修 士 学 位 論 文
Master's Thesis

論文題目

Thesis Title

Deep Learning Techniques for Aerodynamic Wing Shape Optimization

東 北 大 学 大 学 院 工 学 研 究 科

Graduate School of Engineering,

TOHOKU UNIVERSITY

専攻/Department: Aerospace Engineering

学籍番号/ ID No: C1TM9124

氏名/Name: Muhammad Alfiyandy HARIANSYAH

Advising Professor at Tohoku Univ.	Professor Shigeru Obayashi
Research Advisor at Tohoku Univ.	-
Thesis Committee Members Name marked with "○" is the Chief Examiner	○ Prof. Shigeru Obayashi 1 Prof. Tomonaga Okabe 2 Prof. Koji Shimoyama (Kyushu Univ.) 3 Associate Prof. Yuichi Kuya

Author's Profile	
Name Muhammad Alfiyandy HARIANSYAH	Date of Birth June 26, 1998
Nationality Indonesia	

Curriculum Vitae	
Educational Background	
From October 1, 2017 To September 25, 2021	Department of Mechanical and Aerospace Engineering, School of Engineering, Tohoku University (bachelor's program)
From October 1, 2021 To September 25, 2023	Department of Aerospace Engineering, Graduate School of Engineering, Tohoku University (master's program)
From To	
From To	
Work Experience	
From August 10, 2022 To March 31, 2023	Aerodynamics Engineer (part-time), teTra-aviation corp.

TOHOKU UNIVERSITY
Graduate School of Engineering

Deep Learning Techniques for Aerodynamic Wing Shape Optimization
（空力翼形状最適化のためのディープラーニング技術）

A thesis submitted for the Master's degree (Engineering)

Department of Aerospace Engineering

by

Muhammad Alfiyandy HARIANSYAH

July 7, 2023

Deep Learning Techniques for Aerodynamic Wing Shape Optimization

Muhammad Alfiyandy Hariansyah

Abstract

Computational aerodynamic wing shape optimization (AWSO) has gained significant traction in aerospace sectors, emerging as a valuable tool in aircraft design. By integrating advanced numerical simulations and optimization techniques, it offers a powerful approach for enhancing wing performance and efficiency. The process begins with the utilization of computational fluid dynamics (CFD) simulations, which accurately model the airfoil characteristics around different wing configurations. Calculating forces such as lift and drag through fluid flow simulation provides valuable insights into the aerodynamic behavior of different wing shapes. These simulations replace costly wind tunnel tests, serving as ground truth for optimization algorithms to iteratively refine wing geometry, yielding cost-effective and optimized designs.

Optimization techniques like genetic algorithms or gradient-based methods automate the exploration and manipulation of wing shapes to achieve objectives such as fuel efficiency or noise reduction. These numerical algorithms efficiently search for optimal solutions within the design space, minimizing the need for extensive human intervention. Engineers now focus on formulating design problems mathematically, enabling systematic problem-solving. Through iterative process, engineers uncover optimal wing designs with superior aerodynamic performance, resulting in safer, more efficient, and environmentally friendly aircraft.

While computational fluid dynamics (CFD) is more cost-effective than wind tunnel tests, it can still be computationally expensive, especially for high-fidelity simulations. In the context of AWSO, conducting numerous CFD simulations for design exploration using global optimizers like genetic algorithms can be impractical due to computational constraints. This is where surrogate models play a crucial role. Surrogate models are simplified mathematical representations of expensive evaluations (e.g., CFD) that approximate the relationship between input variables (e.g., wing shape parameters) and corresponding outputs (e.g., lift and drag) using a limited number of simulations. By leveraging the real-time prediction capability of surrogate models, they provide a computationally efficient means to explore the design space when coupled with genetic algorithms, forming surrogate-based optimization (SBO). In essence, the optimization technique explores the design space approximated by the surrogate models, enabling efficient AWSO.

Since the data points obtained from the time-consuming CFD simulations are limited, their quality holds significant importance in the representative capability of the surrogate models. In the context of AWSO, it becomes crucial to develop methods that drive the exploration of the surrogate models toward the optimal solutions. This involves strategically selecting data points that are likely to yield valuable insights and drive the search for optimal designs. By intelligently sampling the design space, focusing on critical regions, the surrogate model can be effectively guided to explore and exploit the design space in search of optimal wing shapes.

In the design of experiment step, Latin hypercube sampling (LHS) is commonly used in conjunction with local perturbation techniques like free form deformation (FFD) to offer high geometric flexibility. FFD enables localized modifications to the wing shape, enabling refinement of specific regions of interest. Since LHS ensures a well-distributed parameter sampling, when combined with FFD for wing geometry creation, the resulting shapes can exhibit curviness or abnormal features that compromise aerodynamic performance. These irregularities pose challenges in training surrogate models, particularly in high-dimensional cases, introducing complexities that impact model accuracy. Thus, it is also crucial to develop methods specifically designed to address the challenges associated with curvy wing shapes without decreasing the flexibility, since one can just replace FFD with simpler parameterization that can ensure smoothness of the wing shapes.

Deep learning has gained significant attention as a promising tool across diverse fields. This thesis focuses on utilizing deep learning techniques, including multilayer perceptron (MLP), generative adversarial networks (GANs), and convolutional neural networks (CNNs), to tackle challenges in AWSO. In this research, an MLP is trained as a surrogate model to expedite the prediction of aerodynamic performance for optimization purposes. Additionally, a deep convolutional GAN (DCGAN) is utilized to generate synthetic wing shapes, serving as an alternative to LHS+FFD for initial sampling and enhancing the quality of initial samples. Furthermore, a CNN-based geometric filter is trained to quickly identify and eliminate geometric abnormalities during the infill sampling phase of SBO. By leveraging deep learning, this thesis aims to improve surrogate-based AWSO.

This study evaluates the effectiveness of the aforementioned techniques by defining three optimization methods. The first method employs conventional SBO with LHS for initial sampling. The second method utilizes DCGAN-based initial sampling with the SBO framework. The third method combines DCGAN-based sampling and CNN-based geometric filtering in SBO. These methods are compared based on their performance in solving three AWSO problems of different complexities. The first two problems focus on the transonic airfoil shape optimization with 20 design variables (FFD points), starting from the RAE2822 airfoil. The first problem aims to minimize drag, while the second problem seeks to simultaneously minimize drag and maximize lift, while considering geometric constraints. The third problem addresses a lift-constrained drag minimization of the Common Research Model (CRM) wing with 192 FFD points, subject to specific geometric constraints.

The results indicate that incorporating DCGAN-based sampling and CNN-based geometric filtering significantly accelerates the convergence of optimization processes. These techniques lead to the discovery of more optimal solutions compared to conventional SBO with LHS. The improved performance is attributed to the smoother initial designs generated by DCGAN-based sampling, resulting in better aerodynamic performance and improved surrogate models compared to LHS. The CNN-based filter effectively identifies and eliminates abnormal designs, reducing the complexity of the problem by shrinking the design space. Consequently, the combined utilization of DCGAN-based sampling and CNN-based geometric filtering enhances the optimization process.

The first problem yielded an optimal design with a 14% drag improvement, characterized by a more rounded leading edge, promoting smoother air flow over the airfoil surface. In the second problem, two notable designs were obtained: one with a 16% drag improvement and another with a 58% lift improvement. These designs feature high camber for increased lift and low camber for reduced drag. In the third problem, a design with a modest 0.5 drag count improvement was discovered, characterized by a bifurcated shock region resulting from the oscillation of curvature on the suction side. Furthermore, a comparison with results from other researchers who employed gradient-based methods is discussed. Future work should involve applying these techniques to different use cases or design points to explore their versatility and effectiveness in various optimization scenarios.

Contents

Chapter 1 Introduction 1

1.1 Research background 1

1.1.1 Aerodynamic wing shape optimization 1

1.1.2 Surrogate-based optimization 3

1.1.3 Curse of dimensionality 4

1.1.4 Deep learning techniques 6

1.2 Research objectives 6

1.3 Thesis structure 7

Chapter 2 Design framework 9

2.1 Entire design framework for AWSO 10

2.2 Geometric parameterization 12

2.3 Design of experiment 13

2.4 Numerical simulation 14

2.4.1 Reynolds-Averaged Navier-Stokes (RANS) equations 15

2.4.2 Mesh deformation 15

2.4.3 Grid convergence study 16

2.5 Surrogate modeling 18

2.6 Sub-optimization via a genetic algorithm 19

2.6.1 Genetic algorithm: NSGA-II 19

2.6.2 Combining NSGA-II with a surrogate model 20

2.7 Infilling process 22

2.8 Believer sub-iteration 23

2.9 Stopping criterion 23

2.10 Summary 24

Chapter 3 Deep learning techniques 25

3.1 MLP-based surrogate modeling 25

3.1.1 Artificial neural network 25
3.1.2 Multilayer perceptron 27
3.1.3 MLP for aerodynamic performance approximator 28
3.2 DCGAN-based initial sampling 30
3.2.1 Generative adversarial networks 30
3.2.2 Deep convolutional generative adversarial network 31
3.2.3 DCGAN for producing synthetic airfoil samples 32
3.2.4 Inverse FFD algorithm 35
3.2.5 Sampling flow 36
3.3 CNN-based geometric filtering 37
3.3.1 Convolutional neural network 37
3.3.2 Kernel, padding, and stride in convolution operations 38
3.3.3 CNN for geometric abnormality detection 39
3.4 Summary 42
Chapter 4 Aerodynamic wing shape optimization 43
4.1 Experimental design 43
4.1.1 Optimization problems and methods 43
4.1.2 Settings for MLP-based surrogate modeling 44
4.1.3 Settings for DCGAN-based initial sampling 45
4.1.4 Settings for CNN-based geometric filtering 46
4.1.5 Settings for NSGA-II 47
4.2 DCGAN generative model 48
4.3 Geometric filter score constraint determination 49
4.4 Single-objective airfoil optimization 50
4.4.1 Problem formulation 50
4.4.2 Optimization history 53
4.4.3 CFD visualization 56
4.5 Multi-objective airfoil optimization 59

4.5.1 Problem formulation 59
4.5.2 Optimization history 60
4.5.3 CFD visualization 62
4.6 Single-objective wing optimization 68
4.6.1 Problem formulation 68
4.6.2 Optimization history 70
4.6.3 CFD visualization 73
4.6.4 Comparison with other researchers' results 79
4.7 Surrogate model's accuracy 80
4.8 Computational time 82
4.9 Limitations 84
4.10 Work in progress (WIP) 87
4.11 Summary 91
Chapter 5 Conclusion and future work 93
Acknowledgment 97
References 99
Appendix 107

Chapter 1 Introduction

1.1 Research background

1.1.1 Aerodynamic wing shape optimization

Aerodynamic wing shape optimization (AWSO) is a vital field of study in aerospace engineering, focused on enhancing the performance and efficiency of aircraft wings. The aerodynamic characteristics of an aircraft's wings, whether fixed with respect to the fuselage or capable of various motions like helicopters, are significantly influenced by the shape of the wing section [1]. Engineers and researchers have long sought ways to optimize wing shapes to improve aircraft performance, reduce fuel consumption, and enhance safety.

The fundamental principle behind AWSO is to achieve the most favorable balance between lift and drag forces acting on an aircraft during flight. Lift is the upward force generated by the wings, allowing an aircraft to overcome gravity and stay aloft. Drag, on the other hand, is the resistance encountered by an aircraft as it moves through the air, which hampers its forward motion and increases fuel consumption, see Figure 1.1. By refining the wing shape, engineers can aim to minimize drag or maximize lift, depending on the pre-defined objectives.

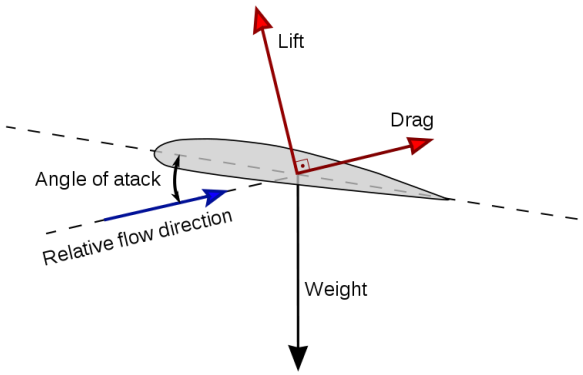


Figure 1.1 Free body diagram of a wing section.

Aerodynamic shape optimization (ASO) has traditionally relied on well-established principles and empirical data. The origins of ASO can be traced back to the 16th century when Newton [2] utilized the calculus of variations to minimize fluid drag on a body of revolution by manipulating its shape. While optimization theory made significant progress in subsequent

years, it wasn't until the 1960s that advancements in both theory and computer hardware made numerical optimization a practical tool for everyday applications [3]. For example, Hicks et al. [4] first tackled airfoil design optimization problems. Hicks and Henne [5] then used a three-dimensional solver to optimize a wing with respect to 11 design variables representing both airfoil shape and the twist distribution.

The recent advancements in computational tools, numerical simulation techniques, and optimization algorithms have truly revolutionized the field of AWSO. These breakthroughs have paved the way for engineers to explore an extensive design space and rapidly iterate through numerous wing shapes. As a result, they can identify optimal configurations that offer improved aerodynamic performance. The utilization of computational tools, such as high-fidelity numerical simulations, has played a pivotal role in enabling this progress. Engineers can now leverage these simulations to obtain detailed information about the flow characteristics and performance of different wing shapes. This allows for more informed decision-making throughout the optimization process.

Moreover, the development of advanced optimization algorithms has further accelerated AWSO. These algorithms, ranging from gradient-based methods to genetic algorithms and surrogate modeling, facilitate the exploration of design variables and help identify optimal wing configurations. With their assistance, engineers can efficiently navigate the vast design space and converge towards solutions that maximize the desired performance objectives. The integration of these computational tools, numerical simulations, and optimization algorithms has created a powerful framework for AWSO. This newfound capability has opened up exciting possibilities for enhancing aircraft performance and improving overall efficiency.

The process of computational AWSO typically begins with defining the design objectives and constraints, such as minimizing drag by meeting specific lift or stability requirements. With these objectives in mind, engineers utilize numerical simulations such as computational fluid dynamics (CFD) to evaluate aerodynamic performance of different wing shapes. CFD plays a key role in AWSO, by solving the governing equations of fluid flow around the wing and providing detailed information on the distribution of pressure, velocity, and other flow parameters. CFD results allow engineers to identify areas of high drag, regions of flow separation, or suboptimal pressure distributions to be improved through design modifications.

Optimization algorithms come into play to search for the best wing shape within the defined design space. These algorithms explore various design variables, such as wing planform,

aspect ratio, sweep, and airfoil profiles, seeking the combination that optimizes the objectives. The optimization process typically involves iterations where the algorithm evaluates different designs, adjusts based on performance metrics, and converges towards an optimal solution.

In the computational design optimization framework, engineers begin by creating an initial design and formulating the design problem by defining design variables, objectives, and constraints. This defined problem is then solved using computer-based optimization algorithms. The resulting optimal solutions are evaluated by the engineers to determine their quality. If the solutions are not satisfactory, the engineers reformulate the design problem or change the initial design and repeat the process until either the computational resources are exhausted, or the desired solutions are obtained. Figure 1.2 illustrates this framework.

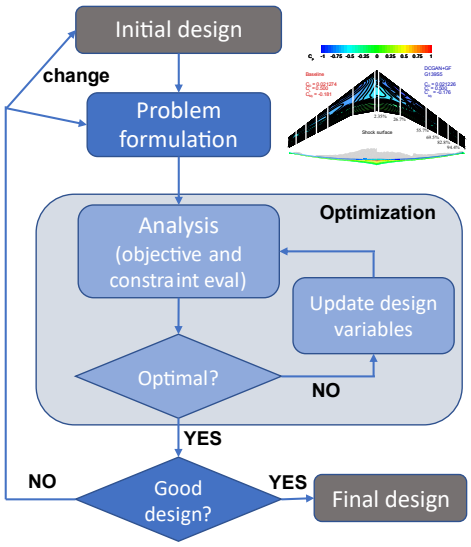


Figure 1.2 Computational design optimization framework.

1.1.2 Surrogate-based optimization

There are two broad categories of optimization algorithms: gradient-free [6], [7] and gradient-based [8]–[10] methods. Gradient-free methods, such as genetic algorithms, usually need to evaluate thousands of different shapes when performing wing optimization, and thus the overall computational cost is prohibitive when using high-fidelity CFD. A common way to handle this is to use them together with surrogate models (SMs). SMs are simplified mathematical representations of expensive evaluations (e.g., CFD) that approximate the relationship between inputs (e.g., wing shape parameters) and corresponding outputs (e.g., lift and drag) using a limited number of simulations. By leveraging the real-time prediction

capability of SMs, they provide a computationally efficient means to explore the design space when coupled with genetic algorithms, forming surrogate-based optimization (SBO) [11].

In contrast, gradient-based methods offer the advantage of faster convergence towards optimal designs. By employing the adjoint method [12], [13], the computational expense of calculating gradients is significantly reduced compared to finite difference methods. Typically, a gradient-based approach with an efficient adjoint solver achieves optimal solutions through a relatively small number of aerodynamic analyses, ranging from dozens to hundreds. Nevertheless, it is worth noting that they are susceptible to becoming stuck in local optimal solutions, although this argument remains open to debate in the context of ASO, read a paper titled Robust aerodynamic shape optimization—From a circle to an airfoil [14].

However, as reviewed by Yondo et al. [15], surrogate-based design optimization is still a popular approach due to the easy implementation, robust, and feature a global search that increases the likelihood of finding the global optimum. It has been applied in airfoil design [16], wing design [17], and aero-structural design optimization [18]. Moreover, this approach does not require gradient information, the computation of which is either time-consuming or requires an adjoint solver. Thus, this thesis focuses on using surrogate-based optimization for aerodynamic wing shape optimization (AWSO).

1.1.3 Curse of dimensionality

The efficiency of surrogate-based optimization (SBO) falls short, particularly in high-dimensional design problems, where surrogate models (SMs) face the curse of dimensionality. This curse implies that the required training data for SMs grows exponentially as the number of dimensions increases. Zhang et al. [19] applied an SBO method using a Kriging model [20] to a wing shape optimization with 39 design variables. They highlighted the challenge of constructing accurate SMs in high-dimensional problems. This aligns with the findings of Diaz-Manriquez et al. [21], who suggested that the Kriging model is suitable for low-dimensional problems but suffers from accuracy degradation as dimensionality increases.

In the context of aerodynamic wing shape optimization (AWSO), it is especially effective for large numbers of shape variables, since small local changes in wing shape have a large effect on performance. Therefore, it is important to develop methods to deal with the curse of dimensionality when utilizing SBO for AWSO. Previous efforts to improve the SBO efficiency mainly emphasize two parts: more accurate surrogate models or smaller design

spaces. For example, the gradient-enhanced Kriging (GEK) [22], [23] uses the gradient information to improve the model accuracy in airfoil design optimization. Another is the active subspace method (ASM), proposed by Li et al. [24] to reduce the high-dimensional inputs.

The accuracy of SMs and the overall efficiency of SBO are heavily influenced by the training data. In SBO, training data consist of initial samples and infill samples. The initial samples are generated using a sampling method during the design of experiment (DoE) process. Unfortunately, existing sampling methods like the Latin hypercube sampling (LHS) [25] do not guarantee realistic wing samples in the context of AWSO, since these methods are designed to generate evenly distributed samples without considering feasibility. As a result, the initial training data often include irregular wing shapes that are impractical for real-world applications, leading to inaccurate SMs for predicting aerodynamic functions of designs with realistic shapes.

In SBO, infill samples are added by solving suboptimization problems, where SMs are used to evaluate aerodynamic functions. However, due to the inaccuracies of SMs, the resulting shapes of new infill samples cannot be guaranteed to be realistic. In fact, a significant number of infill samples exhibit abnormal aerodynamic shapes that do not provide meaningful information to improve the SMs. Consequentially, refining SMs with these abnormal infill samples further diminishes their accuracy, creating a vicious cycle, see Figure 1.3. Thus, it is crucial to avoid abnormal shapes within the training data (both in the initial and infill samples).

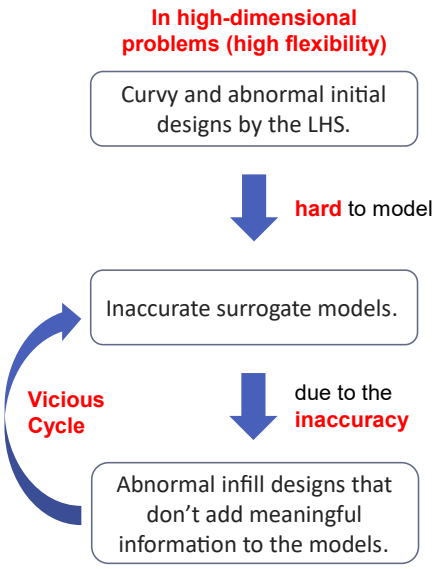


Figure 1.3 Vicious cycle caused by abnormalities in training data.

1.1.4 Deep learning techniques

Deep learning has emerged as a powerful tool with applications spanning various fields. This thesis aims to develop deep learning techniques specifically tailored for surrogate-based aerodynamic wing shape optimization. We identify two key challenges in this domain: 1) the development of surrogate models to replace computationally expensive CFD simulations, and 2) the generation of high-quality and realistic wing samples to enhance the accuracy of surrogate models and overall optimization efficiency. To address these challenges, this study utilizes specific deep learning techniques, namely the multilayer perceptron (MLP), generative adversarial networks (GANs), and convolutional neural networks (CNNs).

In this study, the MLP is employed as a surrogate model to expedite the prediction of aerodynamic performance for optimization tasks. Additionally, a deep convolutional GAN is utilized to generate initial wing samples that are more realistic (smoother) compared to the conventional Latin hypercube sampling (LHS). Furthermore, a CNN-based geometric filter is trained to quickly identify and eliminate geometric abnormalities during the infill sampling phase of SBO. By leveraging these techniques, this study aims to enhance the conventional surrogate-based aerodynamic wing shape optimization, leading to improved results.

1.2 Research objectives

The main objective of this study is to improve the efficiency of the conventional surrogate-based optimization (SBO) method when used in aerodynamic wing shape optimization (AWSO) by leveraging deep learning techniques. Just like any other optimization, it is desirable to find more optimal solutions with as few numbers of evaluations as possible (especially if the evaluation is expensive, like CFD). The success of this research lies in demonstrating that the proposed methods can achieve the same, or even better, optimal solutions compared to conventional SBO approaches with fewer number of CFD simulations (faster convergence).

Other objectives are: 1) to develop deep learning techniques, 2) integrate them into the conventional SBO method, and 3) demonstrate their efficacy by solving three AWSO problems of varying complexities. The first two problems involve low-dimensional cases, each with 20 design variables. The third problem presents a high-dimensional case with 193 design variables. By addressing this high-dimensional case, the study aims to illustrate how the proposed deep learning techniques effectively combat the curse of dimensionality, which is a significant challenge in SBO with a large number of variables.

1.3 Thesis structure

This thesis comprises five chapters, each addressing different aspects of the research. Chapter 1 introduces the background of aerodynamic wing shape optimization (AWSO), surrogate-based optimization (SBO), the challenge posed by the curse of dimensionality, and an overview of deep learning techniques. The objectives of the study are outlined, and the structure of the thesis is presented.

Chapter 2 delves into the entire design framework, offering detailed breakdown of each step involved. It covers aspects such as the wing shape parameterization, the sampling techniques in the design of experiment (DoE) step, the brief theory behind the numerical CFD solver (ground-truth evaluation), surrogate modeling, sub-optimization on the surrogate model using genetic algorithms, the infilling process, believer sub-iteration, and the optimization stopping criterion.

Chapter 3 expands upon the integration of deep learning techniques within the above design framework. It provides a detailed introduction to each deep learning technique, namely multilayer perceptron (MLP), generative adversarial networks (GANs), and convolutional neural networks (CNNs). The chapter then proceeds to explain the application of these techniques in a step-by-step manner, specifically tailored to address the challenges of AWSO.

Chapter 4 provides an overview of the experimental design to demonstrate the efficacy of the proposed deep learning techniques. This chapter focuses on comparing the performance of the proposed methods with the conventional SBO method across three AWSO problems: 1) single-objective transonic airfoil shape optimization, 2) multi-objective transonic airfoil shape optimization, and 3) single-objective optimization of the Common Research Model (CRM) wing. For each problem, the specific problem formulation is outlined, and the optimization performance of each method is compared. The accuracy of the surrogate models in each method is also evaluated and compared. Finally, the optimal results obtained from the different methods are discussed in terms of their physical perspectives.

Chapter 5 concludes the study with a summary of the findings, key arguments, and concluding remarks. The chapter provides a concise explanation to support the arguments made throughout the research. Additionally, it offers recommendations for future work, identifying potential areas for further investigation and improvement based on the outcomes of the study.

Chapter 2 Design framework

The design framework heavily adopts a mathematical optimization framework, which is a technique to find the best possible solutions from a set of available choices. Mathematical optimization finds applications in various domains, including engineering, finance, logistics, etc. In the engineering domain, it is widely known as engineering design optimization: the process of finding design candidates with the best performance subject to some objective function(s) and constraint(s). The search process is done on the design space that has been pre-defined beforehand. Unlike conventional design process that relies on designer experience (draw and build), engineering design now shifts to approaches that are less dependent on the designer's knowledge, experience, and intuition. The advancement of computing power has also driven the design process to use numerical simulations to evaluate the design, e.g., computational fluid dynamics (CFD). The marriage between optimization algorithms and numerical simulations gives rise to the field of engineering design optimization.

With the advent of optimization algorithms, the role of the designer has shifted to formulating the optimization problem rather than manually finding the optimal solution. This formulation process involves defining the design variables $\boldsymbol{x}$, objective function(s) $\boldsymbol{f}(\boldsymbol{x})$, inequality constraint(s) $\boldsymbol{g}(\boldsymbol{x})$, and equality constraint(s) $\boldsymbol{h}(\boldsymbol{x})$. The careful and appropriate formulation of optimization problem is crucial as the quality of the resulting optimal designs heavily relies on it. A general optimization problem formulation can be written as follows.

$$\text{Minimize} \quad f_m(\boldsymbol{x}), \qquad m = 1,2,\dots,\mathrm{M}; \tag{2.1}$$

$$\text{subject to} \quad g_j(\boldsymbol{x}) \leq 0, \qquad j = 1,2,\dots,\mathrm{J}; \tag{2.2}$$

$$h_k(\boldsymbol{x}) = 0, \qquad k = 1,2,\dots,\mathrm{K}; \tag{2.3}$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \qquad i = 1,2,\dots..\mathrm{n}; \tag{2.4}$$

The essence of an optimization problem can be summarized in the following manner: "Find the values of design variables $\boldsymbol{x}$ that minimize the objective function(s) $\boldsymbol{f}$ while satisfying the constraints $\boldsymbol{g}$ and $\boldsymbol{h}$". When there is a single objective function to minimize, it is referred to as a single-objective optimization problem. However, when multiple objective functions are involved, it is known as a multi-objective optimization problem.

2.1 Entire design framework for AWSO

Figure 2.1 illustrates the sequential steps involved in the surrogate-based optimization (SBO) approach applied to aerodynamic wing shape optimization (AWSO). Each step is elaborated as follows:

1. **Geometric parameterization:** The initial step involves the parameterization of the wing geometry using a set of design variables, which determine its shape. In this study, the wing parameterization method employed is the free form deformation (FFD) [26].
2. **Design of experiment (DoE):** The design space is sampled initially to create an initial database for training the surrogate model. Two methods are utilized in this study: Latin hypercube sampling (LHS) [25] and a deep convolutional generative adversarial network (DCGAN)-based sampling. LHS is a conventional method used to directly generate perturbations for FFD points, while the DCGAN-based sampling generates perturbations for FFD points using Algorithm 3 (refer to Chapter 3, Section 3.2.3).
3. **Evaluation of objective functions and constraints:** The objective functions and constraints are computed using true evaluations, involving computational fluid dynamics (CFD) for the aerodynamic functions and analytical evaluations for geometric constraints. These evaluations provide the ground-truth data for the database.
4. **Compilation of solutions in the design database:** The solutions obtained thus far are compiled and stored in a design database. This contains the mapping between design variables and expensive aerodynamic functions to approximate.
5. **Surrogate modeling:** To approximate the computationally expensive aerodynamic functions such as drag (C_D), lift (C_L), moment (C_M) coefficients, a surrogate model based on a multilayer perceptron (MLP) is constructed, utilizing backpropagation technique with the assistance of the Adam optimizer (refer to Chapter 3, Section 3.1.3).
6. **Sub-optimization via genetic algorithms:** A sub-optimization process is performed using a variant of genetic algorithms, NSGA-II, where the aerodynamic functions are evaluated by utilizing the MLP-based surrogate model. If we want to use a convolutional neural network (CNN)-based geometric filter, it is integrated within this sub-optimization phase.
7. **Believer sub-iteration:** If a believer sub-iteration is planned and the believer maximum iteration is not reached, the process proceeds to Step 8. Otherwise, it proceeds to Step 9.

8. **Data appendment:** The predicted candidate data is appended to the surrogate model. Subsequently, a separate model known as the "believer" model is trained. This model earns its name due to the fact that the data is appended using predicted values from the surrogate model. Steps 5-7 are then repeated. This sub-iterative process allows us to efficiently acquire multiple infilling samples that can be computed in parallel.
9. **Stopping criterion:** If there is still available computational budget, proceed to Step 10. Otherwise, move on to Step 11.
10. **Infill sampling:** The newly generated design points (infill samples) need to be evaluated using true evaluations before appended to the database. Steps 3-9 are then repeated.
11. **Achieving convergence or optimized solution(s):** The best solution(s) obtained thus far, which represent the optimized designs, are determined. In the case of single-objective optimization, a single solution is obtained, while for multi-objective optimization, several non-dominated solutions are obtained.

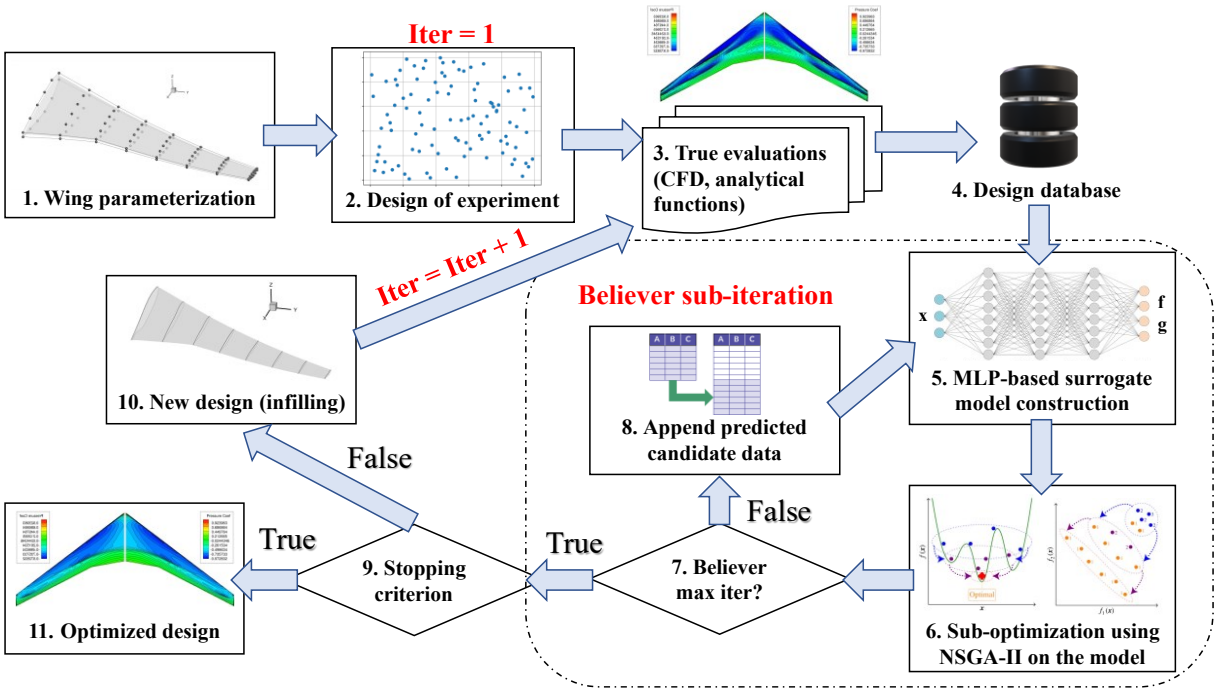


Figure 2.1 Surrogate-based optimization method for AWSO.

2.2 Geometric parameterization

Geometric parameterization involves the characterization and representation of the geometry or shape of a design using a set of design variables. The goal is to capture the essential features of the design that directly influence its performance. In the context of design optimization, geometric parameterization serves as a bridge between design space, where the variables are defined and the physical or virtual representation of the design. It allows systematic exploration and manipulation of the design space to find optimal solutions.

Various methods can be used, depending on the complexity of the design and the specific requirements of the optimization problem. These methods range from simple parameterization techniques like parametric equations and geometric primitives to more advanced approaches such as free-form deformation, Bezier, and spline curves.

In this study, we use free-form deformation (FFD) [26] implemented in pyGeo [27] to parameterize our wing geometry. As the name suggests, the FFD parameterizes the geometry deformation rather than the geometry itself. Engineers should come up with a baseline geometry, which is then embedded in the FFD volume. By perturbing the FFD volume, we have great control of the baseline geometry as the embedded object. For example, one of the optimization problems in this study specifies 192 FFD points to embed the Common Research Model (CRM), as shown in Figure 2.2. The 192 FFD points are distributed in 8 spanwise sections, with 24 points in each section. These points are allowed to move only in z-direction.

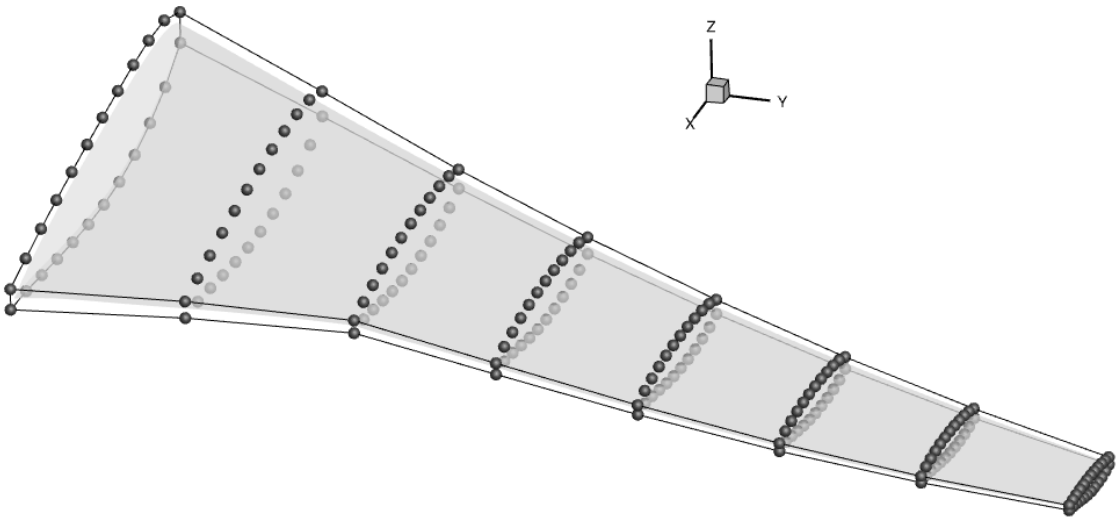


Figure 2.2 FFD control points embedding the baseline geometry.

The changes in z-direction Δz is treated as the design variable. Consequently, the optimizer has control over Δz with some pre-defined boundary $[-Pt, Pt]$, where P is the desired percentage and t is the local thickness of the FFD volume. Figure 2.3 shows the FFD boundaries of the third spanwise section from the wing root. This definition alone can impose great manipulation to the baseline geometry. Twist variations, for example, are possible by allowing the leading edge (LE) to vary and fixing the trailing edge (TE). By setting $\Delta z_{TE,upper} = -\Delta z_{TE,lower}$, fixed TE condition can be imposed. To produce practical wings, geometric constraints can also be defined. For example, the FFD volume can be analytically computed using the convex hull method [28]; the smallest convex set that contains the shape. The thickness constraints can be met by setting the parameter P in the design boundary so that it will not produce unrealistically thin wing sections (in this study, $P = 0.35$).

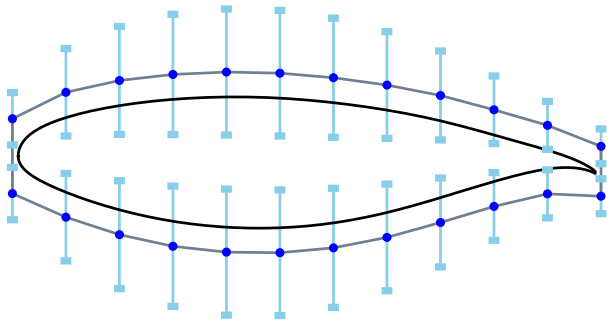


Figure 2.3 FFD boundaries of the third spanwise section from the wing root.

