République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Laboratoire de Génie Mécanique et Développement

Département de Génie Mécanique

End-of-study project dissertation for obtaining the state engineer diploma in Mechanical Engineering

# *Optimization of Wind Turbine Airfoils using Generative Adversarial Networks*

*Lamia BELHASSANI*

Presented and publicly defended on : July 6th, 2023

**Composition of the jury :**

| | | | |
|---|---|---|---|
| President | Mr. Slimane DJELLAL | MCB | ENP |
| Supervisor | Mr. Abdelhamid BOUHELAL | MCA | ENP |
| Co-Supervisor | Mr. Arezki SMAILI | Prof | ENP |
| Examiner | Mr. Samir OUCHENE | PhD student | ENP |
| Guest | Mr. Mohammed Nadjib HAMLAOUI | MAB | UFAS 1 |

ENP 2023

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Laboratoire de Génie Mécanique et Développement

Département de Génie Mécanique

End-of-study project dissertation for obtaining the state engineer diploma in Mechanical Engineering

# *Optimization of Wind Turbine Airfoils using Generative Adversarial Networks*

*Lamia BELHASSANI*

Presented and publicly defended on : July 6th, 2023

**Composition of the jury :**

| | | | |
|---|---|---|---|
| President | Mr. Slimane DJELLAL | MCB | ENP |
| Supervisor | Mr. Abdelhamid BOUHELAL | MCA | ENP |
| Co-Supervisor | Mr. Arezki SMAILI | Prof | ENP |
| Examiner | Mr. Samir OUCHENE | PhD student | ENP |
| Guest | Mr. Mohammed Nadjib HAMLAOUI | MAB | UFAS 1 |

ENP 2023

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Ecole Nationale Polytechnique

Laboratoire de Génie Mécanique et Développement

L G M D

Départment de Génie Mécanique

Mémoire de projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Génie Mécanique

# *Optimisation des Profils Aérodynamiques des Éoliennes en utilisant les Réseaux Antagonistes Génératifs*

Lamia BELHASSANI

Présenté et soutenu publiquement le : 6 Juillet 2023

**Composition du jury :**

| | | | |
|---|---|---|---|
| Président | M. Slimane DJELLAL | MCB | ENP |
| Promoteur | M. Abdelhamid BOUHELAL | MCA | ENP |
| Co-Promoteur | M. Arezki SMAILI | Prof | ENP |
| Examinateur | M. Samir OUCHENE | Doctorant | ENP |
| Invité | M. Mohammed Nadjib HAMLAOUI | MAB | UFAS 1 |

ENP 2023

## ملخص

يتمثل الهدف من هذا العمل في المساهمة في استمثال الاجنحة الحاملة المستخدمة في عنفات الرياح، و ذلك من خلال انشاء تصاميم جديدة بشكل عشوائي بحيث تكون ذات أداء ديناميكي هوائي عالي. و لهذا الغرض، تم تطوير برنامج على Matlab باستخدام الشبكات التضادية التوليدية (GAN) ، تطبيق XFOIL و نظرية BEM . أولا، تم وضع شبكتي GAN و تدريبهما على قاعدة بيانات UIUC . استخدم النموذج المدرب فيما بعد لتوليد أشكال جديدة التي تم معالجتها و من ثم تقييمها بواسطة XFOIL . استنادا إلى نتائج التقييم هذه، تم اختيار الأجنحة ذات أعلى نسبة رفع إلى مقاومة، و طبق أحسن نموذج من بينها على عنفة الرياح NREL phase VI . باستعمال نظرية BEM تم حساب أداء كلتا العنفتين الأصلية و المعدلة، و من خلال مقارنة النتائج لوحظ أن العنفة المعدلة كانت أكثر كفاءة قليلا من العنفة الأصلية، و هذا دون أي تعديل أو استمثال للهندسة الأولية للشفرة، تأكيدا لقدرة البرنامج المطور على إنتاج تصاميم أجنحة ذات كفاءة عالية.

**الكلمات الدالة :** جناح حامل، الشبكات التضادية التوليدية GAN ، XFOIL ، نظرية BEM ، أداء ديناميكي هوائي

## *Résumé*

Ce travail vise à contribuer à l'optimisation des profils des pales d'éoliennes en créant aléatoirement de nouveaux profils avec des performances aérodynamiques élevées. À cet effet, un code Matlab a été établi en utilisant un réseau antagoniste génératif (GAN), XFOIL et la théorie BEM (Blade Element Momentum). Tout d'abord, les deux réseaux du GAN ont été définis et entraînés sur la base de données UIUC. Le modèle entraîné a été utilisé pour générer de nouveaux designs qui , après traitement, ont été évalués par XFOIL. En fonction des résultats de cette évaluation, les profils présentants les rapports portance/trainée les plus élevés ont été sélectionnés. Par la suite, le meilleur profil parmi ceux sélectionnés a été appliqué à une éolienne de référence : NREL phase VI , en remplaçant le profil S809. A l'aide de la méthode BEM, les performances des deux éoliennes, de référence et modifiée, ont été calculées puis comparées. Les résultats ont montré que la nouvelle éolienne était légèrement plus efficace que l'originale, et ce, sans aucune optimisation de la géometrie des pales. Cela valide la capacité du code établi à produire des profils aerodynamiques très performants.

**Mots clés :** Profil d'aile, Réseaux antagonistes génératifs (GAN), XFOIL, théorie BEM, performance aérodynamique

## *Abstract*

This work aims to contribute to the optimization of airfoils in the wind energy domain by randomly creating a database of novel airfoil designs with high aerodynamic performance. For this purpose, a Matlab code was established using Generative Adversarial Networks, XFOIL and the BEM theory. First, the two networks of the GAN were defined and trained over the UIUC database. The trained model was used to generate new shapes that were post-processed and then evaluated in XFOIL. Based on this evaluation results, airfoils with the highest lift to drag ratios were selected. Subsequently, the best airfoil among the selected ones was applied to the NREL phase VI wind turbine. Using the BEM method, performance of both the original and modified turbines were calculated and then compared. The results have shown that the turbine with the selected airfoil was slightly more efficient than the original one, and that without any optimization of the blade design being applied. This serves as a validation of the established code's ability to produce highly efficient airfoils.

**Keywords :** Airfoil, Generative Adversarial Networks (GAN), XFOIL, BEM theory, aerodynamic performance

# *Acknowledgements*

*I would like to express my sincere gratitude to my supervisor* ***Dr. BOUHELAL Abdelhamid*** *for his invaluable guidance and support for the past three years and throughout the duration of this graduation project. His dedication and expertise have been instrumental in shaping the outcome of this work, and I am profoundly grateful for his mentorship.*

*I am equally indebted to my supervisor* ***Pr. SMAILI Arezki*** *for his guidance and his invaluable advice on methodology and the overall structure of this thesis.*

*May* ***Dr. DJELLAL Slimane****, who agreed to preside over the jury, find here my sincere and respectful expressions of gratitude.*

*I would also like to express my deep appreciation and thanks to* ***Mr. OUCHENE Samir*** *for accepting to examine and evaluate this work.*

*My thanks also go to* ***Dr. HAMLAOUI Mohammed Nadjib****, for accepting the invitation to be a part of the jury as a guest and evaluate this work.*

*I extend my heartfelt appreciation to all my professors at the Mechanical Engineering Department, for creating an enriching academic environment and fostering a culture of intellectual curiosity into which was incorporated empathy and compassion, creating a harmonious and inclusive community.*
*As my journey at the department comes to an end, I am sincerely grateful for the knowledge, skills, and inspiration I have gained from them, and which have broadened my horizons and nurtured my passion for mechanical engineering.*

# *Dedications*

***To my beloved parents,***

*I dedicate this work to you with the uttermost love and gratitude.*
*You have been my pillars of strength. Your unwavering support, encouragement, and belief in my abilities have been the driving force behind my academic achievements. Your sacrifices, dedication and relentless commitment to my education have inspired me to always give my best and never settle for anything less.*
*Thank you for instilling in me a thirst for knowledge, a strong work ethic and the courage to pursue my dreams.*
*I am forever grateful to you and I am forever blessed to have you as my parents.*

*To my one and only sister, **Cilia**,*
*Thank you for always being there for me, offering a helping hand, and providing the support I needed. Not only have you been my sister, but you have also been my closest friend and confidant.*

*To my dear brother, **Aghiles**,*
*Thank you for always believing in me and pushing me to reach my full potential. Your support has been a constant reminder that I am not alone on this path, and that I will always have you by my side.*

*To my source of annoyance, my little brother **Salim**,*
*Despite the teasing and disturbance, I appreciate the underlying love and support that always shines through. Your belief in me has provided me with an extra dose of motivation.*

*I therefore, dedicate this work to you, my siblings, as a token of my deepest appreciation and love.*

*To my dearest friends, **Honza**, **my Roomie** and **Kamkam**, with whom I shared incredible and wonderful moments. Your presence in my life has been a constant source of joy, inspiration, and encouragement. Thank you for everything.*

*To my wonderful and unique **classmates**, who have become my second family.*
*The bond we have formed over the years is invaluable. Together, we made this academic journey more enjoyable and meaningful, and we have created memories that are truly a treasure to cherish.*

*To all my family and friends.*

*With love and gratitude,*
*Lamia*

# Contents

# List of Figures

# *List of Symbols and Abbreviations*

| | | |
|---|---|---|
| $A$ | Rotor Area | ($\mathrm{m}^2$) |
| $a$ | Axial induction factor | |
| $a'$ | Angular induction factor | |
| $B$ | Number of blades | |
| $C_d$ | Drag coefficient | |
| $C_l$ | Lift coefficient | |
| $C_n$ | Normal coefficient | |
| $C_p$ | Power coefficient | |
| $C_T$ | Thrust coefficient | |
| $C_{Tr}$ | Local thrust coefficient | |
| $C_t$ | Tangential coefficient | |
| $c$ | Chord length | (m) |
| $D$ | Discriminator | |
| $D^*$ | Optimal Discriminator | |
| $D_{JS}$ | Jensen-Shannon divergence | |
| $F$ | Tip loss correction factor | |
| $F_L$ | Lift force | (N) |
| $F_D$ | Drag force | (N) |
| $G$ | Generator | |
| $L$ | Loss function | |
| $\dot{m}$ | Mass flow rate | (kg/s) |
| $\mathbf{n}$ | Normal vector of a surface | |
| $P$ | Rotor power | (W) |

| | | |
|---|---|---|
| $p$ | Pressure | (Pa) |
| $Q$ | Torque | (Nm) |
| $R$ | Rotor radius | (m) |
| $r$ | Radial distance of the element from the hub | (m) |
| $Re$ | Reynolds number | |
| $s$ | Length of the panel | (m) |
| $T$ | Thrust force | (N) |
| $U$ | Free stream velocity | (m/s) |
| $U_{rel}$ | Relative wind velocity | (m/s) |
| $V_\infty$ | Free flow velocity | (m/s) |

## Greek letters

| | | |
|---|---|---|
| $\alpha$ | Angle of attack | (°) |
| $\alpha_{stall}$ | Stall angle of attack | (°) |
| $\Gamma$ | Total circulation | ($m^2/s$) |
| $\gamma$ | Vortex strength per unit length | (m/s) |
| $\theta_p$ | Section pitch angle | (°) |
| $\theta_{p,0}$ | Blade pitch angle | (°) |
| $\theta_T$ | Section twist angle | (°) |
| $\lambda$ | Tip speed ratio | |
| $\lambda_r$ | Local tip speed ratio | |
| $\mu_D$ | Discriminator's strategy (Discriminative distribution) | |
| $\mu_G$ | Generator's strategy (output distribution) | |
| $\hat{\mu}_D$ | Generator's strategy at Equilibrium | |
| $\hat{\mu}_G$ | Discriminator's strategy at Equilibrium | |
| $\mu_{ref}$ | Reference distribution | |
| $\nu$ | Kinematic viscosity of air | ($m^2/s$) |
| $\rho$ | Air density | ($kg/m^3$) |
| $\sigma$ | Source strength per unit length | (m/s) |
| $\sigma'$ | Local solidity | |
| $\phi$ | Velocity flow potential | ($m^2/s$) |
| $\varphi$ | Angle of relative wind | (°) |
| $\Omega$ | Angular velocity of the rotor | (rpm) |
| $\omega$ | Angular velocity of the wake rotation | (rpm) |

## Abbreviations

| | |
|---|---|
| ANN | Artificial Neural Network |
| AOA | Angle of attack |
| BEMT | Blade Element Momentum Theory |
| CFD | Computational Fluid Dynamics |
| CGAN | Conditional Generative Adversarial Network |
| CNN | Convolutional Neural Network |
| DCGAN | Deep Convolutional Generative Adversarial Network |
| GA | Genetic algorithms |
| GAN | Generative Adversarial Network |
| HAWT | Horizontal Axis Wind Turbine |
| LE | Leading Edge |
| NURBS | Non-uniform rational B-spline |
| ReLU | Rectified linear units layer |
| SAGAN | Self-Attention Generative Adversarial Network |
| TE | Trailing Edge |
| TransGAN | Transformer GAN |
| TSR | Tip Speed Ratio |
| UIUC | University of Illinois at Urbana-Champaign |
| VAEGAN | Variational Auto-Encoder Generative Adversarial Network |
| WGAN | Wasserstein Generative Adversarial Network |
| WT | Wind Turbine |

# *General Introduction*

## Background

Wind energy, a form of renewable energy that harnesses the power of the wind to generate electricity, has emerged as a promising and sustainable power source that is attracting global attention. With its abundant availability almost everywhere on earth and its potential to play a major role in reducing greenhouse gas emissions and combating climate change, this green energy source is rapidly evolving and growing in popularity as the global wind power capacity reached 839 GW by the end of 2021 (Figure 1), with 93.6 GW of new capacity being installed (Figure 2), driven by China and the US. According to the global wind report 2021, Wind energy also accounted for 1870 TWh, 7% of the world's electricity generation in 2021 [1].

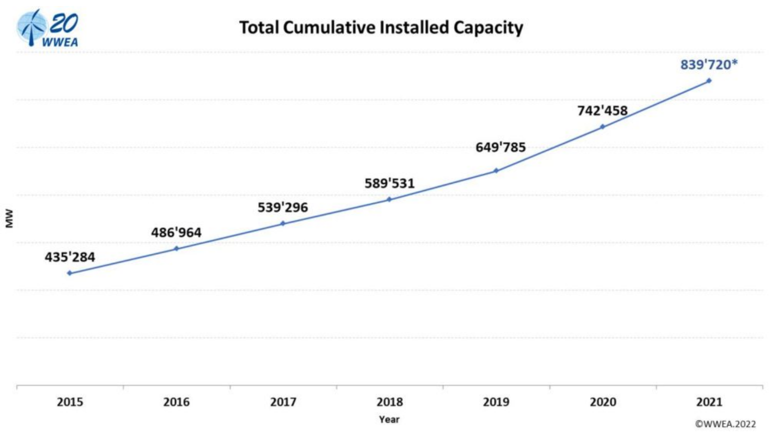


Figure 1: Total cumulative installed wind capacity [2]

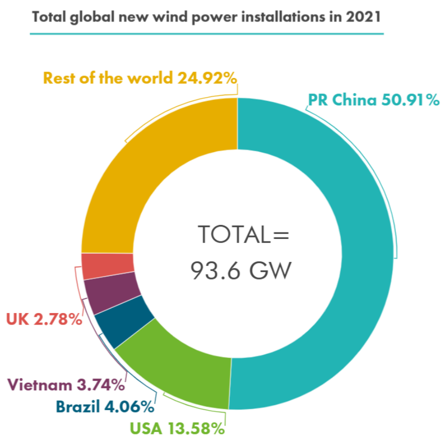


Figure 2: Global new wind power - 2021 [1]

To harness wind energy, wind turbines installed in areas where wind resources are abundant and consistent such as offshore or open fields, are used to convert wind energy into mechanical power and then into electrical energy. The rotating shaft and blades mounted on the wind turbine are crucial components that enable the conversion of wind energy. The blades convert the kinetic energy of the wind into forces due to their curved sectional shape, known as airfoils (Fig: 3).

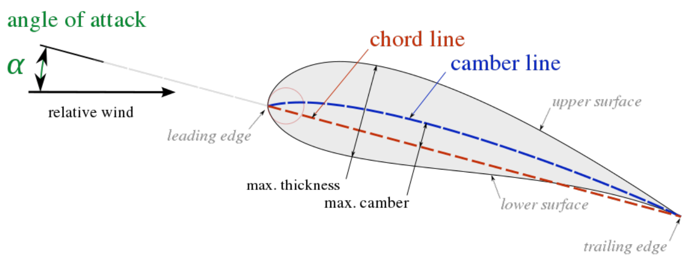


Figure 3: Airfoil geometry [3]

The importance of optimizing wind turbine performance cannot be overstated. Wind turbines are designed to maximize energy production while minimizing the cost of electricity generated. This requires the optimization of several key factors, including wind speed and direction, blade design, and control systems, which can help reducing losses in the power conversion process and improving the reliability of the turbines.

Researchers have also explored multi-objective optimization techniques, aiming to achieve multiple optimization objectives simultaneously. The three primary optimization objectives that have been considered by researchers in this field are : Maximization of energy production, energy cost reduction and blade mass reduction [4].

As it was pointed out earlier, the first step in the energy conversion process from kinetic energy of the wind to forces able to rotate the shaft, is due to the special cross-sections of the blades. A pressure difference is created between the upper and lower surfaces of the blade [5]. The airfoil's shape and orientation allow it to capture the kinetic energy of the wind, deflecting it downwards, and producing lift (Figure 4). This lift force acts perpendicular to the wind direction, causing the blade to rotate around the turbine's hub (Figure 5). As the blades rotate, they turn the turbine's shaft, which is connected to a generator that converts the mechanical energy into electrical energy.

Thus, the airfoil's geometry and parameters play a vital role in the wind turbine's performance, and its shape aims to increase lift, produce thrust, optimize wind energy utilization, and ultimately leading to higher energy output.

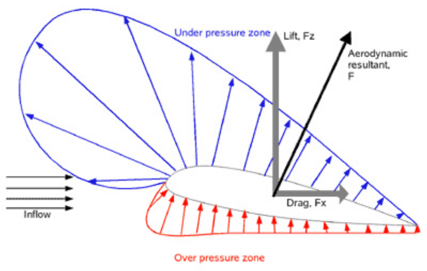


Figure 4: Generation of aerodynamic forces [6]

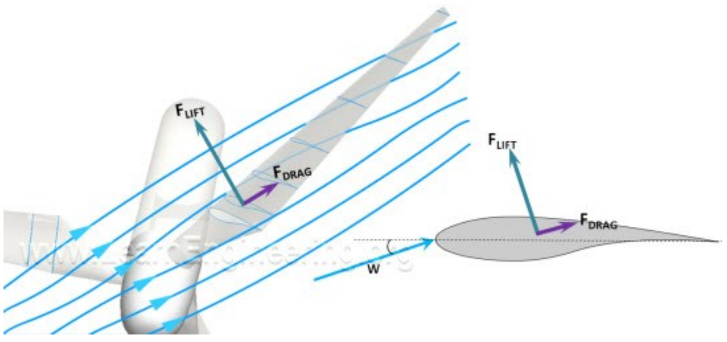


Figure 5: Rotation of wind turbine blade [7]

## Problematic

Optimizing wind turbine performance is essential for ensuring the economic viability and environmental sustainability of wind energy as a major source of electricity. With airfoils being a key factor in wind turbine performance, various mathematical, numerical, and algorithm-based optimization techniques have been considered to explore various design and optimization approaches enabling wind energy to compete with other forms of energy generation.

As such, this work aims to contribute to the optimization of airfoils in the wind energy domain using Generative Adversarial Networks (GANs), a promising approach that has emerged in the past few years with an ability to explore a wider range of airfoil designs.

## Purpose and Objectives

The objective of this work is to randomly generate new airfoil shapes that exhibit very good aerodynamic performance and that can enhance the performance and efficiency of wind turbines. Ultimately, these turbines would compete with existing optimized models, such as the NREL Phase VI .

To achieve this, a MATLAB code ***(airfoil_GAN.m)*** has been developed, in which a GAN is established and trained using the UIUC (University of Illinois at Urbana-Champaign) database. This will allow to generate diverse and realistic airfoil designs, which will then be evaluated for their aerodynamic performance using XFOIL. The ones achieving high lift-to-drag ratios will be selected and applied to the NREL Phase VI , replacing the S809.

Analyzing the performance of the new modified turbine, which normally would be more efficient, would serve as a demonstration for the potential of this approach to yield enhanced airfoil designs, and contribute to the optimization of wind turbine airfoils by providing new designs that would enhance the performance and efficiency of wind turbines.

## Thesis structure

This thesis is structured into 4 chapters as follows :

Chapter 1 presents a literature review on the different methods used in airfoil optimization, along with their associated challenges. GANs are then presented as a promising approach for tackling these challenges, highlighting its advantages and reviewing some previous works that utilized GANs in this context.

The second chapter provides an overview of the theoretical foundations of the methods employed in this work. It covers the theory and mathematical aspects of Generative Adversarial Networks (GANs), the Panels method and boundary layer model used in XFOIL, and the Blade Element Momentum Theory (BEMT). This theoretical background sets the stage for the subsequent chapter.

