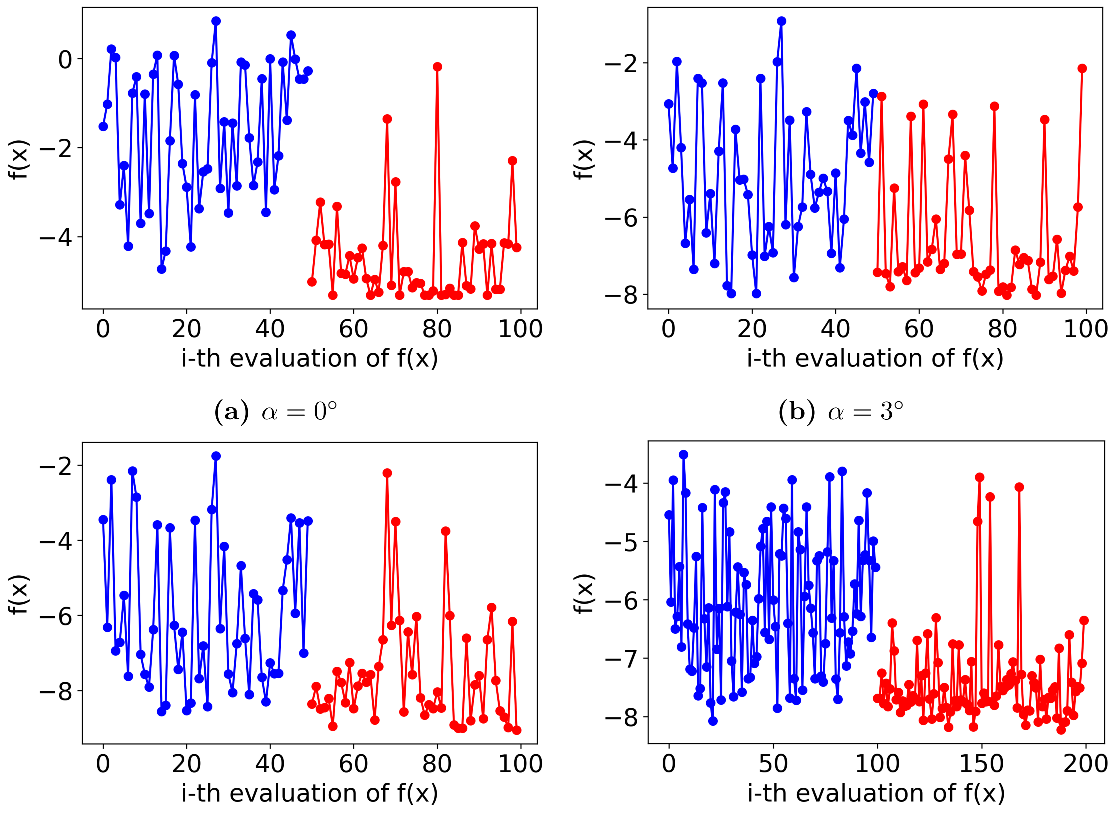
[37] “differential_evolution - SciPy v1.16.1 Manual.” [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

[38] M. Drela, “Pros & Cons of Airfoil Optimization,” *Frontiers of Computational Fluid Dynamics 1998*, 11 1998.

Appendix A

Additional Results Parametric Study

A.1 FMQA Optimization History



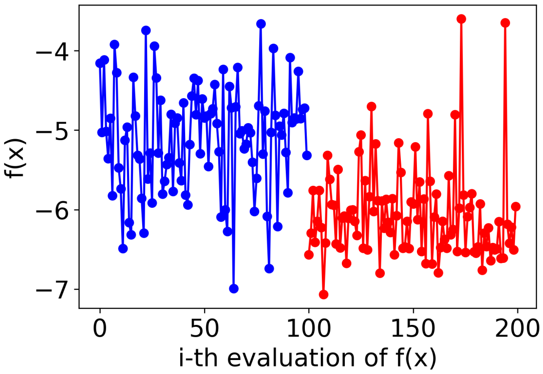
(a) $\alpha = 0°$

(b) $\alpha = 3°$

(c) $\alpha = 6°$

(d) $\alpha = 9°$

Figure A.1: Optimization history of FMQA for the parametric study of the angle of attack with VAE-based parameterization.