2.3 Design of experiment

The design of experiment (DoE) step is a critical component in surrogate-based optimization (SBO) methods, aimed at efficiently exploring the design space and creating an initial dataset to construct surrogate models. One commonly used approach is Latin hypercube sampling (LHS) [25]. LHS is a systematic sampling technique that divides the design space into equally sized intervals along each design variable. The samples are then randomly distributed within each interval, ensuring representative coverage of the entire design space, see Figure 2.4. In this study, LHS is used to represent the conventional technique in the DoE step, which will be compared with the proposed DCGAN-based method, explained in Chapter 3. If analytical constraints are present, the LHS can also be modified to accommodate for them.

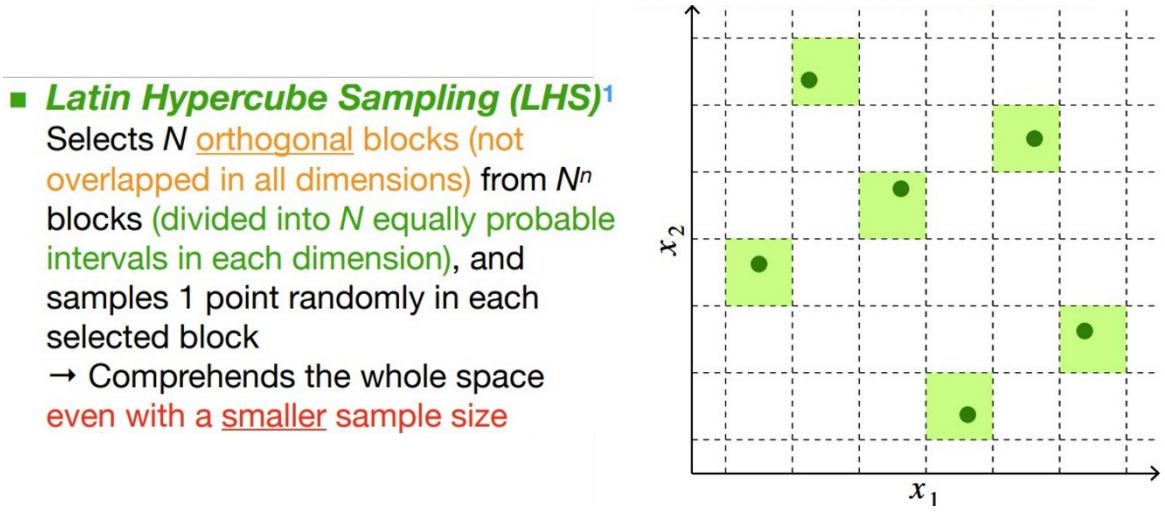


Figure 2.4 Latin hypercube sampling.

2.4 Numerical simulation

In general, the compressible Navier-Stokes equations can be expressed as,

$$\frac{\partial U}{\partial t} + \nabla.\vec{F}^c - \nabla.\vec{F}^v = Q \tag{2.5}$$

where U represents the vector of state variables, $\vec{F}^c(U)$ are the convective fluxes, $\vec{F}^v(U)$ are the viscous fluxes and $Q(U)$ is a generic source term. It is natural to express U in terms of conservative variables $U = (\rho, \rho v_1, \rho v_2, \rho v_3, \rho E)^{\mathrm{T}}$, where ρ is the density of the fluid, E is the total energy per unit mass, and $\vec{v} = (v_1, v_2, v_3) \in \mathbb{R}^3$ is the flow velocity in Cartesian coordinate system. The corresponding flux vectors are given as follows,

$$\vec{F}_i^c = \begin{Bmatrix} \rho v_i \\ \rho v_i v_1 + p\delta_{i1} \\ \rho v_i v_2 + p\delta_{i2} \\ \rho v_i v_3 + p\delta_{i3} \\ \rho v_i H \end{Bmatrix}, \overrightarrow{F_i^v} = \begin{Bmatrix} \cdot \\ \tau_{i1} \\ \tau_{i2} \\ \tau_{i3} \\ v_j \tau_{ij} + \kappa \partial_i T \end{Bmatrix}, Q = \begin{Bmatrix} 0 \\ \rho f_{e1} \\ \rho f_{e2} \\ \rho f_{e3} \\ W_f + q_H \end{Bmatrix}, \qquad i = 1,2,3 \tag{2.6}$$

where p is the static pressure or isotropic part of the stress tensor, δ_{ij} is the Kronecker delta function, H is the total enthalpy or stagnation enthalpy, κ is the thermal conductivity coefficient, T is the temperature, $\vec{f_e} = (f_{e1}, f_{e2}, f_{e3})^{\mathrm{T}}$ is the external force, W_f is work due to the external forces, q_H is the heat source, τ_{ij} is the non-isotropic part of the stress tensor, also known as viscous stress tensor which exists due entirely to the motion of the fluid. For a Newtonian-fluid, the viscous stress tensor is expressed as $\tau_{ij} = \mu\left(\partial_j v_i + \partial_i v_j - \frac{2}{3}\delta_{ij}\nabla.\vec{v}\right)$, where μ is the dynamic viscosity of the fluid.

2.4.1 Reynolds-Averaged Navier-Stokes (RANS) equations

In this study, the numerical simulation solves the compressible Navier-Stokes equations using Reynold-averaging approach, better known as the Reynolds-Averaged Navier-Stokes (RANS). The approach comes from the idea of expressing all quantities as the sum of its mean and fluctuating parts. For an instantaneous quantity $\phi(\boldsymbol{x}, t)$, we have,

$$\phi(\boldsymbol{x}, t) = \bar{\phi}(\boldsymbol{x}) + \phi'(\boldsymbol{x}, t) \tag{2.7}$$

where $\bar{\phi}(\boldsymbol{x})$ is the mean value and $\phi'(\boldsymbol{x}, t)$ is the fluctuating part. Commonly, the mean value is computed using time-averaging technique, in the form of the following equation,

$$\bar{\phi} = \frac{1}{T}\int_{t}^{t+T} \phi(\boldsymbol{x}, t)dt\,; \quad T_1 \ll T \ll T_2 \tag{2.8}$$

where T is assumed very large relative to the maximum period of the turbulent fluctuations, T_1, while it is much smaller relative to the slowest unsteadiness present in the flow, T_2. More details about RANS can be found in [29].

In this study, a finite-volume CFD solver called ADflow [30] is used. ADflow solves RANS equations with the Spalart-Allmaras [31] turbulence model iterated with the diagonally dominant alternating-direction implicit scheme (DDADI). For the startup, an approximate Newton-Krylov (ANK) solver [32] is used, combined with the Newton-Krylov solver for the final stages of convergence. Refer to [30] for more details about the theory behind the solver.

2.4.2 Mesh deformation

When the control points in the free-form deformation (FFD) method are perturbed, the baseline geometry undergoes modifications. To perform numerical simulation of the resulting modified geometry efficiently, it is necessary to deform the baseline volume mesh instead of remeshing for each modified geometry. Mesh deformation techniques are crucial in enabling the mesh to adapt and conform to the evolving object geometry during the optimization process. It is also essential to maintain high-quality mesh throughout the deformation to ensure robust numerical simulations. In the context of this process, an efficient analytic inverse-distance method implemented in IDWarp [33] is utilized for mesh deformation. A notable advantage of this approach is that it does not require volume mesh connectivity information, making it applicable to both structured and unstructured mesh types. Refer to [33] for more details.

2.4.3 Grid convergence study

A grid convergence study (GCS) is a systematic analysis performed to assess the sensitivity of numerical results to the resolution of the computational grid. It is an important step in validating and verifying CFD simulations. In a GCS, the computational domain is discretized into grids of varying levels of refinement. The simulation is then run on each grid, and the results are compared to evaluate the convergence behavior. The primary objective of a GCS is to determine the grid resolution at which the numerical simulation becomes independent of further grid refinement. This resolution is referred to as the grid-independent solution, signifying that the solution has reached an accuracy that is sufficient for the desired application.

This study focuses on optimizing two different geometries: an airfoil and a wing. The initial baseline for the airfoil optimization problem is the RAE2822 airfoil, while for the wing optimization problem, the Common Research Model (CRM) wing [34] serves as the baseline. To assess the sensitivity of the numerical results to grid resolution, a GCS is conducted for both geometries. The grid variations, namely L1, L2, and L3, are depicted in Figures 2.5 and 2.6, representing the different grids used for the RAE2822 airfoil and the CRM wing, respectively.

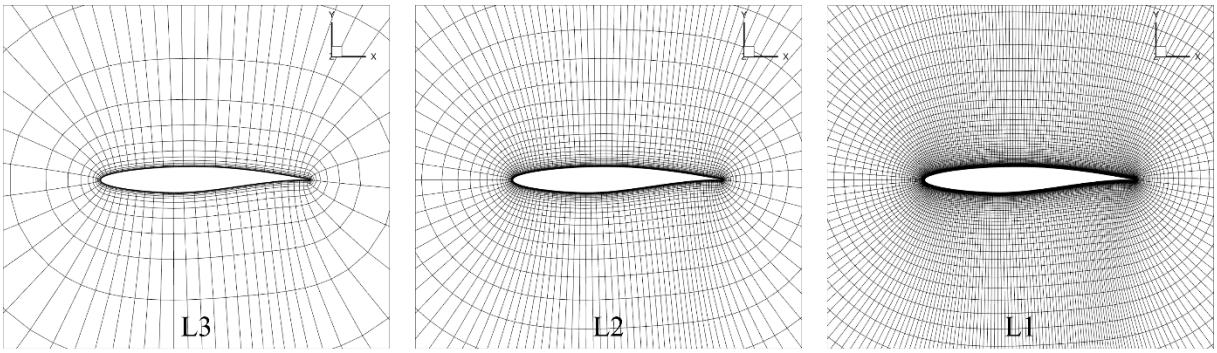


Figure 2.5 Grid variations for the RAE2822 airfoil

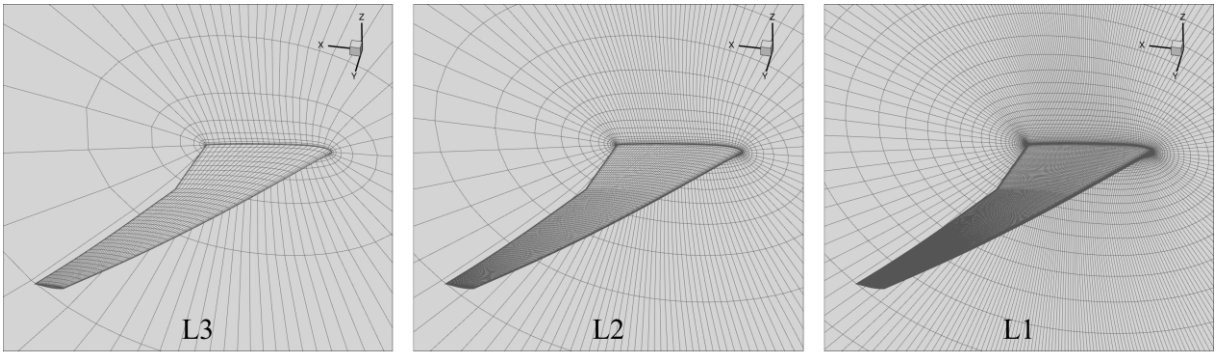


Figure 2.6 Grid variations for the CRM wing

Table 2.1 summarizes the numerical results obtained from the GCS conducted for both the airfoil and wing geometries. In the airfoil case, the flow condition involves a viscous flow with $Mach = 0.73, Re = 7.04E6$, and $AoA = 2^{o}$. For the wing case, the flow condition is also a viscous flow with $Mach = 0.85, Re = 5.0E6$, and $C_L = 0.5$. In the airfoil optimization, the aerodynamic functions of interest are the drag and lift coefficients, C_D and C_L, respectively. Conversely, for the wing optimization, the focus is on the drag coefficient C_D and the moment coefficient C_M, since the fixed lift coefficient constraint is not approximated by the surrogate model, as explained in Chapter 4, Section 4.6.1.

Figures 2.7 and 2.8 depict the variation of these aerodynamic functions against the number of mesh cells. It can be observed from the results that grid independence is marked by the convergence of the aerodynamic functions of interest. Considering the balance between accuracy and simulation runtime, the L1 mesh will be utilized as the baseline mesh for the airfoil optimization, while the L2 mesh will be employed for the wing optimization.

Table 2.1 GCS numerical results.

Geometry	Type	No of cells	Drag C_D	Lift C_L	Moment C_M	Runtime (s)
RAE2822	L1	31,232	0.0120099	0.6953153	-0.0061318	12.26
	L2	7,808	0.0132237	0.7323998	-0.0061723	4.50
	L3	1,952	0.0166476	0.7817169	-0.0053220	3.54
CRM wing	L1	3,604,480	0.0203859	0.4996933	-0.1796512	3877
	L2	450,560	0.0212745	0.4998023	-0.1811899	1807
	L3	56,320	0.0236952	0.4996504	-0.1879029	88

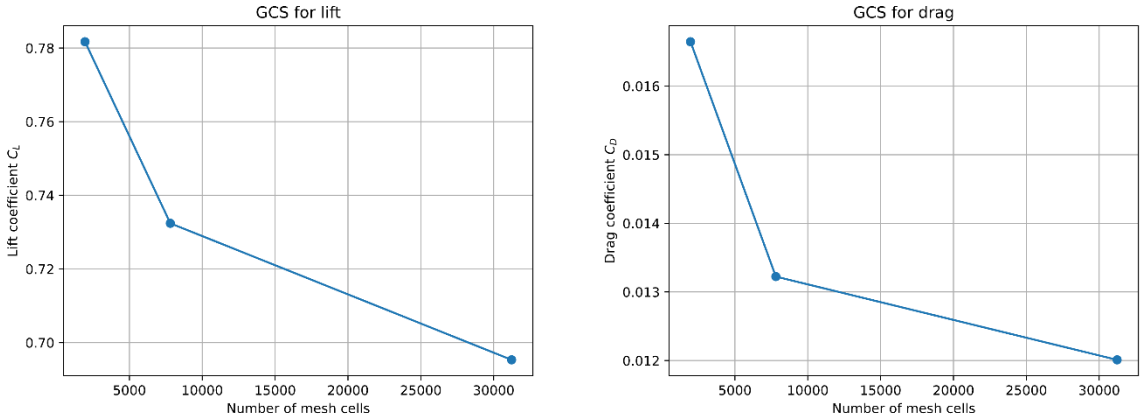


Figure 2.7 GCS plots for the RAE2822 airfoil baseline mesh.

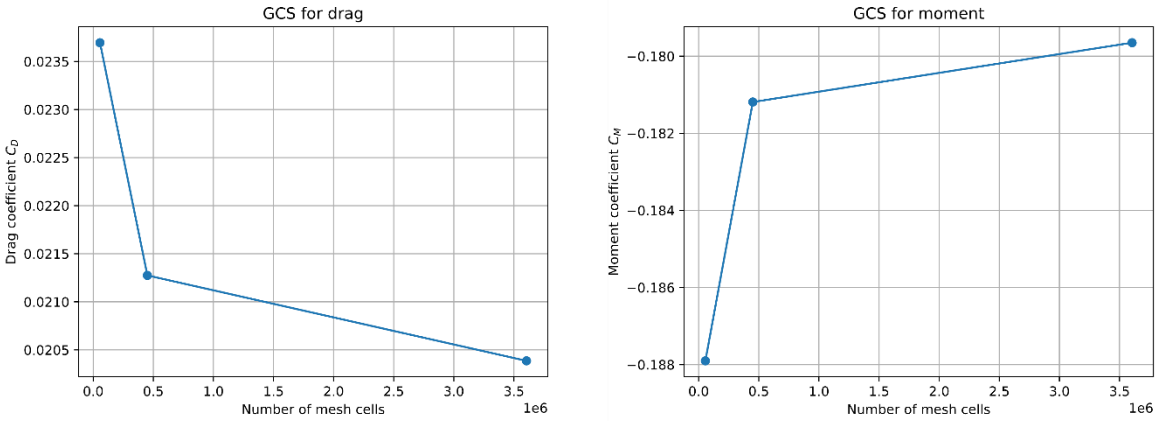


Figure 2.8 GCS plots for the CRM wing baseline mesh.

2.5 Surrogate modeling

In the realm of computational engineering, high-fidelity models often require substantial time and computational resources. Additionally, it is important to have insights into the influence of design variables on different objective functions, known as global sensitivity analysis. To tackle these challenges, the surrogate-based approach for analysis and optimization has emerged as a valuable tool. Surrogates, constructed based on data obtained from high-fidelity models, offer rapid approximations of objectives and constraints at new design points. This enables the feasibility of optimization studies by providing efficient and practical alternatives to directly using the expensive high-fidelity models.

The process of constructing a surrogate model involves sampling the objective function at a limited number of points (DoE step in Section 2.3) and using these samples to train the model. Various techniques can be employed to construct the surrogate model, such as polynomial regression, Kriging (also known as Gaussian process) [20], radial basis functions [35], or neural networks [36]. These models capture the relationship between the input variables and the objective function, allowing for predictions to be made at unobserved points. Once the surrogate model is constructed, it can be used within an optimization algorithm to guide the search for the optimal solution. The surrogate model provides an inexpensive approximation of the functions of interest, allowing the optimization algorithm to explore the search space more efficiently. The surrogate model is periodically refined as more evaluations of the expensive functions of interest become available, improving its accuracy. This thesis focuses on using a neural network-based surrogate model, refer to Chapter 3 for more details.

2.6 Sub-optimization via a genetic algorithm

2.6.1 Genetic algorithm: NSGA-II

Evolutionary algorithms (EAs) are computational techniques that imitate natural evolutionary principles to conduct search and optimization processes. Just as biological evolution and natural selection favor good genes for survival, EAs aim to find optimal solutions by mimicking this process, see Figure 2.9. Humans are the living example of a species that has undergone such evolution.

One key advantage of using EAs for optimization is their ability to discover and maintain multiple optimal solutions within an single optimization run. This feature makes them particularly promising for solving multi-objective optimization problems. EAs employ a population-based approach, where a group of solutions, known as a population, is evaluated in each iteration of the algorithm. One well-known type of EA is the genetic algorithms (GAs), originally conceived by John Holland [37] at the University of Michigan, Ann Arbor. Over the course of several decades, GAs have undergone significant development, leading to the creation of NSGA-II (non-dominated sorting GA-II) by Deb et al. [38]. NSGA-II is specifically designed to tackle multi-objective optimization problems, although it can also be applied to single-objective optimization problems [39]. This algorithm has garnered substantial attention due to its simplicity and broad applicability across various fields. Moreover, the availability of code implementations has further contributed to its popularity. In this study, NSGA-II is chosen as the core optimization for the sub-optimization process. The term “sub” signifies that the entire NSGA-II-based optimization process is executed within a single main iteration.

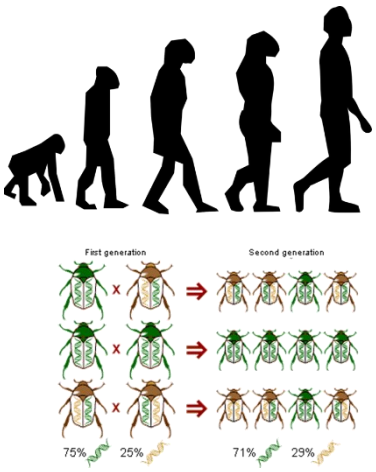


Figure 2.9 Illustrations for biological evolution and natural selection process [40].

All genetic algorithms, including NSGA-II, rely on fundamental operators to drive the evolutionary search process. These operators are inspired by biological mechanisms and include selection, crossover, and mutation. The selection operator, also known as the reproduction operator, performs several tasks. It identifies promising solutions within a population, creates multiple copies of these good solutions, and eliminates weaker solutions. This exploitation feature leverages the existing solutions to improve the overall population quality. The crossover and mutation operators are responsible for generating new solutions in the search process. The crossover operator draws inspiration from the idea of combining genetic material from fit parents, with the aim of producing offspring that inherit favorable traits. On the other hand, the mutation operator introduces genetic variations in the solutions, allowing for the exploration of new solution space. These exploration features are vital for searching beyond the current solutions and finding potentially better solutions.

Maintaining a balance between exploitation and exploration is crucial when solving optimization problems. It is important to have solutions that are both close to optimal solutions (exploitation) and diverse in their characteristics (exploration), to ensure the discovery of global optima. Achieving this balance is a key consideration in the design of genetic algorithms. For a more detailed understanding of these operators, including their mathematical foundations, it is recommended to refer to Kalyanmoy Deb's book [41]. This thesis provides only conceptual aspects, and a comprehensive mathematical background of these operators is beyond its scope.

2.6.2 Combining NSGA-II with a surrogate model

In order to perform a search in an optimization problem, the solutions need to be evaluated. The term "evaluation" refers to the process of assigning a fitness or score to a given solution using analytical functions, numerical simulations, e.g., CFD, or experiments. In test problems, where analytical functions are known, evaluations can be performed quickly and inexpensively. However, in expensive problems, rigorous numerical simulations are required to evaluate the solutions. Since NSGA-II is a population-based approach, a large number of solutions are typically evaluated. While this is not an issue when analytical functions are used for evaluation, it becomes a disadvantage in problems that involve expensive evaluations, leading to longer computational times.

To mitigate the computational cost associated with expensive evaluations, surrogate modeling can be considered. In this study, a neural network, specifically a multilayer perceptron (MLP), is employed as a surrogate model (see Chapter 3).

The abstract flow, as shown in Figure 2.10, presents the general steps of the GA-based optimization process. It begins with initializing the population through an initial sampling step. It is important to note that this step should not be confused with the DoE step mentioned in Section 2.3. In this context, the initial sampling refers to another DoE step conducted in the approximate objectives space using the surrogate model. In this thesis, as the GA is integrated into the sub-optimization process, the initial samples are directly derived from the best population of the previous main iteration (sorted using non-dominated sorting technique [38]).

After the population is initialized, the first population is evaluated to assess their performance. Next, the survival phase is carried out, during which fitness values are assigned to each solution, allowing them to be ranked. All these fitness values are evaluated by the surrogate model. The selection process then takes place based on these ranks, determining which solutions will move forward to the next generation.

Following selection, the crossover and mutation processes occur. These genetic operators generate new solutions by combining genetic information from selected parents and introducing genetic variations, respectively. The resulting solutions are then evaluated again, marking the completion of one optimization iteration. This iterative process continues, with subsequent iterations repeating the steps of population evaluation (by the surrogate model), survival, selection, crossover, and mutation. Through this repetitive cycle, the optimization algorithm aims to enhance the quality of solutions over time.

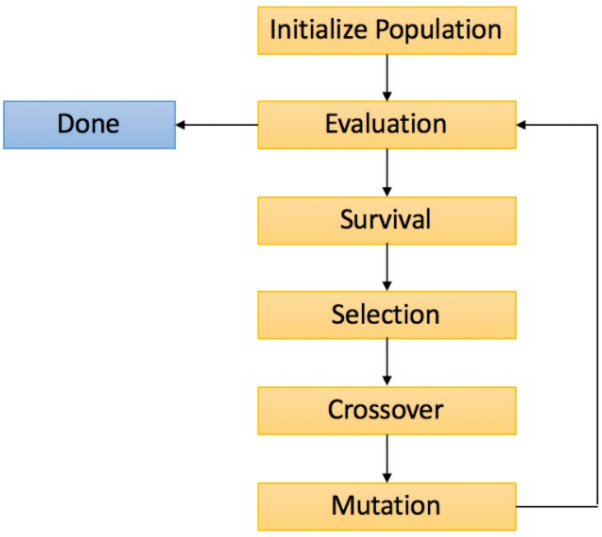


Figure 2.10 General flow of a genetic algorithm.

2.7 Infilling process

The preceding step, which involves sub-optimization via a genetic algorithm, will produce a set of optimal solutions. This is due to the population-based nature of the algorithm. If the problem being tackled has multiple objectives, the resulting population will exhibit trade-offs. In this scenario, an infilling criterion needs to be determined since not all optimal solutions will be evaluated as new data points using CFD. In this thesis, a K-means algorithm [42] for clustering data is used as an infilling criterion. The K-means algorithm helps in downsizing the set of optimal solutions to K points (closest to centroids), which will then be utilized as the new design points. It is important to note that K represents the number of centroids specified by the user (see Figure 2.11 below), and in this thesis, K is set to 8. The clustering process takes place in the approximate objective space, utilizing the surrogate model. Another two points are selected from the extreme solutions on the left and right sides of the objective space (we have 10 new points in total). This selection ensures comprehensive coverage of the objective space.

However, if the problem being addressed has a single objective function, the resulting population will have identical predicted objective function values in the approximate objective space, as determined by the surrogate model. In such cases, one can select a single solution from the set of optimal solutions or employ the K-means infill criterion with $K = 1$ done in the design or decision space. It is noteworthy that the surrogate model exhibits a multi-modality feature: different design variables result in identical predicted objective functions.

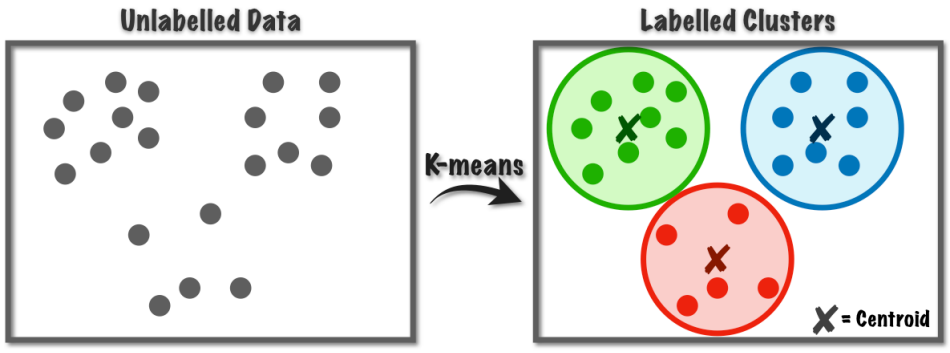


Figure 2.11 Illustration for the K-means clustering algorithm [43].

2.8 Believer sub-iteration

During this sub-iteration, the predicted candidate data are added to the existing database, and the process of refining the surrogate model is repeated. Additionally, a separate "believer" model, distinct from the main surrogate model, is trained and utilized in this sub-iteration. This approach is particularly effective when applied to the Kriging model, which is commonly referred to as the "Kriging believer", see Reference [44]. However, in this thesis, we discovered that we could apply a similar technique to some extent with a neural network-based model.

It is worth noting that using this technique with a neural network-based model poses challenges due to the model's susceptibility to overfitting. Therefore, we employ this technique exclusively in the case of wing optimization, where the problem involves a single-objective function and the CFD simulations are considerably more resource-intensive compared to other optimization problems addressed in this thesis.

The primary motivation for employing this technique is to parallelize the CFD simulations by generating multiple infill points. By leveraging the predicted candidate data (input-output data from the surrogate model) and refining the surrogate model iteratively, we aim to expedite the optimization process and achieve improved efficiency in running the expensive CFD simulations.

2.9 Stopping criterion

The stopping criterion determines when the optimization algorithm should terminate. In this thesis, the computational resource in terms of the maximum number of CFD simulations is considered as a criterion. The goal is to halt the entire algorithm when the available CFD simulations resources are fully utilized. Once this condition is reached, the design framework flow is terminated, and the current optimal solution within the database is considered the best within the given resource constraints. Please refer to Chapter 4, Section 4.1.1 for the specific stopping criterion for each optimization problem.

2.10 Summary

This chapter introduces a design framework that utilizes a surrogate-based optimization (SBO) approach for efficient exploration of the design space. SBO is a numerical optimization technique that employs a surrogate model to approximate costly functions. The proposed framework is specifically tailored for aerodynamic wing shape optimization (AWSO). The chapter provides a detailed step-by-step procedure, including the parameterization of wing geometry, initial sampling using Design of Experiment (DoE) methods, conducting CFD simulations, constructing a surrogate model, performing sub-optimization with genetic algorithms, implementing an infill sampling process, incorporating believer sub-iteration, and defining the stopping criterion.

The procedure provided in this chapter offers comprehensive details for each step. The geometric parameterization step introduces the use of the free-form deformation (FFD) technique. In the Design of Experiment (DoE) step, the conventional Latin hypercube sampling (LHS) is explained. The chapter also provides the theoretical background of Computational Fluid Dynamics (CFD), presenting the mathematical equations of the Reynolds-Averaged Navier-Stokes (RANS) equations. Additionally, a mesh deformation technique and a grid convergence study are presented.

Next, the chapter discusses the motivation for utilizing surrogate modeling and mentions different model choices. In the sub-optimization process, the core technology driving the search is NSGA-II, a variant of genetic algorithms. The chapter also highlights the technique of combining a surrogate model with NSGA-II. For the infilling process, a K-means clustering algorithm is explained. Furthermore, the chapter discusses the difference in the infilling criterion between single-objective and multi-objective optimization. To enhance parallelizability, a believer sub-iteration technique is proposed, which generates multiple infill points. Finally, a stopping criterion based on computational resource constraints is discussed.

Chapter 3 Deep learning techniques

In this chapter, we delve deeper into the incorporation of deep learning techniques within the design framework, which was introduced in Chapter 2. We provide an extensive introduction to three fundamental deep learning techniques: multilayer perceptron (MLP), generative adversarial networks (GANs), and convolutional neural networks (CNNs). Furthermore, we explore the application of these techniques in the context of addressing the specific challenges posed by aerodynamic wing shape optimization. We develop three approaches: MLP-based surrogate modeling, DCGAN-based initial sampling, and CNN-based geometric filtering. The first approach aims to replace the computationally expensive CFD simulations utilized in the sub-optimization process. On the other hand, the second and third approaches are designed to tackle the challenges associated with the curse of dimensionality, as discussed Chapter 1, Section 1.1.3.

3.1 MLP-based surrogate modeling

3.1.1 Artificial neural network

The Artificial Neural Network (ANN) is an abstract computational model inspired by the intricate nature of the human brain (a highly complex, nonlinear, and parallel information-processing system). The ANN provides an analytical representation of the relationship between the input variables, denoted as $\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}^{\mathrm{T}}$, where n represents the dimensionality of the design variables. The output, denoted as $\boldsymbol{f} = \{f_1, f_2, \dots, f_m\}^{\mathrm{T}}$, where m represents the number of expensive-to-evaluate functions to approximate in a single model. Unlike Kriging [20], which primarily interpolates data, the ANN operates as a regression model. Consequently, it possesses the potential for inherent fault tolerance and robust features when used with optimization algorithms [45], implying that the ANN can be directly constructed using multi-fidelity CFD data, without requiring a noise term as necessary in Kriging to handle such data.

An artificial neuron is a fundamental component of an ANN, functioning as an information-processing unit. It comprises two essential elements: an adder and an activation function. The adder performs the task of linearly combining the input signals, while the

activation function enables the ANN to map and process nonlinear functions. Additionally, an externally applied bias, denoted as b_k, can be incorporated into the adder. Multiple neurons collectively form a layer, with each neuron establishing connections to other neurons through links characterized by weight values, as visualized in Figure 3.1 below.

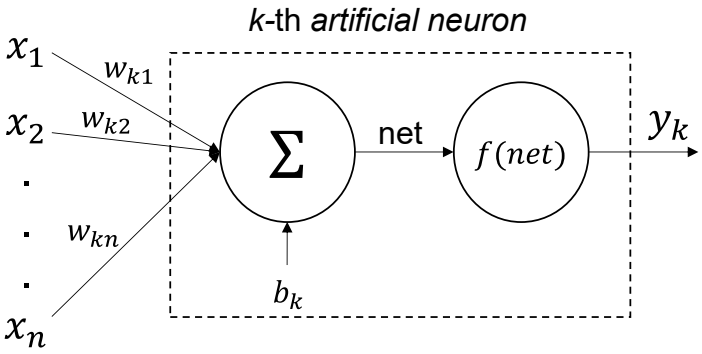


Figure 3.1 Artificial neuron mechanism.

The net of a neuron k can be represented as follows:

$$net_k = x_1 w_{k1} + x_2 w_{k2} + \cdots + x_n w_{kn} + b_k \tag{3.1}$$

$$net_k = \sum_{i=1}^{n} x_i w_{ki} + b_k \tag{3.2}$$

Then the neuron computes the output y_k as a certain function f of net_k value as follows:

$$y_k = f(net_k) \tag{3.3}$$

The function f is called the activation function. Some popular activation functions include sigmoid, hyperbolic tangent, and rectified linear unit (ReLU). The regression model is constructed by training on a collection of data samples that contain input variables (design variables) denoted as $\boldsymbol{x}$ and target outputs (in this case, CFD data) represented as $\boldsymbol{t}$. To evaluate the performance of the model in predicting the desired response, a loss function is employed. In regression tasks, one commonly used loss function is the mean squared error (MSE). The MSE loss function is defined as follows:

$$E(\boldsymbol{w}) = \sum_{j=1}^{J} \|\boldsymbol{t_j} - \boldsymbol{y_j}\|^2 \tag{3.4}$$

where $\boldsymbol{y_j}$ is the predicted value vector, $\boldsymbol{t_j}$ is the desired response vector, J is the number of samples trained in one batch (batch size, and $\boldsymbol{w}$ represents the weight values of the network. Equation (3.4) can also be divided by J to obtain the mean value.

During the training process, the objective is to minimize the loss function by iteratively updating the weight values. This weight adjustment is carried out in a supervised manner using the error backpropagation technique [46] in conjunction with the gradient descent method [47]. This iterative approach allows the model to gradually refine its weights based on the gradients of the loss function, leading to improved performance over time. Refer to [45] for more details.

3.1.2 Multilayer perceptron

A multilayer perceptron (MLP) belongs to the category of feedforward artificial neural networks, where the connections between nodes do not form cycles. An MLP is composed of multiple layers, including an input layer, an output layer, and one or more hidden layers. In this study, a five-layer MLP is utilized, comprising an input layer, three hidden layers, and an input layer. The architecture of this MLP is illustrated in Figure 3.2.

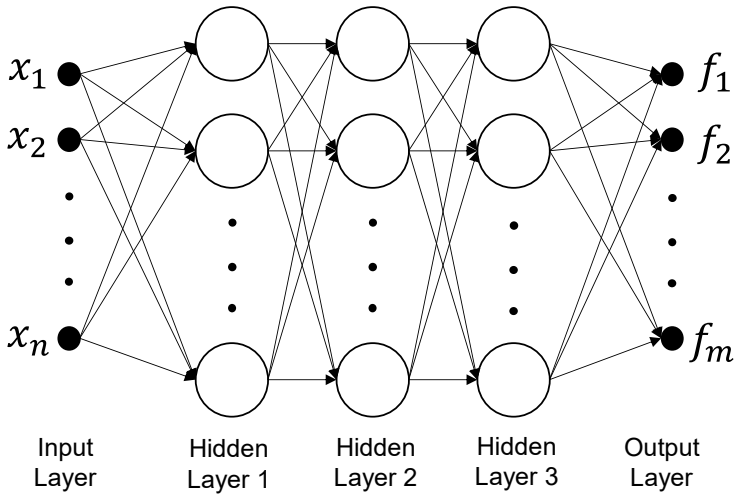


Figure 3.2 A multilayer perceptron architecture in this study.

The input layer of the MLP receives the design variables information, while the output layer processes the expensive-to-evaluation functions. The three hidden layers play a crucial role in mapping the input to the output. Previous research, such as Shen et al. [48], suggests that three hidden layers in a neural network are sufficient for effectively mapping highly non-linear functions.

In this study, the LeakyReLU activation function is employed, which is an extended version of the ReLU activation function. The activation functions are incorporated into each layer of the hidden layers. The ReLU activation function tends to saturate and output zero in

the negative range, resulting in a phenomenon known as the "dying ReLU problem". To address this issue, the LeakyReLU activation function is preferred as it introduces a new term in the negative value region of $\boldsymbol{x}$, preventing saturation. This is particularly important in dense networks where the saturation case frequently occurs. Equations (3.5) and (3.6) provide an explanation of how these activation functions operate.

$$ReLU(x) = \max(0, x) \tag{3.5}$$

$$LeakyReLU(x) = \max(0, x) + 0.01 \cdot \min(0, x) \tag{3.6}$$

3.1.3 MLP for aerodynamic performance approximator

In this thesis, the MLP is utilized for a regression task, specifically to establish an analytical mapping between the design variables of the wing candidate (FFD control points) and its relevant aerodynamic functions of interest, as depicted in Figure 3.3. The training data for the MLP model are acquired through CFD simulations. Once the MLP is trained, it can be employed for rapid prediction of aerodynamic performance. Subsequently, the MLP replaces the need for CFD in the sub-optimization process, acting as a surrogate model.