In chapter 3, a detailed implementation of the methods previously discussed is presented. Starting with a description of the general code structure and parts, and then delving into the practical and technical aspects of each part : GAN training, airfoil generation, evaluation and validation of the model.

The 4th and last chapter is dedicated to the presentation and discussion of results obtained with our code. A comparative analysis is conducted, where we compare the performance of the NREL Phase VI wind turbine and the modified turbine equipped with the best newly generated airfoil.

# Chapter 1
# Literature review

## 1.1 Introduction

This chapter provides an overview of the different methods used for airfoil optimization. For each technique, a description and general steps are presented, along with the advantages and limitations. The subsequent sections present GANs as a promising approach to be used in optimizing airfoil shapes, as well as a review of some of the works that used GANs in this context. Finally, we establish the scope of our study, specifying its parameters and distinguishing it from previous works.

## 1.2 Airfoil optimization techniques

### 1.2.1 Parametrization method

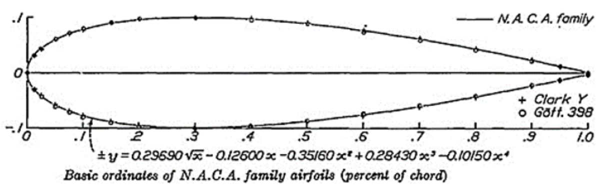
Parametrization is a key technique employed in airfoil optimization, allowing the airfoil's geometry to be represented numerically for manipulation by optimization algorithms. By defining a set of control points on the airfoil's surface, the shape of the airfoil is described in terms of these points. Through varying the positions of these control points, the optimization algorithm seeks to discover the optimal airfoil shape that satisfies the design objectives.

Parametrization can be accomplished using various approaches, such as Bezier curves, B-splines, and other mathematical functions. The selection of the parametrization method depends on the specific requirements of the design problem and the optimization algorithm employed.

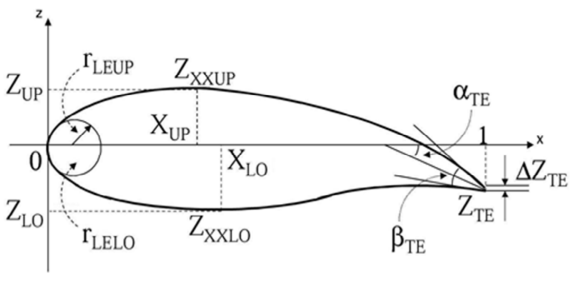
#### Popular examples

- *NACA series* : using a four-digit or five-digit designation [8].
- *FX (F. X. Wortmann) series* : developed at Stuttgart University, it focuses on airfoil designs for wind turbines with high lift and low drag characteristics.
- *RISØ (DTU Wind Energy) series* designed for wind turbine applications [9].
- *PARSEC* a parameterization targeted at representing subsonic and transonic airfoils [10].

The PARSEC parameters, and the NACA representation are described in figure 1.1 below :



(a) NACA 4 digits series [8]



(b) PARSEC parameters [11]

Figure 1.1: Parametrization examples

#### General steps

1. *Identify the design parameters:* These could include thickness, camber, and chord length, as well as more complex geometric features, like those in the PARSEC parametrization.
2. *Choose a parametrization method:* such as B-splines, NURBS, or polynomial curves. The choice of method will depend on factors like the complexity of the desired geometry and the computational resources available.
3. *Create an initial design:* A starting point is needed, so an initial design is created using a combination of empirical data and existing airfoil designs.
4. *Define the optimization objective:* The optimization objective is typically to maximize lift or minimize drag, while meeting other constraints such as thickness, camber, and structural requirements.
5. *Use optimization algorithms:* such as genetic algorithms, gradient-based methods, or particle swarm optimization, to find the optimal values of the design parameters.
6. *Evaluate the optimized airfoil:* Once the optimization process is complete, the resulting airfoil is evaluated using computational fluid dynamics simulations or wind tunnel tests to verify its performance.
7. *Refine and iterate:* The design parameters can be adjusted and the optimization process repeated until a satisfactory design is achieved.

#### Advantages and limitations

- The Parametrization Method has gained acceptance among researchers due to its capability of generating a wider range of new airfoil shapes. This method is based on two fundamental principles: completeness and controllability. These principles ensure that the generated shapes are fully defined and can be easily controlled and modified according to design requirements. Therefore, the parametrization Method is an efficient and reliable approach for airfoil design optimization.
- While the parametrization method offers several benefits, there are also some limitations to consider. One of the main disadvantages is that its accuracy highly depends on the chosen parametrization method and the selected parameters which can lead to suboptimal results.The method may involve a large number of variables, which can make the optimization problem computationally expensive and time-consuming. Moreover, the design space may be limited by the selected parameterization method, which may prevent the exploration of new and unconventional airfoil shapes [12].

### 1.2.2 Genetic Algorithms

Genetic algorithms (GA) are a type of optimization algorithm inspired by the process of natural selection in biology.
The basic idea behind a genetic algorithm is to create a population of candidate solutions, evaluate their fitness (i.e., how well they perform according to the design objectives), and use genetic operators such as selection, crossover, and mutation to generate new candidate solutions. This process is repeated for several generations until an optimal solution is found or a stopping criterion is met.

GA have emerged as a popular technique for achieving highly precise and optimal solutions for aerodynamic optimization problems. This method has been extensively used by researchers in the past few decades, primarily in the context of aircraft wings.

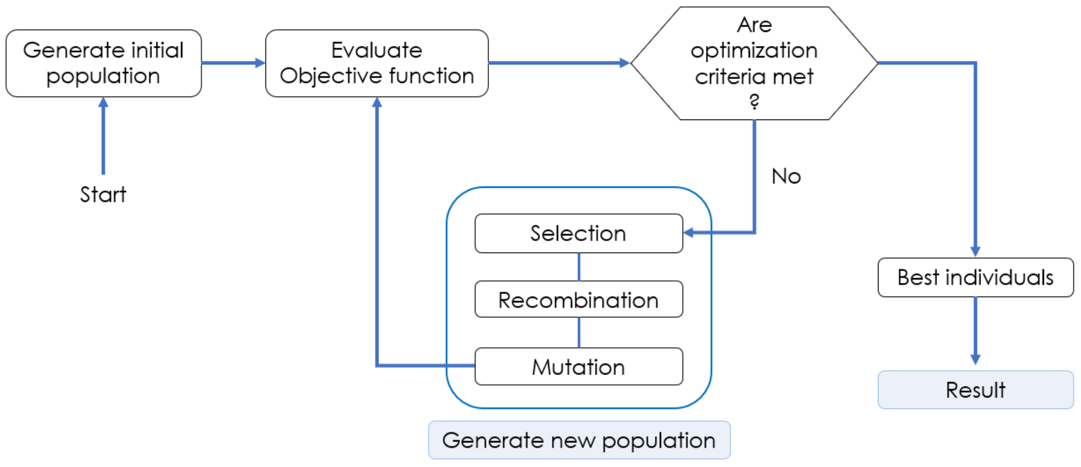

Generate initial population
Evaluate Objective function
Are optimization criteria met ?
Start
No
Selection
Recombination
Mutation
Generate new population
Best individuals
Result

Figure 1.2: Outline of the genetic algorithm

### General steps

- In airfoil optimization, the candidate solutions are typically represented as a set of geometric parameters that define the airfoil shape, such as the camber line, the thickness distribution, and the location of the maximum thickness [13]. These parameters can be varied within a certain range to generate a diverse population of candidate airfoils.
- To evaluate the fitness of each candidate solution, a computational fluid dynamics (CFD) simulation is typically used to calculate the aerodynamic performance of the airfoil, such as its lift and drag coefficients. The fitness value can be calculated based on the design objectives, such as the lift-to-drag ratio or the maximum lift at a given angle of attack.
- The genetic operators are then applied to the population of candidate solutions to generate new solutions.
  - Selection involves selecting the best-performing solutions from the current population to be used as parents for the next generation.
  - Crossover involves combining the genetic material of two parent solutions to create a new offspring solution.
  - Mutation involves randomly changing some of the genetic material of a solution to introduce new variations in the population.

By repeating this process for several generations, the genetic algorithm can search the solution space to find an optimal airfoil shape that satisfies the design objectives. The performance of the genetic algorithm depends on the choice of fitness function, the selection, crossover, and mutation operators, and the parameters that control the algorithm, such as the population size, the mutation rate, and the number of generations.

### Advantages and limitations

- Genetic algorithms are a powerful tool for airfoil optimization, as they can efficiently search the solution space to find an optimal airfoil shape that satisfies the design objectives. The use of genetic algorithms in airfoil optimization has led to the development of innovative airfoil designs that exhibit improved aerodynamic performance over traditional airfoils.
- However, the genetic algorithm can also suffer from issues such as premature convergence (where the algorithm gets stuck in a suboptimal solution) and the difficulty of choosing appropriate parameters and fitness functions. Therefore, careful parameter tuning and optimization of the fitness function are crucial for the success of the genetic algorithm.

## 1.2.3 CFD-based methods

Computational Fluid Dynamics (CFD), is a popular method used in airfoil optimization to simulate the flow of air around an airfoil. CFD simulations can be coupled with optimization algorithms, such as genetic algorithms or gradient-based optimization, to efficiently search for the optimal airfoil design [14].

### Types of the method

There are different types of CFD methods that can be used for airfoil optimization.

1. *Eulerian methods:* These are based on solving the Euler equations for fluid motion, which are a set of equations that describe the behavior of inviscid fluids (i.e. fluids with no viscosity).
2. *Navier-Stokes methods:* These are based on solving the Navier-Stokes equations for fluid motion, which are a set of equations that describe the behavior of viscous fluids (i.e. fluids with viscosity).
   Building upon this method, prominent sub-methods have been developed to correctly model turbulent flows and accurately capture their characteristics.
   - *Reynolds-averaged Navier-Stokes (RANS) methods:* These are based on averaging the Navier-Stokes equations over time and space to simplify the solution process.
   - *Large Eddy Simulation (LES) methods:* These are based on resolving the large-scale turbulent structures of the flow, while modeling the smaller-scale structures.
   - *Hybrid RANS/LES methods:* These are a combination of RANS and LES methods, which can be used to capture both the larger and smaller-scale turbulent structures.

### Advantages and limitations

- CFD methods provide accurate solutions for complex flow phenomena around airfoils. They also allow easy modifications to the geometry and boundary conditions, making it possible to test different design iterations. CFD methods provide detailed visualizations of the flow patterns and pressure distribution around the airfoil, while being cost-effective compared to experimental testing, which involves wind tunnel testing and model fabrication.

- However, CFD simulations require significant computational resources, including high-performance computing clusters and specialized software, which can be expensive and time-consuming. The accuracy is dependent on the chosen numerical models, which may not always represent the real-world behavior accurately.

### 1.2.4 Adjoint methods

Adjoint methods are a class of optimization methods used in airfoil design to efficiently calculate the sensitivity of a design objective to small changes in the design variables. These methods involve solving two separate sets of equations: the flow equations (usually Navier-Stokes or Euler equations) and the adjoint equations [4].

The flow equations describe the physical behavior of the fluid flow around the airfoil, while the adjoint equations describe the sensitivity of the objective function to small changes in the flow variables. The adjoint equations are derived by taking the gradient of the objective function with respect to the flow variables.

Adjoint methods have been applied to a variety of airfoil optimization problems, including the design of low-noise airfoils, high-lift airfoils for wind turbines, and airfoils for micro-air vehicles.

#### Advantages and limitations

- They allow simultaneous optimization of multiple design variables and multiple objective functions, while only requiring one flow solution. This makes them much more computationally efficient than traditional optimization methods that require multiple flow solutions.

  Traditional optimization methods require multiple flow solutions to evaluate each candidate design, which can be computationally expensive and time-consuming. On the other hand, adjoint methods use the solution of the adjoint equation to compute the sensitivities of multiple design variables with respect to multiple objective functions simultaneously. This means that only one flow solution is required to evaluate the sensitivities, which results in faster convergence to an optimal solution and also enabling the optimization of more complex designs with a larger number of design variables and objective functions.

- Adjoint methods require advanced mathematical knowledge and expertise, making them less accessible to those without a strong background in mathematics and optimization. They are also sensitive to initial conditions, and rely on the linearity of the underlying equations, which can limit the design space that can be explored. Another limitation is that adjoint methods are generally limited to steady-state flow problems, which can restrain their applicability to certain airfoil optimization problems.

### 1.2.5 Machine Learning

In recent years, increasing work has been directed to utilizing machine learning algorithms to synthesize accurate airfoil shapes while reducing the required computational cost. There are several machine learning methods that have been used for airfoil optimization, including:

#### Support Vector Machines (SVM)

SVM is a machine learning algorithm that can be used to classify data or predict outcomes. In airfoil optimization, SVM can be used to predict the performance of different airfoil designs based on their design parameters.

#### Kriging

Kriging is a statistical method that can be used to build a surrogate model of the relationship between design parameters and performance metrics. This surrogate model can be used to optimize the airfoil design for a given set of objectives [15].

#### Random Forest

Random Forest is an ensemble learning method that can be used for regression and classification problems. In airfoil optimization, Random Forest can be used to predict the performance of different airfoil designs based on their design parameters.

#### Genetic Programming

Genetic Programming is a machine learning technique that can be used to evolve a population of computer programs that can solve a given problem. In airfoil optimization, Genetic Programming can be used to evolve an optimal airfoil design based on a set of objectives.

#### Bayesian Optimization

Bayesian Optimization is a machine learning technique that can be used to optimize a function that is expensive to evaluate. In airfoil optimization, Bayesian Optimization can be used to find the optimal airfoil design based on a set of objectives while minimizing the number of evaluations required.

#### Artificial neural networks (ANN)

Neural networks are used to build a mapping between airfoil design parameters and performance metrics. The neural network can then be used to optimize the airfoil design for a given set of objectives. ANNs are composed of layers of interconnected nodes, which can be trained to approximate a mapping between input and output data. In the context of airfoil optimization, ANNs can be trained to predict the aerodynamic performance of an airfoil based on its geometry. This can be useful for optimizing airfoil shapes for specific performance objectives.

In the next section, one of these ANN techniques is emphasized and discussed in detail, as it is the main method that will be used in this work.

## 1.3 Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs) are a powerful tool for generating realistic and diverse data samples, such as images, texts, and shapes. They are a type of machine learning technique that can be used for airfoil optimization.
GANs consist of two neural networks, a generator and a discriminator, that are trained together in a game-like setting to generate realistic and novel airfoil designs [16]. In GAN-based airfoil optimization, the generator network takes a random input, or noise vector, and generates an airfoil design as its output. The discriminator network then evaluates the generated airfoil design and provides feedback to the generator network to improve the design. This feedback is based on how well the generated airfoil design resembles the real samples from the training set.

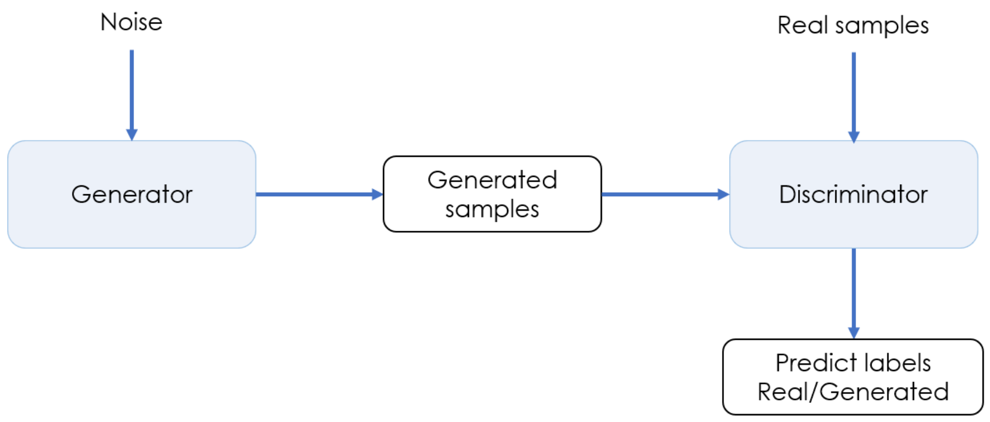


Figure 1.3: Generative Adversarial Network

### 1.3.1 Training

Through the iterative process described above, the generator learns to generate airfoil designs that are similar to the ones in the training set and that meet the optimization criteria. The discriminator network also becomes better at distinguishing generated samples from real ones. This leads to the creation of highly optimized and unique airfoil designs.

Training GANs can be a challenging task. This is because the generator and the discriminator networks compete against each other during the training. In fact, if one network learns too quickly, then the other network may fail to learn. This can often result in the network not being able to converge. To diagnose issues and monitor on a scale from 0 to 1 how well the generator and discriminator achieve their respective goals, their scores can be plotted [17].
In the ideal case, both scores would be 0.5. This is because the discriminator cannot tell the difference between real and fake sample, therefore there is a 50% chance that the sample is from the real or generated set. The ideal value of 0.5 can't be reached, however, a successful GAN is achieved whenever the scores converge closely towards it.

### 1.3.2 Types of GANs

There are many variants of GAN. The most important ones include :

#### Conditional GAN (CGAN)

Conditional GANs are simple GANs with additional input provided for both the generator and discriminator. Instead of generating new samples given a noise vector only, the generator will additionally receive class labels to generate new samples with the desired characteristics. To learn these characteristics, the discriminator is also fed with a labelled training set.

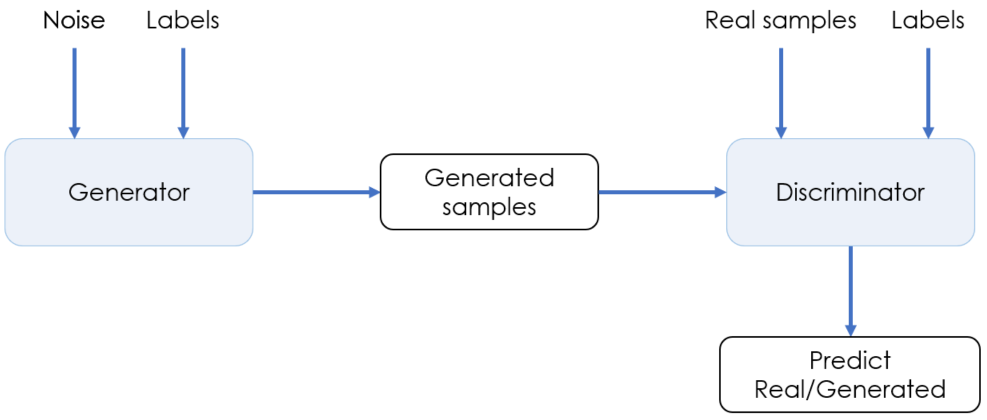


Figure 1.4: Conditional GAN

#### GANs with alternative architectures

In the original paper that introduced GANs, multilayer perceptron networks and convolutional neural networks were used to define the architecture of the generator and discriminator. But many alternative architectures have been used [18], such as :

- Deep convolutional GAN (DCGAN) : using only convolution-deconvolution layers
- Self-attention GAN (SAGAN) : a DCGAN with residually-connected standard self-attention modules at the end of the generator and discriminator.
- Variational auto-encoder GAN (VAEGAN) : defines the generator as a variational auto encoder.
- Transformer GAN (TransGAN): Defines both the generator and discriminator as pure transformers, without any convolution-deconvolution layers.

#### Wasserstein GAN (WGAN)

This type of GANs was proposed to improve the stability of the learning.
In the Wasserstein GAN game, the discriminator provides a better gradient, in other words it provides the generator with a better learning signal. As a result, failure modes are avoided and the learning is more stable especially when it is done with high dimensional spaces.

## 1.4 Previous works

The use of GANs for airfoil optimization is still a relatively new area of research, but has shown promising results in terms of computational efficiency and effectiveness in generating optimized airfoil designs. GANs have been successfully applied to airfoil optimization in various ways, such as inverse design, shape generation, shape optimization, and feature learning. In this bibliographical review, we survey the previous works and papers that have used GANs for airfoil shape optimization.

One of the earliest works that applied GANs to airfoil optimization was in 2019 by Chen et al.[19], who used Bézier-GANs to learn realistic shape variations from the UIUC airfoil database and synthesize a new latent space in which design optimization was conducted using gradient-based methods. This helped to reduce the number of design variables, and accelerated convergence to find optimal designs.

In 2020, Yilmaz and German [20] and Achour et al. [21] have proposed a conditional generative adversarial network (CGAN) framework for airfoil inverse design, by applying the generative model to associate the aerodynamic parameters with the airfoil geometry. In their work, they used a vector of conditional data indicating desired performance metrics, such as stall condition or drag polar, to generate new airfoil shapes via deep convolutional neural networks (ConvNets). They trained their CGAN on a database of airfoil shapes and conditional information, and demonstrated that their framework could create realistic and novel airfoils based on specified inputs. Du et al.[22], also proposed a new hybrid parameterization method by coupling B-Spline and GAN to filter out abnormal shapes. This method was able to achieve a fast aerodynamic optimization.