(e) $\alpha = 12°$

Figure A.1: Optimization history of FMQA for the parametric study of the angle of attack with VAE-based parameterization.

A.2 Optimized Design Variables

Table A.1: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values ($\alpha = 0.00°$).

	FMQA	**sDRL**
z_0	-0.60	0.36
z_1	1.35	1.40
z_2	0.90	0.73
z_3	-1.50	-1.50
z_4	1.20	1.24
z_5	0.30	-0.84
z_6	-0.38	1.50
z_7	-0.69	-1.21
L/D	5.313	5.339

Table A.2: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values ($\alpha = 3.00°$).

	FMQA	**sDRL**
z_0	-0.98	-0.22
z_1	0.53	1.35
z_2	0.67	0.71
z_3	-1.50	-1.50
z_4	1.35	1.50
z_5	-0.38	-1.10
z_6	-0.90	1.07
z_7	1.20	-0.76
L/D	8.023	8.565

Table A.3: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values ($\alpha = 6.00°$).

	FMQA	**sDRL**
z_0	-0.67	-0.64
z_1	1.35	1.33
z_2	1.20	0.61
z_3	-1.50	-1.48
z_4	0.83	0.50
z_5	-0.08	-1.22
z_6	-0.38	0.75
z_7	-0.67	-0.56
L/D	8.735	8.904

Table A.4: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values ($\alpha = 9.00°$).

	FMQA	**sDRL**
z_0	-0.98	0.24
z_1	0.53	1.37
z_2	0.22	-0.42
z_3	-1.27	-1.40
z_4	0.60	1.00
z_5	-1.12	-0.40
z_6	-0.15	0.19
z_7	-1.27	-0.44
L/D	8.228	8.533

Table A.5: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values ($\alpha = 12.00°$).

	FMQA	**sDRL**
z_0	1.05	0.74
z_1	1.35	1.43
z_2	0.00	1.50
z_3	-1.50	-1.45
z_4	1.28	1.00
z_5	-0.08	-0.95
z_6	-0.30	1.11
z_7	-0.60	0.24
L/D	7.068	7.021

Appendix B

Additional Results Benchmark Cases

B.1 Target Lift Coefficient

B.1.1 PARSEC: Optimized Design Variables

Table B.1: Optimized values of the design parameters in PARSEC parameterization and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
r_{le}	0.009	0.0089	0.0075
x_{up}	0.40	0.37	0.43
z_{up}	0.054	0.048	0.055
$z_{xx,up}$	-0.38	-0.49	-0.52
x_{lo}	0.43	0.38	0.33
z_{lo}	-0.053	-0.044	-0.054
$z_{xx,lo}$	0.77	1.08	0.55
z_{te}	-0.011	-0.020	-0.010
Δz_{te}	0.006	0.0081	0.0082
α_{te} (°)	-6.23	-4.71	-6.70
β_{te} (°)	6.79	11.39	12.66
α (°)	9.00	10.00	9.97
$f(\mathbf{x})$	0.04215	0.05053	0.03449

B.1.2 VAE: Optimized Design Variables

Table B.2: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
z_0	-0.82	1.50	1.15
z_1	-0.75	1.47	0.06
z_2	-0.30	1.01	0.60
z_3	0.22	-1.33	-0.84
z_4	1.28	1.23	0.75
z_5	-0.15	-1.15	-1.05
z_6	-0.52	0.44	1.07
z_7	0.97	-1.49	1.40
α (°)	9.25	10.00	9.11
$f(\mathbf{x})$	0.02243	0.01207	0.01262

B.2 Drela's Optimization Problem

B.2.1 Optimized Design Variables

Table B.3: Optimized values of the design parameters in VAE-based parameterization and the corresponding objective function values.

	FMQA	**sDRL**	**SciPy**
z_0	-1.20	-0.55	0.24
z_1	0.90	1.38	0.92
z_2	1.42	-1.00	1.40
z_3	0.75	0.61	0.67
z_4	0.97	1.31	1.39
z_5	1.35	-0.25	1.34
z_6	1.35	1.36	-1.02
z_7	-0.15	-1.47	-1.49
$f(\mathbf{x})$	0.4442	0.4050	0.4156