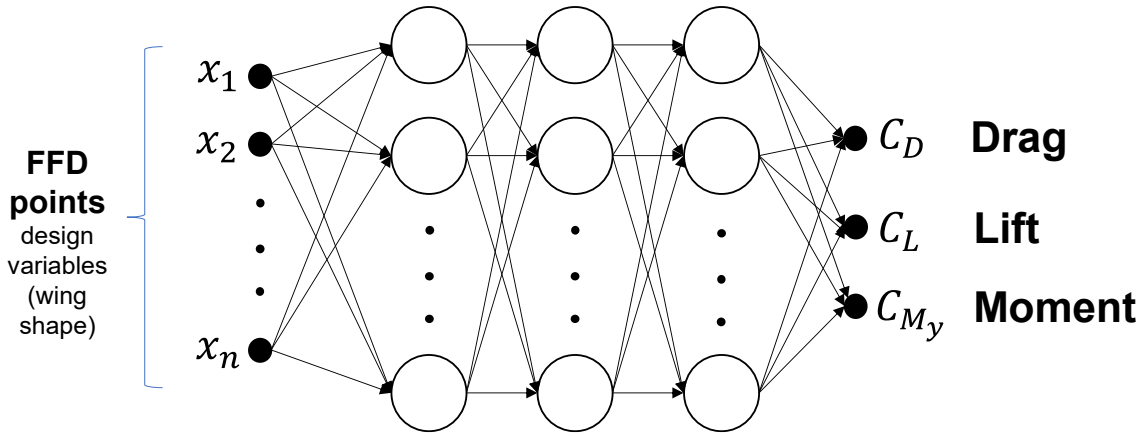


Figure 3.3 MLP for aerodynamic performance approximator.

The initial step involves diving the design database (with CFD data) into two distinct sets: the training set and the validation set. The training set is utilized to train the MLP model, while the validation set is used to evaluate and validate the model's performance. This approach is commonly known as cross-validation. In this study, the training and validation sets are randomly selected from the design database, with a ratio of 90:10.

The MLP model is constructed by training it using the training set. The training process involves sequentially updating the weight values of the model to ensure it captures the intrinsic features in the training data. The training process consists of two phases: the feedforward phase and the backpropagation phase. In the feedforward phase, the training set is fed into the model. Initially, the model has random weight values, and it predicts the output based on these weights. In the subsequent training during the next main iteration of the design framework (Chapter 2), the previous weights obtained from the previous iteration are utilized. These weights serve as the starting point for the training process, allowing the model to build upon the progress made in the previous iteration. By leveraging the previous weights, the MLP model can continue its learning process and further refine its performance in each successive iteration. In the backpropagation phase, the slope of the function is calculated using gradient descent [47], and this value is used to update the weights using the Widrow-Hoff rule, expressed as follows.

$$w_{k,l} \coloneqq w_{k,l} - \eta \frac{\partial E(\boldsymbol{w})}{\partial w_{k,l}} \tag{3.7}$$

where $w_{k,l}$ represents the weight value connecting neuron k in a layer to neuron l in the subsequent layer. The learning rate is denoted by η, and $E(\boldsymbol{w})$ represents the cost function. The optimization task of gradient descent, aiming to minimize the cost function $E(\boldsymbol{w})$, is performed using the Adam optimizer [49], which is a gradient-based optimization algorithm.

During the validation process, only the feedforward phase is executed to compute the cost function. As the training proceeds, the cost function associated with the training set decreases, while the cost function for the validation set eventually starts to increase, see Figure 3.4. This indicates a potential issue of overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. The training process is stopped whenever signs of overfitting are observed. If no signs of overfitting are detected, the training continues, progressing through each epoch. The maximum number of epochs is set to 1000.

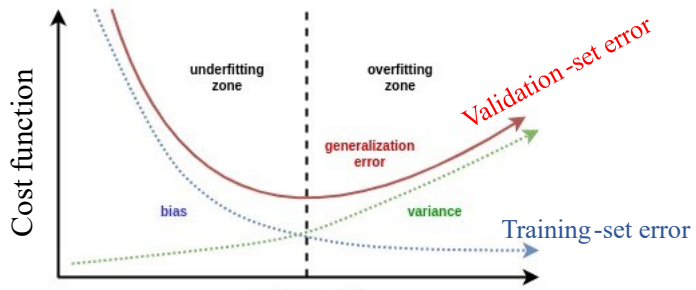


Figure 3.4 Validation error starts to increase as the training error decreases.

3.2 DCGAN-based initial sampling

3.2.1 Generative adversarial networks

Generative Adversarial Networks (GANs) proposed by Goodfellow et al. [50] are a class of deep learning models that consist of two components: a generator and a discriminator. GANs are designed to learn and generate new data samples that resemble a given training dataset. The generator generates synthetic samples, while the discriminator evaluates and distinguishes between real and fake samples. The generator and discriminator are trained simultaneously in an adversarial process. The generative model G is trained to produce a new sample with the same P_{data} distribution with the training samples and the discriminative model D is trained to distinguish whether a sample came from the G model. In other words, the G model aims to produce realistic samples that can deceive the D model, while the D model aims to accurately identify real and fake samples. Through an iterative, the G model improves its ability to generate more realistic samples, while the D model enhances its ability to discriminate between real and fake samples. The training of GANs involves a game-like scenario where the generator and discriminator play against each other, resulting in a dynamic equilibrium. The training loss function is mathematically expressed as a minimax problem, written below.

$$\min_{G}\max_{D} V(D,G) = \boldsymbol{E}_{\boldsymbol{x}\sim P_{data}}[logD(\boldsymbol{x})] + \boldsymbol{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}}[\log\,(1 - D(G(\boldsymbol{x}))] \tag{3.8}$$

where $\boldsymbol{x}$ and $\boldsymbol{z}$ are the training dataset and noisy inputs, respectively, and $\boldsymbol{E}$ is the expected value. By learning the underlying features of P_{data} and as the training progresses, the G model learns to generate samples that are increasingly difficult for the D model to differentiate from real samples. This leads to the creation of novel and high-quality synthetic samples. For illustration, the following analogy of the fake money forger and the police officer is demonstrated, see Figure 3.5. The forger represents the G model, while the police officer represents the D model. The forger's objective is to create counterfeit money that is indistinguishable from real currency, while the police officer's goal is to identify and apprehend counterfeits. During the training phase, the forger's initial attempts at creating fake money are easily detected by the police officer. However, through competitive interaction, both parties improve their skills. The forger refines their counterfeiting techniques to mimic the genuine features of real money, while the police officer becomes more knowledgeable about detecting counterfeit bills. Over time, an equilibrium is reached where the forger's counterfeit money becomes difficult for the police officer to distinguish from genuine currency.

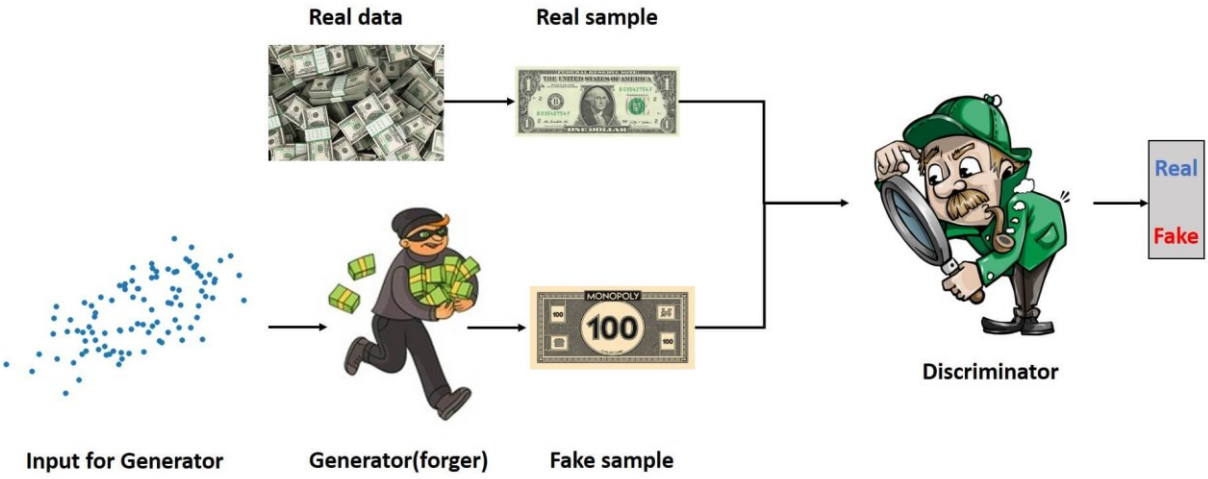


Figure 3.5 Illustration for an adversarial process of training GANs [51].

GANs have demonstrated remarkable success in various domains, including image synthesis, text generation, and vide generation. They have enabled advancements in areas such as computer vision, art generation, and data augmentation. However, training GANs can be challenging and requires careful training to achieve stable and high-quality results. Refer to [50] for more details on the theory and mathematical proofs.

3.2.2 Deep convolutional generative adversarial network

A Deep Convolutional Generative Adversarial Network (DCGAN) is a direct extension of GAN proposed by Radford et al. [52] that utilizes deep convolutional neural networks (CNNs) to generate synthetic data. DCGANs are particularly effective in generating realistic and high-resolution images. Unlike traditional GANs, DCGAN employs convolutional layers in both the generator and discriminator models. The generator network uses transposed convolutions to upscale the input noise into a high-resolution synthetic data. The discriminator network, on the other hand, uses convolutional layers to down sample the synthetic data matrices into a single value, classifying whether the input is real or fake.

The use of convolutional layers helps DCGAN to handle spatial dependencies and maintain the local structure of the generated samples. Training a DCGAN involves a similar adversarial process as other GANs, which is via backpropagation technique [47] (the same technique we used to train an MLP). The suitable loss function of this type of network is the Binary Cross Entropy (BCE), which is a pretty similar expression to Equation (3.8).

3.2.3 DCGAN for producing synthetic airfoil samples

The aim of this section is to develop a generative model capable of producing synthetic airfoil coordinates that possess realistic, smooth, and aerodynamically favorable characteristics. This model will be utilized in the design of experiment (DoE) step phase, potentially replacing the current Latin hypercube sampling method, which tends to generate unrealistic and curvy samples as observed in the results section of Chapter 4.

In this thesis, the DCGAN is employed to generate synthetic airfoil coordinates for initial sampling (DoE step). Figure 3.6 illustrates the architecture of our customized DCGAN, designed specifically for the generation of synthetic airfoil coordinates. The G model, within the DCGAN, utilizes 100-dimensional noise as input and employs deconvolutional layers (transposed convolutions) to perform an upsampling process, ultimately generating airfoil z-coordinates. On the other hand, the D model takes these coordinates and utilizes convolutional layers to perform a downsampling process, resulting in a scalar value that estimates the probability of an airfoil coordinate sample being produced by the G model.

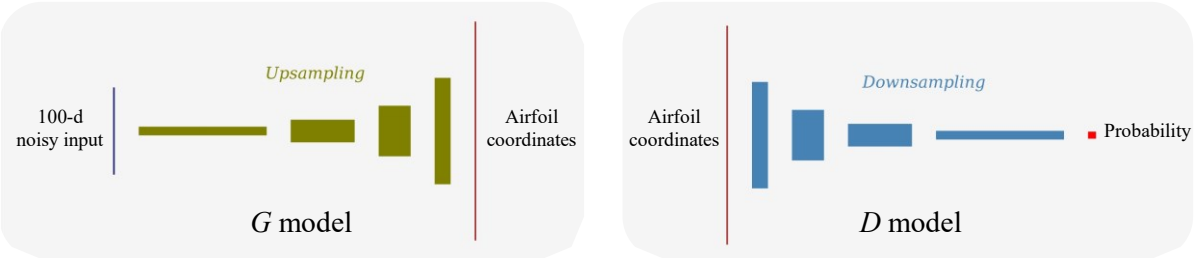


Figure 3.6 DCGAN for synthetic airfoil generation.

To train the DGCAN, we selected 77 transonic airfoils from the UIUC airfoil database [53], see Table 3.1 for the complete list of airfoil names. The choice of transonic airfoils is motivated by the fact that our focus is on aerodynamic wing shape optimization problems in the transonic regime. It is important to note that different applications may require the selection of different airfoil types.

Following the recommendation of Li et al. [54], we normalized the 77 transonic airfoil coordinates to ensure stable training and better generative capability via Algorithm 1. This normalization procedure is needed since the original UIUC airfoils possess different thickness and camber characteristics and have been designed for different purposes.

The normalization first removes the dominant camber information from the data by subtracting the component contributed by the first camber mode. The camber mode is solved by using the camber-thickness shape mode proposed by Li et al. [55]. Then, we scale the remaining component of the airfoil by dividing the maximum absolute value of $\boldsymbol{y}$, referred to as the thickness factor. After normalization, all values are in the $[-1,1]$ interval.

These selected UIUC transonic airfoils are considered to be realistic and smooth, see Figure 3.7. By training our DCGAN model using these airfoils, we aim to capture their intrinsic features and generate new synthetic airfoil coordinates that are both realistic and representative of airfoils with aerodynamic characteristics in the transonic regime.

Table 3.1 Complete list of the 77 UIUC transonic airfoil names

	0	1	2	3	4	5	6
0	rae5212	rae5213	rae5214	rae5215	rae69ck	c141a	c141b
1	c141c	c141d	c141e	c141f	c5a	c5b	c5c
2	c5d	c5e	cast102	kc135a	kc135b	kc135c	kc135d
3	vr11x	ames01	nl722343	k1	k2	nlr7301	giiia
4	giiib	giiic	giiid	giiie	giiif	giiig	giiih
5	giiii	giiij	giiik	giiil	vr11x	vr12	vr13
6	vr14	vr15	apex16	npl9510	dfvlrr4	c5a	c5b
7	c5c	c5d	c5e	nplx	nl722362	n0011sc	rae2822
8	sc20010	sc20012	sc20402	sc20403	sc20404	sc20406	sc20410
9	sc20412	sc20414	sc20503	sc20518	sc20606	sc20610	sc20612
10	sc20614	sc20706	sc20710	sc20712	sc20714	sc21006	sc21010

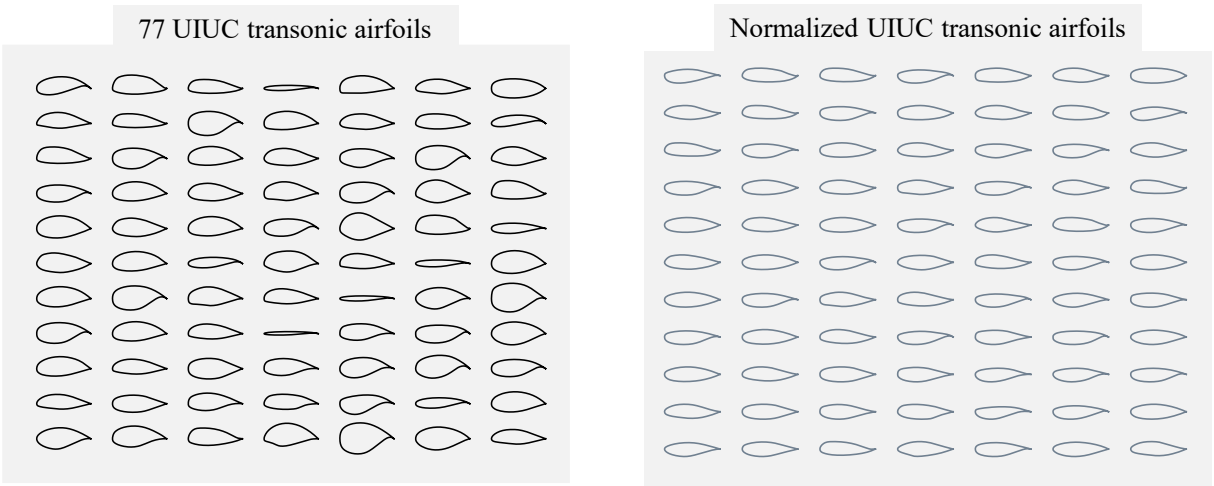


Figure 3.7 The 77 UIUC transonic airfoils: original (left) and normalized (right).

Algorithm 1 Normalization of airfoil coordinates

procedure Normalization$(\boldsymbol{y}, \boldsymbol{C})$ ⇨ $\boldsymbol{y}$ is the airfoil y coordinates, $\boldsymbol{C}$ is the first camber mode
  $\boldsymbol{y}_{camber} = f_c(\boldsymbol{y})$ ⇨ f_c computes the camber line coordinates of this airfoil
  $c_1 = \boldsymbol{C}^T \boldsymbol{y}_{camber}$ ⇨ Compute the first camber mode component
  for $i = 1; i \leq n_f; i++$ **do** ⇨ Cycle to subtract the first camber component
    $y_i = y_i - c_1 \boldsymbol{C}_i$
    $y_{N-i-1} = y_{N-i-1} - c_1 \boldsymbol{C}_i$
  end for
  $t = \max |\boldsymbol{y}|$ ⇨ Compute the thickness factor
  $\boldsymbol{y} = \boldsymbol{y}/t$ ⇨ Normalize
  return $\boldsymbol{y}$ ⇨ Return normalized $\boldsymbol{y}$
end procedure

The G model produces normalized synthetic airfoils by utilizing random noisy inputs. These synthetic airfoils undergo further transformations via Algorithm 2, including the multiplication of a thickness factor and the assignment of a dominant camber component. The thickness factor and dominant camber coefficient are fundamental parameters that determine the overall shape of the airfoils. In a given design problem, the bounds of these parameters can be predetermined based on specific thickness and lift requirements.

To generate the synthetic airfoils, we employ DCGAN to produce the normalized versions, and then employ LHS to generate two preliminary parameters (t and c) for each synthetic airfoil. Subsequently, we scale the thickness and apply the corresponding dominant camber coefficient to transform the synthetic airfoils accordingly. All this procedure is outlined in Algorithm 2, representing the inverse procedure of Algorithm 1.

Algorithm 2 Transformation of DCGAN synthetic airfoils to sampling airfoils

procedure Transformation$(\boldsymbol{y}, t, c)$ ⇨ $\boldsymbol{y}$ is the airfoil y coordinates, t is the thickness factor
  c is the dominant camber coefficient
  $\boldsymbol{y} = \boldsymbol{y} \times t$ ⇨ Scale the synthetic airfoil coordinates
  for $i = 1; i \leq n_f; i++$ **do** ⇨ Cycle to add on the dominant camber coefficient
    $y_i = y_i + c\boldsymbol{C}_i$
    $y_{N-i-1} = y_{N-i-1} + c\boldsymbol{C}_i$
  end for
  return $\boldsymbol{y}$ ⇨ Return the transformed airfoil coordinates
end procedure

After completing the training process, we obtain a G model that has the ability to generate synthetic and realistic airfoils in the transonic regime. At this stage, the D model is no longer necessary for further use. The transformed synthetic airfoils, produced by the G model and Algorithm 2, are referred to as "DCGAN airfoils", as discussed in Chapter 4, Section 4.2. However, these airfoils cannot be directly utilized for wing shape optimization since we still need to determine the corresponding FFD control points. It is important to note that our design variables are represented by these FFD control points and not by the airfoil coordinates. To accomplish this, we employ an inverse FFD algorithm in the subsequent section.

3.2.4 Inverse FFD algorithm

The inverse Free Form Deformation (FFD) algorithm is developed to determine the FFD control points corresponding to the synthetic airfoil shapes generated by the DCGAN model. The FFD control points serve as the design variables in the wing shape optimization process. The inverse FFD algorithm involves finding the set of control points that will produce the desired airfoil shapes. This is achieved by iteratively adjusting the control points until the resulting airfoil shapes closely match the desired synthetic airfoil shapes.

To accomplish this, an optimization algorithm is employed, such as a gradient-based method or a genetic algorithm, which aims to minimize a cost function that measures the difference between the desired airfoil shape and the shape produced by the FFD control points. This study uses NSGA-II [38] to solve this task. By applying the inverse FFD algorithm, the synthetic airfoils generated by the DCGAN model can be used as a starting point for the wing shape optimization process. Please see Algorithm 3 below for the step-by-step procedure.

Algorithm 3 The search of the corresponding FFD points given a set of DCGAN airfoils.

1:	**procedure** INVERSE_FFD($\boldsymbol{z_t}$)	⇨ $\boldsymbol{z_t}$ is the target DCGAN airfoil z-coordinates
2:	**for** $i = 1; i \leq m; i++$ **do**	⇨ loop over the m airfoils
3:	$\boldsymbol{z_t} = \text{scale}(\boldsymbol{z_t})$	⇨ scale the airfoils to have the size like baseline
4:	$\boldsymbol{z_t} = \text{reposition}(\boldsymbol{z_t})$	⇨ translate+rotate so the TE position is like baseline
5:	**end for**	
6:	**start optimization**	⇨ perform an optimization using NSGA-II
7:	minimize $\lvert \boldsymbol{z} - \boldsymbol{z_t} \rvert$	⇨ $\boldsymbol{z}$ is the candidate coordinates after perturbing $\boldsymbol{FFD}$
8:	with respect to $\boldsymbol{FFD}$	⇨ $\boldsymbol{FFD}$ is the candidate FFD control point coordinates
9:	subject to geometric constraint	⇨ the volume constraint
10:	**end optimization**	
11	**return** $\boldsymbol{FFD}$	⇨ return the FFD control point coordinates
12:	**end procedure**	

The number of DCGAN airfoils, denoted as m in Algorithm 3, differs based on the problem at hand (whether it is an airfoil or wing optimization). For airfoil optimization, we work with a single DCGAN airfoil, $m = 1$. In contrast, for wing optimization, we consider multiple DCGAN airfoils $m = 8$. This is because the baseline geometry (the CRM wing), is represented by 8 spanwise sections. Each section is associated with 24 FFD control points, resulting in a total of 192 control points spread over the 8 sections, see Chapter 2, Section 2.2.

3.2.5 Sampling flow

The entire flow of obtaining DCGAN initial wing samples for wing shape optimization is illustrated in Figure 3.8. It begins with a matrix consisting of 8 rows and 100 columns of random noises. These noises serve as input to the pre-trained G model, generating 8 DCGAN airfoils. Each airfoil represents one of the spanwise sections of our DCGAN wing.

Next, Algorithm 3 is executed to identify the corresponding FFD control points that deform the baseline CRM wing geometry to match the 8 spanwise sections of our DCGAN wing. These FFD control points serve as the representation of our first DCGAN wing sample. Ultimately, the obtained FFD control points act as the design variables used as input for our MLP-based surrogate model. This process is repeated until the desired number of initial wing samples is obtained. For airfoil optimization, the flow is similar, but with a starting point of 1 row and 100 columns of random noises.

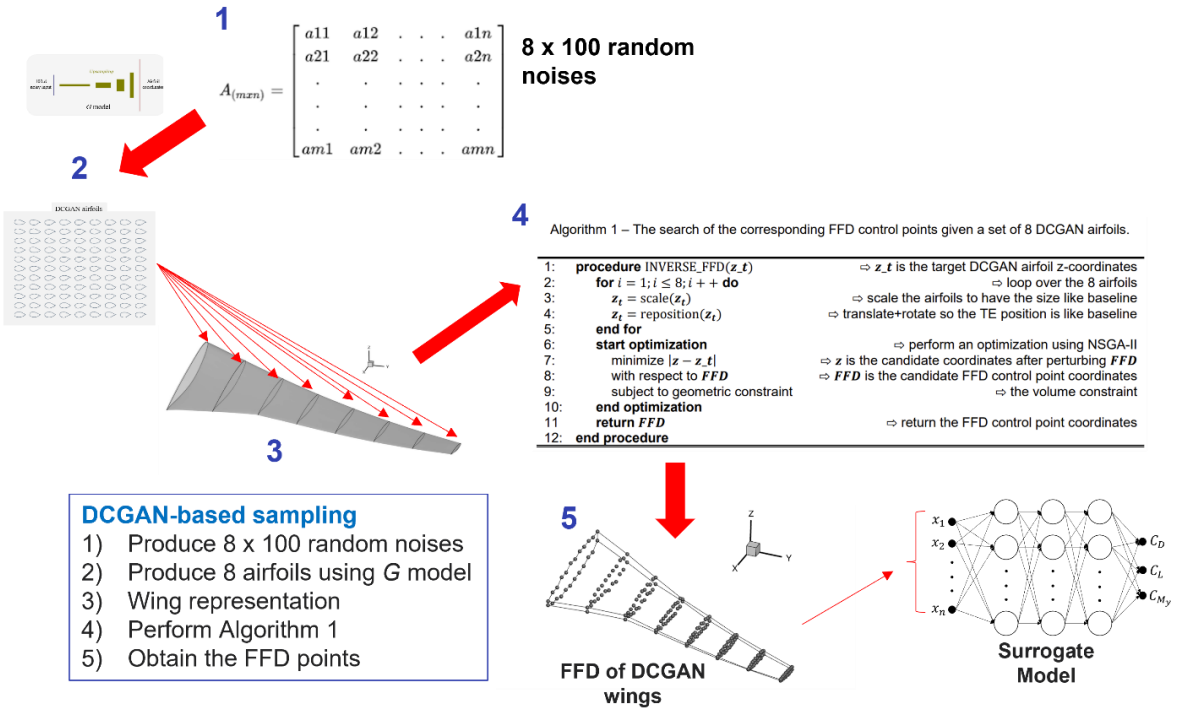


Figure 3.8 Flow for producing DCGAN initial wing samples.

3.3 CNN-based geometric filtering

3.3.1 Convolutional neural network

Convolutional neural networks (CNNs) are a specialized type of neural networks primarily utilized for analyzing visual data. CNNs were proposed in 1989 by LeCun [56], but main advances are realized in last few years. Unlike traditional neural networks, CNNs employ convolution operations instead of general matrix multiplication in at least one of their layers. They are tailored to handle pixel data and find applications in tasks related to image recognition.

Convolution operations are fundamental operations used in CNNs. The convolution operation involves applying a small filter or kernel to the input data by performing element-wise multiplication and summation. In a convolution operation, the kernel is slid over the input data in a sliding window manner. At each position, the filter is multiplied elementwise with the corresponding values in the input data, and the results are summed. This process generates a single value, which represents the response of the filter at that particular position. The key idea behind these operations is to extract local patterns and features from the input data. By sliding the kernel over the entire input, this operation allows the network to automatically learn and capture local information at different spatial locations.

Convolution operations are versatile and can handle different types of input data, such as 1-dimensional (1D), 2-dimensional (2D), and 3-dimensional (3D) data. In 1D convolution, the kernel moves along a single direction, typically used for analyzing time-series data. This allows the CNN to capture patterns and dependencies over time. In 2D convolution, the kernel moves in two directions, horizontally and vertically, making it suitable for processing images. The CNN can extract spatial features from the input images, enabling tasks like image classification, object detection, and image segmentation. In 3D convolution, the kernel moves in three dimensions, which extends the capabilities of CNNs to handle 3D image data. This is commonly applied to medical imaging tasks, such as analyzing MRI scans, CT scans, and videos, where the additional dimension represents depth or time.

The goal is to utilize a CNN architecture to identify irregularities in geometry within the sub-optimization process. Since we only have control over the design variables, specifically the FFD control points, our CNN operates on these 1-dimensional FFD control points as input data. The output of our CNN provides a probability score (we call this geometric filter score) indicating the likelihood of the sample being abnormal. Refer to Section 3.3.3 for details.

3.3.2 Kernel, padding, and stride in convolution operations

The kernel is a small matrix of learnable weights that is convolved with the input data. The size of the kernel determines the receptive field and the type of features captured by the convolution operations. By sliding the kernel over the input, the convolution operation extracts local patterns and features.

Padding is the technique of adding additional pixels or values around the input data before performing the convolution operation. It is used to control the spatial dimensions of the output. Padding can be applied to ensure that the output is the same size as the input or to adjust the output size based on specific use cases. It is often used to preserve information at the edges of the input and maintain spatial resolution during convolutions.

Stride refers to the step size at which the kernel moves across the input data during the convolution operation. A stride value greater than one allows the kernel to skip over pixels, resulting in downsampling of the output feature map. By adjusting the stride, one can control the spatial dimensions of the output. Larger strides reduce the output size, while smaller strides preserve more spatial information.

As an example, let us calculate 1D convolution by hand. The general procedure is to slide our kernel over the input, calculate the element wise multiplications, and sum them up. Say, our $input = [1, 0, 2, 3, 0, 1, 1]$, $kernel = [2, 1, 3]$, $padding = 0$, and $stride = 1$. The result of the convolution is $[8, 11, 7, 9, 4]$, which is calculated in the following way.

- $8 = 1 * 2 + 0 * 1 + 2 * 3$
- $11 = 0 * 2 + 2 * 1 + 3 * 3$
- $7 = 2 * 2 + 3 * 1 + 0 * 3$
- $9 = 3 * 2 + 0 * 1 + 1 * 3$
- $4 = 0 * 2 + 1 * 1 + 1 * 3$

Note that since $padding = 0$, we did not add additional pixel at the start or the end of the $input$ sequence. Since $stride = 1$, we slide over the input by one pixel. One also needs to understand the concept of channel and batch size. The above $input$ and $output$ sequences have $channel = 1$ and $batch = 1$. Since we used the implementation of 1D convolutional layer in PyTorch [57], we followed their requirements for the input and output matrices. The input matrix is in the size of $[N, C_{in}, L_{in}]$, while the output matrix is in the size of $[N, C_{out}, L_{out}]$, where N is the batch size, C is the channel size, and L is the sequence length.

The 1D convolution operation is expressed in the following equation.

$$out\left(N_i, C_{out_j}\right) = bias\left(C_{out_j}\right) + \sum_{k=0}^{C_{in}-1} weight\left(C_{out_j}, k\right) \star input(N_i, k) \tag{3.9}$$

where N_i is i-th batch, C_{out_j} is the j-th channel of the output, C_{in} is the channel of the input, k is the kernel order, $weight$ denotes the kernel learnable weight, and $bias$ is the bias value set by the user. Note that in the previous example, $bias = 0$. The operation $\star$ is the valid cross-correlation operator, refer to [58] for more details.

3.3.3 CNN for geometric abnormality detection

The surrogate-based optimization (SBO) method involves updating the surrogate model using infill samples obtained through sub-optimization with a genetic algorithm, as explained in Chapter 2, Sections 2.6-2.8. However, the accuracy of the model can be compromised due to poor quality training data, leading to infill samples with abnormal shapes and subsequently resulting in inferior aerodynamic performance. This, in turn, worsens the model's accuracy after the update, creating a vicious cycle, as discussed in Chapter 1, Section 1.1.3. To address this issue, there is a need for a geometric filter that can quickly detect shape abnormalities without the need for any Computational Fluid Dynamics (CFD) simulation.

In this study, a CNN-based geometric filter is developed and trained using two sets of samples: 500 samples generated by DCGAN-based sampling and 500 samples obtained through Latin hypercube sampling (LHS). The LHS samples are obtained by directly perturbing the FFD control points to produce airfoil/wing samples. The DCGAN samples are assigned a label of 1 (realistic), while the LHS samples are assigned a label of 0 (abnormal).

After training, the geometric filter can estimate the probability score S (we call it geo filter score) for a given, unseen airfoil/wing sample, and it is ready to be integrated within the sub-optimization process. Integration involves imposing a score constraint, such as $S \geq 0.4$. This constraint is analytical and considered cheap-to-evaluate. The architecture of the CNN used in the geometric filter resembles the D model in the DCGAN. Both models utilize convolutional layers to perform down-sampling, reducing the input dimensions and producing a scalar probability value.

However, there are a couple of differences between the CNN used in the geometric filter and the D model in the DCGAN. Firstly, the input for the CNN in the geometric filter is the FFD control points, whereas the D model in the DCGAN takes airfoil coordinates as input, matching the output size from the G model. The architecture of the CNN used in the geometric filter is illustrated in Figure 3.9. Another distinction is the loss function employed during training. In the DCGAN, the Binary Cross Entropy (BCE) loss function was used. However, for training the geometric filter, the mean squared error (MSE) is utilized. This is because the training process of the geometric filter involves labeled data and is supervised in nature. Cross validation technique explained in Section 3.1.3 was not used, and the training stops when it reaches maximum epoch, which is 150.

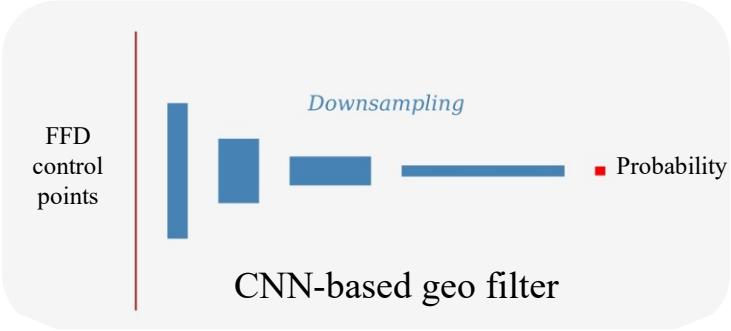


Figure 3.9 CNN architecture for geometric filter.

The determination of the score constraint depends on the distribution of scores produced by the CNN-based geometric filter when applied to the LHS and DCGAN initial samples in each optimization problem. It is important to note that the geometric filter has never seen these samples during training since it was trained using a separate set of training samples. Figure 3.10 provides an illustration of the score distribution generated by the geometric filter for the UIUC, DCGAN, and LHS initial samples in the wing optimization. It can be observed from the score distribution that the majority of LHS initial samples are clustered around a score of 0, whereas the UIUC and DCGAN initial samples are predominantly concentrated around a score of 1. This observation suggests that the CNN-based geometric filter effectively captures the distinguishing features of both the DCGAN and LHS samples, despite not being exposed to these specific initial samples during its training process.

In the current study, the constraint is manually decided prior to commencing the optimization process. A score threshold is chosen based on the premise that LHS samples are

considered abnormal, while DCGAN samples are considered realistic. For instance, in the wing optimization, a value of $S \geq 0.4$ is selected as it appears to effectively separate the DCGAN and LHS samples, as depicted in Figure 3.10. However, it is essential for future research to address how this score can be systematically determined, reducing reliance on subjective user judgment. The author has conducted a preliminary study to mitigate subjectivity in labeling the training data (assigning labels of 1 to DCGAN samples and 0 to LHS samples). To address this issue, an alternative approach utilizing a roughness metric has been proposed. This metric quantitatively measures the roughness or smoothness of the samples, enabling the application of more appropriate labels based on this quantitative assessment. Consequently, samples generated through DCGAN-based sampling are not automatically categorized as realistic, and samples from LHS-based sampling are not automatically classified as abnormal. Further details on this early work can be found in Chapter 4, Section 4.10.

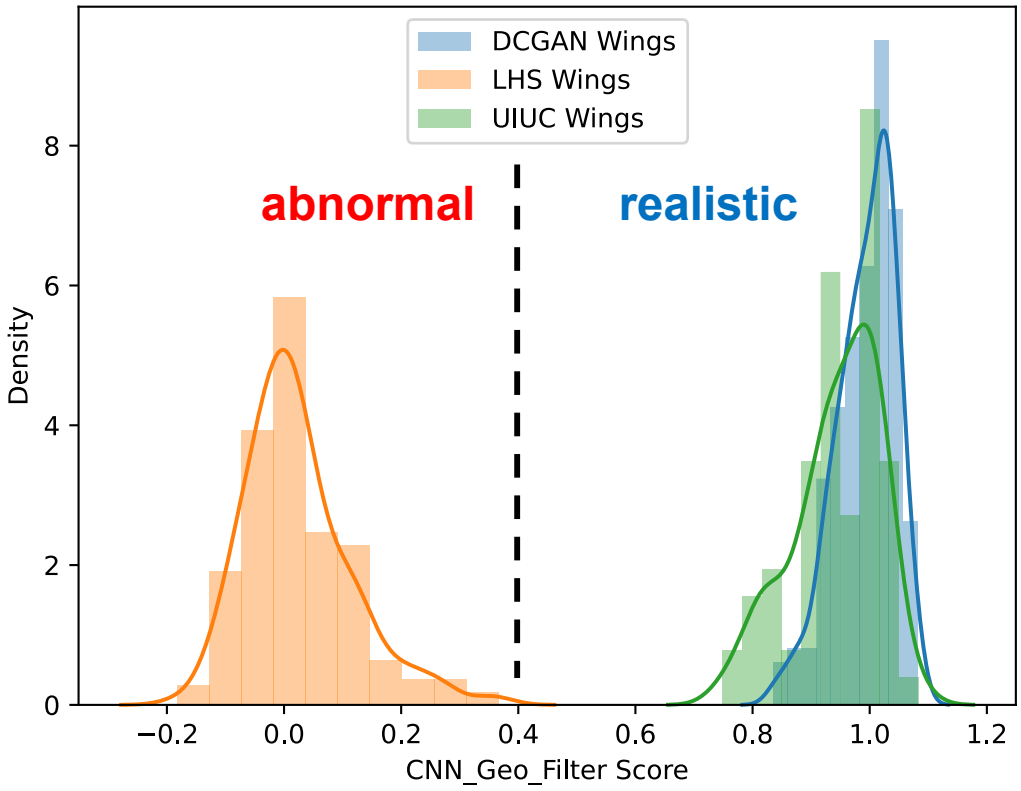


Figure 3.10 The score distribution of the LHS, DCGAN, UIUC initial wing samples

3.4 Summary

This chapter provides a comprehensive introduction of three deep learning techniques: multilayer perceptron (MLP), generative adversarial networks (GANs), and convolutional neural networks (CNNs). The chapter specifically focuses on the application of these techniques in the context of aerodynamic wing shape optimization. The development and adaptation of each deep learning technique for this purpose are thoroughly discussed.

The first section of the chapter introduces an MLP-based surrogate modeling approach, which aims to replace computationally expensive CFD simulations in the sub-optimization process. The theoretical background of neural networks is explained in detail, including the building blocks of these networks and the training process using the gradient descent algorithm. Techniques for obtaining a good model, such as cross-validation and early stopping to prevent overfitting, are also discussed.