More recent works in 2022 have used GANs for airfoil optimization. Wang et al.[23], proposed an airfoil GAN that could encode and synthesize airfoils for aerodynamic-aware shape optimization. Their model was built upon VAEGAN, a neural network that combined variational auto-encoder with GAN, and was trained by the gradient-based technique. Their model could encode existing airfoils into latent vectors and reconstruct them from that, generate novel airfoils by randomly sampling latent vectors and mapping them to the airfoil coordinate domain, and synthesize airfoils with desired aerodynamic properties by optimizing learned features via a genetic algorithm. Another recent work was by Tan et al.[24], who used a conditional GAN (cGAN) based framework with various filter layers for airfoil inverse design problem. They proposed a custom network, by implementing a Wasserstein loss with gradient penalty to the GAN architecture, forming a cWGAN-GP. This model proved to be useful in generating different classes of airfoils with the specified characteristics.

## 1.5 Scope of the study

This work is focused on developing a simple and computationally efficient GAN network for generating new airfoil shapes.

Most of the existing GAN models require high computational resources especially during the training phase. Therefore, we aim to propose a simplified network that is accessible and resource-friendly. This enables the exploration of new airfoil designs while having limited computational resources.

Thus, the MATLAB code ***(airfoil_GAN.m)*** developed in this work, is consisted of a simple GAN that takes only one input which is the airfoils coordinates from the UIUC database, and returns new generated shapes. The code is able to generate as much airfoils as wanted, from which are selected the ones with the desired lift-to-drag ratio.

## 1.6 Conclusion

This chapter provided an overview of the various techniques used in airfoil optimization, with a particular focus on GANs. The reviewed works demonstrated the effectiveness and potential of generative models in optimizing airfoil designs, as they have shown promising results in improving aerodynamic performance, reducing design variables and filtering abnormal geometries. Moreover, by acknowledging the computational complexity and constraints associated with the previously used GAN models, we set the scope of our study which aims to contribute to the field by developing a simplified and computationally efficient model for airfoil optimization using GANs.

# Chapter 2
# Methodology

## 2.1 Introduction

This second chapter provides a theoretical overview of the three key elements used in this work, that is : GANs used to generate new airfoils, the panel method implemented through XFOIL which served as an aerodynamic evaluation tool, and at last the Blade Element momentum theory (BEMT) used in the final validation part.

GANs were already presented in the first chapter, so the focus will now be on their mathematical aspect. The panel method is briefly introduced in a general manner, as it wasn't implemented or programmed, but it is rather to understand how XFOIL works. The BEMT on the other hand, is extensively detailed since it was implemented in the developed code.

## 2.2 Generative Adversarial networks - Mathematical aspect

### 2.2.1 Definition

From a mathematical point of view, a Generative Adversarial Network (GAN) is defined as follows [25] :

A GAN is a minimax game defined within a probability space $(\Omega, \mu_{ref})$, played by two agents : the *Generator* $G$ and the *Discriminator* $D$.

The strategy set of the generator is $\mu_g = \mathcal{P}(\Omega)$, a probability distribution of the new samples $G(z)$ obtained when $z \sim p_z$; an input noise variables.

The strategy set of the discriminator is the set of Markov Kernels $\mu_D : \Omega \to \mathcal{P}[0, 1]$, the discriminative distribution $D(x)$ which represents the probability that $x$ came from $\mu_{ref}$ rather than $\mu_g$.

So the generator generates new samples from a random input, the output is then passed to the discriminator D that will assign a label of whether the input is from the generating or the generative distribution.

During the adversarial training, $D$ seeks to maximize the probability of assigning the correct label, whereas $G$ is trained to minimize $\log(1 - D(G(z)))$.
Thus a single loss function (value function) can be used :

$$L = \min_G \max_D V(D, G) = \mathbb{E}_{x \sim \mu_{ref}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(x)} [\log (1 - D(G(z)))] \tag{2.1}$$

The game terminates at a saddle point that is a minimum with respect to one player's strategy and a maximum with respect to the other player's strategy.

### 2.2.2 Main theorems

Two main theorems for a GAN game were presented and proved in the original article that introduced GANs [25].

#### Theorem 1 : Optimal discriminator

The optimal discriminator computes the *Jensen-Shannon divergence.*

For a given generator strategy $\mu_g$, the optimal reply is :

$$D^* = \arg\max_D L(\mu_G, D) \tag{2.2}$$

$$D^*(x) = \frac{d\mu_{ref}}{d(\mu_{ref} + \mu_G)} \tag{2.3}$$

$$L(\mu_G, D^*) = 2D_{JS}(\mu_{ref}; \mu_G) - 2\log 2 \tag{2.4}$$

Where the derivative is *Radon-Nikodym derivative*, used in measure theory to express the relationship between two measures. And $D_{JS}$ is the *Jensen-Shannon divergence* , a method of measuring the similarity between two probability distributions.

▶ Interpretation

For any given strategy set of the generator, the optimal discriminator calculates the likelihood ratio between the reference distribution of the training set and the generator distribution.

#### Theorem 2 : Unique equilibrium point

For any GAN game, there is a pair of strategies $(\hat{\mu}_D, \hat{\mu}_G)$, that is a sequential equilibrium and a Nash equilibrium at the same time.
In a Nash equilibrium, each player is assumed to know the equilibrium strategies of the other players, and no one has anything to gain by changing only one's own strategy. The sequential equilibrium is a refinement of Nash equilibrium.

$$L(\hat{\mu}_G, \hat{\mu}_D) = \min_{\mu_G} \max_{\mu_D} L(\mu_G, \mu_D) = \max_{\mu_D} \min_{\mu_G} L(\mu_G, \mu_D) = -2\log 2 \tag{2.5}$$

$$\hat{\mu}_D \in \arg\max_{\mu_D} \min_{\mu_G} L(\mu_G, \mu_D), \qquad \hat{\mu}_G \in \arg\min_{\mu_G} \max_{\mu_D} L(\mu_G, \mu_D) \tag{2.6}$$

$$\hat{\mu}_D \in \arg\max_{\mu_D} \min_{\mu_G} L(\hat{\mu_G}, \mu_D), \qquad \hat{\mu}_G \in \arg\min_{\mu_G} \max_{\mu_D} L(\mu_G, \hat{\mu_D}) \tag{2.7}$$

$$\forall x \in \Omega, \hat{\mu}_D(x) = \delta_{\frac{1}{2}}, \qquad \hat{\mu}_G = \mu_{ref} \tag{2.8}$$

▶ Interpretation

The GAN game reaches its end, when the generator perfectly mimics the reference distribution, and the discriminator can no longer distinguish between the reference and generator distribution, and outputs $\frac{1}{2}$ on all inputs.

## 2.3 XFOIL : Panel method and boundary layer model

### 2.3.1 XFOIL

XFOIL is an interactive program widely used for the design and analysis of subsonic airfoils. It was developed by Mark Drela at MIT for the MIT Daedalus project in the 1980s [26]. The version that will be used in this work is the current version 6.99, released in December 2013.

XFOIL uses the panel method coupled with boundary layer calculations to predict the aerodynamic coefficients of airfoils, that is lift, drag, and moment coefficients, as well as pressure and velocity distributions over the airfoil's surface. Given the coordinates specifying the shape of a 2D airfoil, Reynolds and Mach numbers, it can predict accurately and efficiently the aerodynamic performance of the airfoil for various angles of attack.

### 2.3.2 Panel Method

The panel method is a method used to analyze the inviscid flow around an object and determine the pressure forces acting on its surface. It is a numerical approximation that relies on dividing the body's surface into a series of panels or segments and then assigning a potential flow element (such as a vortex, doublet, source or sink) to each element(Figure 2.1). So mathematically, the panels are modeled as singularities that induce flow effects [27].

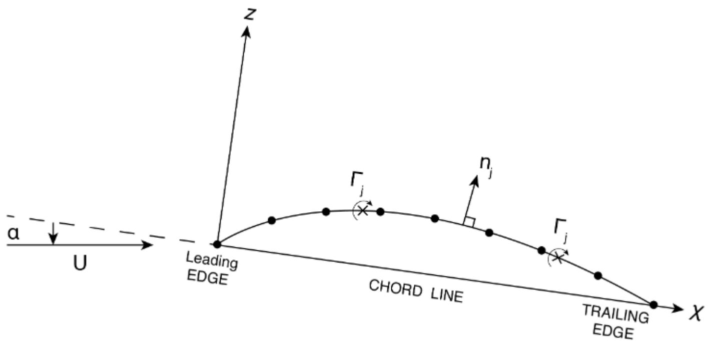


Figure 2.1: Vortex Panels geometry over an airfoil [27]

The following assumption are made :

- Potential flow governed by the laplace equation of the velocity flow :

$$\nabla^2 \phi = 0 \tag{2.9}$$

- The strength of the potential flow remains constant along each panel.
- The interaction of the elements is accounted.
- Far from the object the flow should be equal to the free stream velocity approaching the object.
- The boundary condition is that of an impermeable surface, where the velocity normal to the surface is zero

$$\nabla \phi . \mathbf{n} = 0 \tag{2.10}$$

The goal is to determine the velocity on the surface, and then deduce the local pressure distribution using Bernoulli's equation. The forces being applied on the object can be calculated by integrating the pressure over the surface.

### Source panel method

After dividing the airfoil's surface into $N$ panels, a source distribution of constant strength $\sigma_i$ is placed on each element.

Given that the velocity potential at a point $P$ caused by a source distribution of strength $\sigma(Q)$ positioned at a point $Q$ is :

$$\phi(P) = \int_S \frac{\sigma(Q)}{2\pi} \ln(r(P,Q))ds \tag{2.11}$$

At every panel, $\phi$ is expressed as a function of the source strengths $\sigma$, by taking into account the contribution of all the other panels.

The flow velocity $\partial\phi/\partial\mathbf{n}$ normal to the airfoil is then determined at each panel and set to 0.

This results in $N$ equations with $N$ unknown strengths $\sigma_i$, that can be put in a matrix of the form :

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,N} \\ \vdots & \ddots & \vdots \\ A_{N,1} & \cdots & A_{N,N} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \tag{2.12}$$

The left term of the above equation represents the velocities induced by the source distributions, whereas the right term takes into account the free flow velocity $V_\infty$ and the angle of attack $\alpha$.

However, sources and sinks cannot produce any lift or drag forces, that is why it is necessary to consider a vortex distribution.

### Vortex Panel method

Similarly to the source panel method, the airfoil is divided into $N$ panels, to which are assigned vortices distributions with strengths $\gamma_i$.

At each panel, $\phi$ is expressed as a function of the vortex strengths $\gamma_i$, by taking into account the contribution of all the other panels.

The flow velocity $\partial\phi/\partial\mathbf{n}$ normal to the airfoil is then determined at each panel and set to 0.

This results in $N$ equations with $N$ unknown strengths $\gamma_i$.

For the vortex panel method, an important condition must be incorporated, that is the Kutta condition. This condition states that the flow must leave the trailing edge smoothly. To ensure this, the velocities on both the upper and lower sides of the trailing edge must be equal, as indicated in the figure below :

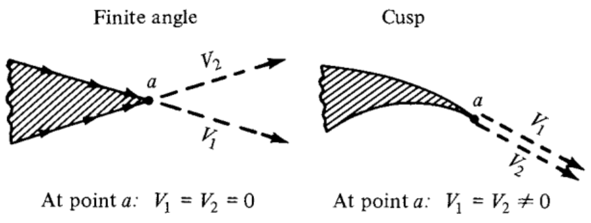

Finite angle
Cusp
$V_2$
$a$
$V_1$
$a$
$V_1$
$V_2$
At point $a$: $V_1 = V_2 = 0$
At point $a$: $V_1 = V_2 \neq 0$

Figure 2.2: Kutta condition [5]

In terms of the strengths of the vortices, the kutta condition is expressed as follows :

$$\gamma(\text{TE}) = 0 \tag{2.13}$$

If both panels around the TE correspond to the first and last ones, then :

$$\gamma_1 + \gamma_N = 0 \tag{2.14}$$

Adding the Kutta condition constitute an over-determined system of $n$ unknowns and $N+1$ equations. Therefore one equation among the $N$ equation obtained above, is abandoned.

The resulting system can be put in a matrix of the form :

$$\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,N-1} & A_{1,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-1,1} & A_{N-1,2} & \cdots & A_{N-1,N-1} & A_{N-1,N} \\ 1 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \vdots \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_{N-1} \\ 0 \end{bmatrix} \tag{2.15}$$

Once the system is solved, the **Kutta-Joukowski theorem** is applied in order to calculate the lift force applied :

$$F_L = \rho_\infty V_\infty \Gamma \tag{2.16}$$

With $\Gamma$ the total circulation :

$$\Gamma = \sum_{j=1}^{n} \gamma_j s_j \tag{2.17}$$

$s_j$ is the length of the $j$th panel.

The lift coefficient can also be calculated :

$$C_l = \frac{2\Gamma}{V_\infty c} \tag{2.18}$$

$c$ is the chord length of the airfoil.

### 2.3.3 Boundary layer model

In addition to the inviscid flow field calculations, integral boundary layer and transition equations are simultaneously solved by XFOIL.
To represent the viscous layers, XFOIL uses a two-equation lagged dissipation integral method [28]. This involves standard compressible integral momentum and kinetic energy shape parameter equations, and a rate equation for the maximum shear stress coefficient to account for deviations of the outer layer dissipation coefficient from the local equilibrium [26].
For further reading, see [26] the original article of Mark Drela that presented XFOIL.

## 2.4 Blade Element Momentum Theory - BEMT

### 2.4.1 Linear Momentum Theory

The simple aerodynamic model of an ideal turbine rotor, is based on a linear momentum theory. This model assumes a control volume, in which boundaries are the surface of a stream tube and two cross-sections of the stream tube (Figure 2.3), and where the flow is one-dimensional. The turbine is represented by a uniform actuator disc that creates a discontinuity of pressure in the stream tube of air flowing through it [29].
The following assumptions are to be considered for this model :

- homogenous, incompressible, steady state fluid flow;
- no frictional drag;
- an infinite number of blades;
- uniform thrust over the disc or rotor area;
- a non–rotating wake;
- the static pressure far upstream and far downstream of the rotor is equal to the undisturbed ambient static pressure

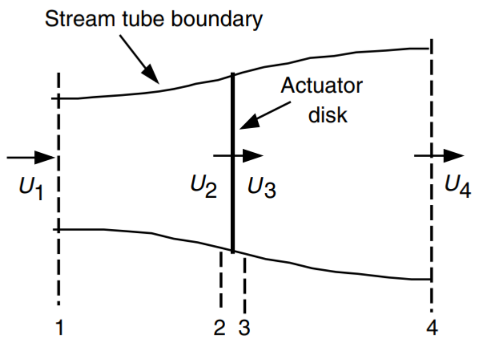


Figure 2.3: Actuator disc model of a wind turbine [29]

From the conservation of linear momentum, the thrust is given by :

$$T = U_1(\rho A U)_1 - U_4(\rho A U)_4 \tag{2.19}$$

where $\rho$ is the air density, $A$ is the cross-sectional area, $U$ is the air velocity, and the subscripts indicate values at numbered cross-sections in (Figure 2.3).

The conservation of mass flow rate through the streamtube would be :

$$\dot{m} = (\rho AU)_1 = (\rho AU)_2 = (\rho AU)_3 = (\rho AU)_4 \tag{2.20}$$

Therefore :

$$T = \dot{m}(U_1 - U_4) \tag{2.21}$$

By applying the Bernoulli equation in the stream tube upstream of the disc :

$$p_1 + \frac{1}{2}\rho U_1^2 = p_2 + \frac{1}{2}\rho U_2^2 \tag{2.22}$$

Similarly, downstream the disc :

$$p_3 + \frac{1}{2}\rho U_3^2 = p_4 + \frac{1}{2}\rho U_4^2 \tag{2.23}$$

Where it is assumed that the far upstream and far downstream pressures are equal ($p_1 = p_4$) and that the velocity across the disc remains the same ($U_2 = U_3$).

Thrust can also be expressed as :

$$T = A_2(p_2 - p_3) \tag{2.24}$$

Considering Bernoulli equations (2.22) and (2.23), we could substitute ($p_2 - p_3$) in equation (2.24) and get :

$$T = \frac{1}{2}\rho A_2(U_1^2 - U_4^2) \tag{2.25}$$

Given equation (2.20) and equating thrust from equations (2.21) and (2.25)

$$U_2 = \frac{U_1 + U_4}{2} \tag{2.26}$$

The actuator disc induces a velocity in the streamtube. The difference between $U_1$ and $U_2$ is called the induced axial velocity. This velocity is represented by a dimensionless factor called the axial induction factor, denoted as $a$.

$$a = \frac{U_1 - U_2}{U_1} \tag{2.27}$$

Thus, the wind velocity at the rotor plane is

$$U_2 = U_1(1 - a) \tag{2.28}$$

And

$$U_4 = U_1(1 - 2a) \tag{2.29}$$

From equations (2.25), (2.28) and (2.29), Thrust is deduced as :

$$T = \frac{1}{2}\rho A U^2 \left[4a(1-a)\right] \tag{2.30}$$

Where $A_2$ is replaced by $A$, the rotor area, and $U_1$ is replaced by $U$, the free stream velocity.

The power output $P$, is equal to the thrust times the velocity at the disc, thus :

$$P = \frac{1}{2}\rho A U^3 \left[4a(1-a)^2\right] \tag{2.31}$$

The power coefficient $C_p$ is defined as the fraction of the power in the wind that is extracted by the rotor. It characterizes the performance of the wind turbine.

$$C_p = \frac{Rotor \quad Power}{Power \quad in \quad the \quad wind} = \frac{P}{\frac{1}{2}\rho A U^3} = 4a(1-a)^2 \tag{2.32}$$

Deriving the power coefficient with respect to $a$ and setting it equal to zero, we can get the maximum value of $C_p$ known as the Betz Limit :

$$C_{p_{max}} = \frac{16}{27} = 0.5926 \tag{2.33}$$

Similarly, a thrust coefficient is defined :

$$C_T = \frac{Thrust \quad force}{Dynamic \quad force} = \frac{T}{\frac{1}{2}\rho A U^2} = 4a(1-a) \tag{2.34}$$

Figure 2.3 illustrates a graph of the power and thrust coefficients for an ideal Betz turbine and the non-dimensionalized downstream wind speed. We can see that the idealized model is not valid for axial induction factors greater than 0.5.

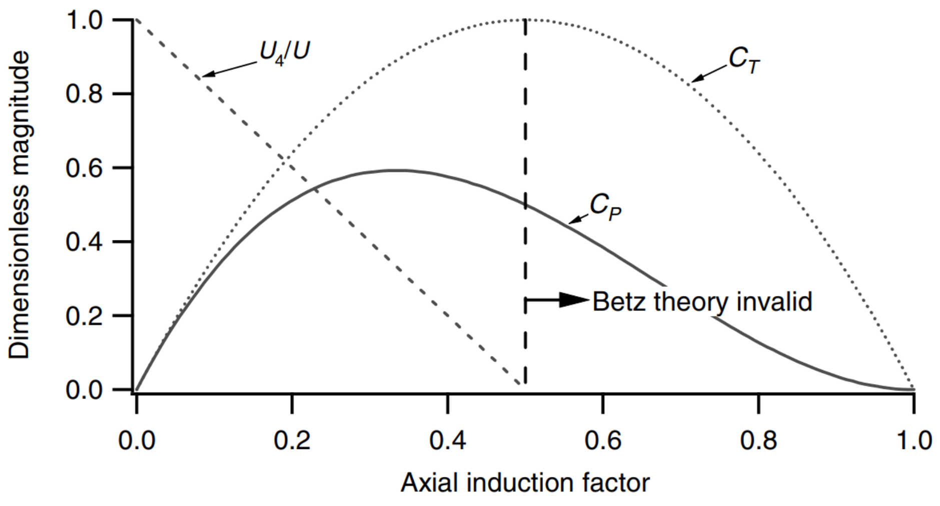


Figure 2.4: Operating parameters for a Betz turbine [29]

### 2.4.2 Ideal Horizontal Axis Wind Turbine with Wake Rotation

In the previous model, the flow was considered one-dimensional. It can be extended to the case where the rotating rotor generates angular momentum, which can be related to rotor torque. In the case of a rotating wind turbine rotor, the flow behind the rotor rotates in the opposite direction to the rotor. Figure 2.5 shows an annular stream tube model of this flow, illustrating the rotation of the wake.

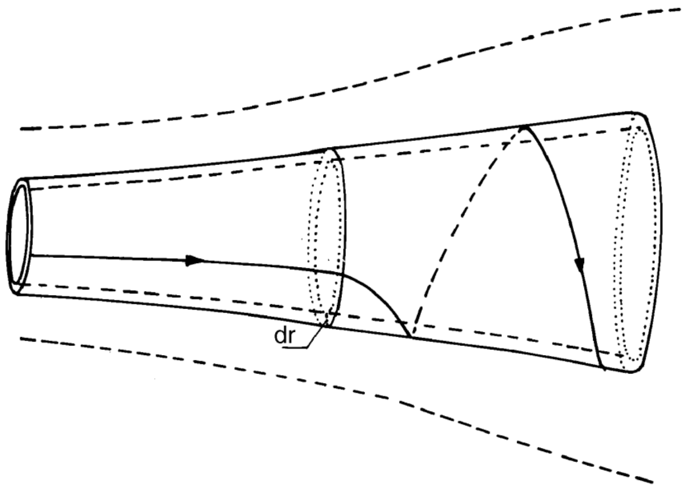


Figure 2.5: stream tube model with wake rotation [29]

The flow is modeled using an annular streamtube with a radius of $r$ and thickness of $dr$, with a cross-sectional area equal to $2\pi r dr$ (Figure 2.6).