In the subsequent section, a deep convolutional generative adversarial network (DCGAN), a variant of GANs, is introduced for generating high-quality synthetic initial samples. The chapter provides an explanation of the underlying theory of GANs, including the generative G and discriminative D models involved. An illustrative analogy involving a fake money forger and a police officer is presented to aid understanding. Furthermore, the application of DCGAN for generating synthetic airfoil samples is discussed. The chapter outlines the procedure for preparing the training dataset, focusing on the preprocessing of transonic airfoils obtained from the UIUC database. Additionally, an inverse FFD algorithm is proposed to transform the samples from DCGAN into a set of design variables suitable for optimization tasks. We then present an end-to-end sampling flow that incorporates these techniques.

Finally, the last section introduces a CNN-based geometric filtering technique designed to efficiently detect geometric abnormalities during the infill sampling process. The chapter introduces CNNs and explains the fundamental operation of convolution, which serves as the core technology within CNNs. Concepts such as kernels, padding, and stride are explained to provide a deeper understanding of this technology. The section then explores the application of CNNs for geometric abnormality detection. The utilization of CNNs is achieved by introducing an analytical constraint within the sub-optimization process. The chapter concludes by discussing the methodology used to select the threshold value for this constraint, ensuring effective detection of geometric abnormalities.

Chapter 4 Aerodynamic wing shape optimization

4.1 Experimental design

4.1.1 Optimization problems and methods

The optimization problems tackled in this study incorporate three distinct AWSO problems, each varying in complexity. These problems include: 1) single-objective airfoil optimization, 2) multi-objective airfoil optimization, and 3) single-objective wing optimization. The complexities of these problems differ in terms of the number of design variables, objective functions, constraints, the level of fidelity in the CFD simulations employed, etc. Consequently, the allocation of computational resources also varies accordingly. A comprehensive summary of these complexities and the corresponding resource allocation can be found in Table 4.1.

Table 4.1 Summary of AWSO problems and their complexities.

	Problem 1	**Problem 2**	**Problem 3**
Geometry	Airfoil	Airfoil	Wing
Baseline	RAE2822	RAE2822	CRM wing
Type	Single-objective	Multi-objective	Single-objective
Design variables	20	20	193
Objective(s)	1	2	1
Constraint(s)	4	3	12
Mesh size	31,232	31,232	450,560
Initial samples	100	100	200
Infill samples	+1	+10	+5
Believer sub-iter	False	False	True
N CFDs / method	600	600	1000

To solve the aforementioned optimization problems, we utilize the design framework based on surrogate-based optimization (SBO) as described in Chapter 2. To assess the effectiveness of the proposed deep learning techniques (DCGAN and CNN), we define three distinct optimization methods within this framework. These methods are designed in a

progressive manner to gradually incorporate each deep learning technique. By adopting this approach, we can evaluate the individual impact of each deep learning technique on the optimization process. These methods are as follows:

1. **LHS**: This method utilizes Latin hypercube sampling as the initial sampling.
2. **DCGAN**: This method employs DCGAN-based sampling as the initial sampling.
3. **DCGAN+GF**: In addition to DCGAN-based sampling as the initial sampling, this method incorporates CNN-based geometric filtering into the sub-optimization process.

The distinguishing features of these three methods are summarized in Table 4.2. We did not specifically investigate the effectiveness of utilizing a multilayer perceptron (MLP) in comparison to NSGA-II-only. It is akin to questioning whether the SBO approach is superior to using NSGA-II alone. For detailed insights into the comparison between MLP+NSGA-II and NSGA-II-only, we refer the readers to the author's previous work [59]. However, for the current study, it is acknowledged that incorporating an MLP in conjunction with NSGA-II has proven to be beneficial and effective. Hence, we accept the premise that utilizing an MLP in combination with NSGA-II yields positive results in the context of our research.

Table 4.2 Summary of optimization methods.

	Method 1	Method 2	Method 3
Name	**LHS**	**DCGAN**	**DCGAN+GF**
Initial sampling	LHS	DCGAN	DCGAN
MLP surrogate	True	True	True
CNN geo filter	False	False	True

4.1.2 Settings for MLP-based surrogate modeling

All three optimization methods in this study are based on surrogate-based optimization (SBO), where a surrogate model is dynamically trained as the design iteration progresses. Consequently, each of these optimization methods incorporates a multilayer perceptron (MLP) in their process, which serves to map the design variables, denoted as $\boldsymbol{x}$, as inputs to the aerodynamic functions of interest, denoted as $\boldsymbol{f}$, as outputs. Before training the surrogate models, the design variables undergo a normalization process, scaling them within the range of 0 to 1. This normalization is achieved by using the lower and upper boundaries of the design variables, denoted as $\boldsymbol{x}^{(L)}$ and $\boldsymbol{x}^{(U)}$, respectively.

Similarly, the normalization of the aerodynamic functions of interest is accomplished by utilizing the minimum ($\min(\boldsymbol{f})$) and maximum ($\max(\boldsymbol{f})$) values from the current database. This normalization procedure helps accelerate the training convergence of the surrogate models [60]. The training process utilizes the cross-validation technique discussed in Chapter 3, Section 3.1.3. Table 4.3 provides a summary of the other parameters utilized during the training process. The number of neurons in the input layer is not the same as the number of design variables due to the presence of constraints limiting the flexibility of some FFD points to achieve fixed trailing edge, see problem formulation for each optimization problem.

Table 4.3 Summary of MLP training parameters

	SO-airfoil-opt	**MO-airfoil-opt**	**SO-wing-opt**
$\boldsymbol{N_{neuron}}$ **(input)**	18	18	183
$\boldsymbol{N_{neuron}}$ **(hidden)**	256	256	1024
$\boldsymbol{N_{neuron}}$ **(output)**	2	2	2
Learning rate	0.001		
Max epoch	1000		
Train ratio	90% of the current database		
Batch size	32		

4.1.3 Settings for DCGAN-based initial sampling

During the training process of DCGAN, two models are employed: the generative G model and the discriminative D model. The G model expands the dimensions of 100-dimensional random noise through deconvolutional operations utilizing ReLU activation functions, except for the last layer, which employs Tanh activation. Conversely, the D model compresses the input dimensions into a scalar probability value by utilizing convolutional operations with LeakyReLU activation functions, except for the last layer, which utilizes Sigmoid activation. To ensure stable and efficient training, some layers incorporate batch normalization techniques [60].

The parameters for these models are summarized in Table 4.4 and 4.5 and were determined through an iterative process of trial and error, while also considering the general guidelines outlined in the original DCGAN paper by Radford et al. [52]. It is also noted that $bias$ is not used in this study.

Table 4.4 Summary of parameters for the generative model.

Layers	***G* model (deconvolutional)**					
	Out channel	**Kernel size**	**Stride**	**Padding**	**Batchnorm**	**Activation**
First	80	10	1	0	True	ReLU
Second	40	10	2	2	True	ReLU
Third	20	10	2	2	True	ReLU
Fourth	10	10	2	1	True	ReLU
Fifth	1	11	2	1	False	Tanh

Table 4.5 Summary of parameters for the discriminative model.

Layers	***D* model (convolutional)**					
	Out channel	**Kernel size**	**Stride**	**Padding**	**Batchnorm**	**Activation**
First	10	10	2	1	False	LeakyReLU
Second	20	10	2	1	True	LeakyReLU
Third	40	10	2	1	True	LeakyReLU
Fourth	80	10	2	1	True	LeakyReLU
Fifth	1	10	1	0	False	Sigmoid

4.1.4 Settings for CNN-based geometric filtering

The CNN architecture varies between the airfoil and wing optimization cases. In both cases, the CNN receives an input matrix of size $[N, C_{in}, L_{in}]$, where N represents the batch size, C denotes the channel size, and L is the sequence length. In both airfoil and wing optimization cases, $N = 100$, which aligns with the population size used in the NSGA-II algorithm, as detailed in Section 4.1.5. Additionally, $C_{in} = 1$, since we are working with a single sequence of design variables. For airfoil optimization, L_{in} is assigned a value of 18, while for wing optimization, L_{in} is set to 183. Consequently, the kernel sizes for these problems differ: K is set to 3 for airfoil optimization and K is set to 7 for wing optimization.

Other parameters are selected in such a way that the output achieves the correct dimensions, which is $[N, 1]$. It is noted that the probability score represents a scalar value for each sample. A summary of all the parameters can be found in Table 4.6 for airfoil optimization and Table 4.7 for wing optimization.

Table 4.6 Summary of parameters for the geometric filter for airfoil optimization

Layers	**CNN architecture (convolutional)**					
	Out channel	**Kernel size**	**Stride**	**Padding**	**Batchnorm**	**Activation**
First	10	3	2	1	False	LeakyReLU
Second	20	3	2	1	False	LeakyReLU
Third	40	3	2	1	False	LeakyReLU
Fourth	80	3	2	1	False	LeakyReLU
Fifth	1	3	2	1	False	-

Table 4.7 Summary of parameters for the geometric filter for wing optimization

Layers	**CNN architecture (convolutional)**					
	Out channel	**Kernel size**	**Stride**	**Padding**	**Batchnorm**	**Activation**
First	10	7	2	1	False	LeakyReLU
Second	20	7	2	1	True	LeakyReLU
Third	40	7	2	1	True	LeakyReLU
Fourth	80	7	2	1	True	LeakyReLU
Fifth	1	7	2	0	False	-

4.1.5 Settings for NSGA-II

The sub-optimization process utilizes an NSGA-II algorithm, with the configuration outlined in Table 4.8, in conjunction with an MLP-based surrogate model. The parameters are all the same for all optimization problems, except for the mutation coefficient, η_m. In the multi-objective case, η_m is set to 20, while in the single-objective case, η_m is set to 15, determined based on the author's best practices [39], [59]. For a detailed explanation of each parameter, please refer to the original paper [38].

Regarding the initial sampling, instead of employing a design of experiment (DoE) technique, the top 100 solutions from the previous main iteration are directly copied to serve as the initial population. The selection of the top 100 solutions is performed by applying a non-dominated sorting technique [38] to the current database.

Table 4.8 Summary of parameters for the NSGA-II in the sub-optimization process

	SO-airfoil-opt	**MO-airfoil-opt**	**SO-wing-opt**
Pop size	100	100	100
Max gen	250	250	250
Crossover operator	Real SBX	Real SBX	Real SBX
Crossover probability	0.9	0.9	0.9
Crossover coefficient	$\eta_c = 15$	$\eta_c = 15$	$\eta_c = 15$
Mutation operator	Real PM	Real PM	Real PM
Mutation coefficient	$\eta_m = 15$	$\eta_m = 20$	$\eta_m = 15$
Initial sampling	The top 100 solutions from the previous main iteration		

4.2 DCGAN generative model

The training process of the DCGAN involves using a dataset of 77 transonic airfoil from the UIUC database. Before training, these UIUC airfoils are normalized. Consequently, the synthetic airfoils generated by the DCGAN are also normalized and require transformation or denormalization via Algorithm 2. The transformed or denormalized airfoils are simply called DCGAN airfoils. The details can be found in Chapter 3, Section 3.2.3.

To demonstrate the generative capability of the DCGAN, we compare 99 airfoils generated by the DCGAN + Algorithm 2 and 99 airfoils obtained through Latin hypercube sampling (LHS). The 99 DCGAN airfoils are produced by the G model using 99 x 100 noisy inputs, while the 99 LHS airfoils result from directly perturbing the FFD control points. These sets of 99 DCGAN and LHS airfoils are presented in Figure 4.1.

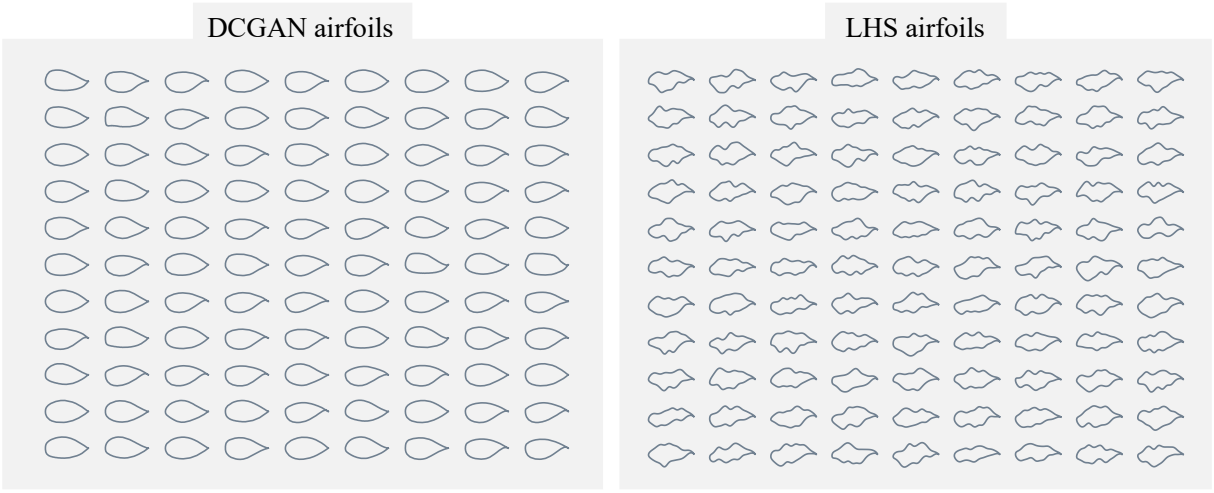


Figure 4.1 Comparison of DCGAN and LHS airfoils

The above figure illustrates a notable observation: the DCGAN-generated airfoils exhibit smoother shapes compared to the LHS-generated airfoils, which display irregular shapes. However, it should be noted that the DCGAN airfoils are influenced by the training airfoils (UIUC transonic airfoils), which can result in limited variability. In contrast, the LHS airfoils provide a broader range of flexibility, covering a vast design space.

4.3 Geometric filter score constraint determination

The CNN-based geometric filter is trained using a separate dataset consisting of 500 DCGAN samples and 500 LHS samples, which are distinct from the initial samples. The DCGAN samples are considered realistic shapes and assigned a score of 1, while the LHS samples are regarded as having abnormal shapes and assigned a score of 0. This filter is designed to map the design variables, i.e., the FFD points, to a geometric filter score, enabling quick detection of whether a given set of design variables will produce abnormal shapes. The details can be found in Chapter 3, Section 3.3.3.

After the CNN training, we apply the trained filter to our initial samples from both LHS and DCGAN, as well as the UIUC samples, for every optimization problem. The resulting score density distributions are visualized in Figure 4.2. It is worth noting that the initial samples for the single-objective airfoil optimization problem are identical with the initial samples for the multi-objective airfoil optimization problem (the sampling done in the design variable space), demonstrated by the left figure in Figure 4.2.

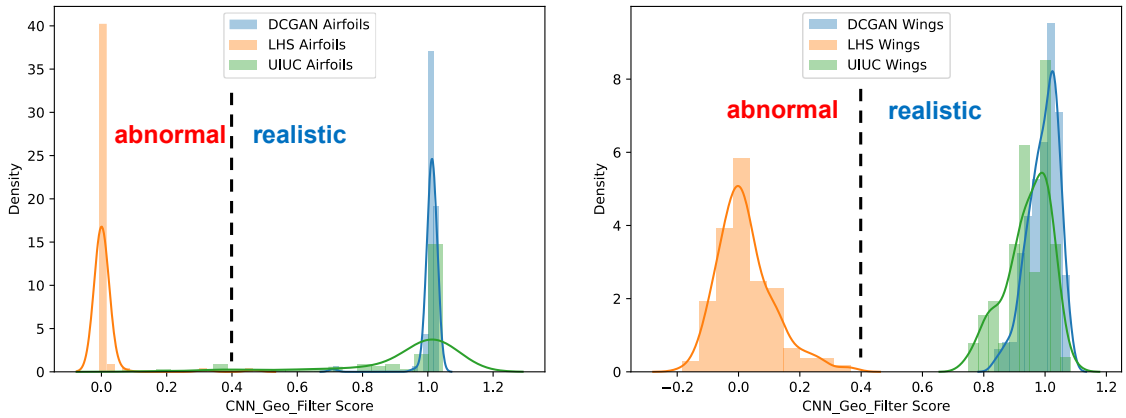


Figure 4.2 Geometric filter score distribution of initial samples: airfoil (left) and wing (right)

The above figure illustrates the effectiveness of the geometric filter in distinguishing samples with abnormal shapes from those with realistic shapes. A geometric filter score close to zero indicates abnormal shapes, while a value close to one indicates realistic shapes. Based on the above score density distribution, a threshold value of 0.4 is selected as it demonstrates effective separation between the DCGAN (realistic) and LHS (abnormal) samples, both in the airfoil and wing cases.

To incorporate this filter into the sub-optimization process, a simple analytical constraint of $S \geq 0.4$ can be integrated. This constraint ensures that only samples with geometric filter scores higher than 0.4 are considered. Since all the realistic shapes from the DCGAN initial samples and the UIUC samples possess geometric filter scores above 0.4, this constraint effectively filters out the samples with abnormal shapes, see the GF scores for the infill samples in Appendix (A1). Importantly, this evaluation is performed analytically and does not impose a significant computational burden.

4.4 Single-objective airfoil optimization

4.4.1 Problem formulation

In the single-objective airfoil optimization, the objective is to minimize the drag coefficient C_D with respect to the displacement of 20 FFD control points. Additionally, an expensive-to-evaluate constraint is imposed on the lift coefficient C_L to ensure it remains equal to or greater than 0.5. It is important to note that there exists a trade-off relationship between

C_D and C_L. By minimizing C_D, C_L is also reduced. Hence the constraint on C_L is imposed to ensure it remains within a certain threshold. The problem also incorporates geometric constraints such as minimum FFD area and fixed leading and trailing edge constraints. The geometric constraints are analytically evaluated. Table 4.9 provides a concise summary of this problem formulation. This problem is considered low dimensional due to the low number of design variables. Usually, for a surrogate-based optimization, a problem is considered high-dimensional if it involves more than 100 design variables.

The initial geometry used as a baseline for the optimization process is the RAE2822 transonic airfoil. This baseline is then embedded by 20 FFD control points. Figure 4.3 illustrates this baseline geometry embedded by the FFD points. Detailed information about the positions of these control points and their boundaries can be found in the Appendix (A2) section. In the figure, the red dashed line represents the resulting airfoil obtained by applying the maximum perturbation to the control points, while the green dashed line represents the resulting airfoil obtained by applying the minimum perturbation to the control points.

Table 4.9 Problem formulation of a single-objective airfoil optimization

	Function/variable	Description	Quantity
Minimize	C_D	Drag coefficient	1
		Total objective function	1
With respect to	Δy	FFD control point displacements	20
		Total design variables	20
Subject to	$C_L \geq 0.5$	Lift coefficient constraint	1
	$A \geq 0.9A_{base}$	Minimum FFD area constraint	1
	$\Delta y_{TE,upper} = -\Delta y_{TE,lower}$	Fixed trailing edge constraint	1
	$\Delta y_{LE,upper} = -\Delta y_{LE,lower}$	Fixed leading edge constraint	1
		Total constraints	4

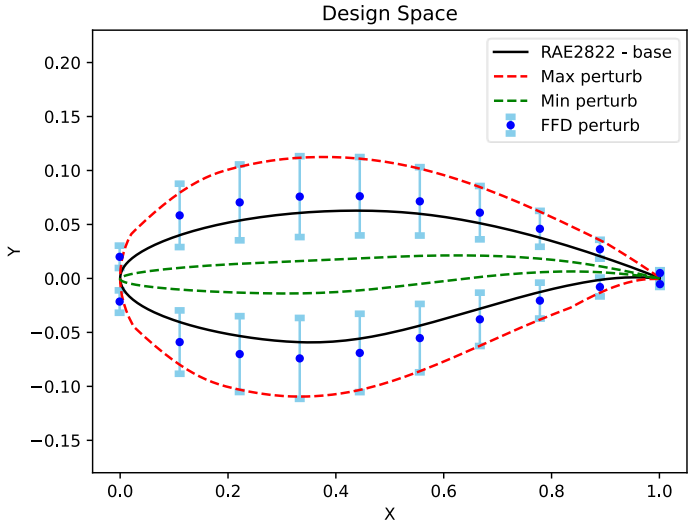


Figure 4.3 Airfoil baseline geometry and the embedded FFD control points.

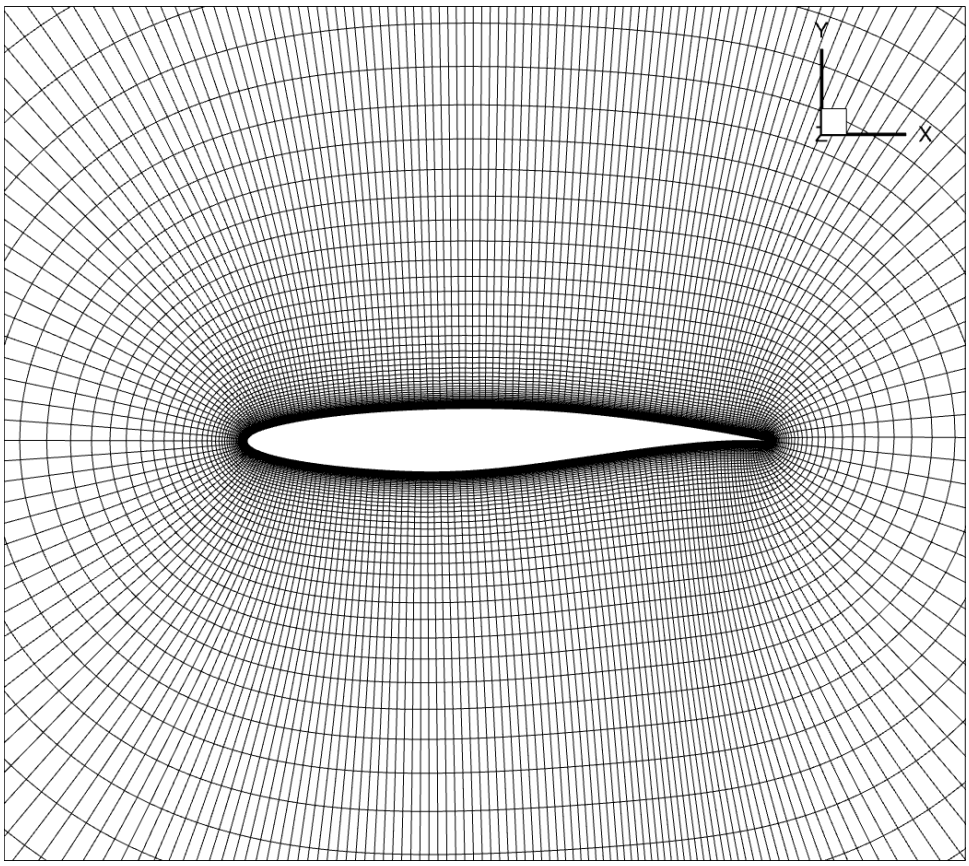


Figure 4.4 Airfoil baseline mesh.

4.4.2 Optimization history

We utilized three optimization methods to tackle this problem, starting with 100 initial samples for each method and adding one infill sample with each iteration. For plotting convenience, we introduced a penalized objective function, as expressed in Equation (4.1), which incorporates both the objective function and the constraint violation. This allowed us to conveniently represent all samples in a single figure, see Figure 4.5.

$$Penalized\ obj = 10{,}000 \cdot C_D + 1000 \cdot \max(0.0, 0.5 - C_L) + 10{,}000 \cdot \max\left(0.0, 0.9A_{FFD_{base}} - A_{FFD}\right) \quad (4.1)$$

Regarding the initial samples, a noticeable distinction can be observed in the distribution between the DCGAN samples (utilized in the DCGAN and DCGAN+GF methods) and the LHS samples. The DCGAN samples tend to occupy the lower region, indicating better aerodynamic performance since the objective is to minimize this value. The disparity in distribution arises from the inherent characteristics of each method in generating samples. The LHS method offers greater flexibility by directly manipulating the local shapes of the wing sections, which can introduce irregularities and bumps, ultimately resulting in poorer aerodynamic performance. For a more detailed analysis, please refer to Section 4.4.3.

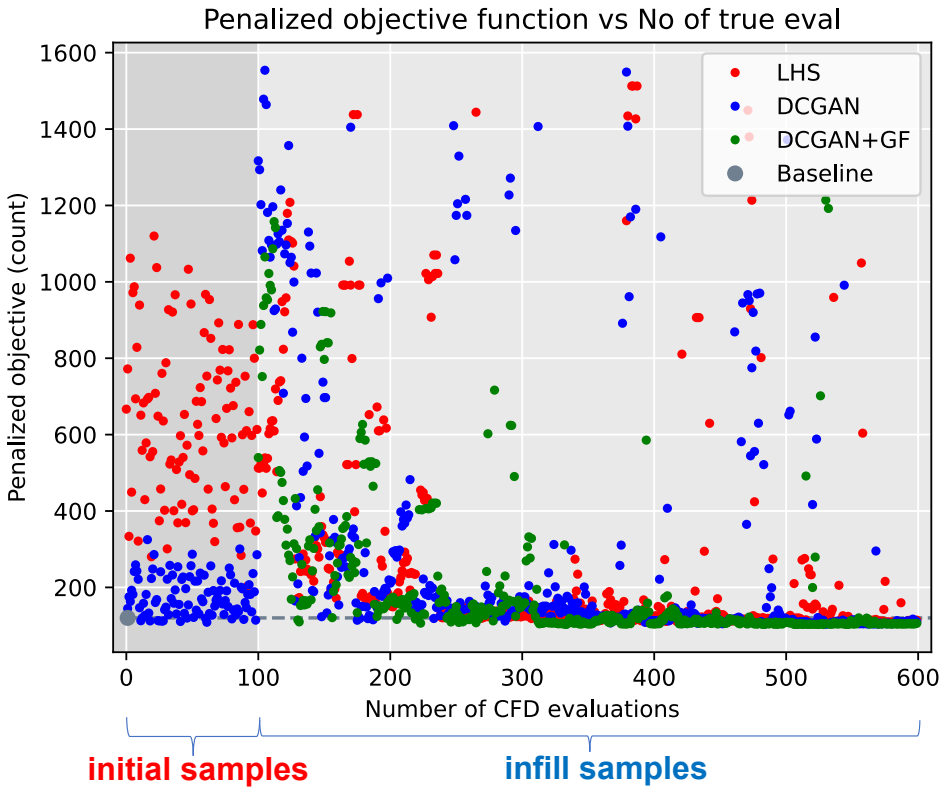


Figure 4.5 Penalized objective function for all samples in the single-obj airfoil opt.

When considering the infill samples, it becomes apparent that the LHS method demonstrates success in finding infill samples that are at least as good as, if not better than, the initial samples. In contrast, the DCGAN method often yields solutions that are worse than its initial samples. This poor performance can be attributed to the fact that the surrogate model underestimates the objective function. In other words, the surrogate model predicts that a sample will have minimal drag, but the CFD analysis proves otherwise, refer to Section 4.7

One possible explanation for this poor performance is the poor diversity among the training samples used for training the surrogate model. This argument gains support when observing the infill samples generated by the DCGAN+GF method. In the early iterations of optimization, this method also struggles to find solutions that outperform its initial samples. However, the incorporation of the geometric filter helps mitigate the occurrence of these high-region infill samples, similar to what was observed in the DCGAN method.

To conduct a more comprehensive comparison of the performance among the three methods, we plotted the optimization iteration history in Figure 4.6. This plot excludes the initial samples and instead focuses on the progression of the minimum feasible drag coefficient as a function of the number of CFD evaluations.

For the LHS method, it initially started with a solution that exhibited a high drag coefficient. However, over subsequent iterations, it gradually discovered improved solutions with reduced drag compared to the baseline. Nevertheless, it was unable to achieve competitive performance when compared to the other two methods (DCGAN and DCGAN+GF) within the first 600 CFD simulations.

In contrast, the DCGAN method encountered difficulties in finding improvements until the 370th iteration. This delay in convergence can be attributed to the poor surrogate model's accuracy, leading to abnormal infill samples. Remarkably, the incorporation of the geometric filter in the DCGAN+GF method significantly accelerated the convergence process. It began to exhibit improvements as early as the 210th iteration and achieved convergence by the 340th iteration. This enhanced performance can be attributed to the capability of the geometric filter to filter out abnormal infill samples. Another explanation can be that the geometric filter shrinks the design space, thanks to the incorporation of the analytical constraint in the sub-optimization process. In other words, the DCGAN+GF method focuses only on not only good feasible solutions, see Figure 4.7. Readers can play around with the optimization data and their visualization by visiting https://airfoil-single-obj-opt-alfi.onrender.com/ [61].

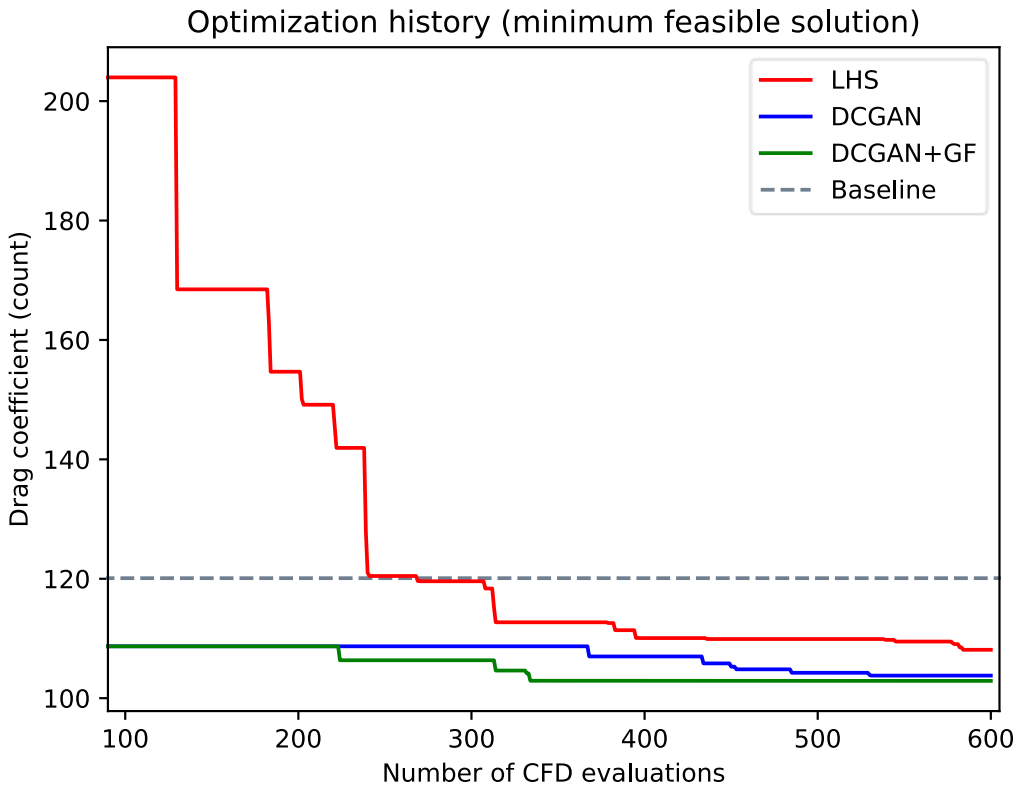


Figure 4.6 Minimum feasible drag coefficient history in single-obj airfoil opt.

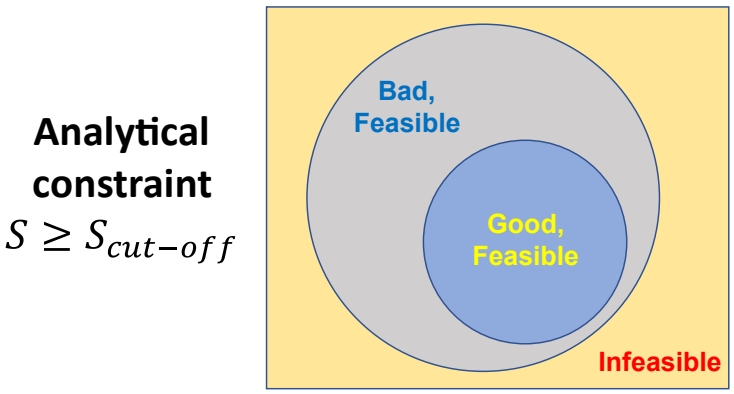


Figure 4.7 The geometric filter shrinks the design space.

4.4.3 CFD visualization

In this section, we present visualization of CFD simulations, specifically contour plots of the Mach number. Figure 4.8 demonstrates that the LHS initial samples experience significant flow separation due to the presence of bumps and irregularities on their surfaces. This outcome is expected since the LHS method offers high flexibility and wide coverage of the design space when modifying local airfoil shapes. This flexibility allows for the creation of irregular airfoil shapes that can lead to flow separation.

In contrast, the DCGAN initial samples exhibit milder flow separation, as illustrated in Figure 4.9. This smoother surface contributes to reduced drag, which explains why the penalized objective function of the DCGAN initial samples is distributed in the lower region, as observed in Figure 4.5. These findings indicate that DCGAN-based sampling has the capability to generate realistic airfoil samples that demonstrate good aerodynamic performance. However, it is important to note that this comes at the cost of sacrificing shape flexibility.

Overall, these results highlight the trade-off between shape flexibility and aerodynamic performance. While the LHS method offers greater design space coverage and flexibility, it often leads to irregular airfoil shapes with significant flow separation. On the other hand, the DCGAN-based sampling approach produces smoother airfoil shapes with milder flow separation, resulting in improved aerodynamic performance.

In Figure 4.10, we visualize the optimal solutions in comparison to the baseline. The LHS method was able to find a solution that reduces the drag by 9.9% compared to the baseline. On the other hand, the DCGAN method achieved a greater drag reduction of 13.5%. However, the best optimal solution was obtained using the DCGAN+GF method, which achieved a remarkable 14.3% drag reduction from the baseline. It is worth noting that although the difference between the best solutions found by DCGAN+GF and DCGAN is relatively small, the DCGAN+GF method converges to this solution much earlier in the iteration process, as depicted in Figure 4.6. This highlights the efficiency of the DCGAN+GF method in finding highly competitive solutions within a shorter iteration. Although the computational time might increase in the case of DCGAN+GF since we have to train a CNN geo filter and a DCGAN generative model prior to solving the optimization. This is discussed in Section 4.8.

In conclusion, the DCGAN+GF method outperforms both the DCGAN and LHS methods by obtaining more optimal solutions within a significantly shorter iteration period.

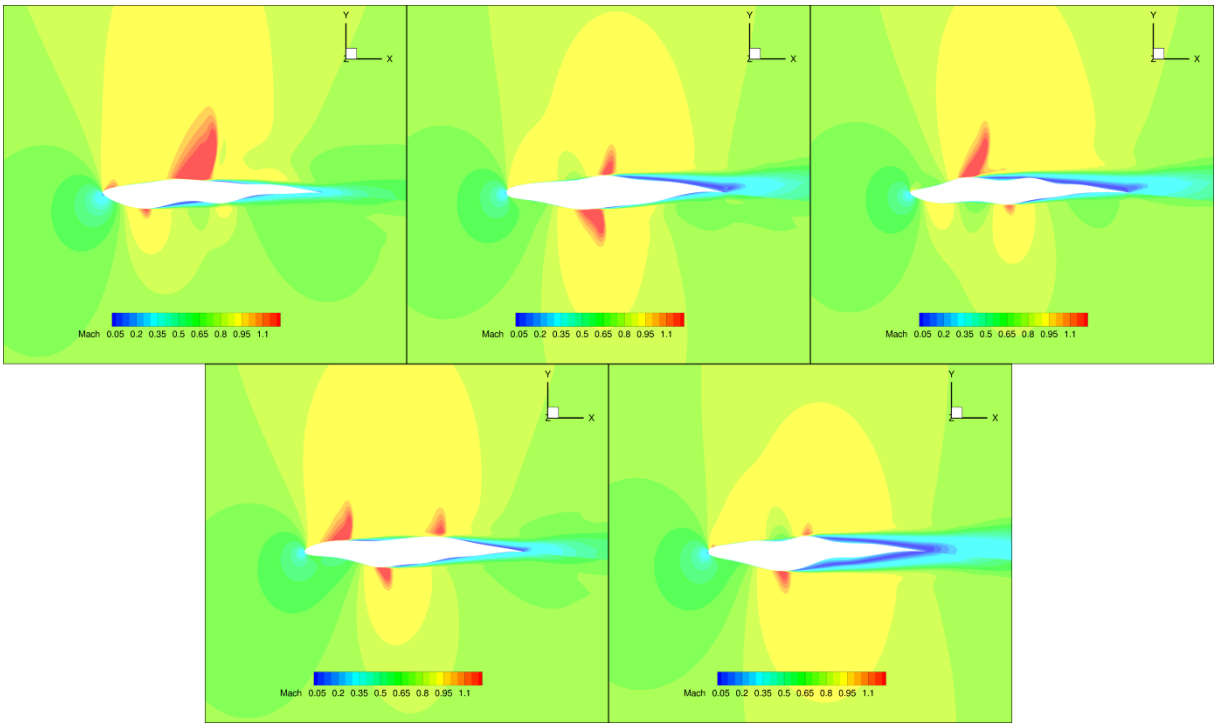


Figure 4.8 CFD visualization of LHS initial samples.

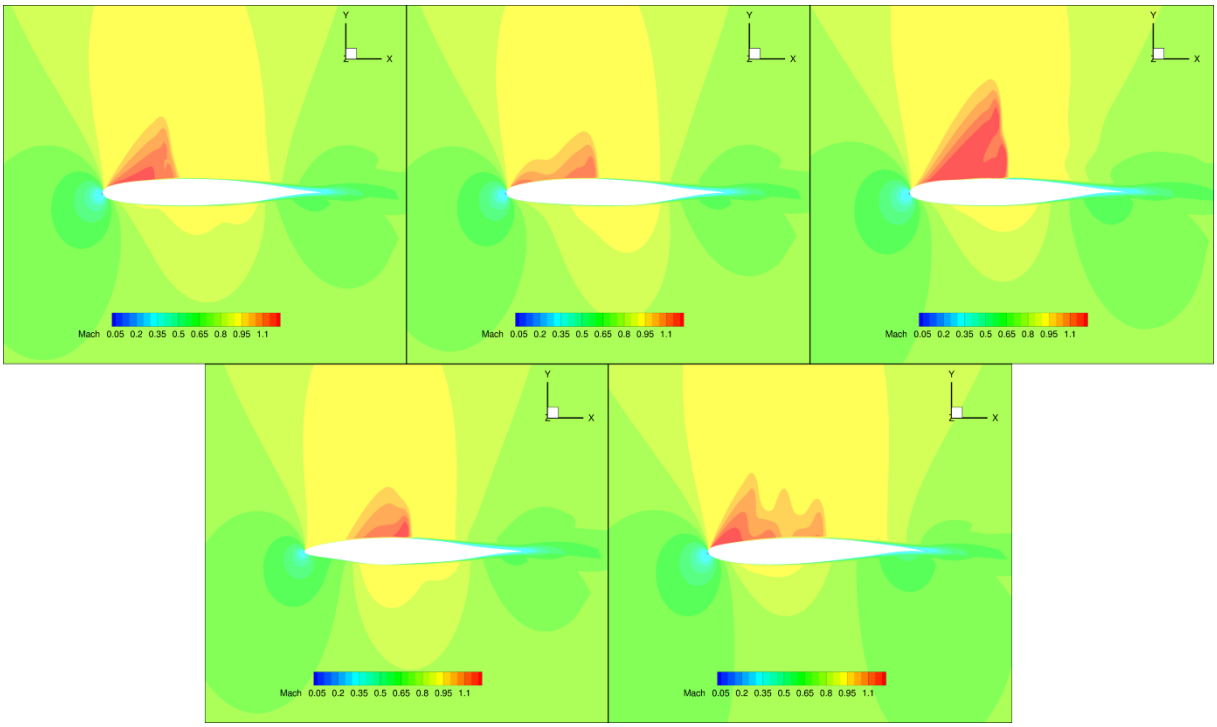


Figure 4.9 CFD visualization of DCGAN initial samples.

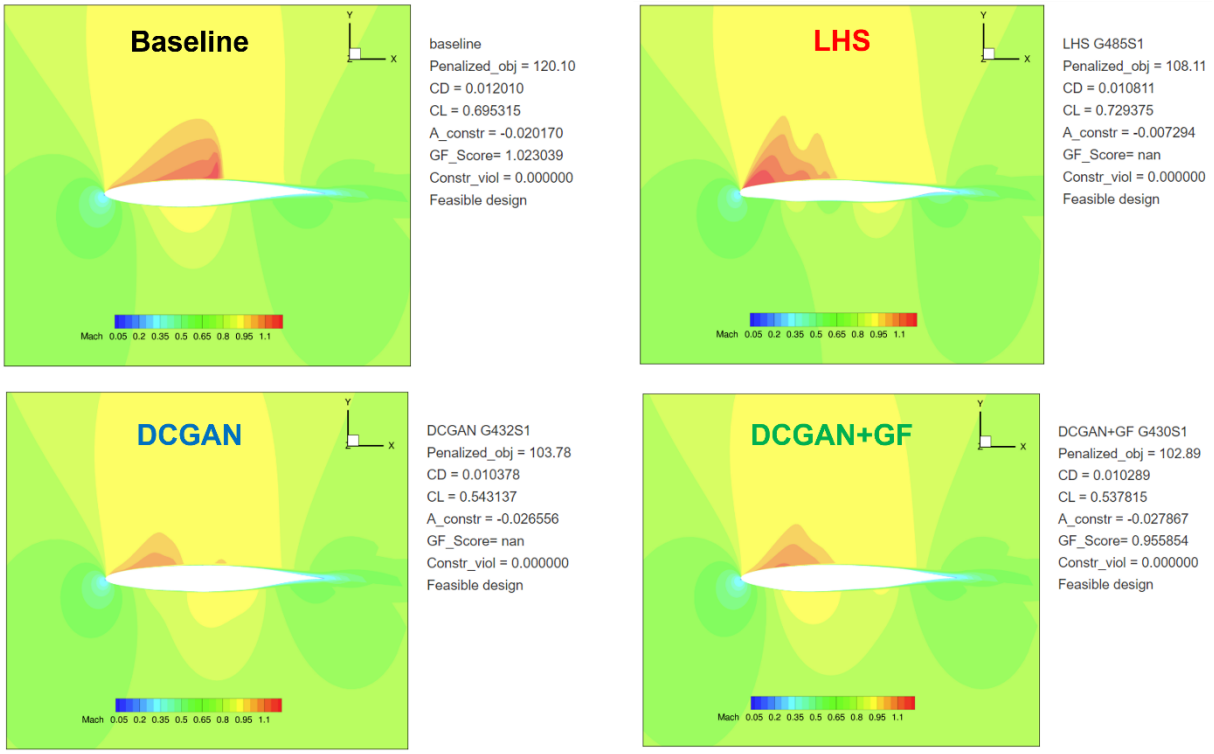


Figure 4.10 CFD visualization of baseline and optimal solutions in the single-obj airfoil opt.

Figure 4.11 displays the airfoil shapes corresponding to the obtained optimal solutions. In the LHS airfoil, drag reduction is achieved by adopting a slenderer shape compared to the baseline. The DCGAN airfoil, on the other hand, reduces drag by modifying the trailing edge to be less cambered. The DCGAN+GF method, which exhibits the lowest drag among the three, achieves drag reduction by both decreasing the camber in the TE and making the LE rounder. This design alteration ensures that the flow remains attached to the airfoil throughout, preventing flow separation downstream.

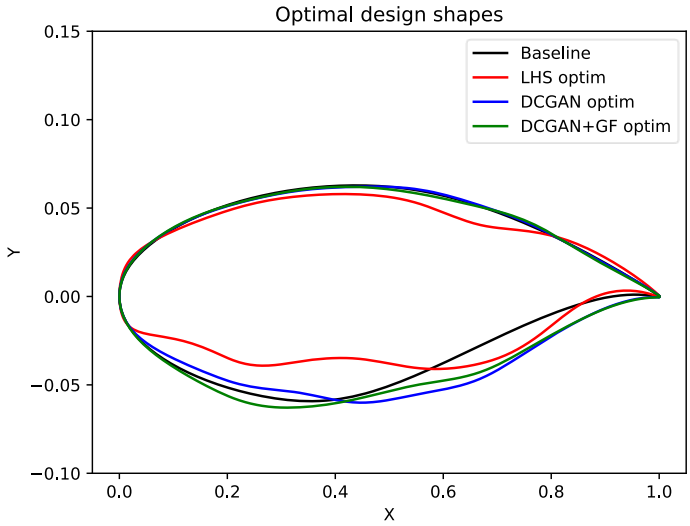


Figure 4.11 Optimal airfoil shapes in the single-objective airfoil optimization.

4.5 Multi-objective airfoil optimization

4.5.1 Problem formulation

In this problem, the objective is to optimize two conflicting objectives simultaneously: minimizing the drag coefficient C_D and maximizing the lift coefficient C_L. For the convenience of optimization problem formulation, the objective of maximizing C_L is expressed as minimizing the negative lift coefficient $-C_L$. The design variables and geometric constraints remain the same as in the single-objective airfoil optimization case.

The baseline geometry for this multi-objective airfoil optimization remains the RAE2822 airfoil, utilizing the same initial mesh as in the single-objective case. The flow conditions also remain unchanged. The primary difference in this problem lies in the treatment of the lift coefficient C_L, which now serves as an objective rather than a constraint. Given these similarities, the initial samples for this problem are directly copied from the single-objective airfoil optimization case. This is feasible because the initial sampling is performed in the design variable space, which does not impact the objective space. A summary of this problem formulation can be found in Table 4.10.

Table 4.10 Problem formulation of a multi-objective airfoil optimization

	Function/variable	Description	Quantity
Minimize	C_D	Drag coefficient	1
	$-C_L$	Negative lift coefficient	1
		Total objective functions	2
With respect to	Δy	FFD control point displacements	20
		Total design variables	20
Subject to	$A \geq 0.9A_{base}$	Minimum FFD area constraint	1
	$\Delta y_{TE,upper} = -\Delta y_{TE,lower}$	Fixed trailing edge constraint	1
	$\Delta y_{LE,upper} = -\Delta y_{LE,lower}$	Fixed leading edge constraint	1
		Total constraints	3

4.5.2 Optimization history

Given that this problem involves two conflicting objectives, we plotted them in the objective space with the negative lift coefficient $-C_L$ plotted on the x-axis and the drag coefficient C_D on the y-axis, as depicted in Figure 4.12. All feasible solutions are represented by black dots on the plot. The initial solutions are indicated by hollow maroon circles, while the non-dominated solutions (also referred to as optimal solutions) are denoted by solid red circles. It is worth noting that the optimal solutions are positioned in the lower left direction on the plot. This is attributed to the fact that the objective is to minimize both C_D and $-C_L$.

In this problem, the initial samples are directly copied from the single-objective case. As we have previously discovered in Section 4.4.3, the LHS initial samples exhibit poorer aerodynamic performance compared to the DCGAN initial samples due to their irregular shapes. This observation is further supported in Figure 4.12. It is evident that the majority of LHS initial samples are positioned significantly distant from the non-dominated solutions, residing in the low-lift high-drag region ($0.0 \leq C_L \leq 0.6$, $0.02 \leq C_D \leq 0.07$). In contrast, the DCGAN initial samples are in close proximity to the non-dominated solutions. This close proximity to the optimal solutions enables easier convergence for the DCGAN and DCGAN+GF methods when compared to the LHS method.

Regarding the infill samples, the LHS method manages to gradually discover solutions that are better than its initial samples. On the other hand, the DCGAN method often encounters solutions that are worse than its initial samples, specifically in the low-lift region ($0.0 \leq C_L \leq 0.4$). The reason for this is because the surrogate model in the DCGAN method has never seen those regions and it suffers from underestimation, see Section 4.7. However, the introduction of geometric filter in the DCGAN+GF method aims to minimize the occurrence of these suboptimal infill samples. As a result, it is observed that the infill samples obtained using the DCGAN+GF method exhibit fewer points in the low-lift region compared to those generated by the DCGAN method. This indicates the success of the geometric filter in mitigating the production of undesirable infill samples and improving the overall quality of the solutions, particularly in the low-lift region.

To further compare the performance of the three methods, we utilized a hypervolume (HV) indicator, which serves as a performance measure for multi-objective optimization problems. The HV indicator quantifies both the diversity and proximity of optimal solutions found by an algorithm.

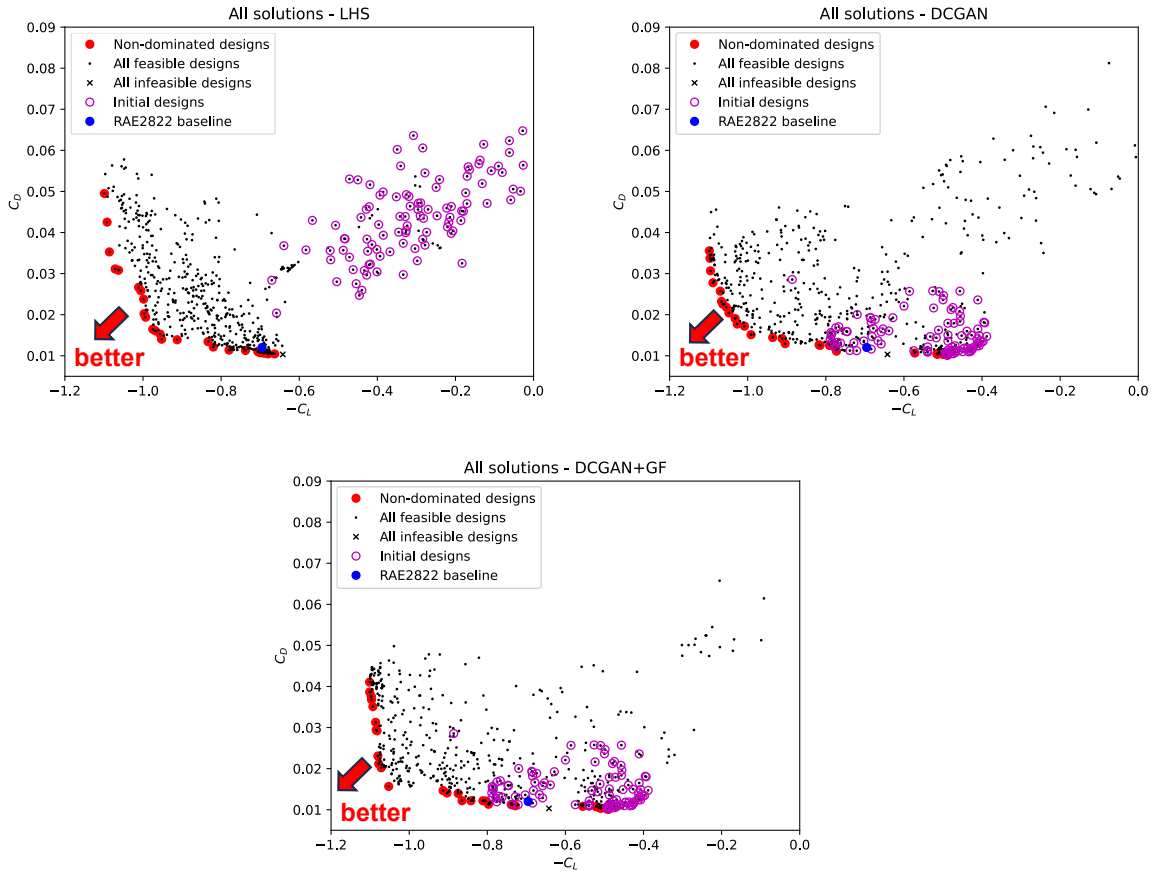


Figure 4.12 Objective space results in the multi-objective airfoil optimization problem.

By utilizing reference points, the HV indicator calculates the hatched area, as illustrated in Figure 4.13. In the context of minimizing both objectives, a larger hatched area is desirable, indicating a higher value of HV and better performance.

In Figure 4.14, we present the progression of the HV value for all the three optimization methods. It is evident that all three methods demonstrate a close convergence to a value of approximately 0.65. Furthermore, this figure reinforces our confidence in the superiority of the DCGAN+GF method, as it achieves faster convergence compared to the other two methods. This finding solidifies the assertion that the DCGAN+GF method outperforms the other two methods in terms of optimization convergence speed.

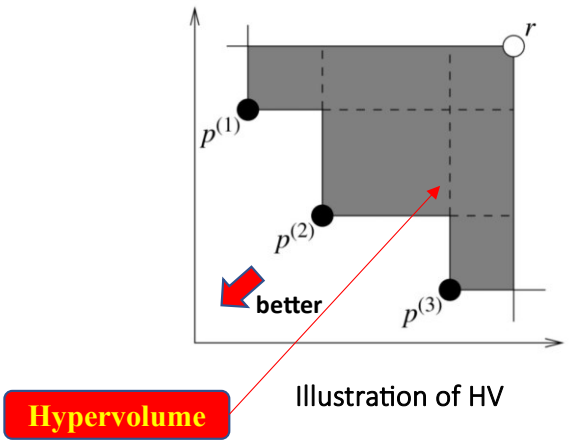


Figure 4.13 Hypervolume illustration.

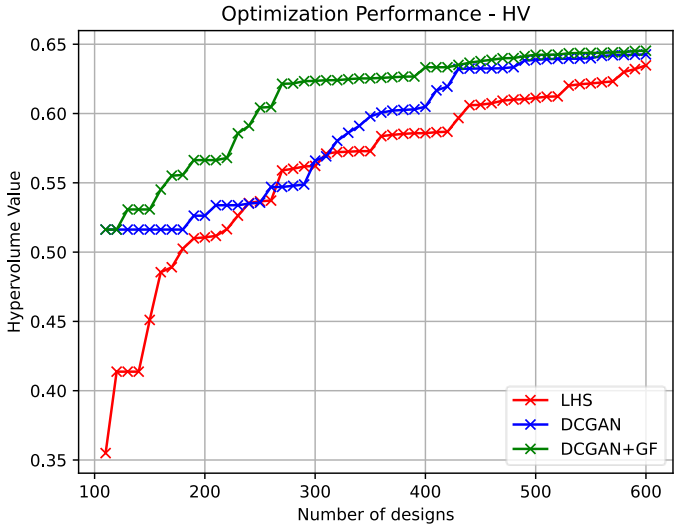


Figure 4.14 HV performance history in the multi-objective airfoil optimization problem.

4.5.3 CFD visualization

Figures 4.16-4.18 present visualizations of CFD simulations for different samples in the objective space. We specifically focus on five types of samples: the baseline, one of the initial samples, two extreme solutions, and one sample that dominates the baseline (the true optimal solution). The baseline, located in the low drag region, exhibits a drag coefficient C_D value of 0.0495. The LHS initial sample demonstrates severe flow separation, while the DCGAN initial sample experiences milder flow separation, as described in detail in Section 4.4.3.

The extreme solutions consist of the non-dominated solution with the highest lift and lowest drag. Airfoils with the highest lift coefficient possess significant shock formations on the upper side, reducing the pressure above the airfoil and generating more lift. Conversely, airfoils with the lowest drag exhibit mild shock waves, effectively reducing wave drag. The baseline-dominating airfoils surpass the baseline by not only achieving lower drag, but also generating higher lift compared to the baseline. It is important to note that there is a trade-off relationship between drag and lift; reducing drag usually results in decreased lift. All these samples and their corresponding aerodynamic performance are summarized in Table 4.11, providing an overview of their characteristics in terms of drag and lift coefficients.

The values $\%C_D$ and $\%C_L$ represent the percentage difference from the baseline values drag and lift coefficients, respectively. In the visualizations, the color red indicates unfavorable conditions, such as higher drag and lower lift compared to the baseline. Conversely, the color green represents favorable conditions, such as lower drag and higher lift compared to the baseline. These color-coded indications allow for a clear understanding of the relative importance of different samples in terms of their deviation from the baseline values of drag and lift coefficients. Now, it becomes clear why the optimal solutions have green colors in both their $\%C_D$ and $\%C_L$, signifying their superiority over the baseline in terms of both drag and lift.

Table 4.11 Summary of aerodynamic performance for various airfoil samples

Method	**Feature**	**Name**	C_D	C_L	$\%C_D$	$\%C_L$
LHS	Initial	G1S22	0.0648	0.0277	+440%	-96%
	High lift	G51S5	0.0495	1.0989	+313%	+58%
	Low drag	G51S10	0.0104	0.6628	-13%	-5%
	Optimal	G25S10	0.0110	0.7054	-8%	+1%
DCGAN	Initial	G1S87	0.0257	0.4566	+114%	-34%
	High lift	G47S9	0.0355	1.0983	+196%	+58%
	Low drag	G40S10	0.0102	0.4874	-15%	-30%
	Optimal	G50S8	0.0111	0.7728	-8%	+11%
DCGAN+GF	Initial	G1S87	0.0257	0.4566	+114%	-34%
	High lift	G44S7	0.0411	1.1013	+243%	+58%
	Low drag	G49S5	0.0101	0.4887	-16%	-30%
	Optimal	G22S3	0.0110	0.7281	-8%	+5%
Baseline			0.0120	0.6953		

In Figure 4.15, we can see the geometric features of the airfoil shapes in comparison to the baseline. Starting with the initial airfoil shapes, it is evident that the LHS airfoil exhibits a rough and bumpy surface, while the DCGAN airfoil appears smoother. Regarding the high lift airfoils, they generally exhibit increased camber throughout the length of the airfoil and a more curved downward (drooped) leading edge (LE). This drooped LE design helps to delay flow separation and enhance lift. In contrast, the low drag airfoils show some variations. The LHS airfoil maintains a similar shape to the baseline at the trailing edge (TE) but features modifications from the center towards LE. On the other hand, the DCGAN and DCGAN+GF airfoils converge to a similar shape, reducing drag by lowering the camber at the trailing edge.

For the optimal airfoil shapes, they undergo small modifications to the baseline. The LHS airfoil further lowers the camber at the TE. The DCGAN and DCGAN+GF airfoils tend to increase the radius of the LE, making it more rounded, and slightly increase the camber at the TE. Readers can play around with these airfoil data, optimization data, and their CFD visualization by visiting https://airfoil-multi-obj-opt-alfi.onrender.com/ [62].

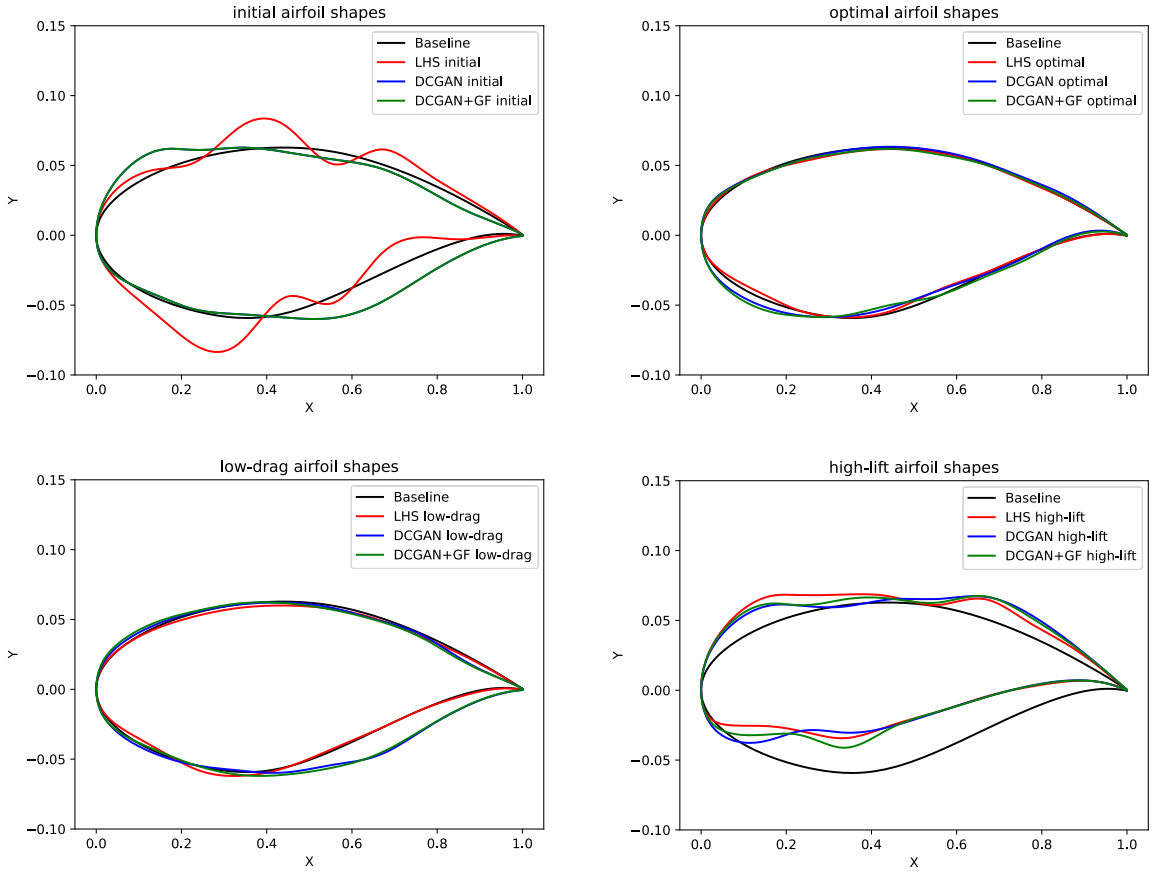


Figure 4.15 Airfoil shapes of various solutions in the multi-objective airfoil optimization.

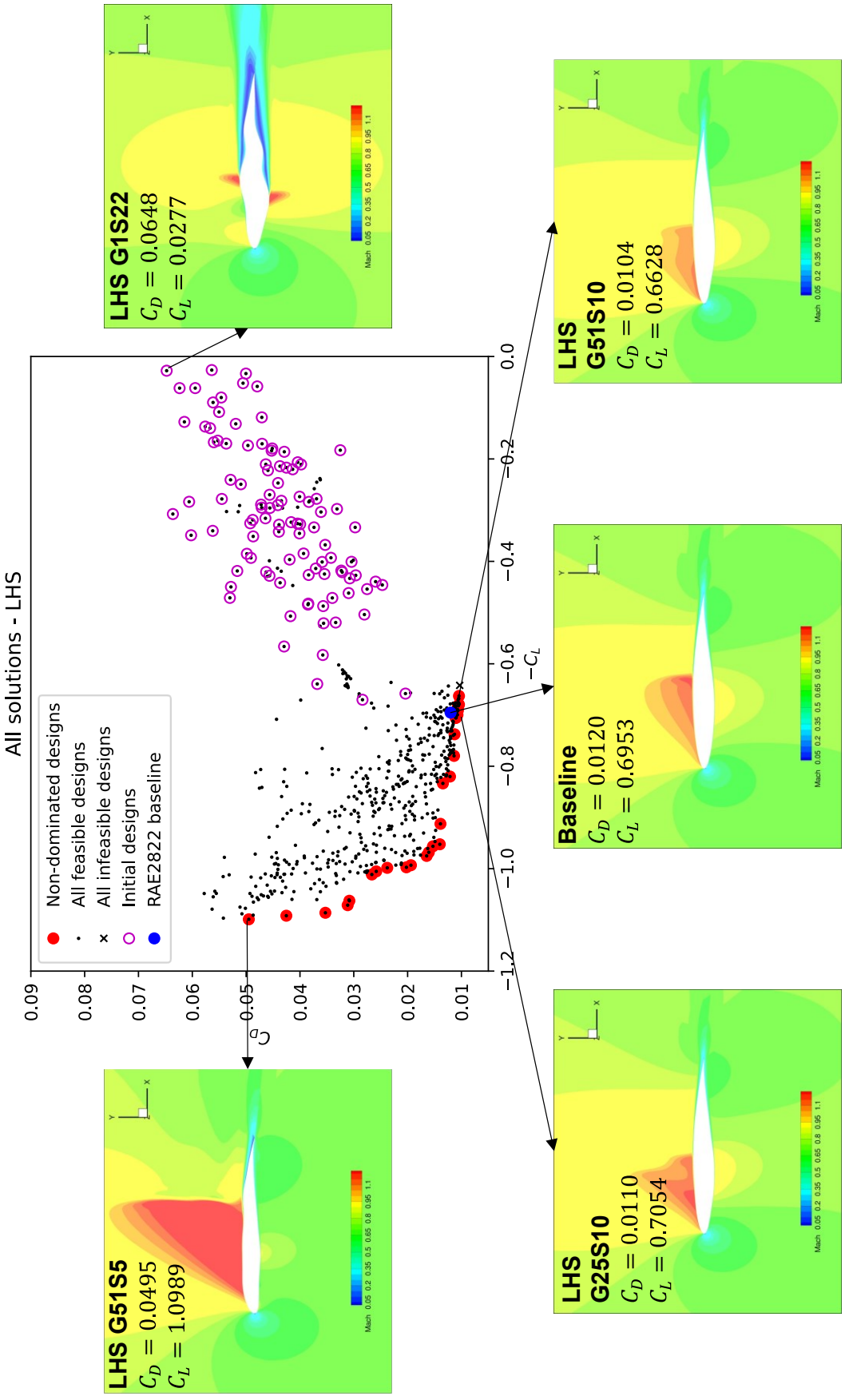


Figure 4.16 CFD visualization of different solutions by LHS in the objective space.

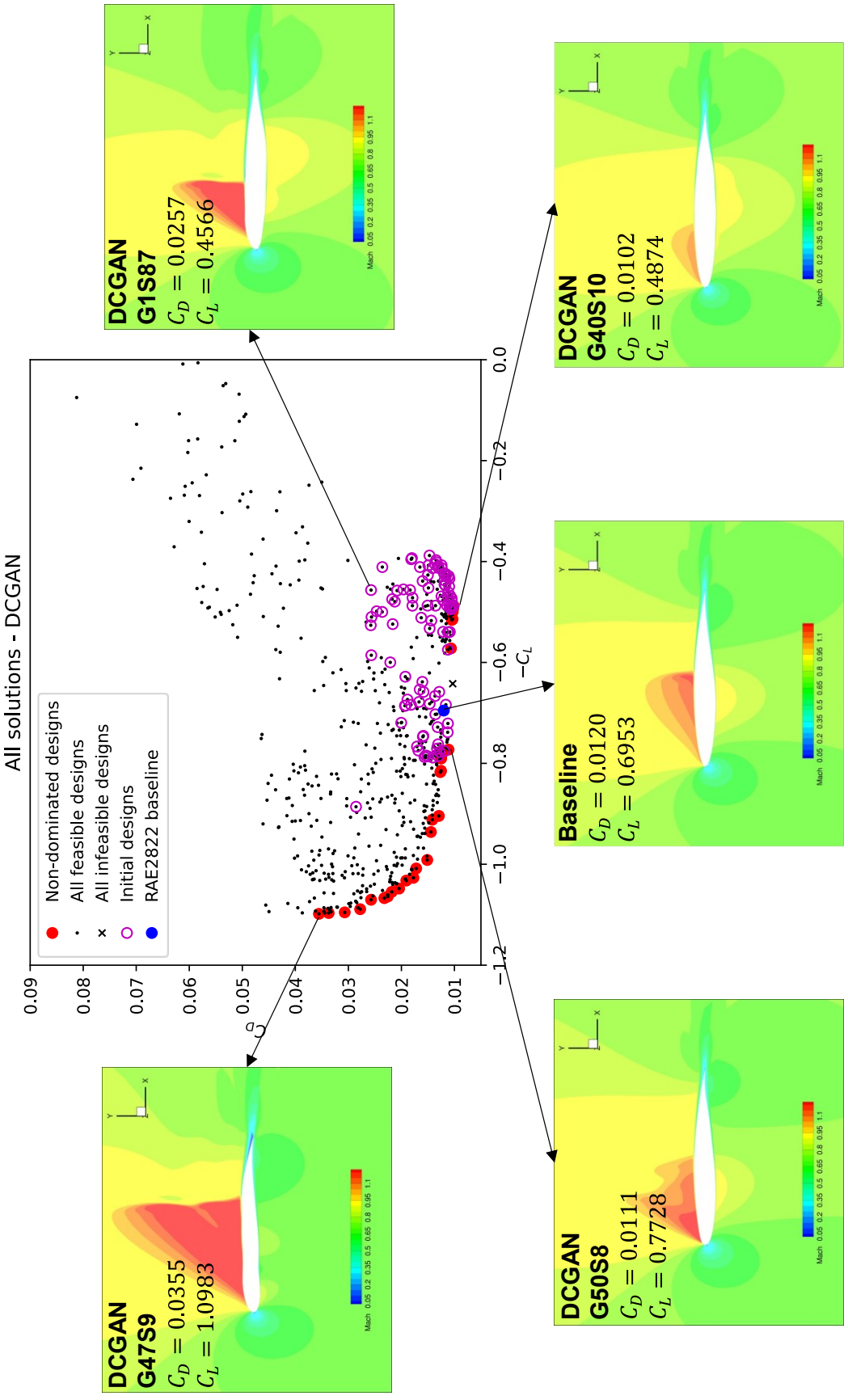


Figure 4.17 CFD visualization of different solutions by DCGAN in the objective space.

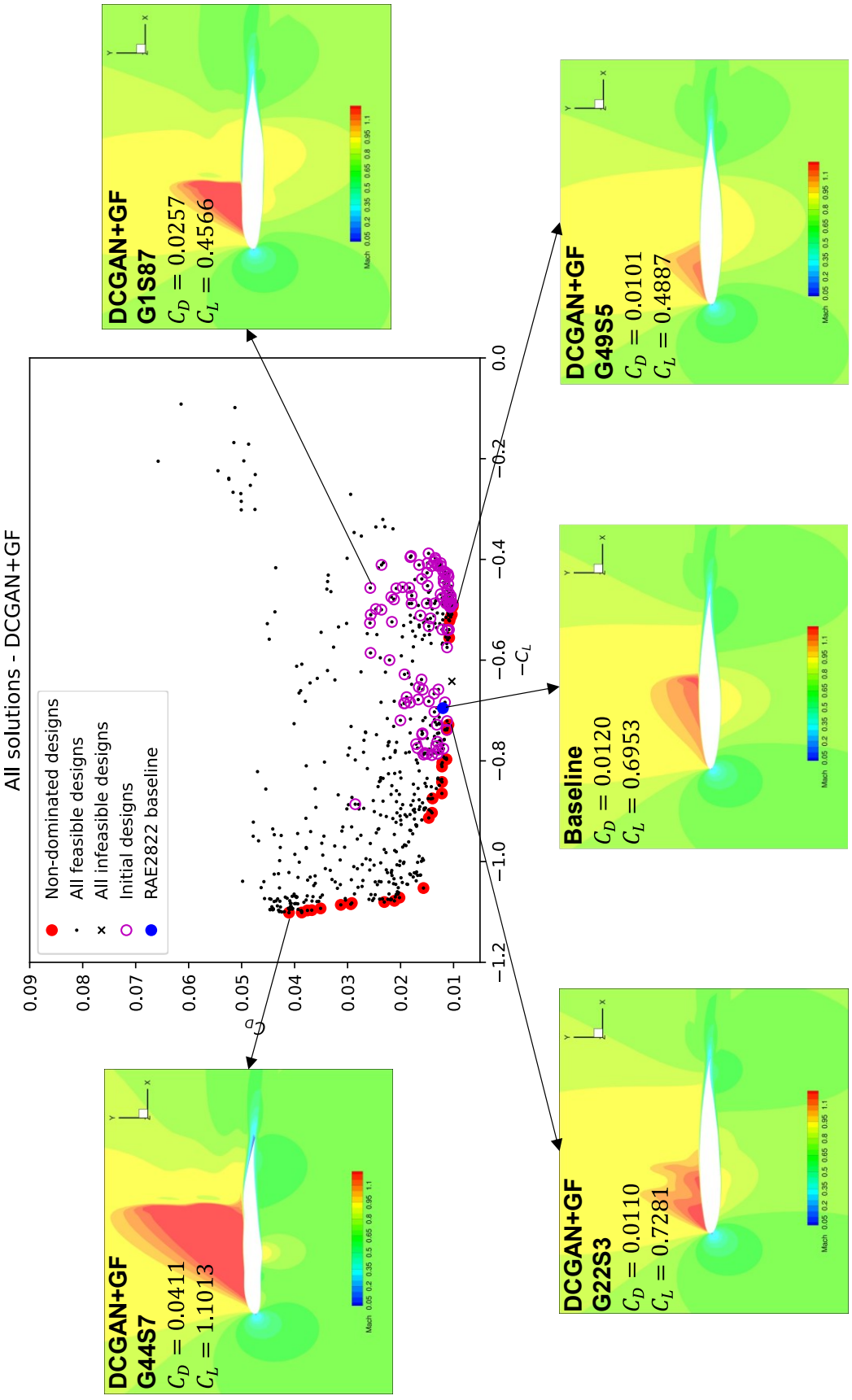


Figure 4.18 CFD visualization of different solutions by DCGAN+GF in the objective space.

4.6 Single-objective wing optimization

4.6.1 Problem formulation

The objective of this problem is to minimize the drag coefficient C_D while imposing aerodynamic and geometric constraints. The design variables consist of a set of 192 FFD control points and an angle of attack. The aerodynamic constraints consist of a lift constraint and a moment constraint. The lift constraint requires the design to satisfy $C_L = 0.5$. To achieve this fixed lift, a simple secant method search is conducted on the angle of attack α for a given design candidate. The control over the design variable α lies with the CFD solver, and the optimizer does not have influence over it. Another aerodynamic constraint involves the moment coefficient in the y-direction C_{My}, which must exceed the baseline value. A geometric constraint is imposed on the FFD volume to prevent excessively thin wings. Additionally, fixed trailing edge (TE) constraints are applied to all wing sections, while a fixed leading edge (LE) constraint is applied only to the wing root to maintain a consistent incidence with the fuselage. Overall, the problem involves one objective function, 193 design variables, and 12 constraints, as summarized in Table 4.12. Due to the involvement of over 100 design variables, this problem is classified as high-dimensional, posing a significant challenge for any SBO method.