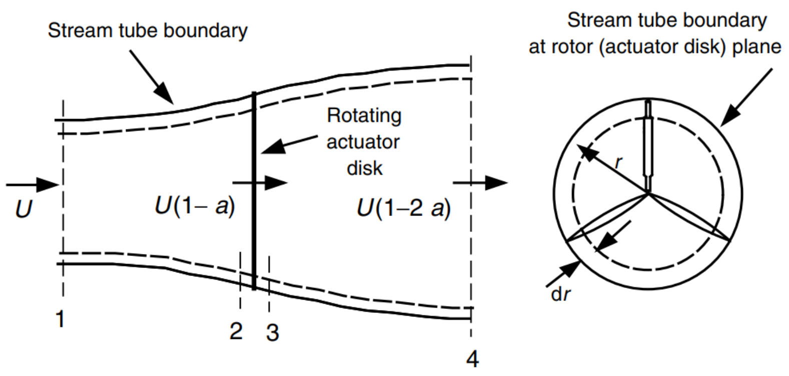


Figure 2.6: Geometry for rotor analysis [29]

The angular velocity of the wake rotation, $\omega$, is lower than that of the rotor, $\Omega$. Assuming a control volume with a rotation equal to that of the blades and applying the energy conservation equation, Glauert was able to determine the pressure difference at the turbine by noting that the axial velocity of the air remains constant, and that its tangential component increases from $\Omega$ to $\Omega + \omega$. This results in :

$$p_2 - p_3 = \rho \left(\Omega + \frac{1}{2}\omega\right) \omega r^2 \tag{2.35}$$

The trust on an annular element is then :

$$dT = \left[\rho \left( \ \Omega + \frac{1}{2}\omega\right) \omega r^2\right] 2\pi r dr \tag{2.36}$$

An angular induction factor $a'$ is defined as :

$$a' = \frac{\omega}{2\Omega} \tag{2.37}$$

Now that the wake rotation is included, the induced velocity at the rotor consists of another component in addition to the axial one $U\,a$, that is a component in the rotor plane $r\,\Omega\,a\prime$. The thrust becomes :

$$dT = \rho \Omega^2 r^2 \left[4a'(1 + a')\right] \pi r dr \tag{2.38}$$

The thrust can also be determined using the induction factor $a$, by replacing $A$ in equation (2.30) with the area of an annular cross-section $2\pi r\,dr$ :

$$dT = 4a\,(1 - a)\,\rho U^2\,\pi r\,dr \tag{2.39}$$

Equating (2.38) and (2.39), we get :

$$\frac{a(1-a)}{a'(1+a')} = \frac{\Omega^2 r^2}{U^2} = \lambda_r^2 \tag{2.40}$$

Where $\lambda_r$ is the local speed ratio.
The tips speed ratio (TSR) $\lambda$ is defined as the ratio of the blade tip speed to the free stream wind speed :

$$\lambda = \frac{\Omega R}{U} \tag{2.41}$$

An expression of the torque exerted on the rotor can be derived. The torque $Q$ is equal to the change in angular momentum of the wake. This gives :

$$dQ = d\dot{m}(\omega r)(r) = (\rho U_2 2\pi r dr)(\omega r)(r) \tag{2.42}$$

Considering equations (2.28) and (2.37), the expression of the torque becomes :

$$dQ = \rho U \Omega r^2 \left[4a'(1 - a)\right] \pi r dr \tag{2.43}$$

The power generated at each element is given by :

$$dP = \Omega dQ \tag{2.44}$$

### 2.4.3 Blade Element Theory

In this theory, the blade is assumed to be divided into $N$ elements, as shown in figure 3.5b, and the following assumptions are made :

- No aerodynamic interaction between the elements (no radial flow).
- The forces on the blades are determined solely by the lift and drag characteristics of the airfoil shape of the blades.

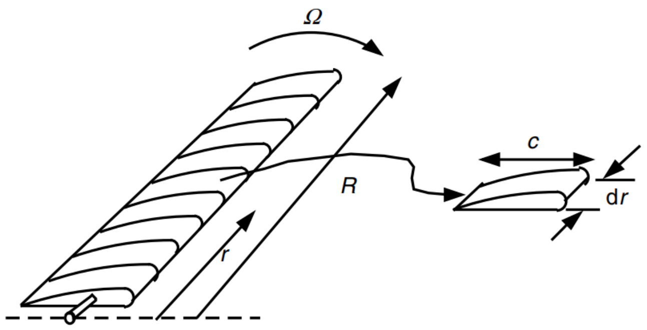


Figure 2.7: Blade elements [29]

As it was explained in the general introduction, the lift and drag forces are perpendicular and parallel, respectively, to the flow direction. In this analysis, the flow in question is an effective, or relative, wind. The relative wind is the vector sum of the wind velocity at the rotor $U(1-a)$, and the velocity due to the blade rotation, which itself is a vector sum of the blade section velocity $\Omega r$ and the induced angular velocity from the conservation of angular momentum $\omega r/2$

$$\Omega r + (\omega/2)r = \Omega r + \Omega a' r = \Omega r(1 + a') \tag{2.45}$$

Figure 2.8 shows the different forces, angles, velocities at the blade, and the relationships between them.

$dF_L$ and $dF_D$ are the incremental lift and drag forces, respectively. $dF_N$ is the incremental force normal to the plane of rotation and which contributes to thrust. $dF_T$ is the incremental force tangential to the circle swept by the rotor, and which is responsible of creating useful torque.

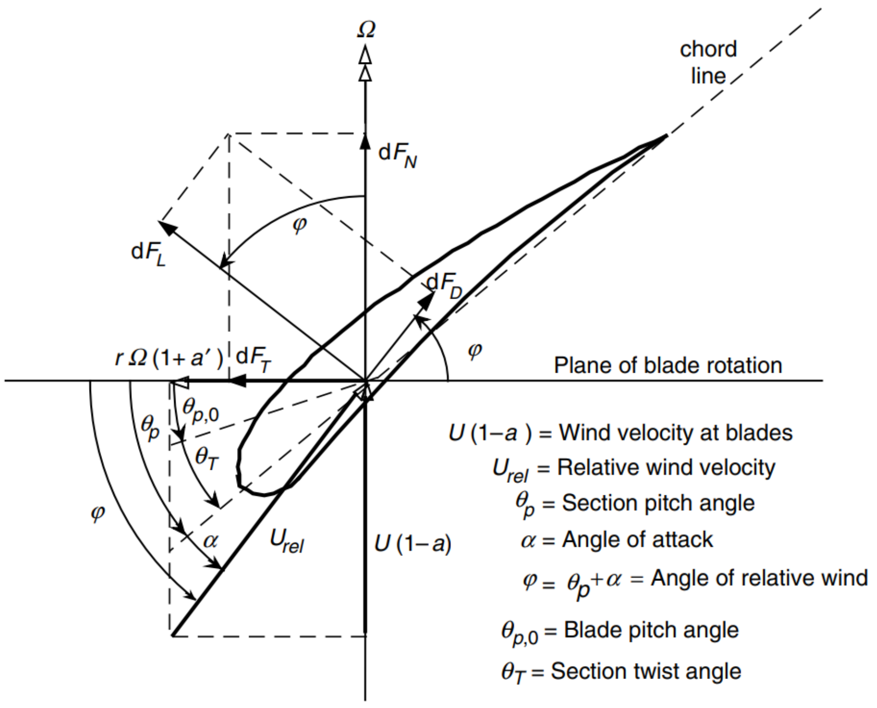

Ω
chord line
$dF_N$
$dF_L$
$\varphi$
$dF_D$
$\varphi$
Plane of blade rotation
$r\Omega(1+a')$
$dF_T$
$\theta_{p,0}$
$\theta_p$
$\theta_T$
$\varphi$
$\alpha$
$U_{rel}$
$U(1-a)$
$U(1-a)$ = Wind velocity at blades
$U_{rel}$ = Relative wind velocity
$\theta_p$ = Section pitch angle
$\alpha$ = Angle of attack
$\varphi = \theta_p + \alpha$ = Angle of relative wind
$\theta_{p,0}$ = Blade pitch angle
$\theta_T$ = Section twist angle

Figure 2.8: Geometrical variables for analysis of a HAWT [29]

The twist angle $\theta_T$, which is a function of the blade geometry, is defined as :

$$\theta_T = \theta_p - \theta_{p,0} \tag{2.46}$$

From the figure above, we can set the following equations :

$$\tan\varphi = \frac{U(1-a)}{\Omega r(1+a')} = \frac{1-a}{(1+a')\lambda_r} \tag{2.47}$$

$$U_{rel} = \frac{U(1-a)}{sin\varphi} \tag{2.48}$$

$$dF_L = C_l\,\frac{1}{2}\rho U_{rel}^2 cdr \tag{2.49}$$

$$dF_D = C_d\,\frac{1}{2}\rho U_{rel}^2 cdr \tag{2.50}$$

$$dF_N = dF_L\cos\varphi + dF_D\sin\varphi \tag{2.51}$$

$$dF_T = dF_L\sin\varphi - dF_D\cos\varphi \tag{2.52}$$

For $B$ number of blades, and substituting equations (2.49) and (2.50) in (2.51), the thrust $dT$ is given by :

$$dT = B\,dF_N = B\,\frac{1}{2}\rho U_{rel}^2\,(C_l\cos\varphi + C_d\sin\varphi)\,c\,dr \tag{2.53}$$

The differential torque due to the tangential force, at a distance $r$ is :

$$dQ = Br\,dF_T \tag{2.54}$$

$$dQ = B\,\frac{1}{2}\rho U_{rel}^2\,(C_l\sin\varphi - C_d\cos\varphi)\,c\,r\,dr \tag{2.55}$$

Using equation (2.48), we get :

$$dT = \sigma'\pi\rho\,\frac{U^2(1-a)^2}{\sin^2\varphi}\,(C_l\cos\varphi + C_d\sin\varphi)\;r\,dr \tag{2.56}$$

$$dQ = \sigma'\pi\rho\,\frac{U^2(1-a)^2}{\sin^2\varphi}\,(C_l\sin\varphi - C_d\cos\varphi)\;r^2\,dr \tag{2.57}$$

Where $\sigma'$ is the local solidity defined as :

$$\sigma' = \frac{Bc}{2\pi r} \tag{2.58}$$

### 2.4.4 BEMT

The BEMT results from coupling the two theories presented earlier.
By equating the torque expressions from equations (2.57) and (2.43), and the thrust expressions from equations (2.40) and (2.56), we could calculate the aerodynamic performance of a wind turbine through an iterative process that allows to determine the induction factors $a$ and $a'$. The algorithm of the method is presented in the next section. But first, the evaluation of the lift and drag coefficients, as well as the correction models used are to be discussed.

#### Evaluation of $C_l$ and $C_d$

The lift and drag coefficients depend on two parameters : the angle of attack ($\alpha$) and the Reynolds number ($Re$) defined as :

$$Re = \frac{U_{rel}c}{\nu} \tag{2.59}$$

where $\nu$ is the kinematic viscosity of air.

These coefficients are determined by conducting flow simulations over the two dimensional airfoil shape using dedicated software like XFOIL. The calculated values are referred to as the 2D coefficients.

When the angle of attack exceeds the critical angle of stall ($\alpha_{stall}$), the airflow will separate from the surface of the airfoil, resulting in a drop in lift, as shown in Figure 2.9.

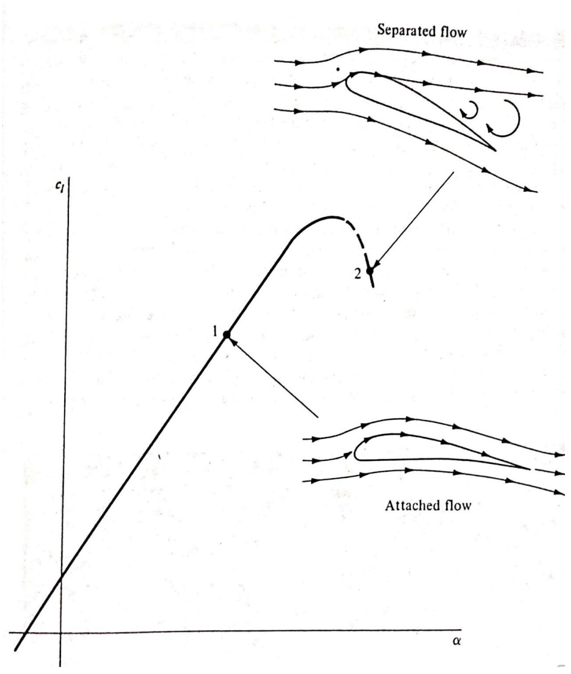


Figure 2.9: Airfoil stall[30]

Around this stall region, the dedicated software like XFOIL are unable to converge and to predict correct lift and drag coefficients. However, it is necessary to have these values for a wider range of angles of attack when calculating a turbine's performance with the BEMT algorithm. For this reason, many extrapolation techniques were established. The most recognized among them are the Viterna and the flat plate methods, which are detailed below.

- Viterna extrapolation

The Viterna method allows extrapolation of the $C_l$ and $C_d$ coefficients beyond the stall region, up to an angle of attack less than or equal to 90°, by using the following equations :

$$C_l = A_1 \sin(2\alpha) + A_2 \frac{\cos^2\alpha}{\sin\alpha} \tag{2.60}$$

$$C_d = B_1 \sin^2\alpha + B_2 \cos\alpha \tag{2.61}$$

Where :

$$C_{d_{max}} \approx 1.11 + 0.018\, AR \tag{2.62}$$

$$A_1 = \frac{C_{d_{max}}}{2} \tag{2.63}$$

$$B_1 = C_{d_{max}} \tag{2.64}$$

$$A_2 = [C_{l_{stall}} - C_{d_{max}}\ \sin(\alpha_{stall})\ \cos(\alpha_{stall})] \frac{\sin(\alpha_{stall})}{\cos^2(\alpha_{stall})} \tag{2.65}$$

$$B_2 = \frac{C_{d_{stall}} - C_{d_{max}} \ \sin^2(\alpha_{stall})}{\cos(\alpha_{stall})} \tag{2.66}$$

With $\alpha_{stall}$ the stall angle of attack, $C_{l_{stall}}$ and $C_{d_{stall}}$ the values of lift and drag coefficients at this stall angle. $AR$ is the aspect ratio, ranges from 0 to 50, and has a slight effect on the extrapolated curve.

- Flat plate extrapolation

For a complete extrapolation from -180° to 180°, the airfoil shape is assumed to behave like a flat plate, and the lift and drag coefficient are calculated as follows :

$$C_l = 2\sin(\alpha) \ \cos(\alpha) \tag{2.67}$$

$$C_d = \sin^2(\alpha) \tag{2.68}$$

### BEMT correction models

The variation in the relative velocity of the flow, due to a change in wind speed, has an effect on the boundary layer that is not taken into account in static simulation of the 2D coefficients. The encountered phenomenon is known as dynamic stall, and it is observed in the stall region, where the boundary layer experiences a stall delay. As a result, the aerodynamic coefficients are underestimated. To correct this, several empirical models have been developed to approximate the 3D boundary layer and take into account the stall delay phenomenon.
One of the most used models is the Du and Selig 3D correction.

- Du and Selig stall delay model

For an airfoil section with a chord length $c$, at a distance $r$ from the center of a rotor with a radius $R$, rotating at a speed $\Omega$ and subjected to a wind speed $U$, the corrected 3D lift and drag coefficients are given by :

$$C_{l,3D} = C_{l,2D} + \Delta C_l \tag{2.69}$$

$$C_{d,3D} = C_{d,2D} - \Delta C_d \tag{2.70}$$

With :

$$\Delta C_l = f_l \left(C_{l,p} - C_{l,2D}\right) \tag{2.71}$$

$$\Delta C_d = f_d \left(C_{d,2D} - C_{d,0}\right) \tag{2.72}$$

Where : $C_{l,p} = 2\pi(\alpha - \alpha_0)$, $\alpha_0 = \alpha$ for $C_{l,2D} = 0$, and $C_{d,0} = C_{d,2D}$ for $\alpha = 0$.
The factors $f_l$ and $f_d$ are given by :

$$f_l = \frac{1}{2\pi}\left[\frac{1.6(c/r)a - (c/r)^{\frac{d}{\Lambda}\frac{R}{r}}}{0.1267\,b + (c/r)^{\frac{d}{\Lambda}\frac{R}{r}}} - 1\right] \tag{2.73}$$

$$f_d = \frac{1}{2\pi}\left[\frac{1.6(c/r)a - (c/r)^{\frac{d}{2\Lambda}\frac{R}{r}}}{0.1267\,b + (c/r)^{\frac{d}{2\Lambda}\frac{R}{r}}} - 1\right] \tag{2.74}$$

$$\Lambda = \frac{\Omega R}{\sqrt{U^2 + (\Omega R)^2}} \tag{2.75}$$

with $a$, $b$ and $d$ empirical factors, usually equal to 1.

► Tip loss correction

The pressure difference between the suction and pressure sides of a blade causes an airflow around the tip from the lower to upper surface, resulting in reduced lift and power production near the tip of the blade.
Many methods have been suggested to include this effect . The most widely used method, developed by Prandtl, involves introducing a correction factor $F$ into the equations of torque and thrust, to characterize the reduction in these forces due to the tip loss.

$$F = \left(\frac{2}{\pi}\right)\cos^{-1}\left[\exp\left(-\left\{\frac{(B/2)\,[1-(r/R)]}{(r/R)\sin\varphi}\right\}\right)\right] \tag{2.76}$$

$B$ is the number of blades, $\varphi$ the relative wind angle, and $r/R$ the position on the blade.

► Glauert correction

For a tubulent wake state, the thrust calculated from the momentum theory is no longer valid. To take this case into account, Glauert developed an empirical relationship between the axial induction factor and the thrust coefficient.

$$a = \frac{1}{F}\left[0.143 + \sqrt{0.0203 - 0.6427(0.889 - C_T)}\right] \tag{2.77}$$

Equation (2.77) is only valid for $C_{Tr} > 0.96$.
With :

$$C_{Tr} = \frac{\sigma'(1-a)^2 C_n}{sin^2\varphi} \tag{2.78}$$

### 2.4.5 BEMT Algorithm

The BEMT algorithm enables the determination of the aerodynamic performance of a wind turbine through an iterative calculation process, as follows :

1. For each element and each wind speed, calculate the local solidity and the local speed ratio, using equations (2.58) and (2.41) (replacing $R$ with $r$)

2. Set initial guesses for the induction factors $a$ and $a'$ (usually 0)
3. Calculate the angle of relative wind $\varphi$ from equation (2.47)
4. Calculate the angle of attack using the following equation :

$$\alpha = \varphi - (\theta_t + \theta_p) \tag{2.79}$$

5. Determine the lift and drag coefficients corresponding to the calculated angle of attack, and apply the Viterna extrapolation and stall delay correction model.
6. Deduce the normal and tangential coefficients :

$$C_n = C_l \cos(\varphi) + C_d \sin\varphi \tag{2.80}$$

$$C_t = C_l \sin(\varphi) - C_d \cos\varphi \tag{2.81}$$

7. Calculate the tip loss correction factor from equation (2.76)
8. Calculate the local thrust coefficient from equation (2.78)
9. Update $a$ and $a'$ from the equations below :

$$\begin{cases} a = \left[1 + \dfrac{4F\sin^2\varphi}{\sigma' C_n}\right]^{-1} & if \quad C_{Tr} \leq 0.96 \\ a \text{ from equation (2.77)} & if \quad C_{Tr} > 0.96 \end{cases} \tag{2.82}$$

$$a' = \left[-1 + \frac{4F\sin\varphi\ \cos\varphi}{\sigma' C_t}\right]^{-1} \tag{2.83}$$

10. Repeat steps from 3 to 8 until the values of $a$ and $a'$ converge to an acceptable level of tolerance.
11. Calculate the thrust, torque and generated power for each blade element, using equations (2.56), (2.57) and (2.44).
12. Calculate the total torque, thrust and generated power of the turbine :

$$\begin{cases} T = \sum\ dT \\ Q = \sum\ dQ \\ P = \sum\ \Omega dQ \end{cases} \tag{2.84}$$

## 2.5 Conclusion

This chapter presented the three key elements utilized in this study, that is, GANs, XFOIL(panel method) and the BEM theory, from a theoretical and mathematical perspective.

The combination of these methods will allow for an exploration of unique airfoil designs with a potential of enhancing wind turbine performance.

The next chapter will cover the details of the adopted numerical approach, the implementation of the three methods as well as their combination.

# Chapter 3
# Numerical approach

## 3.1 Introduction

In this chapter, a detailed implementation of the methods discussed in the previous chapter will be presented. We start with a description of the general code structure and parts, and then delve into the practical and technical aspects of each part in the upcoming sections. By describing the underlying code structure and its functioning, readers are provided with the necessary insights to reproduce the experimental setup. Thus, enabling them to build upon and enhance this code, to extend the research and explore further aspects of the topic.

## 3.2 Description

The code generates random airfoil shapes using a Generative Adversarial Network (GAN), then evaluates the new generated airfoils by calculating their aerodynamic performance, in order to select the shapes that meet the desired lift-to-drag ratio criterion.