Table 4.12 Problem formulation of a single-objective wing optimization

	Function/variable	Description	Quantity
Minimize	C_D	Drag coefficient	1
		Total objective function	1
With respect to	Δz	FFD control point displacements	192
	α	Angle of attack	1
		Total design variables	193
Subject to	$C_L = 0.5$	Lift coefficient constraint	1
	$C_{My} \geq C_{My,base}$	Moment coefficient constraint	1
	$V \geq 0.8V_{base}$	Minimum FFD volume constraint	1
	$\Delta z_{TE,upper} = -\Delta z_{TE,lower}$	Fixed trailing edge constraints	8
	$\Delta z_{LE,upper} = -\Delta z_{LE,lower}$	Fixed leading edge constraint	1
		Total constraints	12

The baseline geometry for this problem is the Common Research Model (CRM) wing, which was originally proposed by Vassberg et al. [34]. The CRM wing is part of a configuration that includes a wing, body, nacelle, pylon, and horizontal tail, and its size is similar to that of a Boeing 777. The design of the CRM transonic supercritical wing is known for its well-behaved and high-performance aerodynamic characteristics, both with and without the nacelle and pylon group. However, for the purposes of this paper, we only consider the CRM wing-alone as the baseline, excluding the other groups, see Figure 4.19 for the geometry and its 8 representative wing sections. Since the CRM wing is already a well-performing design, we do not anticipate significant aerodynamic improvements from it in this study.

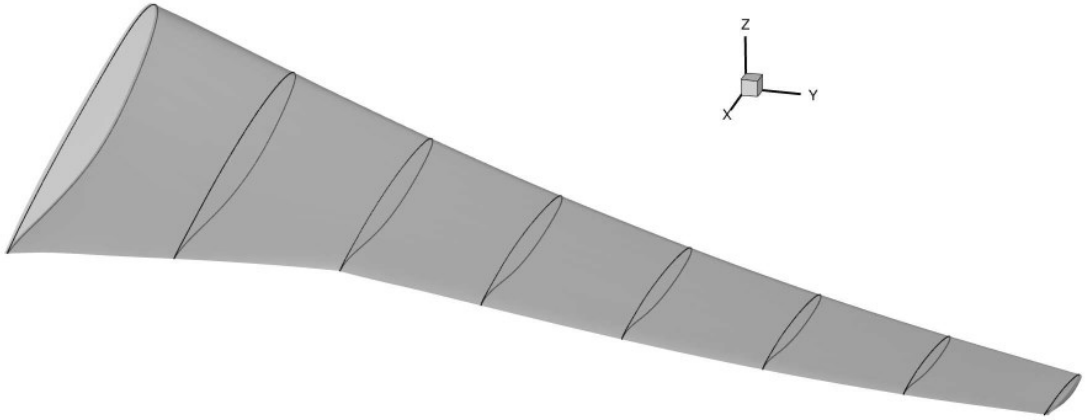


Figure 4.19 CRM wing-alone and its 8 representative wing sections.

In Figure 4.20, the initial mesh for the optimization process is displayed, consisting of 450,560 cells. This baseline mesh has been determined after conducting a grid convergence study, as detailed in Chapter 2, Section 2.4.3. The flow conditions for the entire analyses in the current optimization problem are characterized by a transonic turbulent flow, with an angle of attack a Mach number $M = 0.85$ and a Reynolds number $Re = 5E6$, with an initial angle of attack guess $\alpha = 2^{o}$. The CFD solver then performs a secant method to find the corresponding α that results in $C_L = 0.5$. The allowed maximum iteration of the α search is 3.

The problem formulation in this single-objective wing optimization is based on a benchmark case set by the AIAA Aerodynamic Design Optimization Discussion Group (ADODG) [63], with some minor modifications such as the FFD volume constraints.

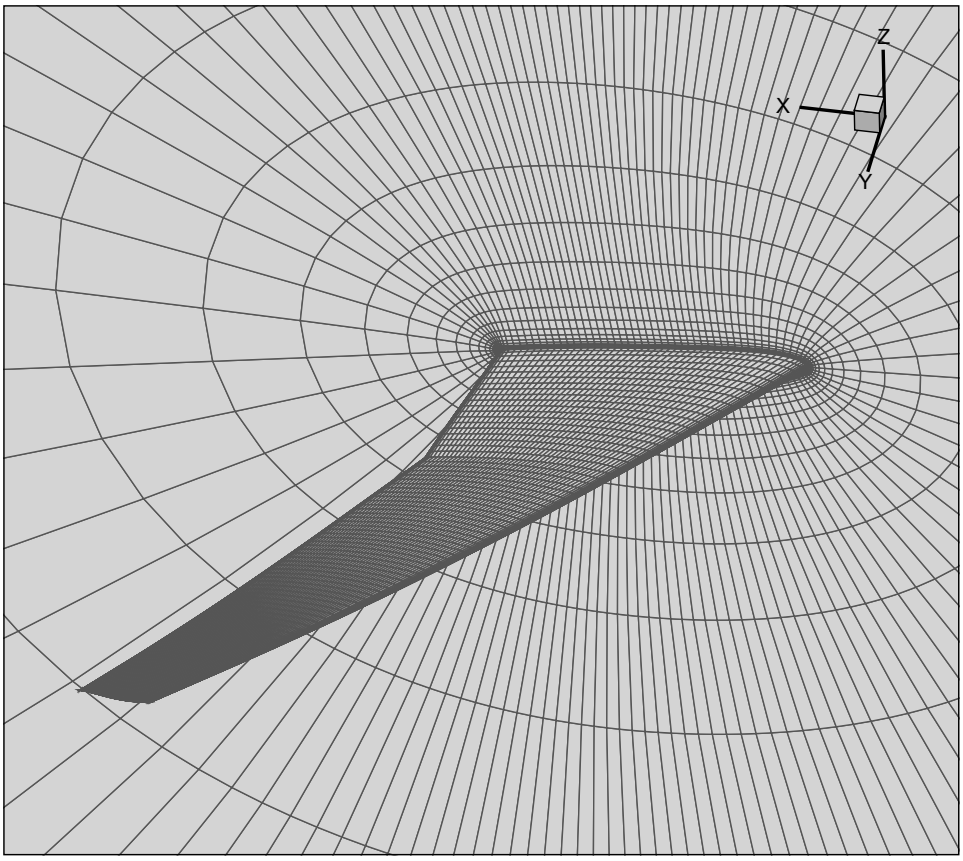


Figure 4.20 Initial baseline mesh of the CRM wing with 450,560 mesh cells.

4.6.2 Optimization history

We again utilized three optimization methods to tackle this problem, starting with 200 initial samples for each method and adding 5 infill samples with each iteration via believer sub-iteration. For plotting convenience, we introduced a penalized objective function, as expressed in Equation (4.2), which incorporates both the objective function and the constraint violation, as was done in the case of single-objective airfoil case. This allowed us to conveniently represent all samples in a single figure, see Figure 4.21.

$$Penalized\ obj = 10{,}000 \cdot C_D + 1000 \cdot |0.5 - C_L| + 1000 \cdot \left|\min\left(0.0, C_M - C_{M,base}\right)\right| \tag{4.2}$$

In Figure 4.21, we observe similar trends in the optimization results compared to the single-objective airfoil optimization discussed in Section 4.4.2. For example, a distinct distribution pattern can again be noticed between the initial samples generated by the DCGAN and the LHS methods. The DCGAN samples tend to be concentrated in the lower region,

suggesting improved aerodynamic performance since the objective is to minimize this value. The variation in this distribution can be further understood by analyzing the CFD results, as outlined in Section 4.6.3. Similar trends to the single-objective airfoil optimization are once again evident when examining the infill samples. Specifically, the LHS method continues to exhibit success in discovering infill samples that are at least as good as, if not better than, the initial samples. In contrast, the DCGAN method frequently produces solutions that perform worse than its initial samples. This behavior can be attributed to the accuracy of the surrogate model, which will be elaborated on in Section 4.7.

As mentioned earlier, the inadequate performance of the DCGAN method can be attributed to the lack of diversity among the training samples used to train the surrogate model. This explanation gains further support when examining the infill samples generated by the DCGAN+GF method. In the initial iterations of optimization, this method also faces challenges in discovering solutions that surpass its initial samples. However, the inclusion of the geometric filter aids in reducing the occurrence of these infill samples in the higher regions.

Figure 4.22, similar to Figure 4.6, illustrates the history of the minimum feasible drag coefficient as a function of the number of CFD evaluations for this problem. In the LHS method, at the second iteration, a solution with a remarkable improvement from the initial samples was discovered. This can be attributed to the inclusion of the baseline data in the initial samples of the NSGA-II sub-optimization, as discussed in Section 4.1.5 regarding the initial population setting. However, limited progress was made in subsequent iterations.

On the other hand, the DCGAN method consistently identified better solutions as the iterations progressed. It surpassed the LHS method and found a superior solution at the 310th CFD evaluation. The DCGAN+GF method exhibited faster convergence with superior solutions compared to the other two methods. It converged close to, or slightly better than, the baseline with only a one-drag count improvement. This demonstrates the challenging nature of optimizing this wing design since the baseline CRM wing is already a well-performing design. To determine the significance of this improvement, it is necessary to compare these results with those obtained by other researchers, as discussed in Section 4.6.5.

The obtained results provide further support for our claim that the DCGAN+GF method is superior in terms of optimization convergence, even in high-dimensional scenarios such as this problem. This can be attributed to the combination of the DCGAN-based sampling and the CNN-based geometric filtering techniques incorporated in the DCGAN+GF method.

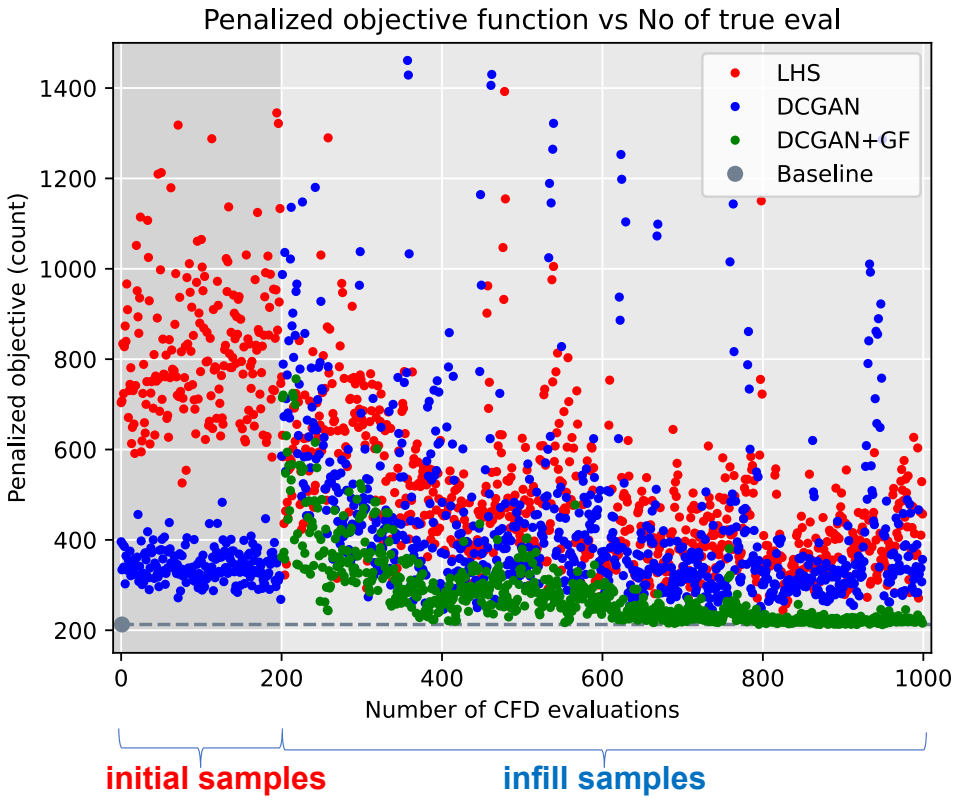


Figure 4.21 Penalized objective function for all samples in the single-obj wing opt.

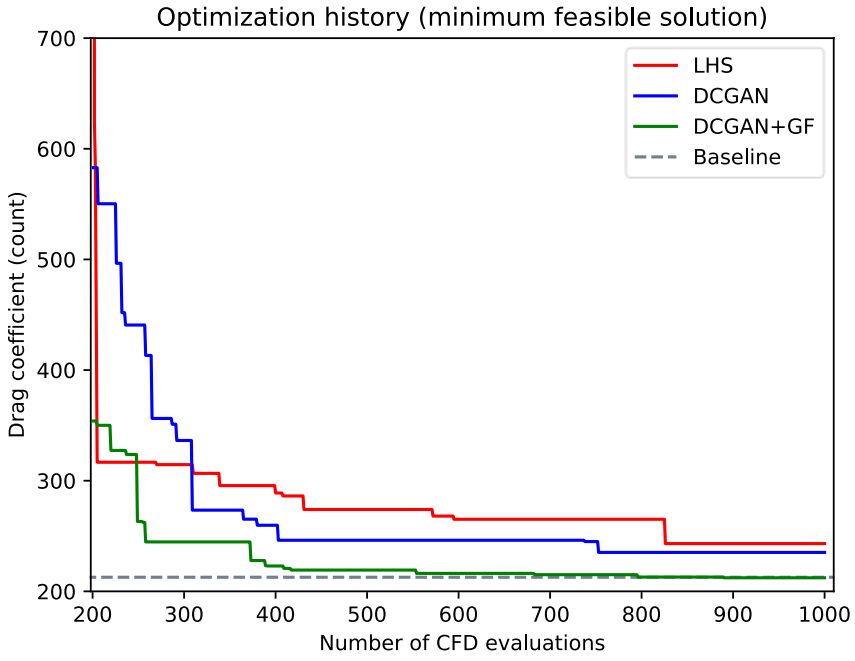


Figure 4.22 Minimum feasible drag coefficient history in single-obj wing opt.

4.6.3 CFD visualization

Figures 4.23 and 4.24 summarize the true evaluations of the best designs obtained by the three methods, as well as the baseline design. Remarkably, with the same CFD budget, the DCGAN method was able to find a solution with a drag coefficient that is 8 counts lower than the one discovered by the LHS method. The incorporation of the geometric filter (DCGAN+GF) resulted in a significant improvement of 23 counts compared to the DCGAN-only method, with a modest 0.5 count improvement (G139S5) over the baseline. We also presented another design alternative (DCGAN+GF G136S2) that outperforms the baseline by 1 drag count. However, this particular design violates the moment constraint by 0.001. These results effectively demonstrate the advantages of utilizing DCGAN and geometric filtering techniques in the design framework over the conventional LHS method.

		Feasible		best	Infeasible
	Baseline	LHS	DCGAN	DCGAN+GF	DCGAN+GF (lowest drag)
Design	-	G127S1	G112S3	G139S5	G136S2
C_D (counts)	212.745	243.160	235.179	212.261	211.788
C_L	0.500	0.499	0.500	0.500	0.500
C_{My}	-0.181	-0.168	-0.174	-0.176	-0.182
AoA	2.212°	2.286°	2.267°	2.207°	2.217°
FFD Volume	0.689	0.722	0.709	0.700	0.687
Wing Volume	0.231	0.232	0.232	0.230	0.230
		conventional		proposed	

Figure 4.23 Summary of true evaluations of optimal designs by all three methods.

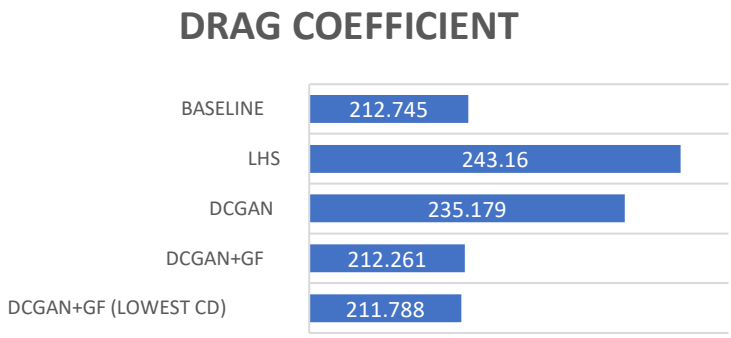


Figure 4.24 Summary of drag coefficients of optimal designs by all three methods.

With the exception of the last design, all the optimized designs successfully meet the imposed constraints. The lift constraint was satisfied by the CFD solver within only 3 iterations, with a tolerance of 0.001. It is important to note that our formulation of the moment and volume constraints differs from the definition provided by the ADODG. Our moment constraint ($CM_y \geq CM_{y,base}$) is less strict compared to the ADODG formulation ($CM_y \geq -0.170$). Consequently, the solutions obtained using the DCGAN and DCGAN+GF methods meet our constraint criteria but may not meet the criteria defined by the ADODG.

Regarding the volume constraint, instead of the wing's internal volume criterion, we utilized the FFD volume criterion ($V_{FFD} \geq 0.8\, V_{FFD,base}$), as evaluating the former is more computationally expensive than the latter. However, we discovered a strong correlation between the FFD volume and the win's internal volume, see Appendix (A3). In fact, our best designs differ only slightly in terms of wing internal volume when compared to the baseline.

The distribution difference of initial samples can be attributed to the distinct sampling techniques employed by the LHS and DCGAN methods, as discussed earlier. The LHS-based method allows for greater flexibility by directly modifying the local shapes of the wing sections. Consequently, this can introduce irregularities and bumps in the wing sections. These bumps cause the flow to separate more quickly, leading to intensified shock waves and increased drag. This phenomenon is illustrated in Figure 4.25, which presents a CFD comparison between a sample from each method. These two samples were randomly selected for comparison, providing visual evidence of the impact of the sampling techniques on the flow behavior.

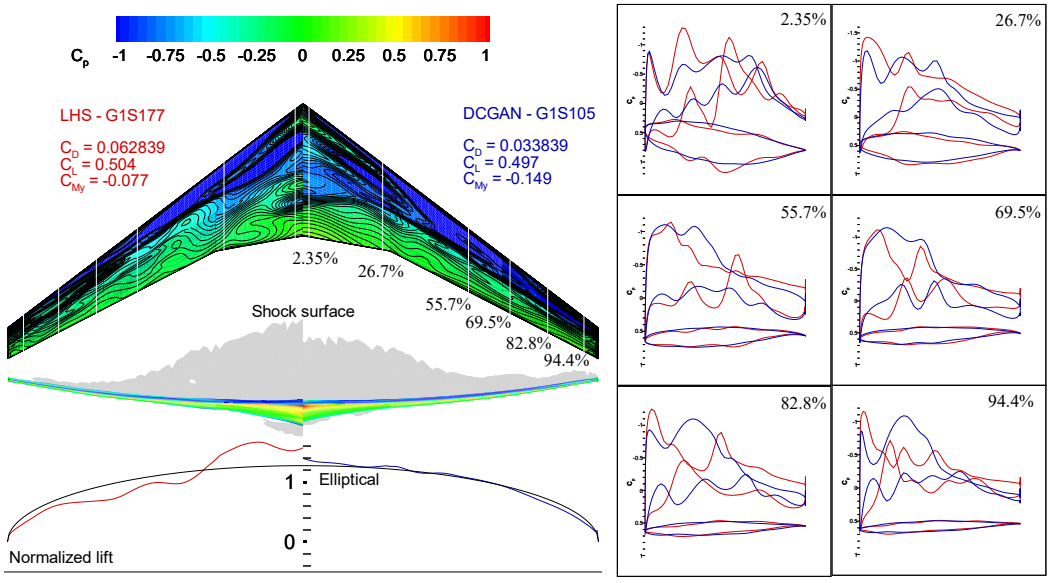


Figure 4.25 CFD comparison between one of the LHS and DCGAN initial samples.

From Figure 4.25, it is evident that the DCGAN sample exhibits a closer adherence to the elliptical normalized lift distribution compared to the LHS sample. Additionally, the DCGAN wing sections appear smoother in comparison to the LHS wing sections, leading to lower drag. This explains why the initial samples generated by the DCGAN method are distributed in a lower drag region compared to the initial samples generated by the LHS method.

Figure 4.26 and Figure 4.27 present the CFD results for the baseline design and the optimal and best solutions obtained by the DCGAN+GF method, respectively. It is observed that both solutions exhibit relatively similar wing sections. However, there are noticeable differences in the shock surface characteristics. Specifically, the shock surface of the DCGAN+GF solution appears to be less intense in the region between 0% and 45% of the wing, but not in the region between 45% and 90%. Additionally, some pressure oscillations are still evident on the surface of the DCGAN+GF solution, indicating the presence of minor bumps. Despite these differences, both solutions display comparable lift distribution profiles. However, both solutions achieved lower drag values compared to the baseline design, with a reduction of approximately 0.5 to 1 drag count. In order to provide a detailed explanation for these findings, further analysis of the 3D shock regions will be conducted shortly.

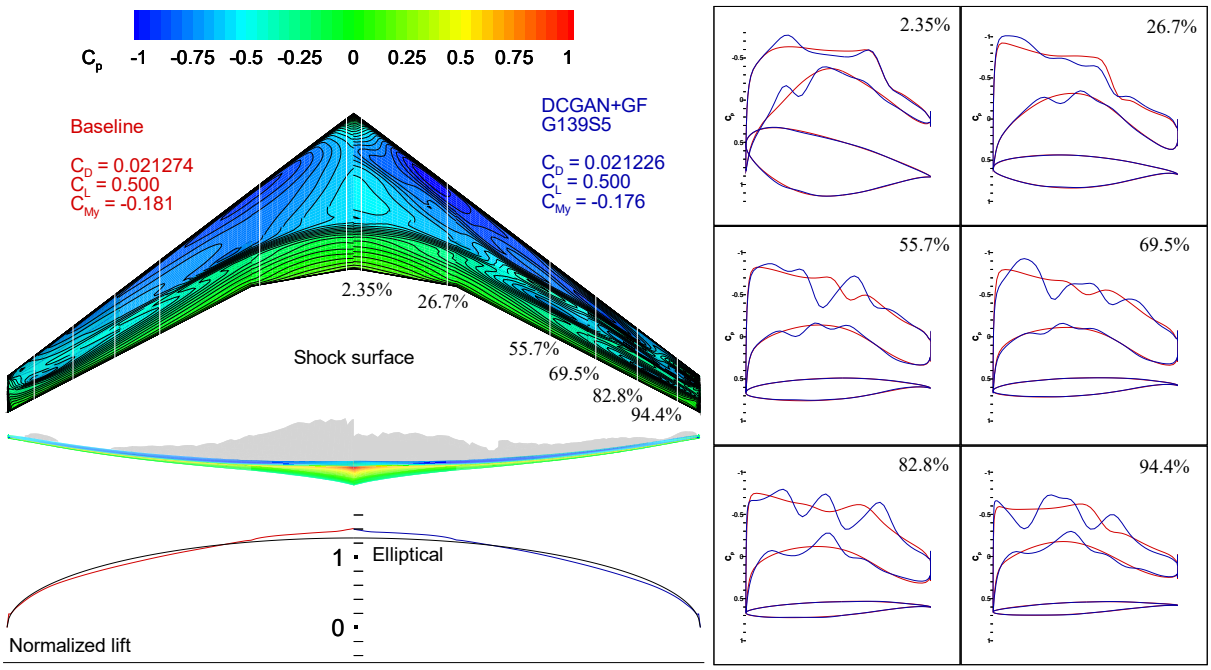


Figure 4.26 CFD comparison between the baseline and the feasible DCGAN+GF optimal sol.

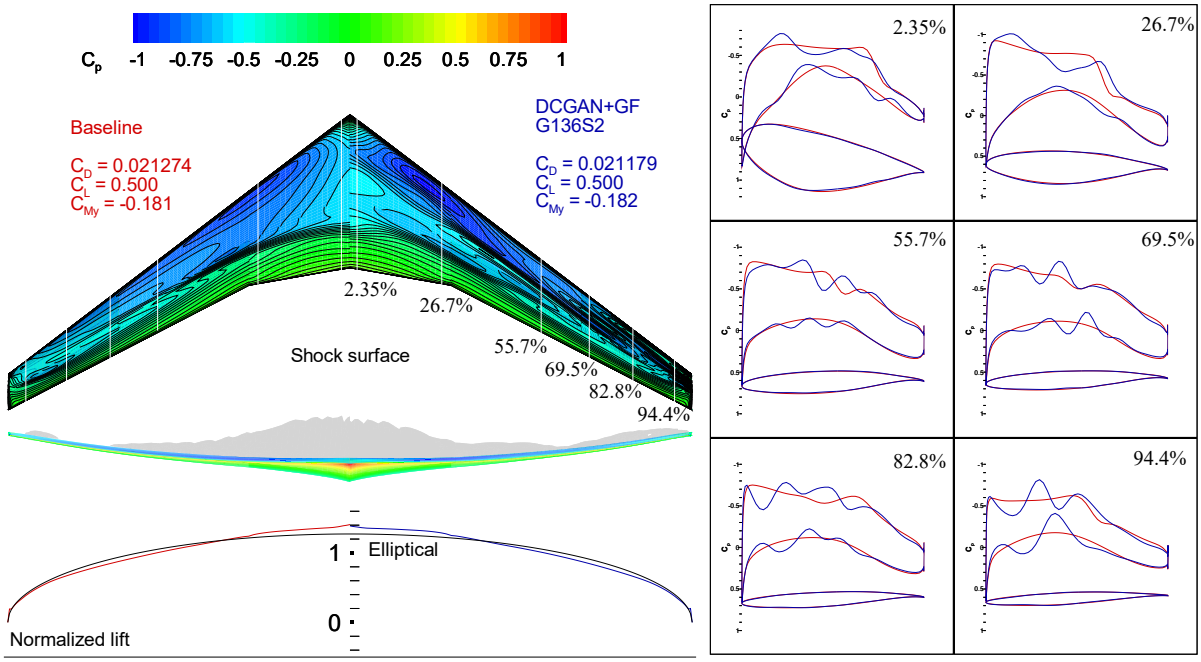


Figure 4.27 CFD comparison between the baseline and the feasible DCGAN+GF best sol.

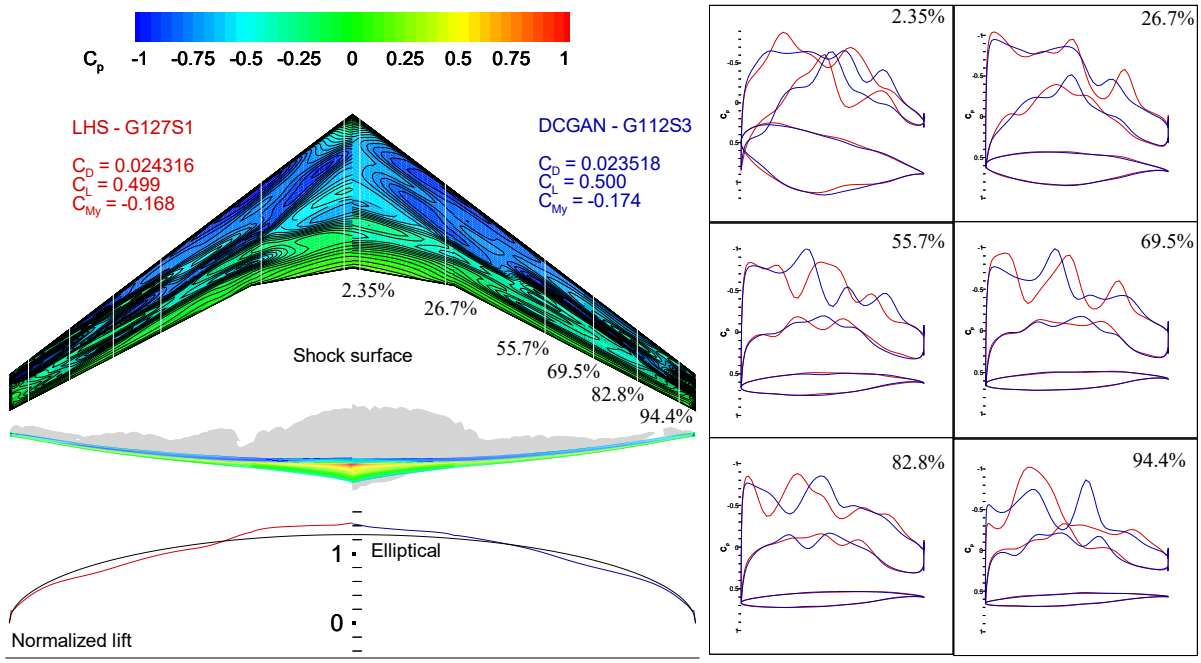


Figure 4.28 CFD comparison between the DCGAN and LHS optimal solutions.

The CFD results depicted in Figure 4.28 reveal noticeable pressure oscillations and bumps for both the LHS and DCGAN solutions, with the LHS solution exhibiting more severe conditions. These irregularities contribute to the development of more severe shock waves and higher drag. By comparing optimized designs obtained by each method with their respective initial samples in Figure 4.25, it becomes evident that less severe shocks, resulting in improvements in drag coefficient, were achieved through the smoothing of wing section surfaces and adherence to the elliptical normalized lift distribution.

In Figure 4.29, the 3D shock regions are depicted for both the baseline design and the obtained optimized designs. These shock regions are represented as transparent grey areas on the suction side of the wing. While the baseline design features a single shock region, the optimized designs clearly display bifurcated shock regions. Specifically, the DCGAN+GF G139S5 and G136S2 designs exhibit a shock region with a Lambda-configuration. This means that instead of having a single shock region like the baseline, these designs showcase two shock regions at the wing root. The two shock regions appear to merge near the wing tip, forming a λ-structure. It is important to note that despite these optimized designs exhibiting a completely different shock configuration, they still manage to achieve competitive drag performance compared to the baseline design.

The development of double shock regions in the optimized designs can be attributed to the presence of a small curvature in the suction side profiles within the range of $0.2 < x < 0.5$, as illustrated in Figure 4.30. This small curvature leads to a flow deceleration, resulting in the formation of a shock wave that terminates the supersonic region at the leading edge (LE). Simultaneously, a larger curvature within the range of $0.5 < x < 0.8$ induces a flow acceleration, creating another supersonic region that is subsequently terminated by the second shock wave. The combined effect of these curvatures gives rise to the formation of the double shock regions observed in the optimized designs.

The curvature $\kappa(x)$ is expressed in the following equation,

$$\kappa(x) = \frac{d^2z}{dx^2} \cdot \left(1 + \left(\frac{dz}{dx}\right)^2\right)^{-1.5} \tag{4.3}$$

where z is the airfoil z-coordinates, and x is the airfoil x-coordinates.

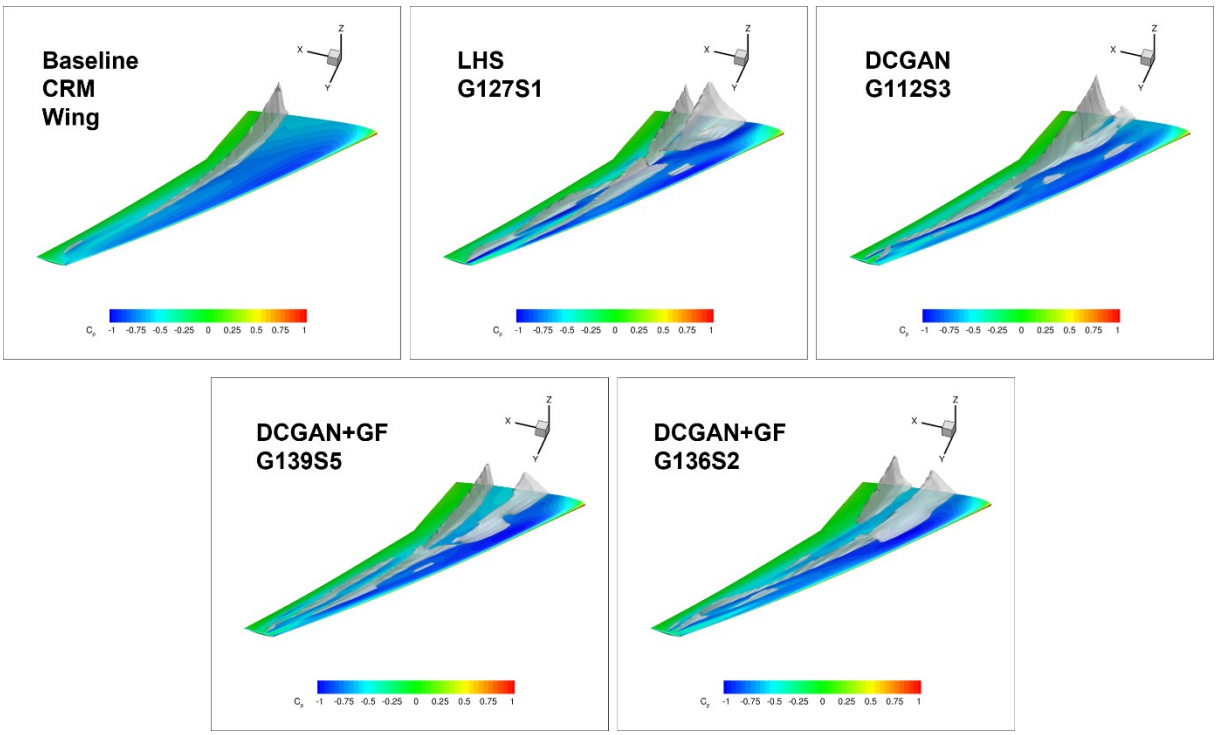


Figure 4.29 Visualization of 3D shock regions for baseline and optimized designs.

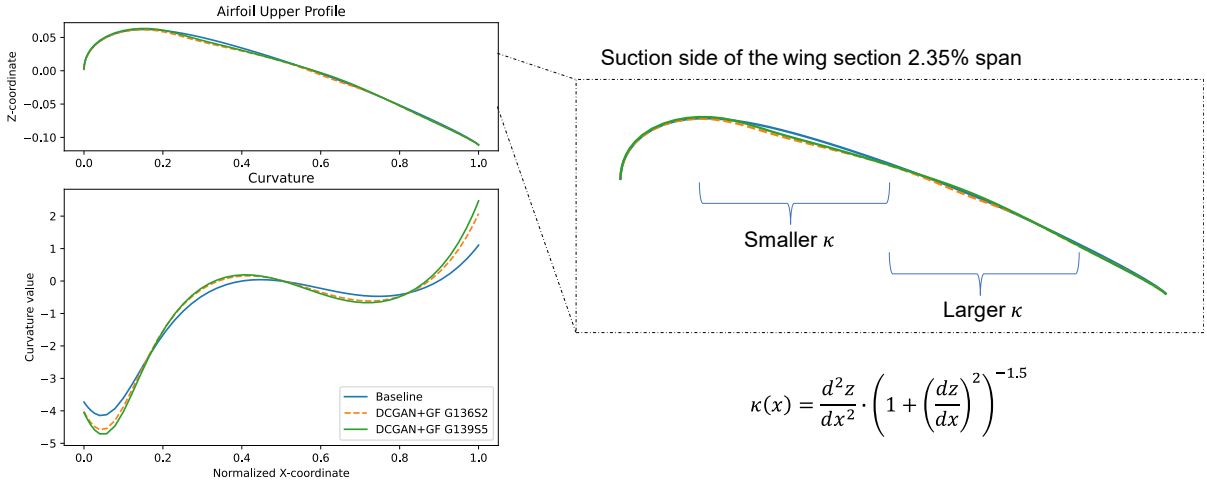


Figure 4.30 Curvatures of baseline and optimized designs at the 2.35% span wing section.

4.6.4 Comparison with other researchers' results

The lift-constrained drag minimization of the CRM wing has been extensively explored as a benchmark problem, and other researchers have conducted similar studies [3], [64], [65]. However, a common approach in these studies is the utilization of local gradient-based optimizers in combination with adjoint solvers for gradient computation. In their work [64], Lyu et al. emphasized that gradient-based optimization is often deemed necessary when dealing with a large number of design variables in aerodynamic shape optimization. Consequently, this thesis aims to enhance the SBO approach combined with a genetic algorithm to effectively handle a problem with over 100 design variables.

We conducted a comparison of our results from the single-objective wing optimization with findings from other researchers who have explored similar studies. The summarized comparison is presented in Table 4.13, which also includes key features of their respective methods. As previously acknowledged, the CRM wing optimization problem poses considerable challenges. Li et al. [65] managed to achieve a drag coefficient improvement of less than 2 counts compared to the baseline using a combination of adjoint-based and adjoint-free methods, along with gradient-based optimizers. Their approach involved 192 FFD points and 1000 CFD samples. In contrast, Lyu et al. [3] attained a more substantial drag coefficient improvement of 16 counts. Their methodology involved a gradient-based optimization approach combined with an adjoint solver, utilizing 720 FFD points and 800 CFD samples.

Table 4.13 Comparison summary with other researchers' results in CRM wing opt.

Researcher	Improvement (drag count)	Number of design vars (FFD points)	CFD samples	Using adjoint	Optimizer
This study	0.5	8 x 24 = **192**	1000	No	Gradient-free
Li et al. [61]	0.7 – 2.0	8 x 24 = **192**	1000	Mixed	Gradient-based
Lyu et al. [3]	16.0	15 x 48 = **720**	800	Yes	Gradient-based

We believe that the number of design variables, specifically the FFD control points in this case, has a direct impact on the quality of optimal designs achievable by an algorithm. The greater the number of FFD points, the more flexibility there is to deviate from the baseline, resulting in the potential for more optimal designs that an algorithm can converge to.

However, it is important to note that increasing the number of FFD control points also adds complexity to the problem. Furthermore, when dealing with a large number of design variables, the use of gradient-based optimization methods proves to be superior compared to gradient-free optimization. Gradient-based approaches leverage the power of gradient information to guide the optimization process efficiently. Additionally, employing an adjoint solver can significantly enhance the efficiency of gradient computations. Although an adjoint simulation typically consumes around half or less of the computational resources of a CFD simulation [66], it allows for more efficient and accurate computation of gradients.

4.7 Surrogate model's accuracy

The performance of the surrogate-based optimization (SBO) approach heavily relies on the accuracy of the surrogate model. Achieving sufficient accuracy of the surrogate model becomes challenging when data samples are limited or expensive to evaluate. Therefore, it is crucial to monitor and assess the accuracy of the surrogate model as the iteration progresses.

In this section, we will specifically evaluate the surrogate model's accuracy in the context of wing optimization. This is particularly important because, in this study, the wing optimization represents a high-dimensional problem, where SBO approaches often face challenges associated with the curse of dimensionality, as discussed in Chapter 1, Section 1.1.3. The accuracy of the surrogate models for the airfoil optimization is provided in the Appendix (A4). To measure the accuracy, we calculate the root mean square error (RMSE) for the modeled function: drag and moment coefficient, as follows,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{4.4}$$

where y_i is the ground-truth evaluation by CFD, $\hat{y}_i$ is the predicted value by the model, and N is the number of samples. We compute the RMSE at each main iteration after we evaluate new design candidates using CFD. In this case, $N = 5$ since we consider a total of five infilling samples per iteration. The RMSE serves as a metric to quantify the disparity between the predicted values generated by the surrogate model and the ground-truth values obtained from CFD. Essentially, the RMSE provides a measure of the accuracy of the surrogate model.

In Figure 4.31, we plotted the RMSE values for both the drag and moment coefficients as the iterations progress for all three methods. This allows us to visualize the changes in surrogate model accuracy over time. Note that the drag and moment coefficients are the expensive functions in the wing optimization that the surrogate model tries to approximate.

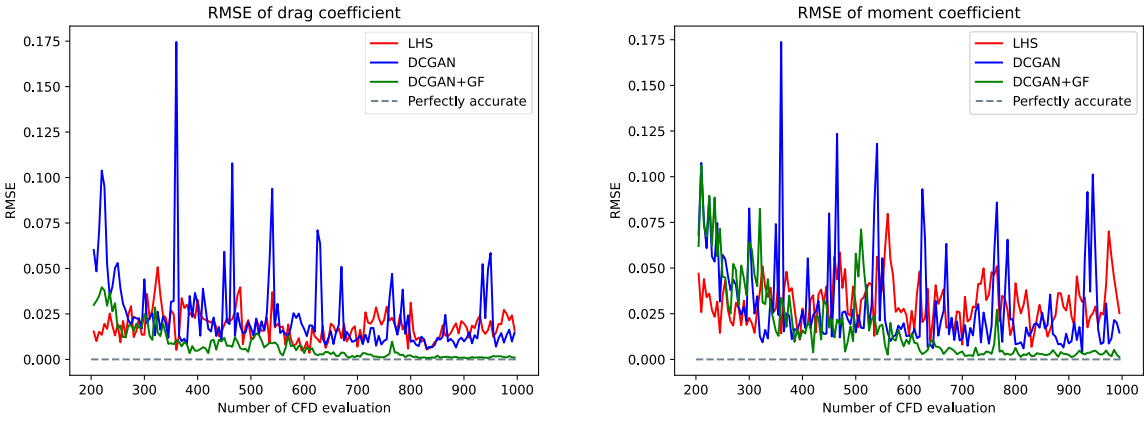


Figure 4.31 RMSE of drag (left) and moment (right) to quantify surrogate model's accuracy.

The historical progression of the model's accuracy in the above figure allows us to gain insights into the optimization performance of each method. For the LHS method, there is no clear trend observed in the RMSE values, which correlates with its optimization performance showing limited improvement after the second iteration (the 202nd CFD evaluation).

In contrast, the DCGAN method exhibits mostly lower RMSE values compared to the LHS method. However, it does display sporadic spikes at certain iterations, indicating instances of low model accuracy. During these iterations, the model underestimates the data, leading to suboptimal designs that are worse than the initial samples generated by DCGAN. This suggests that the model has limited information about the design candidates, likely due to the poor diversity of the DCGAN initial samples. Nonetheless, the DCGAN method manages to discover improved solutions compared to the LHS method.

The drawback observed in the DCGAN method is mitigated by including the geometric filter (GF), as evident from the RMSE history of the DCGAN+GF method. The GF effectively filters out irregular infilling samples with low performance, enhancing the model's accuracy (lower RMSE) and expediting the convergence process. It is worth noting that the selection of the GF score criterion could impact the performance of the DCGAN+GF method, as the GF imposes constraints directly in the design space, potentially restricting the discovery of new designs. Further research should carefully examine this aspect in future studies.

4.8 Computational time

While the DCGAN+GF method demonstrates faster convergence compared to the conventional method, it is important to consider the computational resources allocated for training the DCGAN and CNN models. The overall computational time for a single method can be divided into seven parts.

The first part (**DCGAN**) involves training the DCGAN model for initial sampling purposes. In this study, this process took approximately 0.747 hours. However, it is worth noting that this step only needs to be performed once. Once the DCGAN model is trained, it can be used for all subsequent optimization problems.

The second part (**CNN**) entails training the CNN for the geometric filter. Training for this model was relatively quick, taking only a few minutes. However, the time required to collect the data (500 DCGAN and 500 LHS samples) for training purposes, which is the third part (**CNN_data**), can be time-consuming. Collecting LHS samples is typically fast, but gathering DCGAN samples involves performing the inverse FFD procedure outlined in Algorithm 3 (refer to Chapter 3, Section 3.2.3). For the airfoil optimization this procedure took approximately 6 hours, while for the wing optimization, it took around 31 hours.

The fourth part (**sampling**) involves the initial sampling stage, which can be carried out using either the LHS algorithm or the DCGAN-based sampling. This step also includes the inverse FFD procedure. The fifth part (**sbo_iter**) involves the sub-optimization process utilizing the NSGA-II and the dynamic training of MLP-based surrogate model. The duration of this part is relatively consistent across all methods.

The sixth part (**cfd_initial**) represents the computational time required to execute the CFD simulations on the initial samples. The seventh part corresponds to the computational time needed for performing CFD simulations on all the infill samples. The duration of these computational times is summarized in Table 4.14. It is important to note that the duration of the sixth and seventh parts may vary depending on the hardware used. For the airfoil optimization, the CFD simulations were performed on an Intel(R) Core(TM) i9-1090XE 3.0 GHz with 36 processors (local computer, shown in Figure 4.32 (right)). On the other hand, for the wing optimization, the CFD simulations were conducted on an Intel Xeon Gold 6148 2.4 GHz with 40 processors (IFS supercomputer, shown in Figure 4.32 (left)).

It should also be noted that the computational time required for the CFD simulations also depends on the complexity of the geometry being analyzed. Generally, the more curvy or irregular the local shapes of the design, the longer it takes for the CFD simulations to converge. As indicated in Table 4.14, the CFD simulations in the LHS method generally take longer compared to the other two methods. This can be attributed to the irregular geometries resulting from the LHS-based sampling.

Table 4.14 Summary of computational time (hours)

	problem	method	DCGAN	CNN	CNN_data	sampling	sbo_iter	cfd_initial	cfd_infill
0	airfoil_single	LHS	0.000	0.000	0.000	0.004	1.789	0.685	4.121
1	airfoil_single	DCGAN	0.747	0.000	0.000	1.259	1.794	0.554	3.002
2	airfoil_single	DCGAN+GF	0.747	0.048	6.318	1.259	1.873	0.527	2.863
3	airfoil_multi	LHS	0.000	0.000	0.000	0.004	0.181	0.683	4.364
4	airfoil_multi	DCGAN	0.747	0.000	0.000	1.259	0.178	0.517	4.540
5	airfoil_multi	DCGAN+GF	0.747	0.048	6.318	1.259	0.186	0.520	4.739
6	wing_single	LHS	0.000	0.000	0.000	0.032	9.089	10.547	70.988
7	wing_single	DCGAN	0.747	0.000	0.000	12.643	8.567	7.316	60.121
8	wing_single	DCGAN+GF	0.747	0.032	31.689	12.643	8.702	7.316	62.138

The above computational times are also visualized using bar plots in Figures 4.33-4.35. It is evident that the DCGAN+GF requires the longest computational time among the three methods for all the optimization problems. However, it is worth noting that it is possible to calculate the required CFD time specifically until the optimization convergence is achieved. This long time can be attributed to the need to gather 500 DCGAN and 500 LHS samples for training purposes. However, it should be noted that in practice, this step only needs to be performed once. For example, the CNN-based geometric filter used in the multi-objective airfoil optimization is identical to the filter used in the single-objective airfoil optimization. Hence, as long as the problem employs the same geometric parameterization, the same filter can be reused for different problem formulations. It is also important to ensure that the flow regime remains consistent across different problems in order to use the same filter effectively.

It is important to note that the computational time required to obtain a CNN-based filter does not depend on the fidelity of the CFD simulations. Consequently, increasing the CFD fidelity does not affect the training time for the CNN, as it solely relies on the design variables and does not require any CFD data for training. This unique characteristic of training the filter provides an advantage when transitioning from lower-fidelity CFD to higher-fidelity CFD. For example, when transitioning from airfoil to wing optimization, the DCGAN+GF can compete with the other two methods, see Figure 4.35. In scenarios where they employ even higher-fidelity CFD simulations that have millions of mesh sizes, the DCGAN+GF method will likely be superior in terms of optimization convergence and overall computational time.

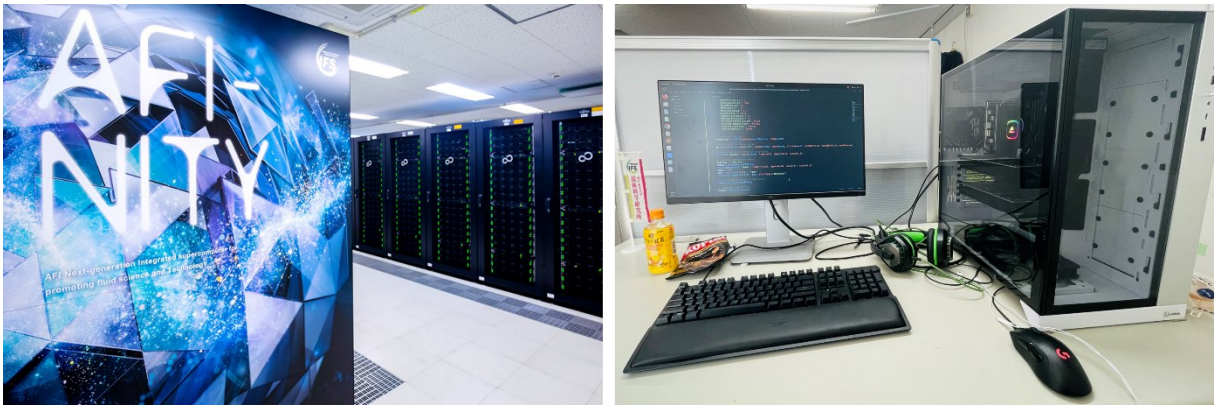


Figure 4.32 Computational hardware: IFS supercomputer [67] (left) and local comp (right)

4.9 Limitations

It is important to acknowledge that while the DCGAN+GF demonstrates the fastest optimization convergence, it may not necessarily have the shortest overall computational time. This is primarily due to the data preparation required for training the DCGAN and CNN models, which can be time-consuming. For instance, in our study, we needed to prepare a dataset of 77 transonic airfoils from the UIUC database, which required preprocessing before training the DCGAN. Additionally, datasets for the CNN were prepared by performing the inverse FFD procedure to obtain realistic wing shapes and using LHS to gather abnormal wing shapes.

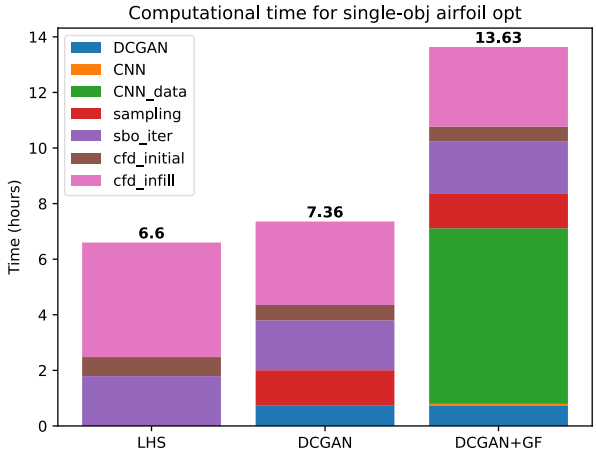


Figure 4.33 Computational time for single-objective airfoil optimization.

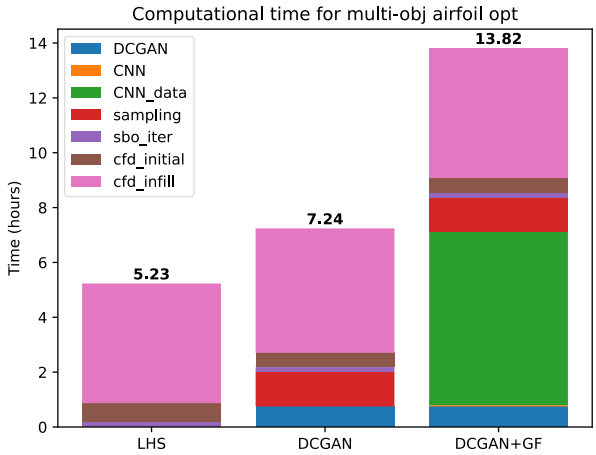


Figure 4.34 Computational time for multi-objective airfoil optimization.

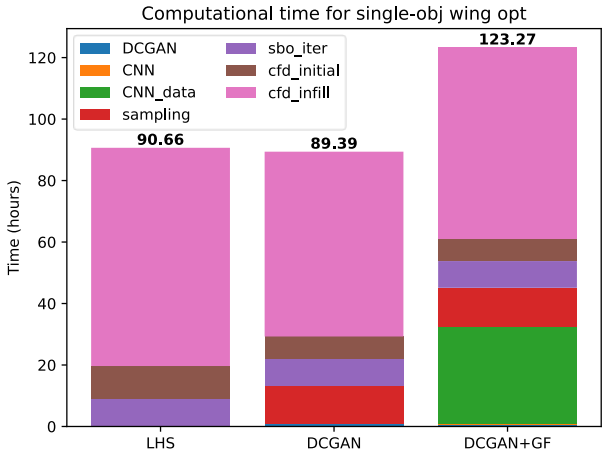


Figure 4.35 Computational time for single-objective wing optimization.

However, once these DCGAN and CNN models are trained, they can be reused for different problem formulations as long as the design variables remain the same. The training time for these models is not influenced by the fidelity of the CFD solvers. Therefore, with the DCGAN+GF approach, it becomes feasible to increase the fidelity of the CFD solvers without affecting the training time of the models. In our study, the highest fidelity of the solver included 450,560 mesh cells, and the evaluation time on a supercomputer took approximately 30 minutes. However, it is common in both the research community and industry to employ baseline meshes with millions of cells and take hours if not days of computational time. Considering these factors, it is arguable that the DCGAN+GF method can achieve a lower overall computational time, especially when dealing with higher fidelity CFD solvers. However, further investigation is required to explore this aspect thoroughly in future work.

Another important limitation of our study is that the DCGAN and CNN models utilized are specifically designed for the transonic regime. This is due to the fact that the airfoils selected from the UIUC database were specifically designed for transonic flow conditions. Therefore, applying these deep learning techniques to other use cases or flow regimes would necessitate the selection of airfoils that are better suited for those specific purposes. This would require additional time and research to identify and gather the appropriate airfoil dataset. It is crucial to consider this limitation when considering the generalizability of the DCGAN and CNN models to different flow conditions or specific use cases.

The last limitation of our study relates to how we labeled the dataset used for training the CNN. Specifically, we labeled 500 DCGAN samples as 1 to indicate realistic shapes, while 500 LHS samples were labeled as 0 to represent abnormal shapes. However, it is important to note that we did not employ a quantitative measure to determine which samples are realistic or abnormal. Instead, the labeling was based solely on subjective assumptions, assuming that all DCGAN samples are realistic and all LHS samples exhibit curvy and abnormal characteristics. There are numerous approaches available to define a quantitative measure for assessing the smoothness of the samples. However, exploring these measures and their sensitivity requires further research. In Section 4.10, we provide an ongoing work in progress (WIP) that addresses this issue and investigates the sensitivity of different techniques to deal with this limitation.

4.10 Work in progress (WIP)

To overcome the aforementioned limitation, we propose a quantitative measure to assess the smoothness of airfoils. We introduce a "roughness metric" defined as the root mean squared difference between the second derivative of the coordinate points and their mean value and then averaged between the upper and lower sides of the airfoil, expressed as follows,

$$Roughness\ metric = 0.5 \cdot \left(\sqrt{\frac{1}{n_{upper}} \sum_i \left(y''_{upper} - \overline{y''}_{upper} \right)^2} + \sqrt{\frac{1}{n_{lower}} \sum_i \left(y''_{lower} - \overline{y''}_{lower} \right)^2} \right) \quad (4.5)$$

where n is the number of coordinate points, y'' is the second derivative of the coordinate points, and $\overline{y''}$ is the mean of second derivative of the coordinate points. This metric measures the roughness of the airfoil. The greater the value, the rougher the surface is. To visualize the inner mechanics of the roughness metric, we plotted the second derivative of one of the LHS (left) and DCGAN (right) samples in Figure 4.36.

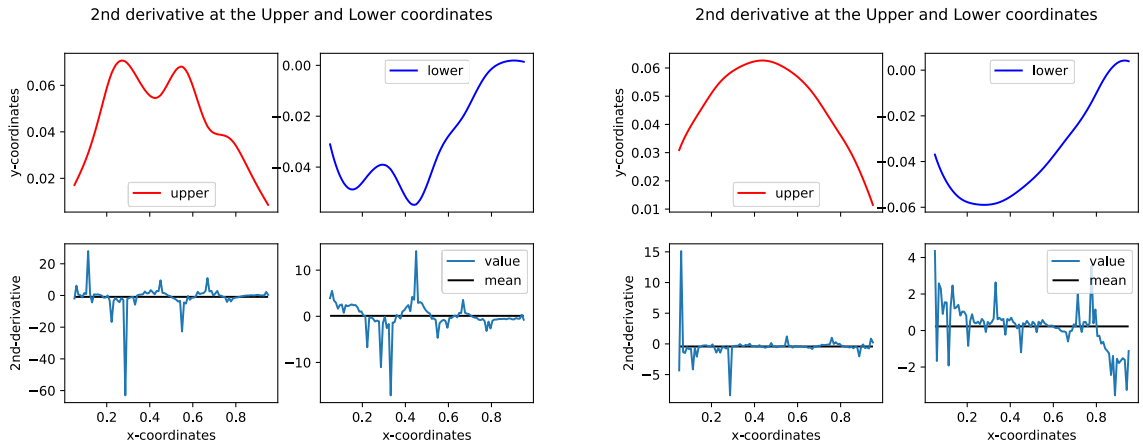


Figure 4.36 Plot for second derivative: LHS (left) and DCGAN (right) airfoils.

Next, we utilize the roughness metric to evaluate all the DCGAN and LHS initial samples, as well as the UIUC transonic airfoils and the RAE2822 baseline airfoil. The distribution of roughness metric values is depicted in Figure 4.37. It becomes apparent that some roughness metric values for DCGAN and LHS closely overlap with each other, making it challenging to establish a definitive threshold for distinguishing between abnormal and realistic samples. For the purposes of this work-in-progress (WIP), it is determined that airfoils with a roughness metric value of $S \leq 4$ are considered realistic.

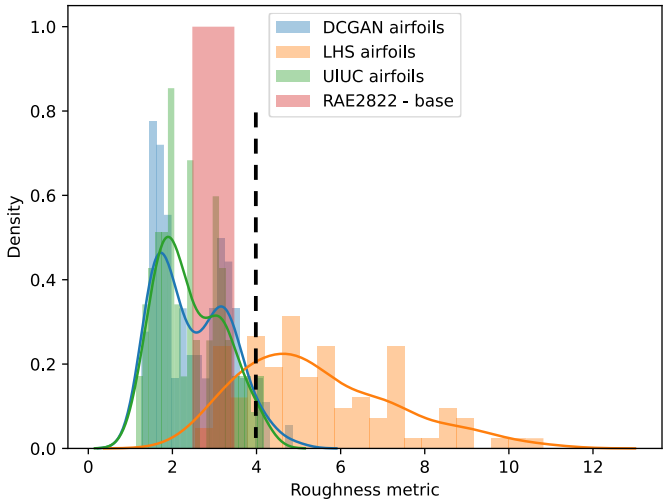


Figure 4.37 Plot for roughness metric distribution.

In the first approach, we train a CNN-based geometric filter using the actual roughness metric values. Subsequently, we incorporate this filter into the single-objective and multi-objective airfoil optimization processes by imposing an analytical constraint of $S \leq 4$. The results of this approach demonstrate promising performance for both optimization problems, as illustrated in Figures 4.38-4.39. Although this approach does not surpass the original DCGAN+GF method, it outperforms both the LHS and DCGAN techniques.

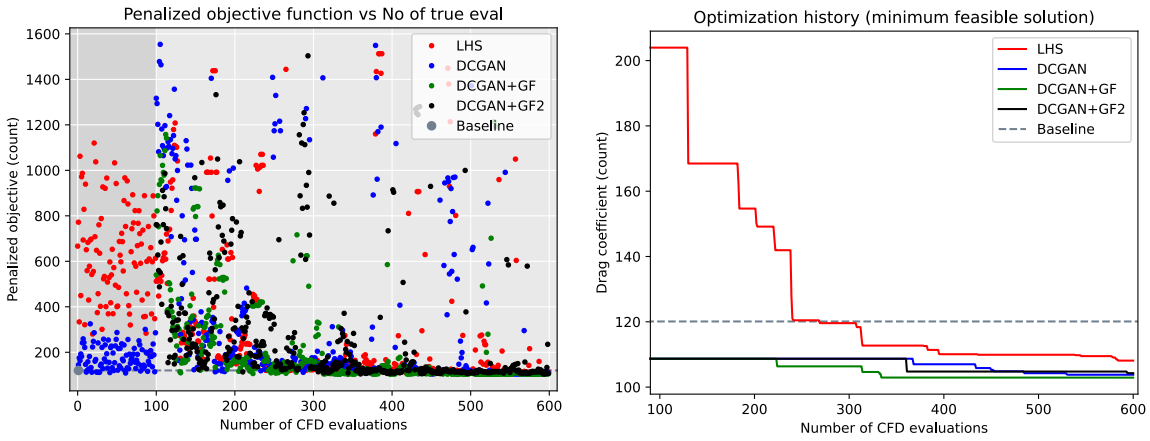


Figure 4.38 Results for the single-objective airfoil optimization.

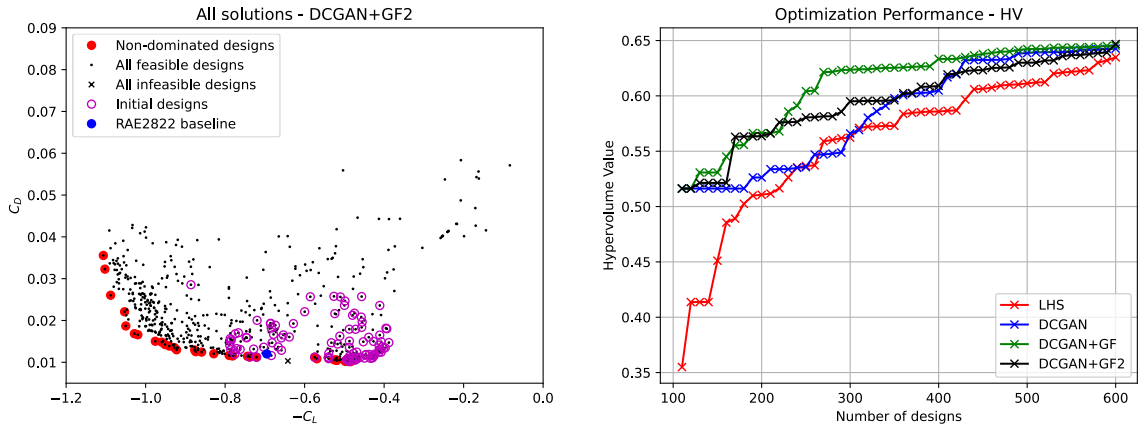


Figure 4.39 Results for the multi-objective airfoil optimization.

In the second approach, we utilize the roughness metric to determine the smoothness of airfoils, assigning discrete labels of 1 and 0 accordingly. Specifically, airfoils with a $roughness \leq 4$ are labeled as 1, indicating realistic airfoils, while airfoils with higher roughness values are labeled as 0, representing abnormal airfoils. This approach follows a similar framework to the original DCGAN+GF method, but now employs the roughness metric as the criterion for determining realistic and abnormal samples.

Once the filter was trained, we proceeded to examine the distribution of filter scores on the initial samples, as depicted in Figure 4.40. Based on this distribution, we determined that an analytical constraint of $S \geq 0.5$ would be appropriate to apply. Subsequently, we incorporated this constraint into both the single-objective and multi-objective airfoil optimization problems, considering the previously defined threshold.

In the single-objective optimization (Figure 4.41), we observe that this new approach (DCGAN+GF3) outperforms the previous approach (DCGAN+GF2), as well as the DCGAN and LHS methods. However, it still falls short of surpassing the performance of the original DCGAN+GF method. Regarding multi-objective airfoil optimization (Figure 4.42), this approach (DCGAN+GF3) performs even worse than the LHS method. However, it gradually converges to an HV value of 0.63, which is still the poorest among all the compared methods. The results obtained in this section emphasize the sensitivity of the filter's effectiveness to both the roughness metric and the training methodology of the CNN-based geometric filtering. This work-in-progress (WIP) further reinforces our confidence in the original approach described in Chapter 3, Section 3.3.3 as the most effective way to train the CNN filter thus far.

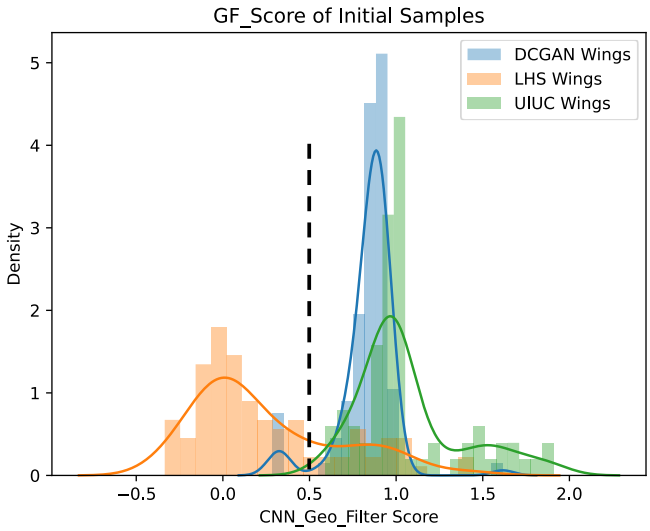


Figure 4.40 Distribution of GF score on the initial samples.

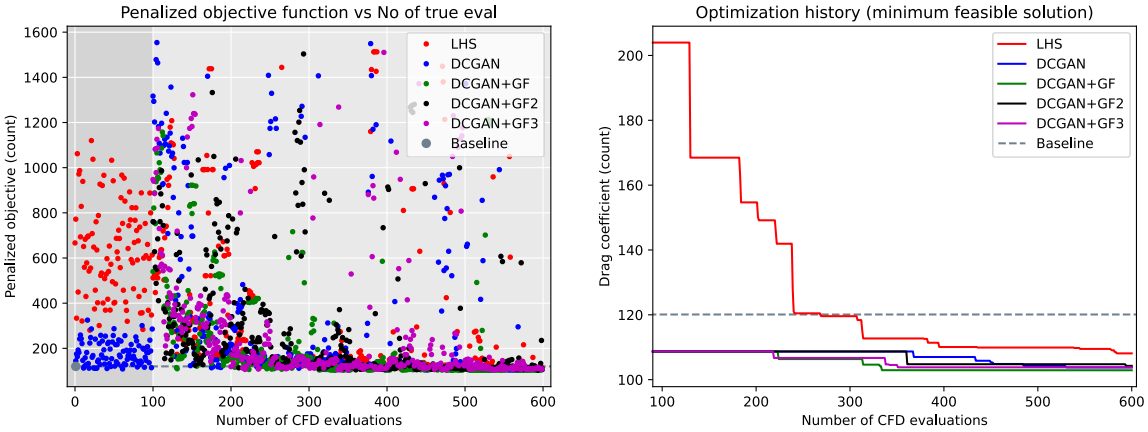


Figure 4.41 Results for the single-objective airfoil optimization.

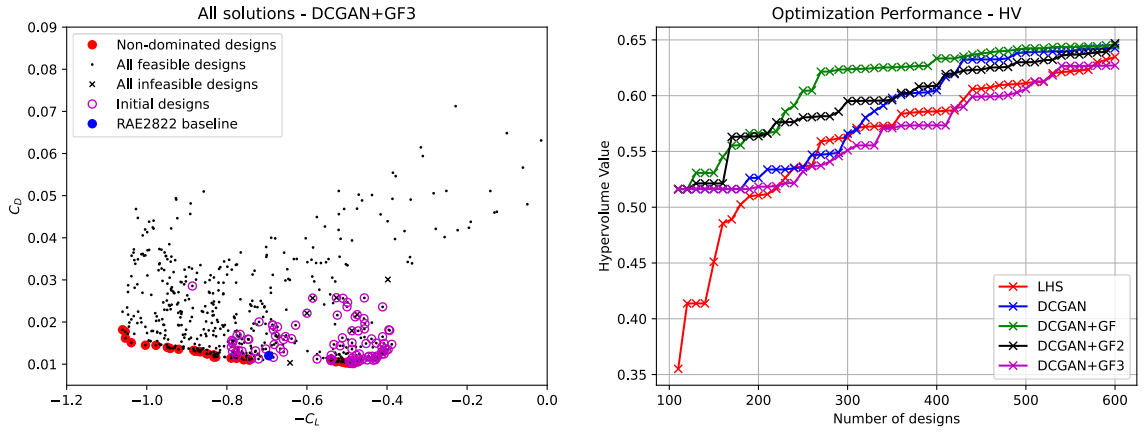


Figure 4.42 Results for the multi-objective airfoil optimization.

4.11 Summary

In this chapter, we present the practical implementation of the deep learning techniques and design framework discussed in the previous two chapters. The main objective is to demonstrate that our proposed methods are superior to the conventional method. Specifically, we apply these methods to a series of aerodynamic wing shape optimization (AWSO) problems. We provide a comprehensive overview of the experimental setup, including the AWSO problem formulations, the methodologies employed, and the specific configurations for each algorithm utilized, namely MLP, DCGAN, CNN, and NSGA-II.

Subsequently, we highlight the efficacy of the DCGAN-based sampling technique in generating high-quality initial samples for the optimization process. Additionally, we analyze the distribution of geometric filter scores among the initial samples and determine appropriate threshold values for these scores, which are employed as constraints during the optimization.

To further investigate each optimization problem, we delve into their respective problem formulations, optimization histories, and the accuracy of surrogate models employed. We also provide a detailed analysis of the optimized designs, focusing on their physical interpretations and implications. Furthermore, we evaluate the convergence capability of the optimization methods used in each problem and compare their performance. We assess their ability to converge towards optimal solutions and discuss any variations in their effectiveness. Given that the CRM wing optimization is a widely studied benchmark problem, we perform a comparative analysis of our results with those obtained by other researchers in the field.

From all the optimization results, we found that our proposed method (DCGAN+GF) can achieve the fastest optimization convergence. However, it may not be superior in terms of the overall computation time, due to the required time to gather datasets for the purpose of training DCGAN and CNN. But we argued that in cases where CFD simulations take hours or even days, our methods will be superior. In the subsequent section, we highlight the limitations of our techniques such as the required time for preparing the datasets to train DCGAN and CNN models, the fact that the models are limited to only transonic regime, and the qualitative judgement in labeling the airfoil smoothness for CNN datasets. In the last section, a work in progress (WIP) was provided in an attempt to address the last limitation by defining a quantitative roughness metric. However, the results are still vague, and further studies should be done to understand the sensitivity of this metric.

Chapter 5 Conclusion and future work

In this thesis, we have developed a design framework specifically tailored for aerodynamic wing shape optimization (AWSO) using surrogate-based optimization (SBO) as the core technology. Effective AWSO often requires a large number of design variables, which poses a significant challenge for SBO methods due to the curse of dimensionality. One of the main difficulties lies in training the surrogate model to achieve sufficient accuracy, as the accuracy of surrogate model directly impacts the optimization performance. Therefore, the main objective of this study is to improve the efficiency of conventional SBO methods in the context of AWSO by leveraging deep learning techniques.

In this research, we have employed three deep learning techniques, namely multilayer perceptron (MLP), deep convolutional generative adversarial network (DCGAN), and convolutional neural network (CNN), to address the challenges in AWSO. The MLP serves as a surrogate model within our SBO framework. It replaces the computationally expensive CFD simulations for evaluating aerodynamic functions during the sub-optimization process. Additionally, the DCGAN is employed to generate synthetic wing shapes, acting as an alternative to the conventional Latin Hypercube Sampling (LHS) method for initial sampling. Lastly, a CNN is trained as a geometric filter. This filter is utilized during the infill sampling phase of the SBO framework. Its purpose is to quickly identify and eliminate abnormalities in the design candidates, enhancing the quality and speeding up the optimization convergence.

To assess the effectiveness of the proposed deep learning techniques, we conducted three AWSO problems with varying complexities. Two of these problems focused on airfoil optimization, representing low-dimensional cases, while the third problem involved wing optimization, a high-dimensional challenge. The results demonstrate the efficacy of the MLP-based surrogate modeling in replacing expensive CFD simulations during the sub-optimization process. The conventional SBO method, empowered by the MLP, successfully converged to optimal solutions in the AWSO problems. This highlights the capability of the MLP-based surrogate model in accurately approximating aerodynamic functions. Moreover, the DCGAN-based sampling produced initial samples with superior aerodynamic performance compared to the traditional LHS-based sampling. This can be attributed to the smoother and more regular nature of the initial samples generated by the DCGAN. In contrast, the LHS samples often exhibited curvy and irregular shapes, leading to suboptimal aerodynamic performance.

Furthermore, the CNN-based geometric filtering played a crucial role in enhancing the accuracy of the surrogate models by effectively filtering out abnormal infill samples with poor aerodynamic performance. By introducing an analytical constraint, the filter shrinks the design space to focus solely on feasible and high-performing infill samples. As a result, the surrogate models achieved higher accuracy due to the reduced design space.

The integration of all three deep learning techniques is essential to improve the efficiency of the conventional SBO method and achieve the fastest optimization convergence. This assertion is substantiated by the superior performance of the DCGAN+GF method, which combines all the techniques, compared to the individual DCGAN and LHS methods, even in a high-dimensional wing optimization case. The superiority of the DCGAN+GF can be attributed to the generation of high-quality initial samples by the DCGAN and the effectiveness of the geometric filter (GF) in eliminating abnormal infill samples. By leveraging the strengths of each technique, the DCGAN+GF method achieves faster convergence and improved optimal solutions in aerodynamic wing shape optimization.

Among all the optimization results obtained, the DCGAN+GF demonstrated the fastest convergence. However, it is important to note that in terms of overall computation time, it may not outperform the conventional method in this study. This is primarily due to the time required to collect and prepare datasets for training the DCGAN and CNN models. Nonetheless, we argue that the DCGAN+GF offers significant advantages in scenarios where CFD simulations demand hours or even days to complete.

In addition to the time required for preparing DCGAN and CNN datasets, our method has another limitation. It is specifically tailored for the transonic regime, which means that if these deep learning techniques are to be applied in different scenarios, additional research is necessary to identify airfoils that are better suited for those specific purposes. Another limitation is related to the subjective nature of qualitative judgment when labeling airfoil smoothness for CNN datasets. This labeling process still relies on subjective judgment, assuming that all DCGAN airfoils are realistic, while all LHS airfoils are abnormal. Although we attempted to address this limitation by defining a quantitative roughness metric, the results of this ongoing experiment are currently being evaluated. Further studies are needed to comprehensively understand the sensitivity of this metric in terms of the optimization convergence.

Based on the previously mentioned limitations and observations from the results, we propose several directions for future research:

- Further application of the deep learning techniques proposed in this study to different use cases or flow regimes should be explored. This would help determine the versatility and effectiveness of these techniques in various optimization scenarios.
- The use of higher fidelity CFD simulations is encouraged to validate our argument that the DCGAN+GF is likely to outperform other methods not only in terms of optimization convergence, but also in overall computational time.
- It is actually possible to gather all the airfoils from the UIUC database and establish a general DCGAN model that can produce a synthetic airfoil for any flow regime. However, this should be further tested and studied.
- Research should be conducted to develop a quantitative measure for assessing airfoil smoothness. Although work in progress has been presented, the obtained results are not satisfactory. Defining another quantitative measure would improve our understanding of the sensitivity of the smoothness or roughness measurement.
- In the last wing optimization, a slight improvement was achieved. To explore the design space further and increase the likelihood of finding more optimal designs, it is recommended to either increase the computational budget beyond 1000 or reformulate the problem by incorporating a larger number of FFD control points. Figure 5.1 illustrates the reformulation of the optimization problem as a thick red arrow.