MATLAB was used as the primary tool for implementation, a powerful software widely used in scientific and engineering research. Additionally, XFOIL was called and executed from the MATLAB environment, for aerodynamic analysis.

The code mainly consists of 4 parts as shown in figure (3.1) below :

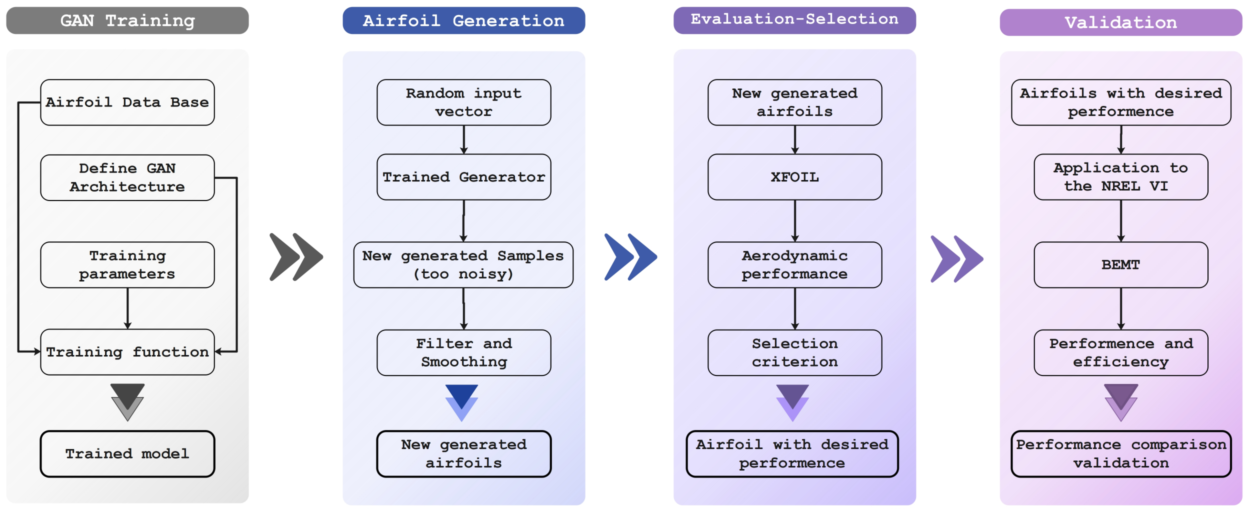


Figure 3.1: Global structure of the code

- To begin, a set of airfoils from the UIUC database [31],is loaded as a cell array containing the x and y coordinates for each airfoil. Pre-processing is then carried out to standardize the training set. This involves adjusting the y coordinates of the airfoils to align with a common x-coordinate vector.
  Next, two networks : the generator and discriminator, are defined for the Generative Adversarial Network (GAN). The networks are subsequently trained using parameters that were optimized in such a way to achieve convergence of the training process.

- Once the training is complete, the trained generator is used to create new airfoil designs randomly. However, the generated coordinates are initially too noisy. To address this, a Savitzky-Golay filter is applied along with a smoothing of both trailing and leading edges using spline functions and Bezier curves. For a better visualization, the resulting shapes are plotted.
- Then, an evaluation of the generated and smoothed airfoils is carried out using XFOIL. This involves calculating the lift to drag ratio for various angles of attack. Subsequently, a selection criterion is defined, specifying a minimum value for $C_l/C_d$. The airfoils that satisfy this criterion are then chosen, and their coordinates are exported to text files in the selig format.
- In the last section of the code, we seek to validate the selected airfoil, by applying it to the NREL phase VI . The performance of the modified wind turbine is then calculated using the BEMT method, and results are compared to the ones of the original NREL phase VI .

## 3.3 GAN Training

This first part mainly includes : preparing the training set, building the GAN network, and then passing the two into the training function to get the trained model that will be able to generate new realistic airfoil shapes. Figure (3.2) gives an overview of this part.

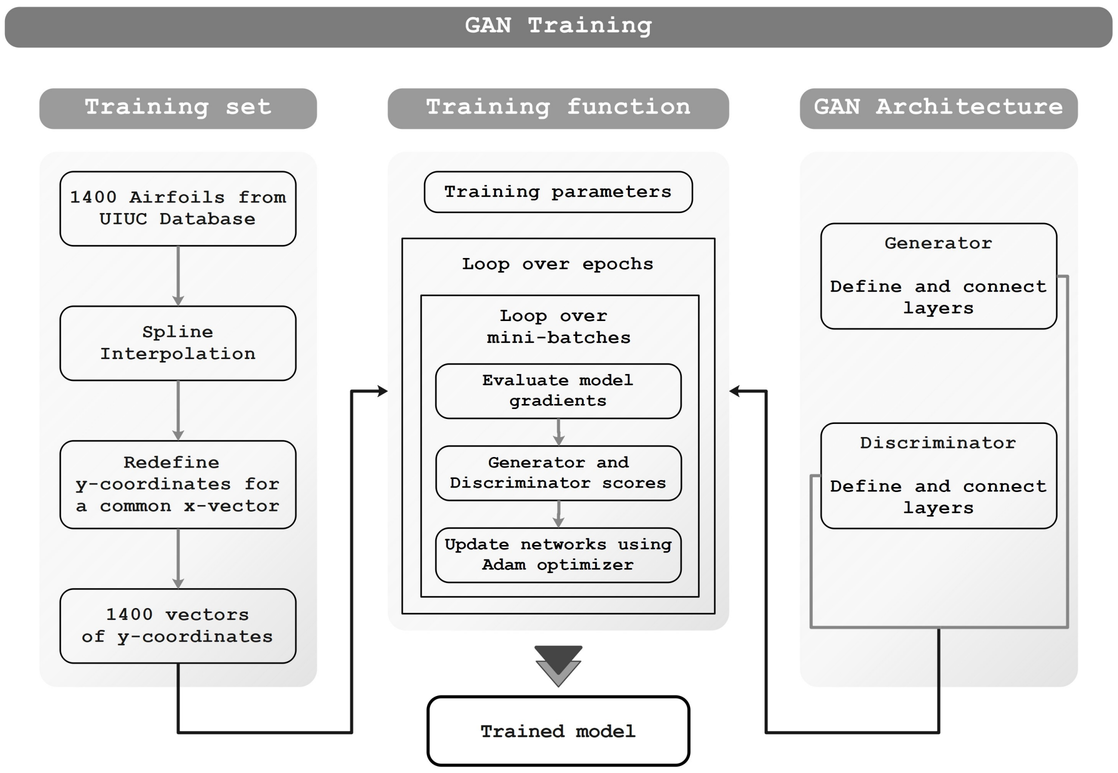

GAN Training
Training set
Training function
GAN Architecture
1400 Airfoils from UIUC Database
Spline Interpolation
Redefine y-coordinates for a common x-vector
1400 vectors of y-coordinates
Training parameters
Loop over epochs
Loop over mini-batches
Evaluate model gradients
Generator and Discriminator scores
Update networks using Adam optimizer
Trained model
Generator
Define and connect layers
Discriminator
Define and connect layers

Figure 3.2: Organigram of the GAN training phase

### 3.3.1 Training set

The airfoils that served as a training set were downloaded from the UIUC (University of Illinois at Urbana-Champaign) airfoil database [31] which covers a wide range of applications. The 1400 airfoils were then loaded into MATLAB and saved as a cell array to facilitate further processing. Each cell contains a 2 columns matrix in which is stored the x and y coordinates.

#### Pre-processing

To standardize the training set, a pre-processing step was performed. This involved redefining the y coordinates of all airfoils for a common x-coordinate vector. With a consistent x vector, the representation of the airfoil profiles became uniform. With a spline interpolation, each airfoil was redefined to have 600 panels on both the intrados and extrados. The final representation of the coordinates was a $(1201, 2)$ matrix.

This standardization step not only simplified the data representation, but also allowed us to reduce the dimensionality of the input data by eliminating the variability in the x direction and focus only on the variations in the y direction. This dimensionality reduction enhanced computational efficiency during the training process, as it reduced the number of patterns and features the network needed to capture.

### 3.3.2 GAN Architecture

The GAN architecture consists of two main components: the generator and the discriminator, both employing a sequential structure. They are based on a deep convolutional neural network (CNN) architecture, composed of multiple deconvolutional layers, batch normalization, convolutional layers with increasing complexity, and also Rectified linear units (ReLU) layers, allowing the networks to capture hidden patterns effectively.

The generator network takes a vector of latent inputs and aims to synthesize realistic samples similar to the training set. This is achieved through a series of transposed convolutional layers that gradually up-sample the learned features. On the other hand, the discriminator, aims to distinguish between real and generated data. It takes the samples (real and generated) as an input and passes them through a series of convolutional layers that progressively extract hierarchical features.

***Note :*** The architecture of our GAN is a conditional GAN (CGAN), however all the labels were assigned the same value 1 so that it produces the same result as a simple GAN. For future works, it would be very easy to adapt the code and turn it into an inverse design optimization by introducing the training set with the corresponding labels, and then specify the labels in the generation step to obtain samples with the desired characteristics.

### Generator

The Generator is a two-input network, which creates new airfoil shapes given $(1, 1, 100)$ arrays of random values and corresponding labels (in our case : ones vector).

- It begins with an input layer that takes in a vector of latent inputs that are then projected and reshaped using the custom layer projectAndReshapeLayer (used in Matlab examples [32]). This layer uses a fully connected layer to upscale the input and reshape it, resulting in a feature map with dimensions [4, 1, 1024].
- Similarly, the labels are taken as an additional input through an image input layer. The categorical labels are passed through the custom layer embedAndReshapeLayer (used in Matlab examples [32]), that outputs arrays of size $(4, 1, 1)$.
- A concatenation layer is used to concatenate the results from both inputs, and outputs an array of dimension $(4, 1, 1025)$.
- The concatenated features are then fed into a series of transposed convolution layers to increase the spatial resolution of the feature maps, followed each time by intermediate batch normalization layers and rectified linear unit activation to improve stability and introduce non-linearity.
- The up-sampled output is an array of dimension (1201,1,1). The final generator network is created as a dlnetwork (dlnetGenerator) for training and synthesis purposes, as this enables automatic differentiation.

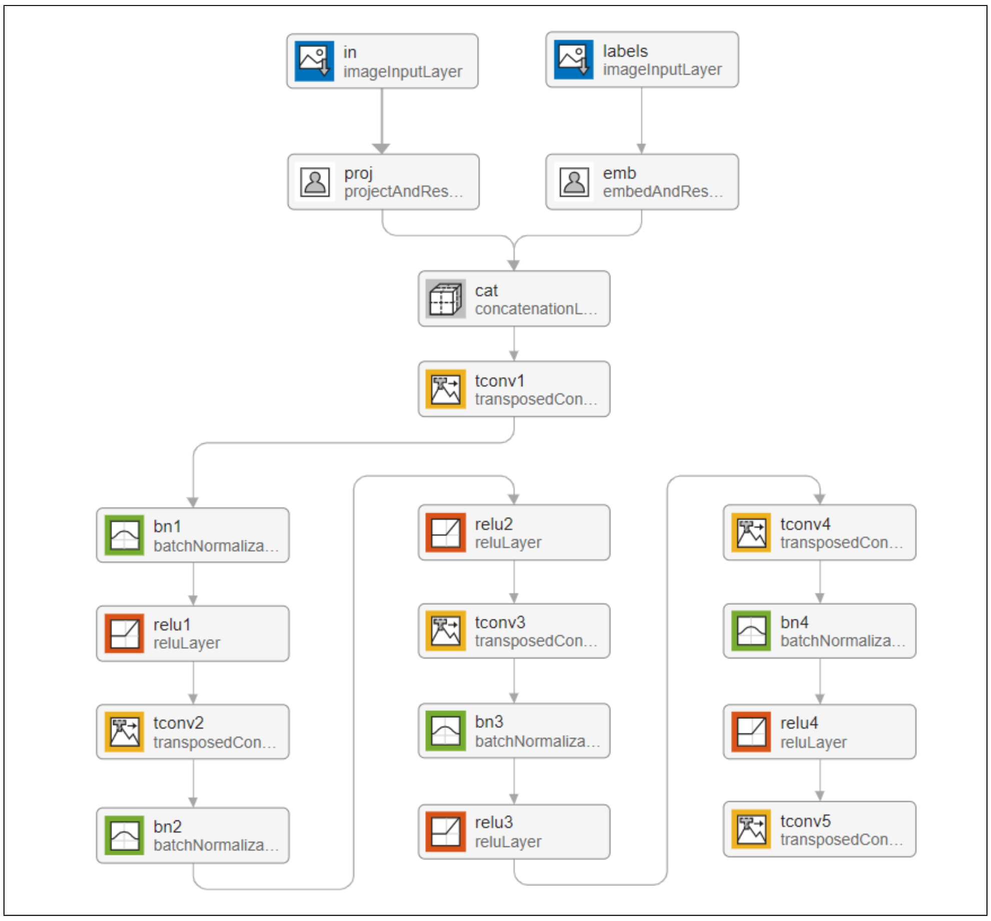


Figure 3.3: Generator network's architecture

## Discriminator

The discriminator is a two-input network that plays the role of an adversary in the GAN architecture, aiming to distinguish between real and generated samples, by producing a probability score indicating the authenticity of the input image.

- It takes as input an image with dimensions [1201, 1, 1], corresponding to the y-coordinates vector.
- Similar to the generator, the discriminator also takes an additional input, the labels, which are passed through an embedding layer (emb). The embedded labels are then concatenated with the other inputs using the concatenation layer.
- The concatenated features are then passed through a series of convolutional layers that further process the data to capture hierarchical features. Each convolutional layer is followed by a leaky ReLU activation layer.
- The last layer is a convolutional layer, which produces a single-channel feature map.
- The final discriminator network is created as a dlnetwork (dlnetDiscriminator) for training and adversarial purposes within the GAN framework.

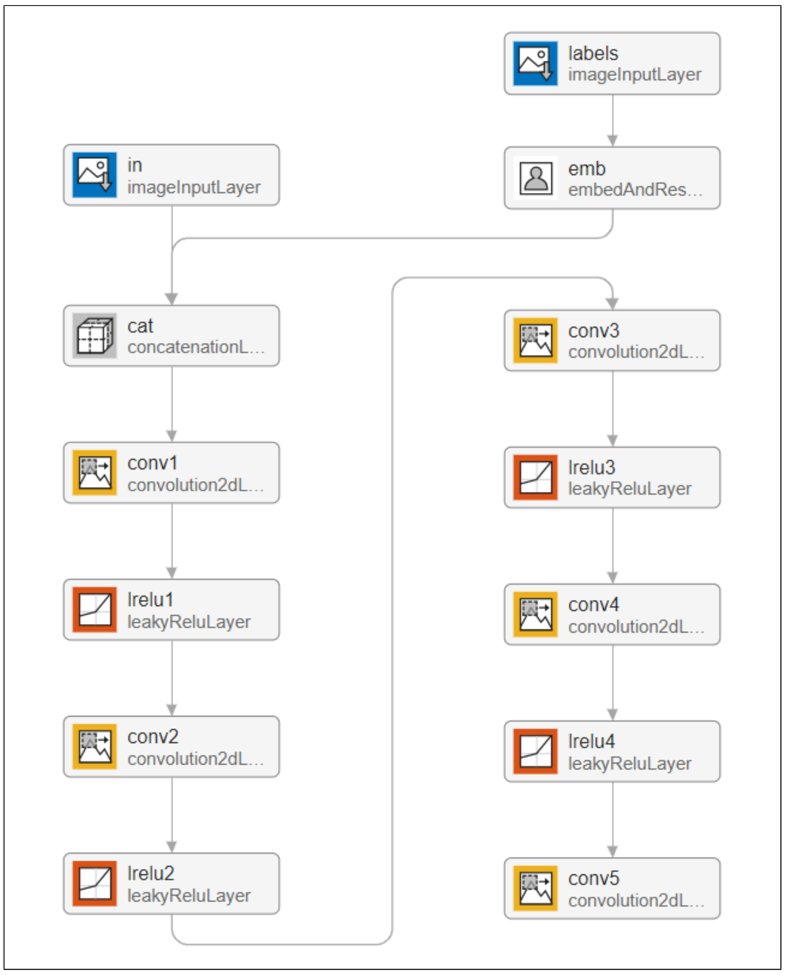


Figure 3.4: Discriminator network's architecture

### 3.3.3 Training function

Training the GAN was the most delicate and challenging task in this work. It requires a careful choice of parameters to optimize the training process and achieve convergence of the adversarial training. And since there is no direct or definitive method to find the optimal parameters, we relied on experimenting various configurations and combinations, while observing the impact on the GAN's performance and convergence, in order to adjust the parameters until getting satisfactory results.

#### Training Parameters

After several trials and iterative adjustments, the following set of training parameters demonstrated satisfactory performance in terms of convergence and stability of the training process, and the generation of realistic airfoil profiles.

```
% numLatentInputs = 60
params.numLatentInputs = numLatentInputs;
% inputSize = [1201 1 1];
params.sizeData = [inputSize length(labels)];
params.numEpochs = 200;
params.miniBatchSize = 300;

% Specify the options for Adam optimizer
params.learnRate = 0.0001;
params.gradientDecayFactor = 0.3;
params.squaredGradientDecayFactor = 0.999;

```

- numLatentInputs: The number of latent inputs, which represent the random noise vectors sampled from a normal distribution and serve as input to the generator network.
- sizeData: The size of the input data, including the input size (inputSize) and the length of the labels.
- numEpochs: The total number of training epochs, determining the number of times the entire training dataset is traversed during training.
- miniBatchSize: The size of each mini-batch used for training, indicating the number of samples processed in each iteration.

The optimization algorithm used in training is the Adam optimizer, which is a popular choice for training deep neural networks. The following options were specified for the Adam optimizer:

- learnRate: The learning rate or step size, which determines the magnitude of weight updates during each iteration.
- gradientDecayFactor: The factor that controls the decay of the first-order moment estimates in the Adam optimizer.
- squaredGradientDecayFactor: The factor that controls the decay of the second-order moment estimates in the Adam optimizer.

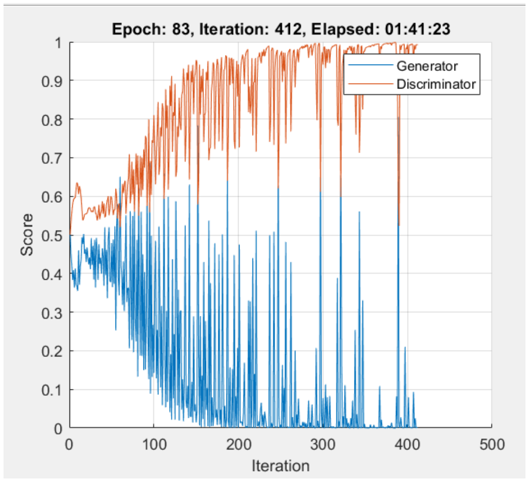
The figures below show different score plots that were obtained during the training process for different sets of training parameters.

a) For the first set, we can see that training failed to converge while being very oscillatory, and the discriminator overpowering the generator. This is a type of failure modes known as : Discriminator domination. The discriminator learns too quickly and classifies most of the data correctly, which results in the generator network failing to learn.

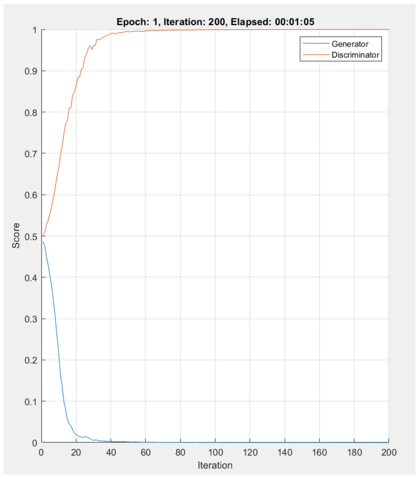
b) We can see that after changing the parameters the training became more stable. However, the same failure mode occurred. The discriminator dominated the generator too quickly during the first epoch. In this case, we had to stop the training, because it cannot recover from this failure mode.

c) After many changes in the parameters' set and several trials, the training converged as indicated in the plot (c). But it didn't reach a satisfactory level, as the scores were still a bit far from 0.5 and slightly oscillatory.

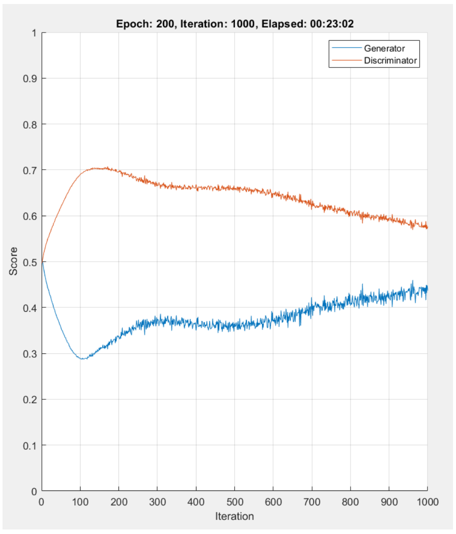
d) Finally, for the optimized parameters that were presented earlier, the training achieved convergence at a very satisfactory level. The progress was very stable and the scores of both the generator and discriminator converged very close to 0.5, indicating that the training performance is good.