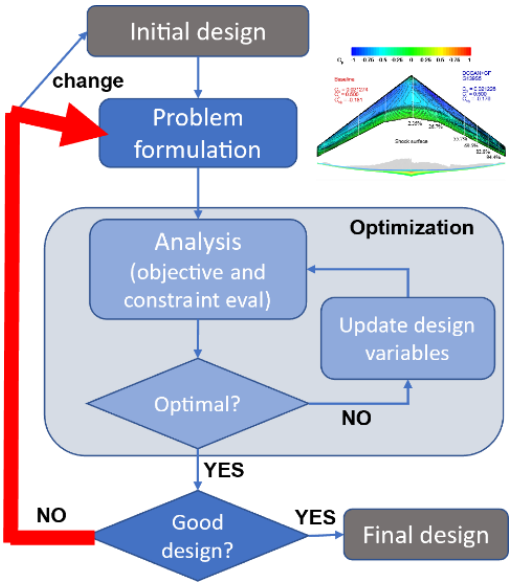


Figure 5.1 Reformulating the design problem.

Acknowledgment

First and foremost, I am immensely grateful to my supervisors, Prof. Koji Shimoyama and Prof. Shigeru Obayashi (committee chair), for their unwavering commitment, expertise, and patience. Their insightful feedback, constructive criticism, and constant availability have been instrumental in shaping the direction and quality of this thesis. I am truly fortunate to have had their guidance and mentorship.

I would like to express my sincere appreciation to the other committee members, Prof. Tomonaga Okabe, and Assoc. Prof. Yuichi Kuya. Their expertise and valuable insights have added depth and breadth to the research, enriching the discussions and conclusions of this thesis. I am grateful for their willingness to serve on my committee and for their valuable contributions to the academic rigor of this work. I extend my heartfelt thanks to the faculty members of the Department of Aerospace Engineering at Tohoku University for their dedication to imparting knowledge and providing a conducive learning environment.

I would also like to acknowledge and express my sincere gratitude for the support received from the Boeing Higher Education Graduate Research Project, the Mizuho International Foundation, and the MEXT (Ministry of Education, Culture, Sports, Science, and Technology) as part of the “Program for Promoting Researches on the Supercomputer Fugaku.” This work would not have been possible without their financial support and assistance.

Lastly, but certainly not least, I would like to express my profound gratitude to my family (my dad, my mom, my sister) and friends (lab: Naoki, Adachi, Iijima, Akashi, Fujimoto, Sekinishi, Nakamura; lyfe: Mba Farah, Hanif, Mba Halida, Ijat, Adam, Uti, Aghi, Kang Shiddiq, Teh Nadin, Mas Pras, Mba Yanti, Farros, and other @apato.kok.apotek members) for their unwavering love, encouragement, and companionship. Their belief in my abilities and their constant presence in my life have been my greatest source of strength and motivation.

To everyone mentioned above and to those who may have inadvertently been left unnamed, please accept my sincere appreciation for your invaluable contributions to the completion of this master’s thesis.

Sendai, July 10, 2023

Alfi

References

[1] I. H. Abbott and A. E. von Doenhoff, *Theory of Wing Sections*. New York: Dover, 1959.

[2] I. Netwon, *Philosophiae Naturalis Principia Mathematica*. London: Imprimatur. s. Pepys Reg. Soc. Praeses, 1686.

[3] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, vol. 53, no. 4, pp. 968–985, Apr. 2015, doi: 10.2514/1.j053318.

[4] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats, "An Assessment of Airfoil Design by Numerical Optimization," 1974.

[5] R. M. Hicks and P. A. Henne, "Wing Design by Numerical Optimization," *J Aircr*, vol. 15, no. 7, pp. 407–412, Jul. 1978, doi: 10.2514/3.58379.

[6] S. Jeong, M. Murayama, and K. Yamamoto, "Efficient Optimization Design Method Using Kriging Model," *J Aircr*, vol. 42, no. 2, pp. 413–420, Mar. 2005, doi: 10.2514/1.6386.

[7] Z.-H. Han and S. Görtz, "Hierarchical Kriging Model for Variable-Fidelity Surrogate Modeling," *AIAA Journal*, vol. 50, no. 9, pp. 1885–1896, Sep. 2012, doi: 10.2514/1.j051354.

[8] A. Jameson, "Aerodynamic design via control theory," *J Sci Comput*, vol. 3, no. 3, pp. 233–260, Sep. 1988, doi: 10.1007/bf01061285.

[9] G. K. W. Kenway and J. R. R. A. Martins, "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *J Aircr*, vol. 51, no. 1, pp. 144–160, Jan. 2014, doi: 10.2514/1.c032150.

[10] S. Chen, Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic Shape Optimization of Common Research Model WingBodyTail Configuration," *J Aircr*, vol. 53, no. 1, pp. 276–293, Jan. 2016, doi: 10.2514/1.c033328.

[11] N. V Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, Jan. 2005, doi: 10.1016/j.paerosci.2005.02.001.

[12] S. Nadarajah and A. Jameson, “A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization,” in *38th Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 2000. doi: 10.2514/6.2000-667.

[13] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide, “ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers,” *AIAA Journal*, vol. 46, no. 4, pp. 863–873, Apr. 2008, doi: 10.2514/1.29123.

[14] X. He, J. Li, C. A. Mader, A. Yildirim, and J. R. R. A. Martins, “Robust aerodynamic shape optimizationFrom a circle to an airfoil,” *Aerosp Sci Technol*, vol. 87, pp. 48–61, Apr. 2019, doi: 10.1016/j.ast.2019.01.051.

[15] R. Yondo, E. Andrés, and E. Valero, “A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses,” *Progress in Aerospace Sciences*, vol. 96, pp. 23–61, Jan. 2018, doi: 10.1016/j.paerosci.2017.11.003.

[16] S. Koziel and L. Leifsson, “Surrogate-Based Aerodynamic Shape Optimization by Variable-Resolution Models,” *AIAA Journal*, vol. 51, no. 1, pp. 94–106, Jan. 2013, doi: 10.2514/1.j051583.

[17] Z.-H. Han, M. Abu-Zurayk, S. Görtz, and C. Ilic, “Surrogate-Based Aerodynamic Shape Optimization of a Wing-Body Transport Aircraft Configuration,” in *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Springer International Publishing, 2018, pp. 257–282. doi: 10.1007/978-3-319-72020-3_16.

[18] Y. Zuo, P. Chen, L. Fu, Z. Gao, and G. Chen, “Advanced Aerostructural Optimization Techniques for Aircraft Design,” *Math Probl Eng*, vol. 2015, pp. 1–12, 2015, doi: 10.1155/2015/753042.

[19] Y. Zhang, Z.-H. Han, and L. T. Leifsson, “Surrogate-Based Optimization Applied to Benchmark Aerodynamic Design Problems,” in *35th AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, Jun. 2017. doi: 10.2514/6.2017-4367.

[20] D. G. Krige, “A statistical approach to some basic mine valuation problems on the Witwatersrand,” *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, vol. 52, no. 6, pp. 119–139, 1951.

[21] A. Diaz-Manriquez, G. Toscano-Pulido, and W. Gomez-Flores, “On the selection of surrogate models in evolutionary optimization algorithms,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*, IEEE, Jun. 2011. doi: 10.1109/cec.2011.5949881.

[22] Z.-H. Han, Y. Zhang, C.-X. Song, and K.-S. Zhang, “Weighted Gradient-Enhanced Kriging for High-Dimensional Surrogate Modeling and Design Optimization,” *AIAA Journal*, vol. 55, no. 12, pp. 4330–4346, Dec. 2017, doi: 10.2514/1.j055842.

[23] M. A. Bouhlel and J. R. R. A. Martins, “Gradient-enhanced kriging for high-dimensional problems,” *Eng Comput*, vol. 35, no. 1, pp. 157–173, Feb. 2018, doi: 10.1007/s00366-018-0590-x.

[24] J. Li, J. Cai, and K. Qu, “Surrogate-based aerodynamic shape optimization with the active subspace method,” *Structural and Multidisciplinary Optimization*, vol. 59, no. 2, pp. 403–419, Aug. 2018, doi: 10.1007/s00158-018-2073-5.

[25] M. D. McKay, R. J. Beckman, and W. J. Conover, “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code,” *Technometrics*, vol. 21, no. 2, p. 239, May 1979, doi: 10.2307/1268522.

[26] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 151–160, Aug. 1986, doi: 10.1145/15886.15903.

[27] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins, “A CAD-Free Approach to High-Fidelity Aerostructural Optimization,” in *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth,~TX, Sep. 2010.

[28] R. L. Hardy, “Multiquadric equations of topography and other irregular surfaces,” *J Geophys Res*, vol. 76, no. 8, pp. 1905–1915, Mar. 1971, doi: 10.1029/jb076i008p01905.

[29] O. Reynolds, “IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion,” *Philos Trans R Soc Lond A*, vol. 186, pp. 123–164, Dec. 1895, doi: 10.1098/rsta.1895.0004.

[30] C. A. Mader, G. K. W. Kenway, A. Yildirim, and J. R. R. A. Martins, “ADflow—An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization,” *Journal of Aerospace Information Systems*, 2020, doi: 10.2514/1.I010796.

[31] P. Spalart and S. Allmaras, “A one-equation turbulence model for aerodynamic flows,” in *30th Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 1992. doi: 10.2514/6.1992-439.

[32] A. Yildirim, G. K. W. Kenway, C. A. Mader, and J. R. R. A. Martins, “A Jacobian-free approximate NewtonKrylov startup strategy for RANS simulations,” *J Comput Phys*, vol. 397, p. 108741, Nov. 2019, doi: 10.1016/j.jcp.2019.06.018.

[33] N. R. Secco, G. K. W. Kenway, P. He, C. Mader, and J. R. R. A. Martins, “Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization,” *AIAA Journal*, vol. 59, no. 4, pp. 1151–1168, Apr. 2021, doi: 10.2514/1.j059491.

[34] J. Vassberg, M. Dehaan, M. Rivers, and R. Wahls, “Development of a Common Research Model for Applied CFD Validation Studies,” in *26th AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, Jun. 2008. doi: 10.2514/6.2008-6919.

[35] M. J. D. Powell, “Restart procedures for the conjugate gradient method,” *Math Program*, vol. 12, no. 1, pp. 241–254, Dec. 1977, doi: 10.1007/bf01593790.

[36] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull Math Biophys*, vol. 5, no. 4, pp. 115–133, 1943.

[37] J. H. Holland, “Genetic Algorithms,” *Sci Am*, vol. 267, no. 1, pp. 66–73, 1992, [Online]. Available: http://www.jstor.org/stable/24939139

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[39] M. A. Hariansyah and K. Shimoyama, “On the use of a multilayer perceptron based surrogate model in evolutionary optimization,” *The Proceedings of The Computational Mechanics Conference*, vol. 2021.34, no. 0, p. 235, 2021, doi: 10.1299/jsmecmd.2021.34.235.

[40] “Nature of nurture: is violence in our genes?,” Sep. 28, 2016. https://phys.org/news/2016-09-nature-nurture-violence-genes.html (accessed Aug. 04, 2023).

[41] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[42] J. B. MacQueen, "Some Methods for Classification and Analysis of MultiVariate Observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Le Cam and J. Neyman, Eds., University of California Press, 1967, pp. 281–297.

[43] Alan Jeffares, "K-means: A Complete Introduction," Nov. 19, 2019. https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c (accessed Aug. 04, 2023).

[44] D. Ginsbourger, R. Le Riche, and L. Carraro, "Kriging Is Well-Suited to Parallelize Optimization," in *Computational Intelligence in Expensive Optimization Problems*, Springer Berlin Heidelberg, 2010, pp. 131–162. doi: 10.1007/978-3-642-10701-6_6.

[45] M. Kantardzic, "ARTIFICIAL NEURAL NETWORKS," in *Data Mining*, John Wiley & Sons, Ltd, 2019, pp. 231–277. doi: https://doi.org/10.1002/9781119516057.ch7.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[47] S. Amari, "A Theory of Adaptive Pattern Classifiers," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 299–307, Jun. 1967, doi: 10.1109/pgec.1967.264666.

[48] Z. Shen, H. Yang, and S. Zhang, "Neural network approximation: Three hidden layers are enough," *Neural Networks*, vol. 141, pp. 160–173, Sep. 2021, doi: 10.1016/j.neunet.2021.04.011.

[49] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv, 2014. doi: 10.48550/ARXIV.1412.6980.

[50] I. J. Goodfellow *et al.*, "Generative Adversarial Networks." arXiv, 2014. doi: 10.48550/ARXIV.1406.2661.

[51] "GANs - Face editing with Generative Adversarial Networks," Feb. 04, 2022. https://datahacker.rs/005-gans-face-editing-with-generative-adversarial-networks/ (accessed Aug. 04, 2023).

[52] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” arXiv, 2015. doi: 10.48550/ARXIV.1511.06434.

[53] M. Selig, “UIUC Airfoil Coordinates Database.” https://m-selig.ae.illinois.edu/ads/coord_database.html (accessed Jul. 03, 2023).

[54] J. Li, M. Zhang, J. R. R. A. Martins, and C. Shu, “Efficient Aerodynamic Shape Optimization with Deep-Learning-Based Geometric Filtering,” *AIAA Journal*, vol. 58, no. 10, pp. 4243–4259, Oct. 2020, doi: 10.2514/1.j059254.

[55] J. Li, M. A. Bouhlel, and J. R. R. A. Martins, “A data-based approach for fast airfoil analysis and optimization,” in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Jan. 2018. doi: 10.2514/6.2018-1383.

[56] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.

[57] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” arXiv, 2019. doi: 10.48550/ARXIV.1912.01703.

[58] Wikipedia, “Cross-correlation — Wikipedia, The Free Encyclopedia.” 2023.

[59] M. A. Hariansyah and K. Shimoyama, “An Artificial Neural Network-Assisted Genetic Algorithm with Application to Multi-Objective Transonic Airfoil Shape Optimization,” *JAXA Special Publication: Proceedings of the 53rd Fluid Dynamics Conference / the 39th Aerospace Numerical Simulation Symposium*, no. 21–008, pp. 115–124, Feb. 2022, Accessed: Jul. 04, 2023. [Online]. Available: http://id.nii.ac.jp/1696/00048365/

[60] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” arXiv, 2015. doi: 10.48550/ARXIV.1502.03167.

[61] M. A. Hariansyah, “Single-Objective Airfoil Optimization,” 2023. https://airfoil-single-obj-opt-alfi.onrender.com/ (accessed Jul. 05, 2023).

[62] M. A. Hariansyah, “Multi-Objective Airfoil Optimization,” 2023. https://airfoil-multi-obj-opt-alfi.onrender.com/ (accessed Jul. 05, 2023).

[63] S. Nadarajah *et al.*, “AIAA Aerodynamic Design Optimization Discussion Group.” https://sites.google.com/view/mcgill-computational-aerogroup/adodg (accessed Jul. 06, 2023).

[64] Z. Lyu, Z. Xu, and J. R. R. A. Martins, “Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization,” in *Proceedings of the 8th International Conference on Computational Fluid Dynamics*, Chengdu, Sichuan, China, Jul. 2014.

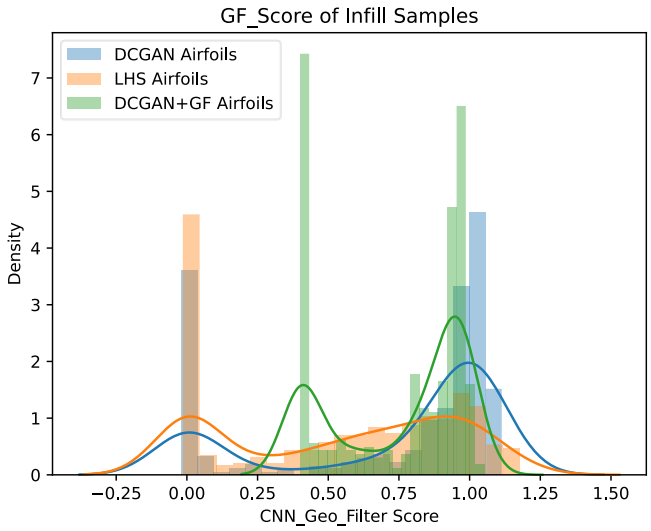
[65] J. Li and M. Zhang, “Adjoint-Free Aerodynamic Shape Optimization of the Common Research Model Wing,” *AIAA Journal*, vol. 59, no. 6, pp. 1990–2000, Jun. 2021, doi: 10.2514/1.j059921.

[66] Z. Lyu, G. K. Kenway, C. Paige, and J. R. R. A. Martins, “Automatic Differentiation Adjoint of the Reynolds-Averaged Navier-Stokes Equations with a Turbulence Model,” in *21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Jun. 2013. doi: 10.2514/6.2013-2581.

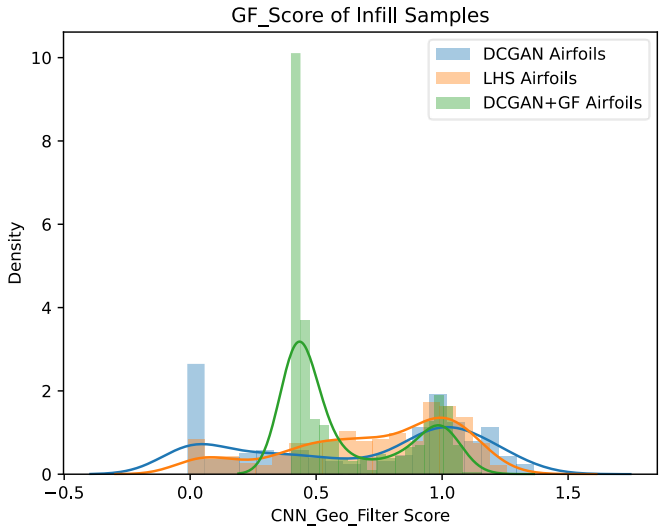
[67] “Advanced Fluid Information Research Center, Institute of Fluid Science, Tohoku University.” http://www.ifs.tohoku.ac.jp/~afirc/afirc_eng/supercomputer/ (accessed Jul. 07, 2023).

Appendix

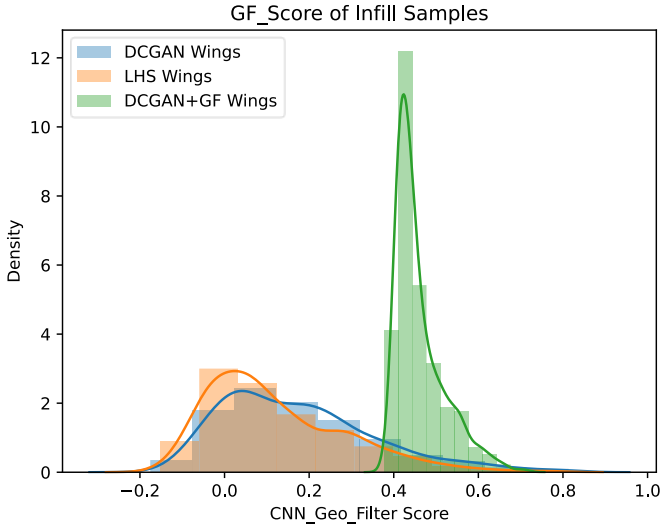
A.1. Geometric filter scores of the infill samples (DCGAN+GF has $S \geq 0.4$)



A.1.1 Geometric filter scores on the infill samples of single-obj airfoil opt.



A.1.2 Geometric filter scores on the infill samples of multi-obj airfoil opt.

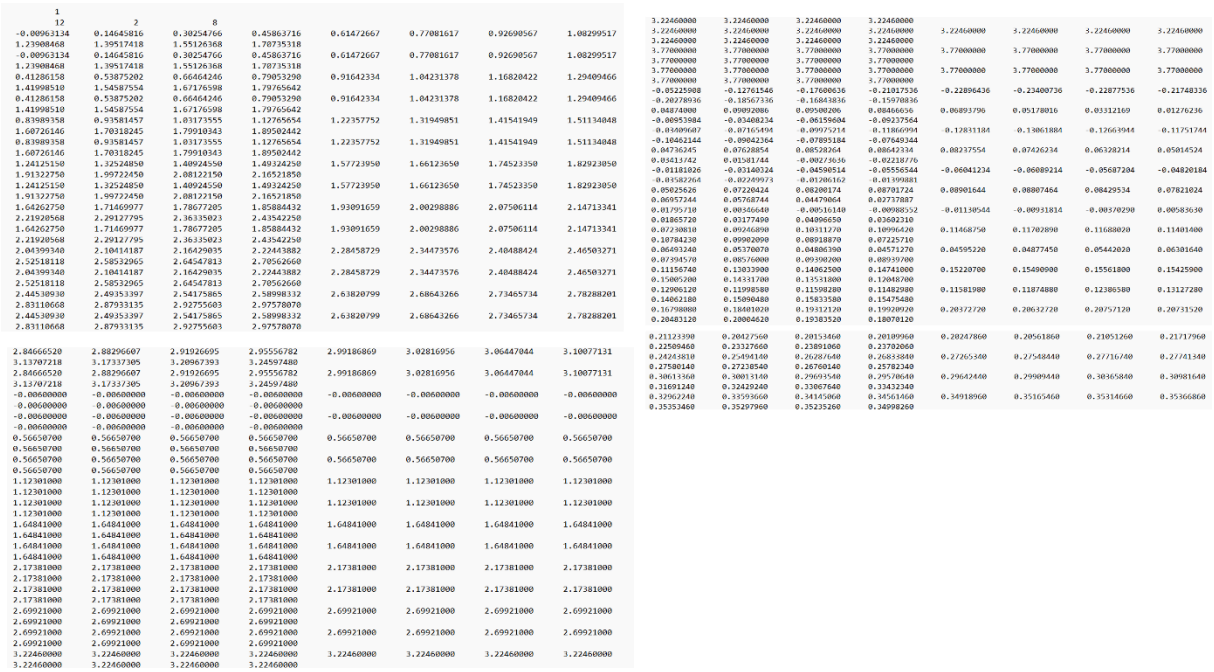


A.1.3 Geometric filter scores on the infill samples of single-obj wing opt.

A.2. FFD control points and their boundaries

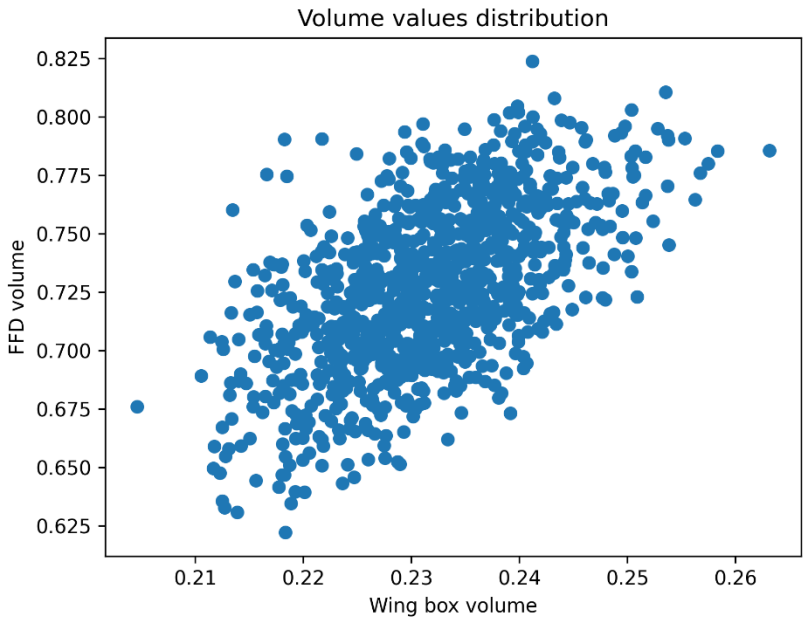
	x	y	y_lower	y_upper
0	1.001000	-0.005302	-0.007874	-0.002731
1	0.889667	-0.007932	-0.016677	0.000814
2	0.778333	-0.020555	-0.037183	-0.003926
3	0.667000	-0.037905	-0.062611	-0.013199
4	0.555667	-0.055300	-0.086976	-0.023624
5	0.444333	-0.069036	-0.105328	-0.032745
6	0.333000	-0.074091	-0.111558	-0.036624
7	0.221667	-0.070094	-0.105227	-0.034962
8	0.110333	-0.058979	-0.088322	-0.029636
9	-0.001000	-0.021443	-0.031808	-0.011079
10	-0.001000	0.020015	0.009650	0.030380
11	0.110333	0.058393	0.029050	0.087736
12	0.221667	0.070436	0.035304	0.105569
13	0.333000	0.075776	0.038310	0.113243
14	0.444333	0.076130	0.039838	0.112422
15	0.555667	0.071402	0.039727	0.103078
16	0.667000	0.060921	0.036214	0.085627
17	0.778333	0.045959	0.029330	0.062587
18	0.889667	0.027050	0.018304	0.035795
19	1.001000	0.004985	0.002413	0.007557

A.2.1 FFD for airfoil

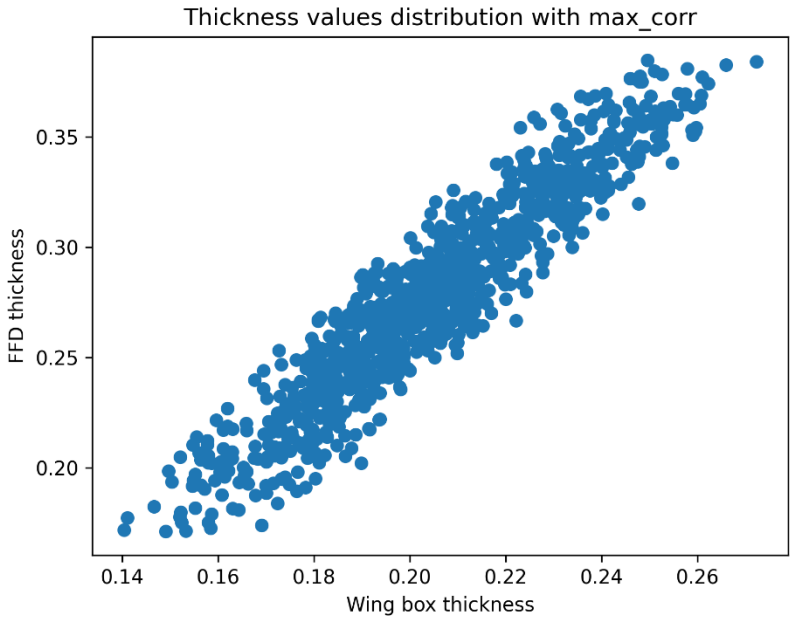


A.2.2 FFD for wing

A.3. Correlation between FFD volume and the wing's internal volume



A.3.1 Correlation for FFD volume

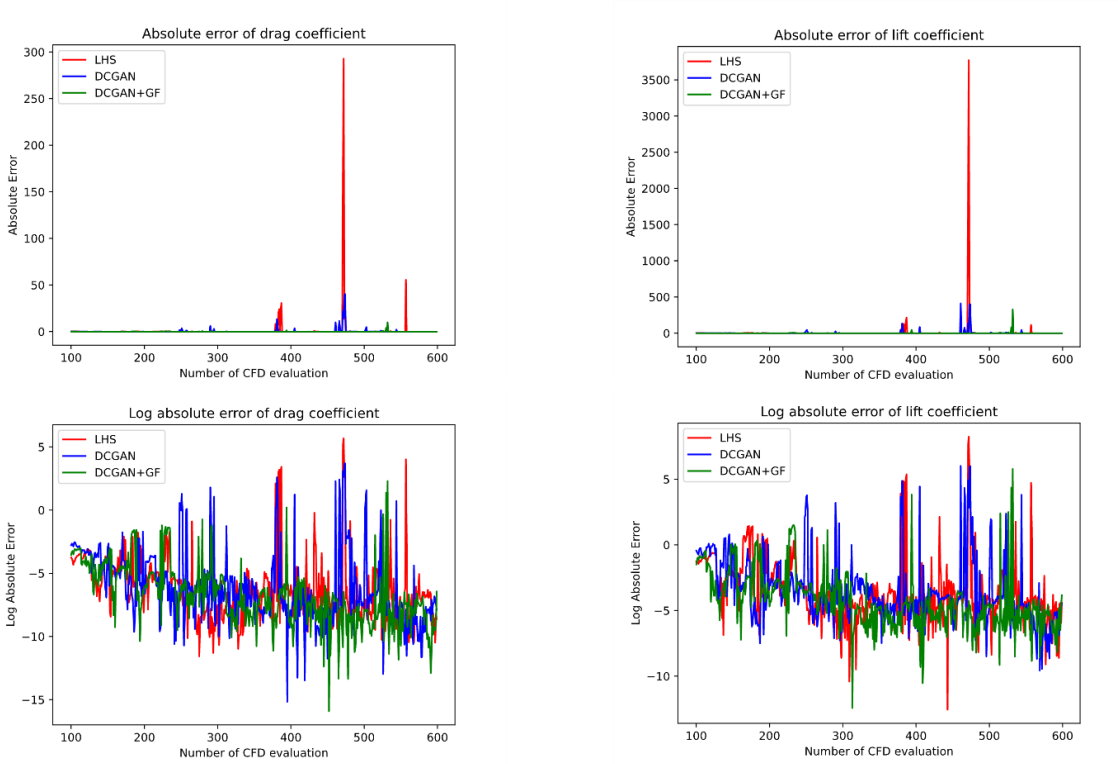


A.3.2 Correlation for FFD thickness

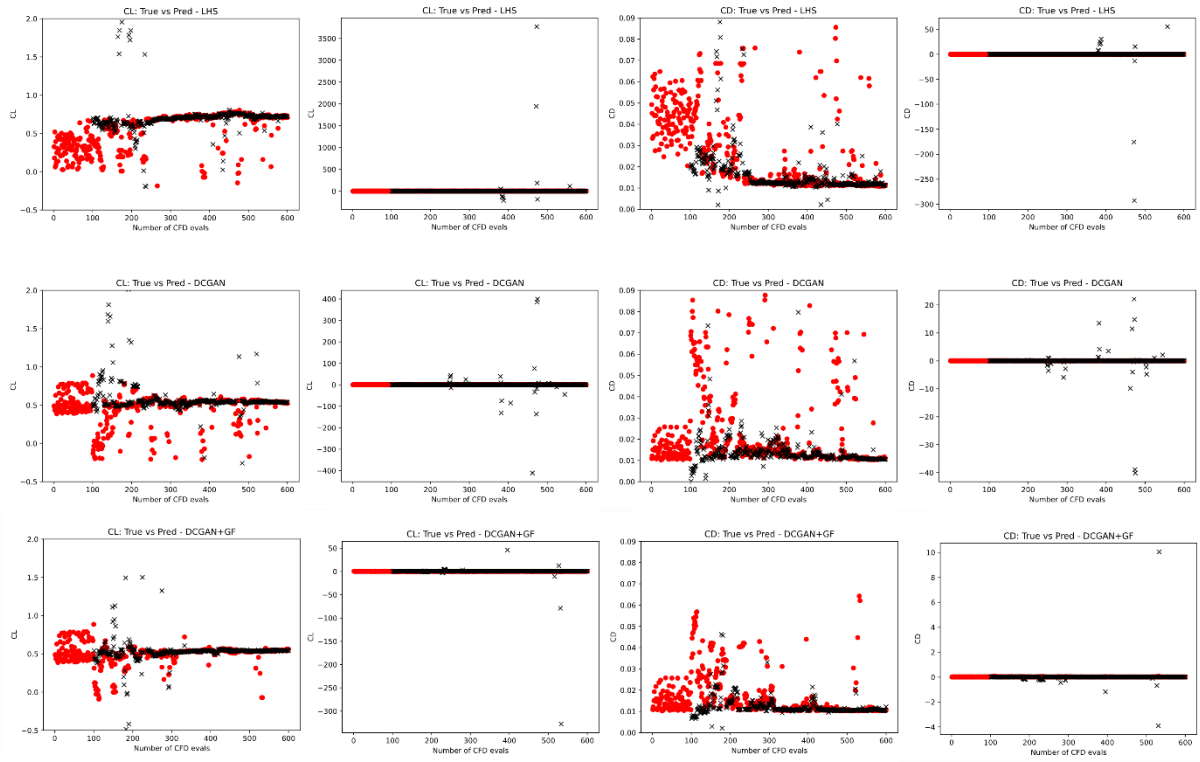
Correlation coefficient for volume	
Pearson	0.579781
Spearman	0.562638
Average correlation coefficient for thickness	
Pearson	0.647498
Spearman	0.641237

A.3.3 Correlation coefficient summary

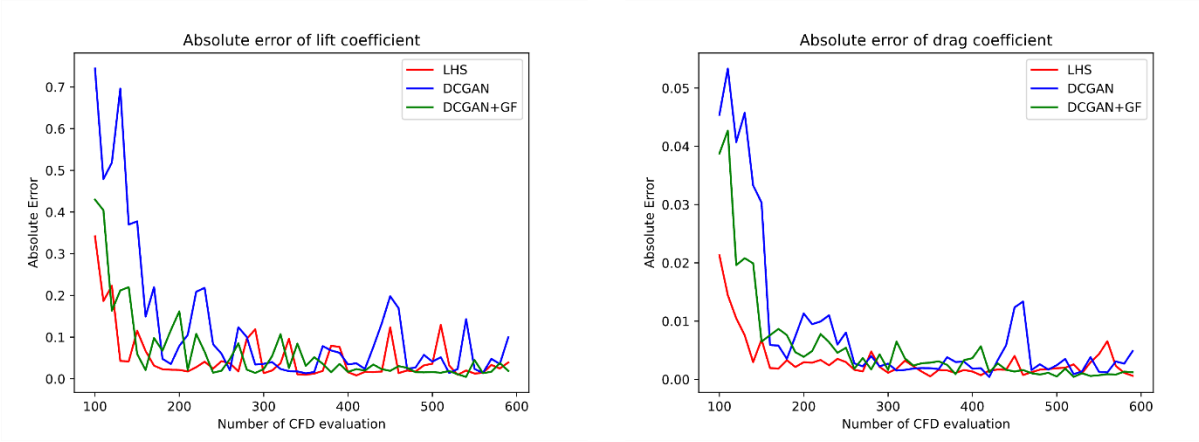
A.4. Accuracies of surrogate models in airfoil optimization



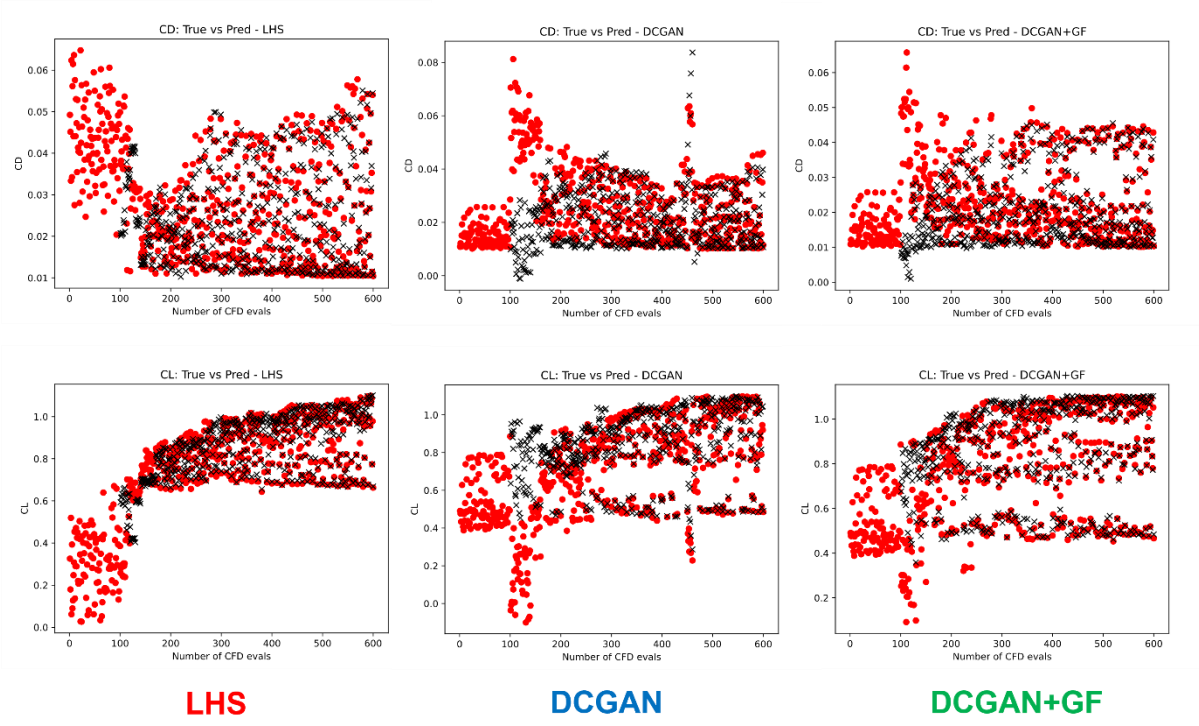
A.4.1 Surrogate model's accuracy for single-obj airfoil opt.



A.4.2 Underestimation in single-obj airfoil opt.



A.4.3 Surrogate model's accuracy for multi-obj airfoil opt.



A.4.4 Underestimation in multi-obj airfoil opt.