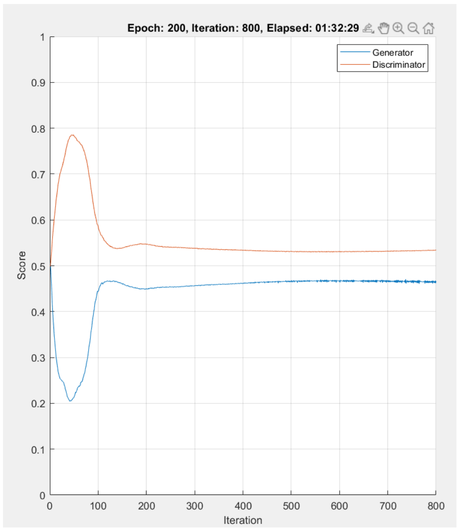
(a) Divergence - Oscillations



(b) Discriminator domination



(c) First convergence



(d) Optimal parameters

Figure 3.5: Training performance

### Training Loop

To train the network, a custom training function ***trainGan*** was used. The function relies on an iterative process over the training set, while updating the network parameters at each iteration, to optimize the generator and discriminator networks through adversarial training. For real-time insights into the performance and evolution of the training, the scores of both the generator and discriminator were dynamically plotted after each iteration.

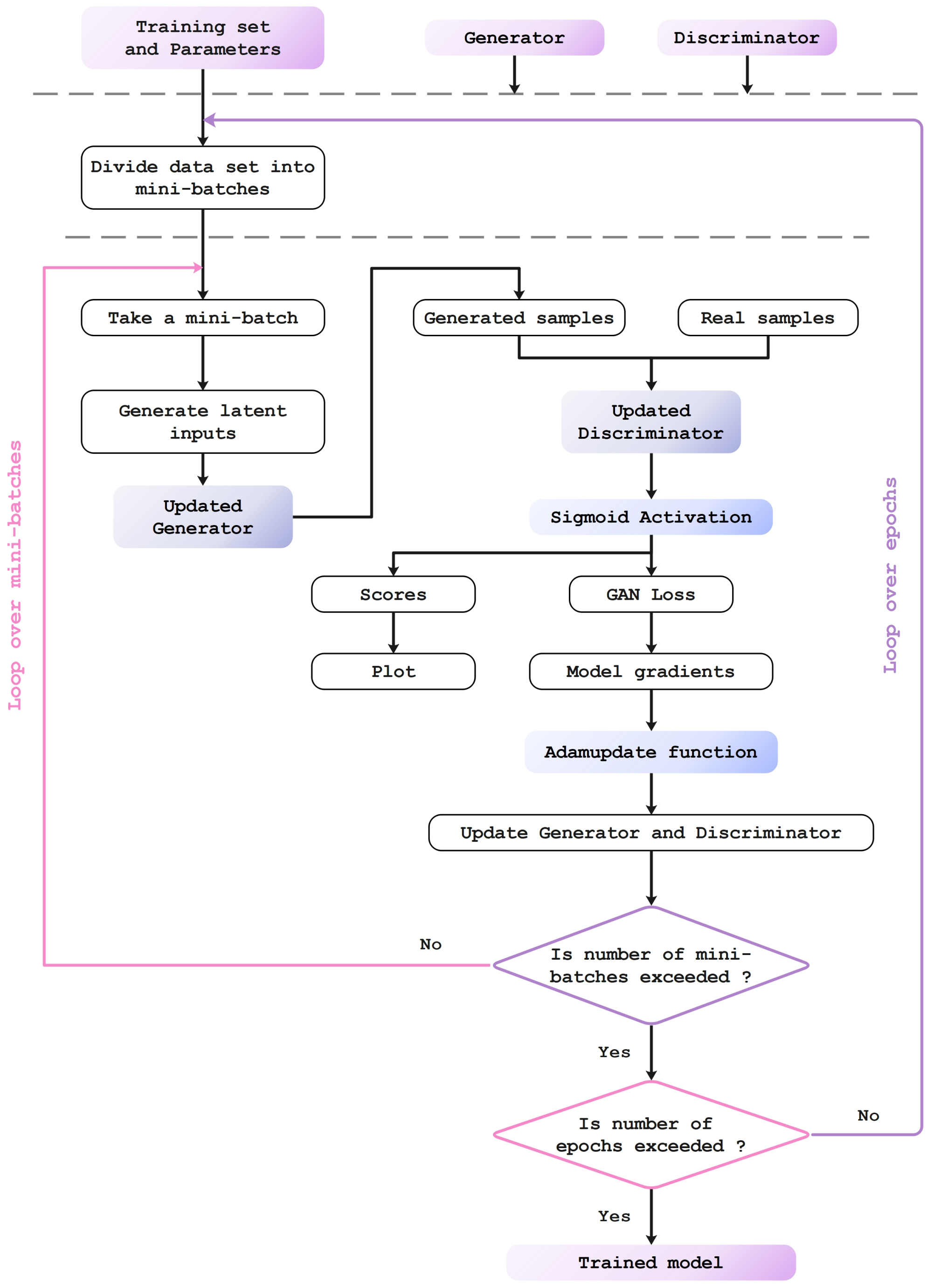


Figure 3.6: Flowchart of the training function

- For each epoch, the training set is divided into mini-batches.
- For each mini-batch :
    - A set of random latent inputs is sampled from a normal distribution using the dlarray (Deep Learning Toolbox) object.
    - Compute the gradients of the loss with respect to the learnable parameters :
      The generator takes the latent inputs and generates new airfoil profiles. Then the discriminator is presented with a combination of real airfoil shapes from the mini-batch and the generated ones. It evaluates the authenticity of each input, and assigns probability scores using a sigmoid activation. Based on this, scores of both the generator and the discriminator are calculated, and the GAN loss is deduced. At last, gradients with respect to loss are calculated for each network.
    - The model gradients, which indicate the direction and magnitude of the parameter updates, are evaluated using the dlfeval function.
    - Update the network parameters using the adamupdate (Deep Learning Toolbox) function.

## 3.4 Airfoil Generation

Now that the generator is trained, it can up-sample new airfoil shapes. The new generated samples are too noisy to be considered as airfoils, that is why a Savitzky-Golay filter was applied to the y-coordinates vector. The resulting airfoils have very sharp leading edges and irrelevant trailing edges. Therefore, smoothing using spline functions and Bezier curves was applied. Finally, both x and y coordinates undergo normalization.

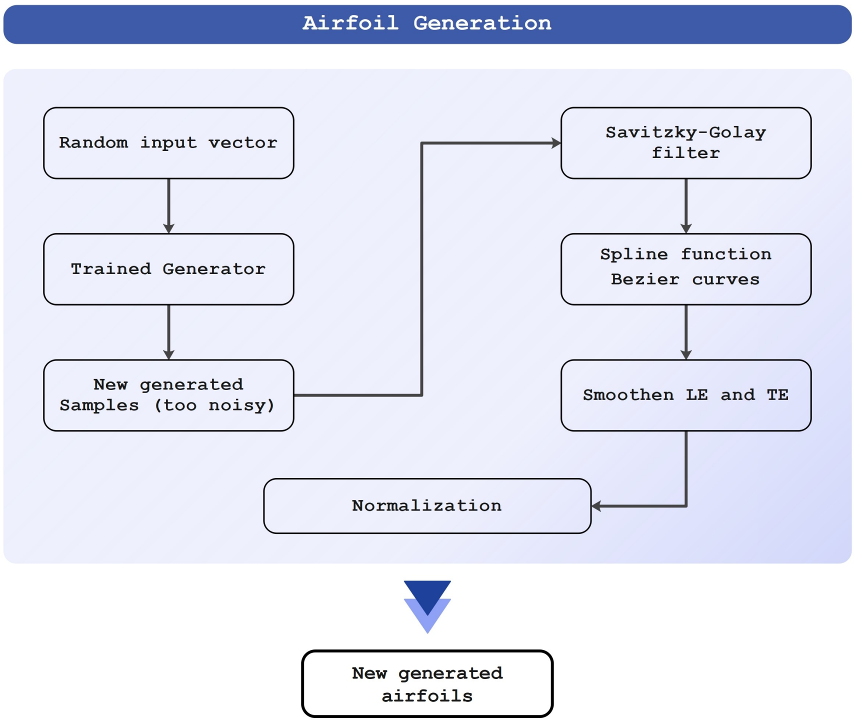


Figure 3.7: Airfoil generation

- The Savitzky-Golay filter applied is a 6-th order filter with a frame length of 1201.
- For the TE, a spline smoothing was applied to the extrados, with a smoothing parameter of 0.99999.
- As for the sharp LE, new coordinates were calculated using Bezier curves in the LE zone.

The figure (3.8) below, shows the transformation that the generated airfoils undergo during this smoothing step.

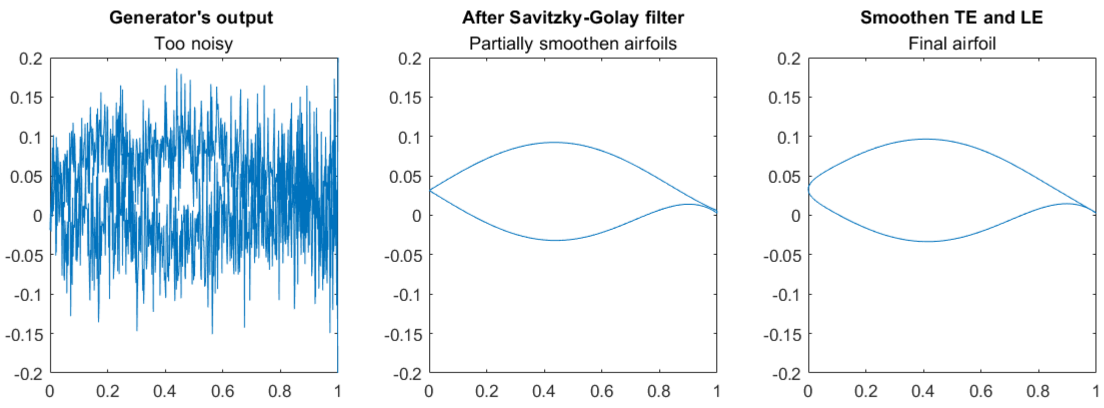


Figure 3.8: Smoothing of the generated airfoils

## 3.5 Evaluation and selection

In this part, the generated airfoils are evaluated by calculating their lift to drag ratio for various angles of attack, using XFOIL. Afterwards, we set a selection criterion, that is a minimum value for $C_l/C_d$. The airfoils that satisfy this criterion are chosen, and their coordinates are exported to text files in the Selig format.

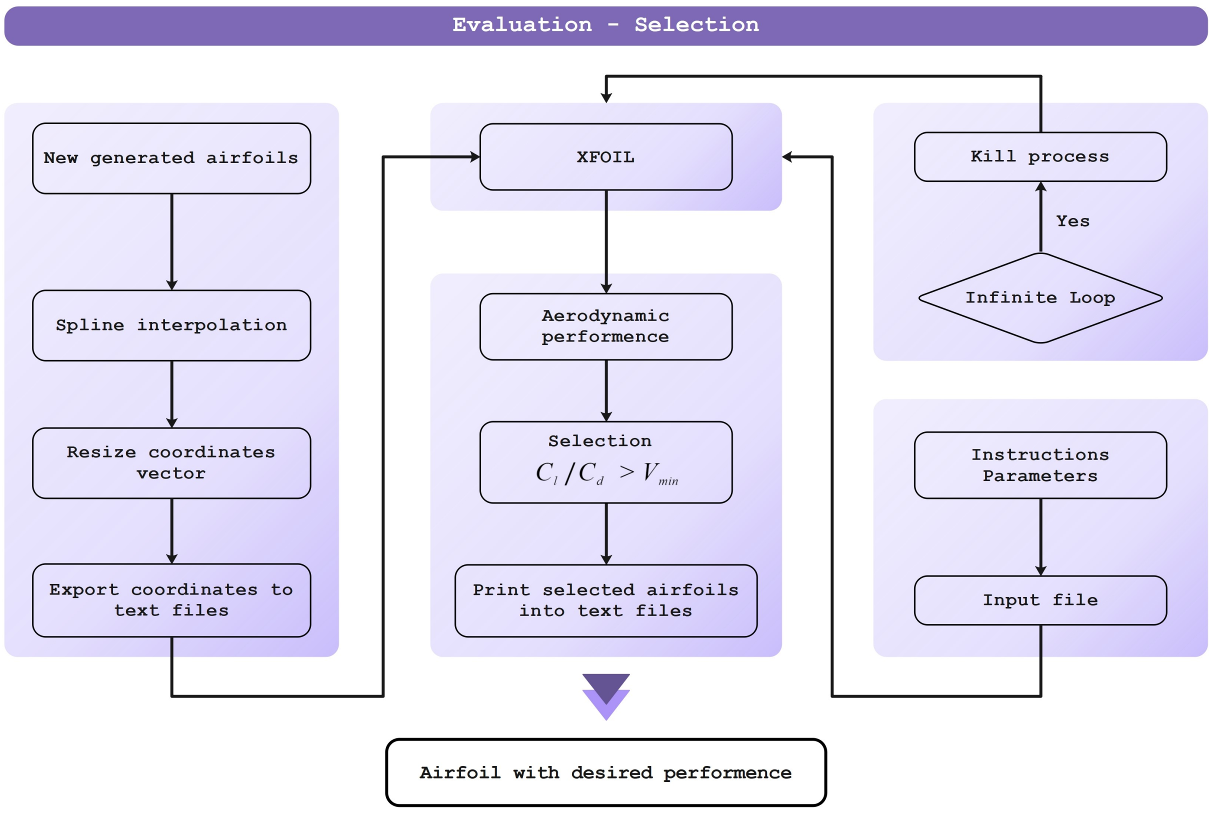

Evaluation - Selection
New generated airfoils
Spline interpolation
Resize coordinates vector
Export coordinates to text files
XFOIL
Aerodynamic performence
Selection $C_l/C_d > V_{min}$
Print selected airfoils into text files
Kill process
Yes
Infinite Loop
Instructions Parameters
Input file
Airfoil with desired performence

Figure 3.9: Evaluation and selection

### 3.5.1 Re-adapting the coordinates

The resulting airfoils, defined as arrays of size 1201 by 2, must be resized before loading them into XFOIL, as this latter takes a number of coordinates limited to 400.
Thus, the coordinates will undergo the same pre-processing step that was applied in the preparation of the training set, by only changing the number of panels from 600 to 200 for both the intrados and extrados.

### 3.5.2 Calling XFOIL

To execute XFOIL from Matlab's environment, a helper function ***"xfoil.m"*** was defined.

- This function takes as input the coordinates of the airfoil and the parameters for the PPAR menu : Number of panel nodes, Panel bunching parameter, TE/LE panel density ratios, Refined area/LE panel density ratio, Top and Bottom sides refined area x/c limits

- The function then prints the admitted coordinates into a text file that will later be loaded to XFOIL. It also prepares the input file containing the commands to execute.
- Once the airfoil and input files are ready, we call the batch file below to run XFOIL, using the command ***dos***.

```
@echo off

call xfoil.exe < xfoil_input.inp > convergence_data.txt

taskkill /F /T /IM "cmd.exe"

```

- Finally, the function returns the lift and drag coefficients along with the corresponding angles of attack, after reading them from the output file in which they were stored.

## Input file

The following input file is created inside the function ***"xfoil.m"***.

- This file loads the airfoil in question (for each iteration), assigns a name to it, and then normalizes it.
- It also applies changes to the coordinates as the airfoil is re-panelled and customized according to the parameters of the PPAR menu.
- It then sets the parameters of the operational menu, by choosing the viscous mode and defining the Reynolds number as well as the maximum number of iterations.
- Finally, it prescribes a set of angles of attack for which the aerodynamic performance would be calculated, and save the results in a text file.

```
LOAD airfoil.txt
Airfoil1000
norm
PPAR
N 400
P 1
T 1
R 1
XT 1 1
XB 1 1


PSAV Save_airfoil.txt
OPER
visc
0.8e6
iter 50
Pacc 1


aseq -5 20 0.5
CPWR Save_airfoil_Cp.txt
PWRT
Save_airfoil_Pol.txt
```

### Infinite loop problem

In some cases, XFOIL goes into an infinite loop, and thus the code keeps running without making any further progress. So the execution of XFOIL must be terminated manually. However it is not practical to keep following all the iterations and wait for the problem to happen to end the task manually.

To overcome this problem, another batch file is executed simultaneously with the first one.

```

@echo off
start "temp" cmd /c "timeout /T 60> NUL & taskkill /F/IM xfoil.exe"

```

- The above script waits for 60 seconds and then forcefully terminates the ”xfoil.exe” process.
- Meanwhile, a command is added at the end of the first batch file, so that when XFOIL finishes calculating aerodynamic performance of a certain airfoil, the task of the second batch file is killed. Otherwise, this file would terminate the XFOIL process of the next airfoil without any infinite loop problem taking place.
- Setting the timeout for 60 seconds is a choice based on obervations that were made on the time execution of XFOIL for the first generated airfoils. For most of them, the process took 15 seconds on average. This may not be a general rule, and some airfoils could be eliminated during this step. But the time gained by overcoming this problem compensates the fact that very few airfoils are being lost.

### 3.5.3 Selection

At this stage, the generated airfoils are all evaluated, and their lift and drag coefficients with respect to the angles of attack are stored in cell arrays.

The glide ratio $C_l/C_d$ is calculated, and for each airfoil the maximum value $(C_l/C_d)_{max}$ is stored along with the corresponding angle of attack.

A selection criterion is then defined, by specifying a minimum value for $C_l/C_d$, that is the selection glide ratio $(C_l/C_d)_S$.

The airfoils that have a maximum glide ratio greater than the minimum value set as a criterion ($(C_l/C_d)_{max} > (C_l/C_d)_S$), are selected. Their coordinates are then exported to text files in the Selig format and are placed in a folder named ’Selection’.

## 3.6 Validation

This is the last part of the ***(airfoil_GAN.m)*** code, where we seek to validate this work in general. The new selected airfoil is applied to the NREL phase VI , replacing the S809.
By using the BEMT method, power output and efficiency of both original and modified wind turbines are calculated. We can then compare the results and validate the work.
The validation process and steps are indicated in the flowchart below.

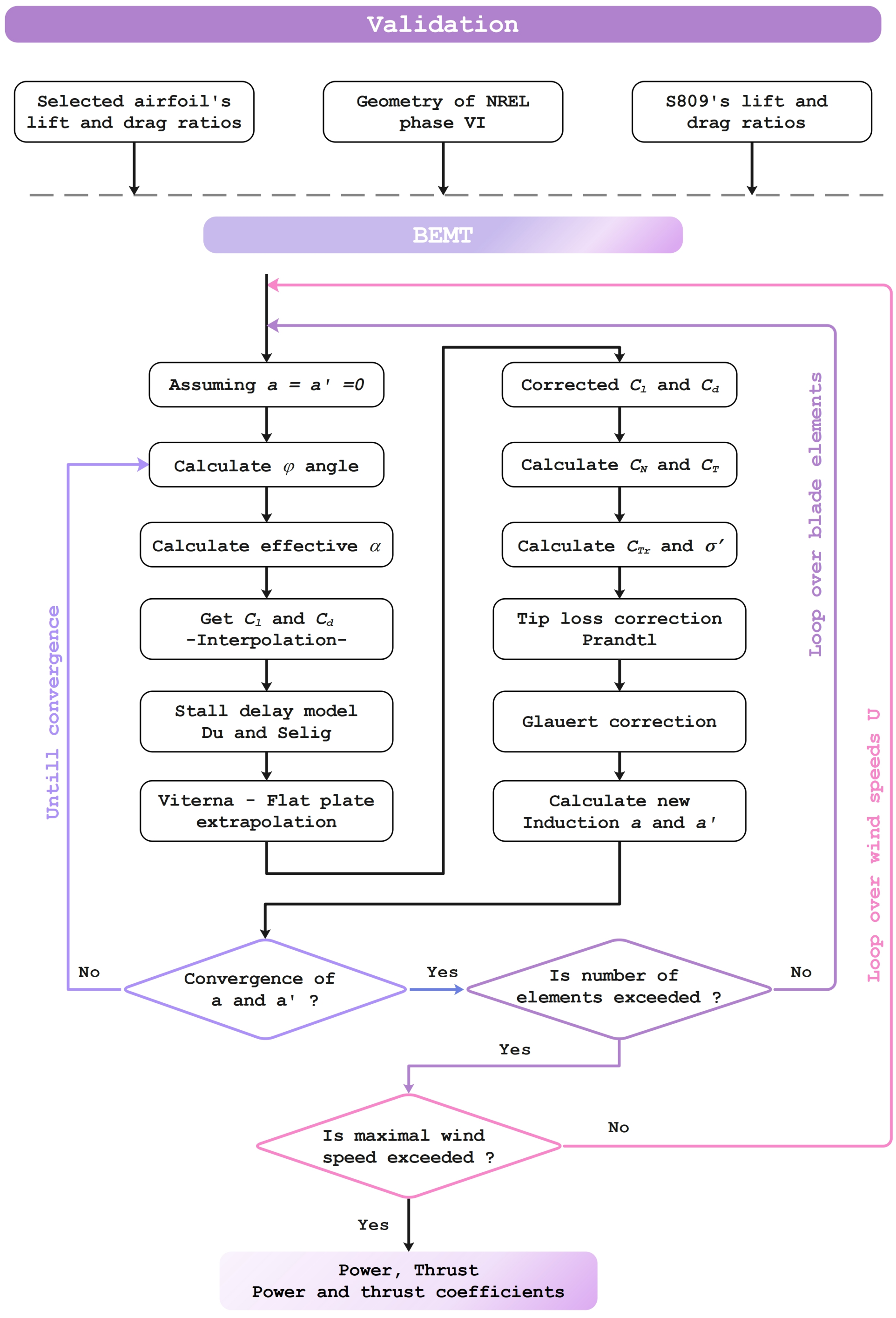

Validation
Selected airfoil's lift and drag ratios
Geometry of NREL phase VI
S809's lift and drag ratios
BEMT
Assuming a = a' =0
Calculate φ angle
Calculate effective α
Get $C_l$ and $C_d$ -Interpolation-
Stall delay model Du and Selig
Viterna - Flat plate extrapolation
Corrected $C_l$ and $C_d$
Calculate $C_N$ and $C_T$
Calculate $C_{Tr}$ and σ'
Tip loss correction Prandtl
Glauert correction
Calculate new Induction a and a'
Untill convergence
Loop over blade elements
Loop over wind speeds U
No
Convergence of a and a' ?
Yes
Is number of elements exceeded ?
No
Yes
Is maximal wind speed exceeded ?
No
Yes
Power, Thrust
Power and thrust coefficients

Figure 3.10: Flowchart of the validation process

## 3.7 Conclusion

In summary, this chapter has provided a detailed account of the implementation of the code ***(airfoil_GAN.m)*** for airfoil generation using a Generative Adversarial Network (GAN). The structure and functioning of the code were described, highlighting the key components and steps involved in the generation, evaluation, selection and validation processes.

The chapter began by presenting the training process, where the architecture of both the generator and discriminator networks was presented, outlining the sequential layers and their configurations. The iterative process and the intricate steps involved in training the model were also detailed.

After generating new airfoils, XFOIL was used as an evaluation tool to calculate aerodynamic performance, by running it from the Matlab environment. Based on these results, a subset of airfoils that achieved specific lift-to-drag ratios was chosen for further analysis and validation.

To validate the effectiveness of the selected airfoil, it was applied to the NREL phase VI wind turbine, and Blade Element Momentum Theory (BEMT) method was selected for calculating the efficiency of the original and modified wind turbine. By comparing the results of the two, we ensure that the generated airfoils are thoroughly analysed and validated.

The detailed explanations and insights presented here serve as a valuable resource for reproducing the experimental setup, extending the code for further research, and exploring potential enhancements in the generation of realistic and diverse airfoil profiles.

# Chapter 4
# Results and discussion

## 4.1 Introduction

This chapter presents and discusses the results of the present work, which are new airfoil shapes generated with the code ”airfoil_GAN”.

Furthermore, it aims to validate the implemented methodologies and techniques, ensuring the reliability of the code, through a comparative analysis that will be conducted between a reference turbine and a modified one to which the selected airfoil is applied.

## 4.2 New Generated Airfoils

The GAN training phase was executed on an intel i5 8th generation CPU, and lasted for one and half hours. At the end of the training, both the discriminator and generator converged very close to 0.5 which is the equilibrium of the GAN game. Figure 4.1 shows a score plot that provides a better visualization of the training progress.

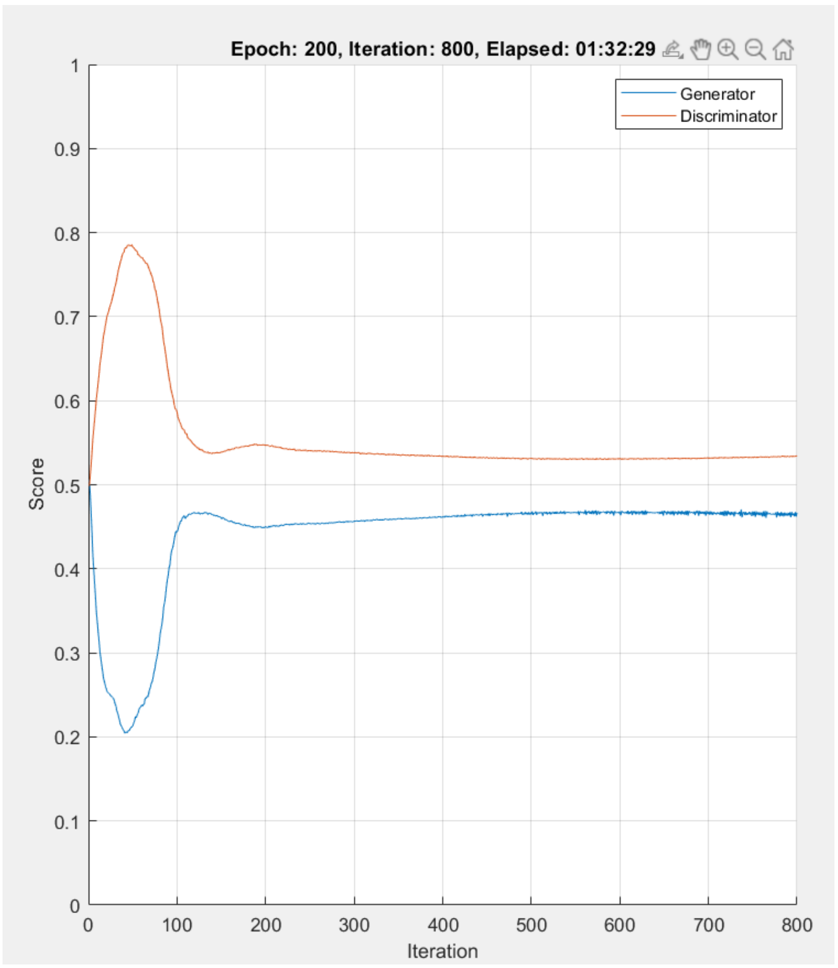


Figure 4.1: Training scores

Now that the generator has been trained, it is ready to generate a large number of novel airfoil shapes by receiving a random noise vector as an input. In our case, 1000 airfoils were generated within a second. Filtering and smoothing were then applied to the initially noisy shapes, resulting in the creation of new airfoils as indicated in figure 4.2, which shows a plot of the first 12 generated airfoils.

Some of the shapes are unlikely to be considered as valid airfoils, for example Airfoil N°1 in the plot. However, this does not cause any worries as these shapes would most likely be eliminated during the evaluation process since they won't be able to converge in XFOIL.

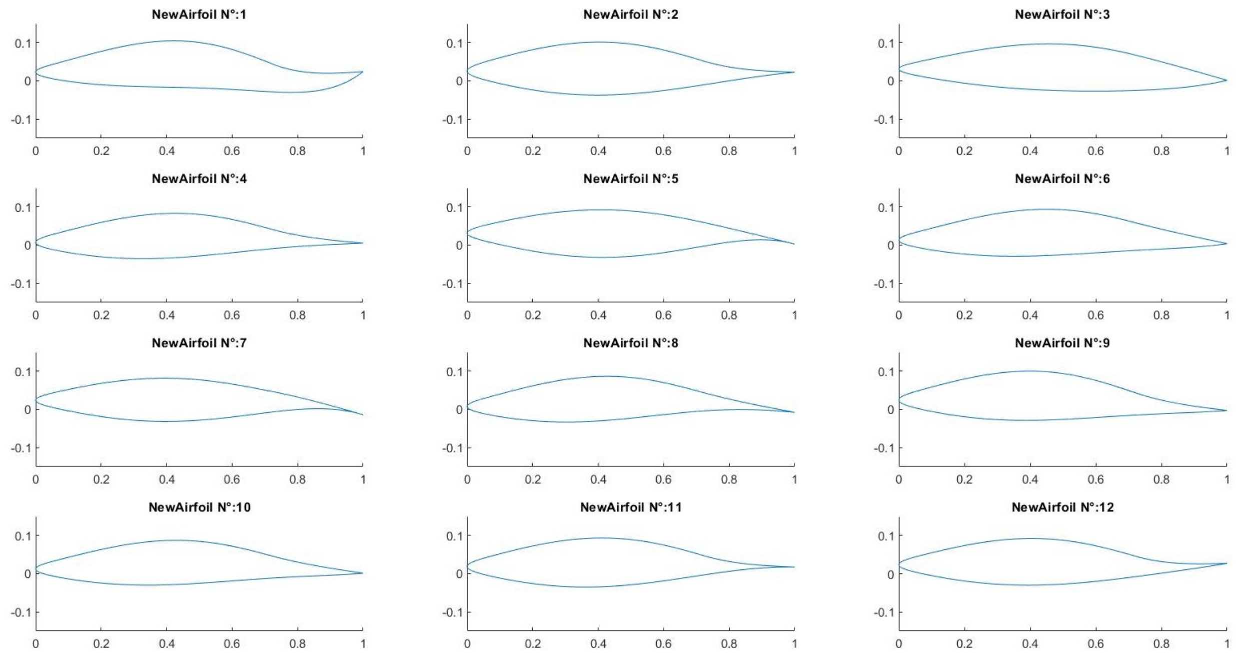


Figure 4.2: New generated airfoils

## 4.3 Evaluation and selection

### 4.3.1 Choice of Reynolds number

Before runing the xfoil.m function that evaluates the aerodynamic performance of airfoils, an important input parameter needs to be set, that is the Reynolds number. The selection of this parameter relies on the intended application. The aim could be getting airfoils that exhibit better aerodynamic performance at low speeds, or on the contrary, airfoils that achieve high-speed aerodynamic efficiency.

The aim for now is to validate the established code by applying the best generated airfoil to the NREL phase VI , a two bladed HAWT with a rotor diameter of 11 m, a rated power of 20 kW and a rotational speed of 72 rpm.
Considering the fact that Reynolds number varies along the span of the blade [33], due to the

variation in the chord and the relative wind velocity , and assuming that the WT will operate at low wind speeds ranging from 4 to 14 m/s, the following plot can be obtained using the BEMT code, thus giving a better overview on this variation for the NREL phase VI .

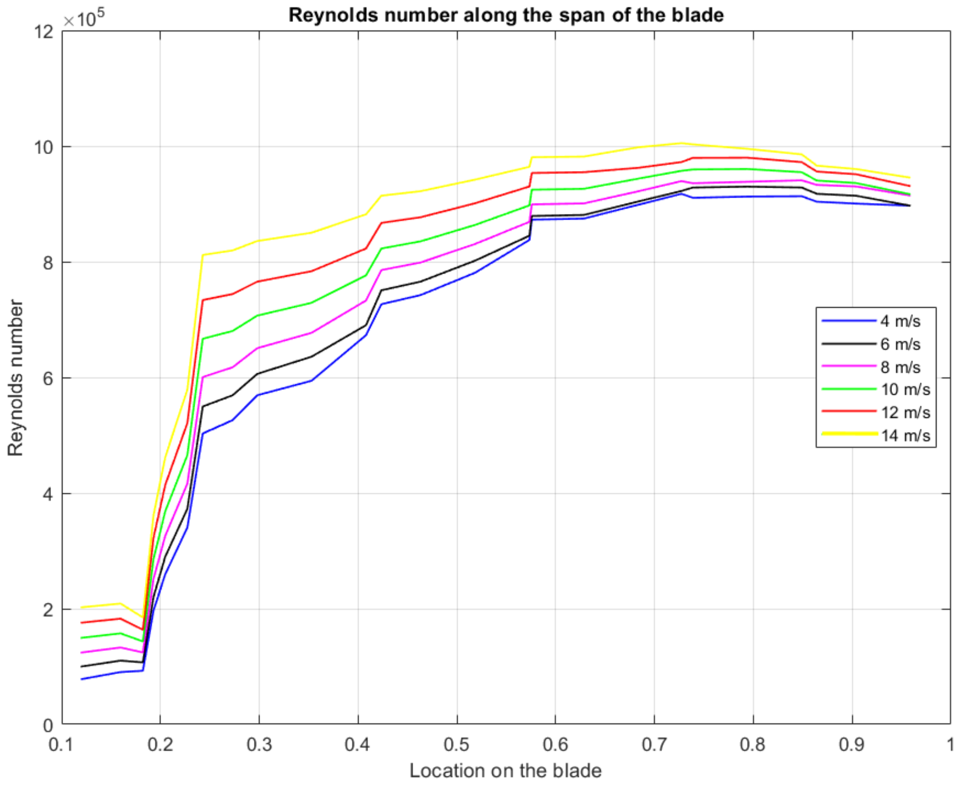


Figure 4.3: Range of Reynolds number along the blade span

It can be observed that for around 60% of the blade (from a location of 0.4 to 1), the Reynolds number ranges from $0.8 \times 10^6$ to $1 \times 10^6$. So it would be reasonable to select a Reynolds number of $0.9 \times 10^6$ on which the evaluation would be based, since the airfoil would mostly be operating in this flow condition.

It is important to note that this is only done for the evaluation part. The calculation of the WT's performance with the BEMT code will be multipolar, i.e aerodynamic performance of the airfoil would be calculated for different Reynolds number on each element.

### 4.3.2 Evaluation

The generated airfoils were evaluated in XFOIL for a Reynolds number of $0.9 \times 10^6$. The process lasted for about 2 hours.
Among the 1000 shapes that were evaluated, 594 successfully got through this phase. The rest of the shapes were eliminated as they failed to converge in XFOIL, or whenever an infinite loop problem was encountered.

### 4.3.3 Selection

To perform a selection, a criterion is defined based on the maximal glide ratio of the airfoils. A value of 110 was chosen so that only the ones having a superior maximal $C_l/C_d$ ratio are selected.The best airfoil among the selected ones has a maximal glide ratio of **116**, which is a very interesting result.

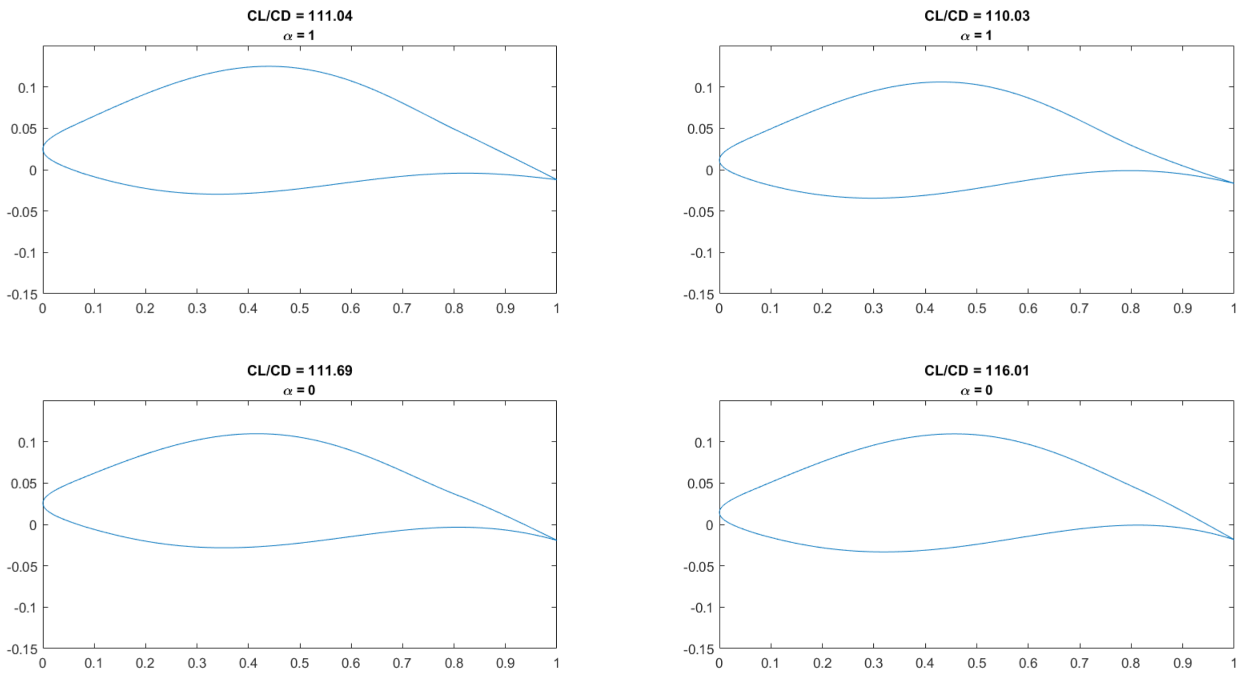


Figure 4.4: Selected airfoils

Figures 4.5, 4.6 and 4.7 illustrate the aerodynamic coefficients of both the S809 airfoil (used in NREL phase VI ) and the new selected one.

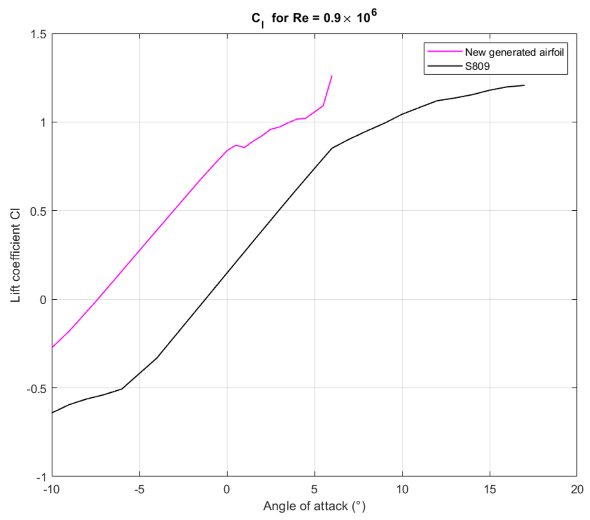


Figure 4.5: Lift coefficients

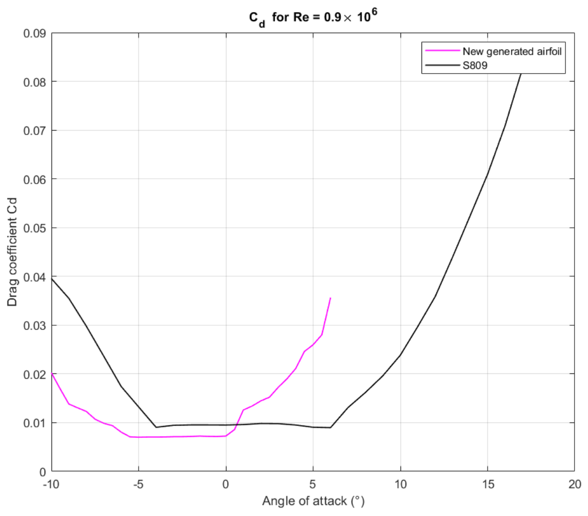


Figure 4.6: Drag coefficients

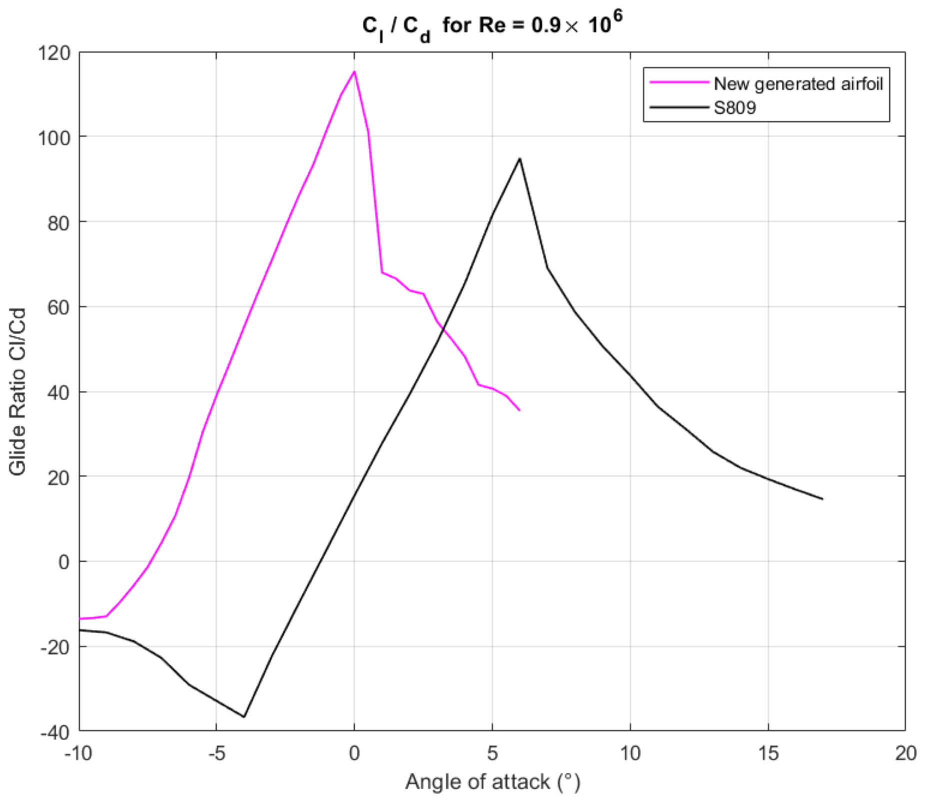


Figure 4.7: Glide ratios

The figure above highlights the difference between the performance of the two airfoils. The S809 reaches its maximum glide ratio at an AOA equal to 6°, while for the new airfoil it is reached at 0° AOA with a value **22%** higher than the one of S809. However, as can be seen in the plot, this difference is only this high around the peak value, as the lift to drag ratio of the new airfoil exhibits a relatively sharp drop compared to the S809.

## 4.4 Application to the NREL phase VI

The NREL Phase VI is taken as the reference blade [34], and the new blade is obtained by replacing the original S809 airfoil with the selected aerodynamic shape, while keeping the chord and twist distribution unchanged. The performance of both the turbines are then evaluated and compared.

Figures 4.8 and 4.9 illustrate the power output and power coefficients of the NREL Phase VI and the new modified turbine.

It is notable from the figures that the new modified turbine is only more efficient at higher wind speeds starting from 16 m/s equivalently for a TSR less than 2.5 . For lower wind speeds, the power output of the NREL phase VI is 2 kW greater than that of the other turbine, and with a power coefficient difference reaching 15%.

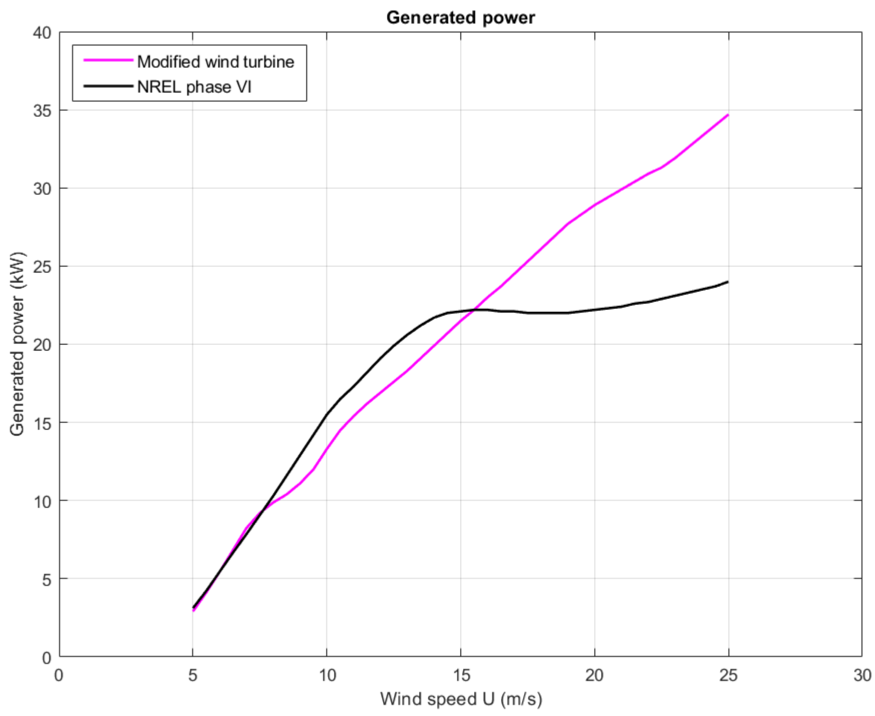


Figure 4.8: Power output

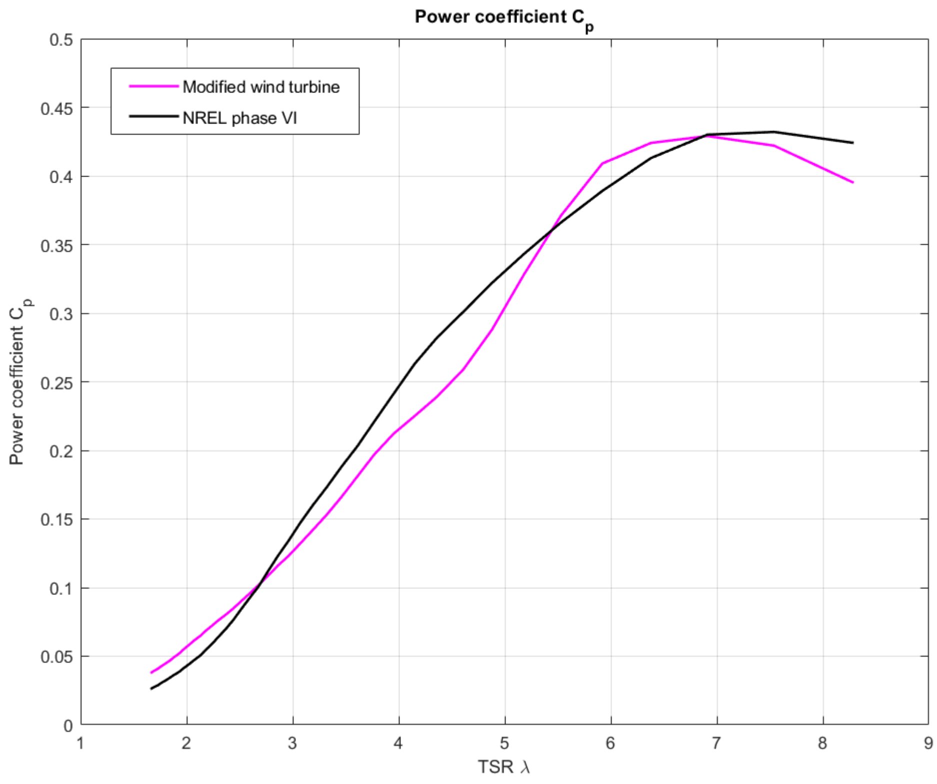


Figure 4.9: Power coefficient

It isn't surprising that the results for the new modified wind turbine aren't quite good, since no optimization of the chord and twist angle along the span was undertaken.
However, one particular modification that can be made is to change the blade pitch angle $\theta_{p,0}$ which is 3° for the reference turbine.

In fact, a close examination of the lift to drag ratios of both turbines, reveals that there's a 6° difference between the ranges of AOA for which the airfoils exhibit high aerodynamic performances, as indicated in figure 4.10 below.

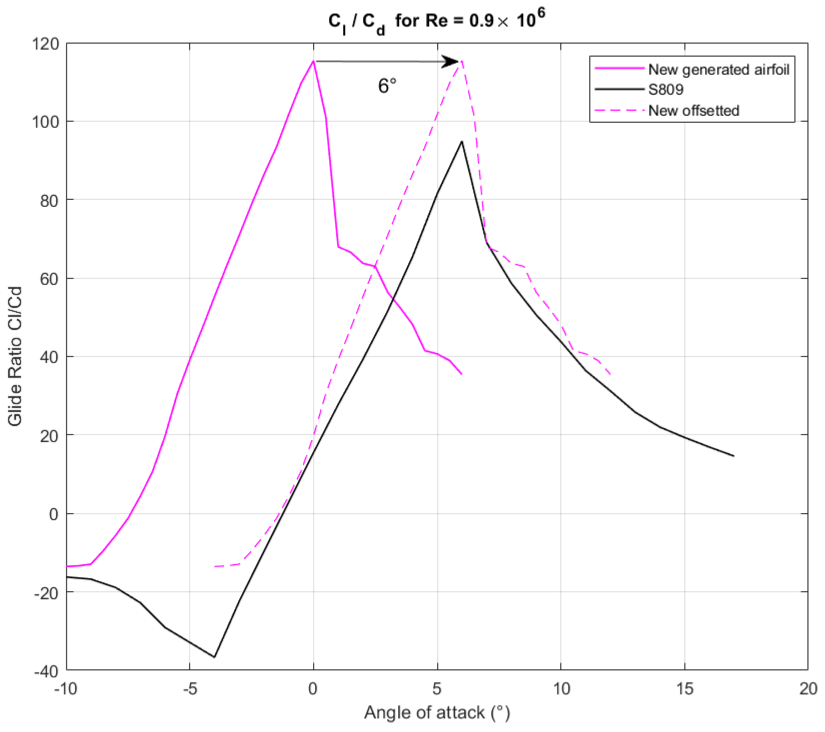


Figure 4.10: Offsetted lift to drag ratio

This 6° difference is added to the original blade pitch angle, to result in $\theta_{p,0} = 9°$.

A performance calculation is conducted for the modified turbine with the new pitch angle, and the results are then compared to the ones of NREL in the same way as the first case.

Figures 4.11 and 4.12 illustrate the power output and power coefficients of the NREL Phase VI and the new modified turbine with a blade pitch angle of 9°.

It can be seen in figure 4.11 that the generated power of the new turbine has improved compared to the previous one. The power output is now slightly greater than that of the reference turbine for low wind speeds, and much greater for high speeds. The range of wind speeds for which the NREL's output exceeds the new turbine's has narrowed from [7.5 – 15.5] m/s to [10 – 13] m/s, and the power difference dropped from 2 kW to 1.5 kW.

Figure 4.12 clearly shows that the new turbine is more efficient than the NREL phase VI except for the range of TSR values corresponding to wind speeds between [10 − 13] m/s. For very low wind speeds (TSR ranging from 7 to 8) the power coefficient is greater than 0.45 and 7% higher than that of the NREL, while in the first case it was 6% lower than it.

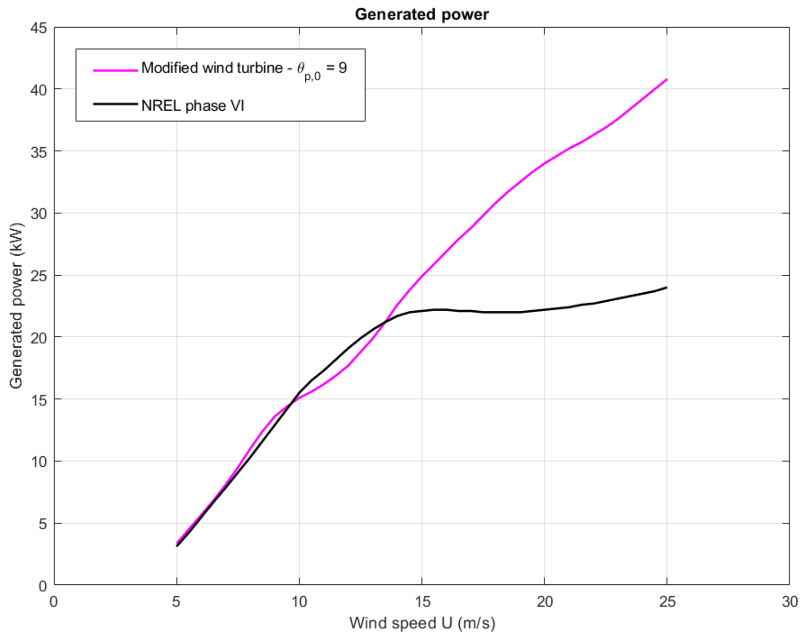


Figure 4.11: Power output for a 9° pitch angle

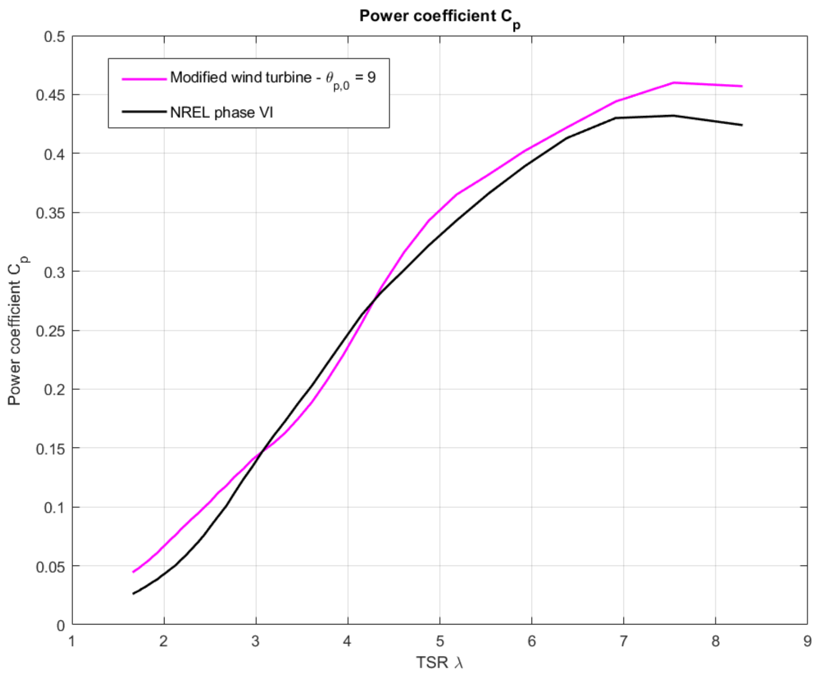


Figure 4.12: Power coefficient for 9° pitch angle

## 4.5 Conclusion

This chapter presented the results of the implemented ”airfoil_ GAN” code, from generation to validation. Performance of the selected airfoil was discussed and compared to that of the S809. A comparative analysis was then conducted on the performance of NREL phase VI and the new modified turbines.

Considering that no optimization was undertaken for the blade design, the results obtained when replacing the S809 airfoil with the best selected airfoil generated by the developed code are very promising. A simple small modification to the blade pitch angle was sufficient to improve the efficiency of the turbine, so an optimization of the blade design would definitely achieve better results. Thus, this chapter provided a successful validation of the code’s ability to accurately generate novel airfoil shapes with high aerodynamic efficiency that would contribute to the enhancement and optimization of wind turbines’ overall performance.

# General Conclusion

The purpose of this thesis work was to randomly generate novel airfoil designs with high aerodynamic performance using a Generative Adversarial Network. A resource-friendly Matlab code was developed for this purpose, using 3 key element methods that is, GANs, Panel method (through XFOIL) and the BEMT method.

The first part of the code established the GAN network by defining the generator and discriminator. The GAN was then trained over a diverse airfoil database. This part was quite challenging, because finding the right architecture and the right training parameters that achieved convergence was the hardest task in the entire work.

Next, the trained generator was used to create new shapes that were post-processed , resulting in unique airfoils. These airfoils were evaluated using XFOIL, and based on the calculated aerodynamic performance the best ones were selected.

By applying the selected airfoil to a reference wind turbine (the NREL phase VI ), and without any optimization of the blade design, the results of the new turbine's performance calculated with BEMT method were very promising.

So this work has not only contributed to the creation of highly efficient airfoils with limited computational resources, but also has practical implications for enhancing the efficiency and energy production of wind turbines, enabling the development of wind energy as a sustainable and efficient clean energy source. Furthermore, this work's implications extend to the field of aviation, if applied for wing optimization.

## Recommendations for future works

Many future works could be pursued and built upon the code developed in this thesis, to extend the research and explore further aspects. For this reason, the following recommendations and perspectives are proposed :

- ▶ The first futur work to be carried out is to extend the exploration of airfoil shapes. This involves generating and evaluating a significantly larger number of airfoils, up to a million novel designs, using the developed GAN code and clusters.

- ▶ An easy upgrade of the code can be done, by assigning labels to the training data set and leveraging the conditional architecture of the GAN. To assign labels, the airfoils in the database should be classified according to characteristics of interest depending on the study, this can be thickness, camber, aerodynamic coefficients or any other parameter. As a result, a selection phase will no longer be needed as the desired characteristics would be specified during the generation step.

- ▶ It would be interesting to consider multiple trainings, followed each time by an extension of the training set. This can be achieved by adding the new generated airfoils to the initial database, perform a new training, and then repeat this operation several times. This would introduce more variability and diversity in the generated samples. Thus, extending the exploration domain for aerodynamic shapes.

- A similar idea to the one above, is to create a new training set containing the new generated airfoils only. This would most likely achieve good results when a conditional GAN is used. Since the generated samples would surely have the desired characteristics, and introducing them as the training data would ensure that the generator learns from the best samples. Thus the generated shapes can only be as good as the training set or even better.

- Since the main reason of optimizing the airfoil shapes is to enhance the wind turbine's performance, a conditional GAN could be applied to generate new blade designs instead of airfoil shapes. This would naturally require a different set, containing optimized blade designs to which labels are assigned representing their performance and efficiency.

# Bibliography

[1] "Global wind report 2021, global wind energy council." https://gwec.net/global-wind-report-2021/. [Accessed: March 29, 2023].

[2] "World wind energy association." https://wwindea.org/world-market-for-wind-power-saw-another-record-year-in-2021-973-gigawatt-of-new-capacity-added/, May 2022. [Accessed: March 29, 2023].

[3] O. Cleynen, "Wing profile nomenclature." https://commons.wikimedia.org/wiki/File:Wing_profile_nomenclature.svg, 2011. [Accessed: April 22, 2023].

[4] P. Sharma, B. Gupta, M. Pandey, A. K. Sharma, and R. Nareliya Mishra, "Recent advancements in optimization methods for wind turbine airfoil design: A review," *Materials Today: Proceedings*, vol. 47, pp. 6556–6563, 2021. International Conference on Advances in Design, Materials and Manufacturing.

[5] J. D. Anderson, *Fundamentals of Aerodynamics*. McGraw-Hill Education, 6 ed., 2016.

[6] C. G. Velasco, "Aerodynamic forces on an airfoil." https://www.zonagravedad.com/modules.php?name=News&file=article&sid=781. [Accessed: March 30, 2023].

[7] S. Mathew, "Forces on wind turbine blade." https://www.lesics.com/wind-turbine-design.html, April 2014. [Accessed: March 30, 2023].

[8] E. N. Jacobs, K. E. Ward, and R. M. Pinkerton, "The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel," 1932.

[9] P. Fuglsang and C. Bak, "Development of the risø wind turbine airfoils," *Wind Energy*, vol. 7, no. 2, pp. 145–162, 2004.

[10] H. Sobieczky, "Parametric airfoils and wings," 1999.

[11] J. Hájek, "Parameterization of airfoils and its application in aerodynamic optimization," 2007.

[12] S. Sivanandam and S. Deepa, *Genetic Algorithms*, pp. 15–37. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[13] N. Bizzarrini, F. Grasso, and D. Coiro, "Genetic algorithms in wind turbine airfoil design," *EWEA, EWEC2011*, 01 2011.

[14] S. Giraldo, "Coupled cfd shape optimization for aerodynamic profiles," Master's thesis, Universidad EAFIT Medellin, 2015.

[15] R. Mukesh, K. Lingadurai, and S. Ulaganathan, "Kriging methodology for surrogate-based airfoil shape optimization," *Arabian Journal for Science and Engineering*, vol. 39, 10 2014.

[16] "Train conditional generative adversarial network (cgan)." https://www.mathworks.com/help/deeplearning/ug/train-conditional-generative-adversarial-network.html. [Accessed: April 6, 2023].

[17] "Monitor gan training progress and identify common failure modes." https://www.mathworks.com/help/deeplearning/ug/monitor-gan-training-progress-and-identify-common-failure-modes.html. [Accessed: April 15, 2023].

[18] "Generative adversarial networks." https://en.wikipedia.org/wiki/Generative_adversarial_network. [Accessed: April 2, 2023].

[19] W. Chen, K. Chiu, and M. Fuge, "Aerodynamic design optimization and shape exploration using generative adversarial networks," in *AIAA Scitech 2019 Forum*, p. 2351, 2019.

[20] E. Yilmaz and B. German, "Conditional generative adversarial network framework for airfoil inverse design," in *AIAA aviation 2020 forum*, p. 3185, 2020.

[21] G. Achour, W. J. Sung, O. J. Pinon-Fischer, and D. N. Mavris, "Development of a conditional generative adversarial network for airfoil shape optimization," in *AIAA Scitech 2020 Forum*, p. 2261, 2020.

[22] X. Du, P. He, and J. R. Martins, "A b-spline-based generative adversarial network model for fast interactive airfoil aerodynamic optimization," in *AIAA Scitech 2020 Forum*, p. 2128, 2020.

[23] J. Wang, R. Li, C. He, H. Chen, R. Cheng, C. Zhai, and M. Zhang, "An inverse design method for supercritical airfoil based on conditional generative models," *Chinese Journal of Aeronautics*, vol. 35, no. 3, pp. 62–74, 2022.

[24] X. Tan, D. Manna, J. Chattoraj, L. Yong, X. Xinxing, D. M. Ha, and Y. Feng, "Airfoil inverse design using conditional generative adversarial networks," in *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 143–148, 2022.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014.

[26] M. Drela, "Xfoil: An analysis and design system for low reynolds number airfoils," vol. 54, 06 1989.

[27] "The Panel Method - An Introduction." https://eng.libretexts.org/@go/page/77055, May 2 2022. [Accessed: May 15, 2023].

[28] "Xfoil docs : Formulations." https://v0xnihili.github.io/xfoil-docs/formulations/. [Accessed: May 21, 2023].

[29] A. L. R. James F. Manwell, Jon G. McGowan, *Wind Energy Explained: Theory, Design and Application, 2nd Edition.* Wiley, 2 ed., 2010.

[30] "Airfoils wings and other aerodynamic shapes." https://slidetodoc.com/chapter-5-airfoils-wings-and-other-aerodynamic-shapes/. [Accessed: June 10, 2023].

[31] "Uiuc airfoil coordinates database." https://m-selig.ae.illinois.edu/ads/coord_database.html. [Accessed: March 4, 2023].

[32] "Generate synthetic signals using conditional gan." https://www.mathworks.com/help/deeplearning/ug/generate-synthetic-pump-signals-using-conditional-generative-adversarial-network.html. [Accessed: April 17, 2023].

[33] M. Ge, L. Fang, and D. Tian, "Influence of reynolds number on multi-objective aerodynamic design of a wind turbine blade," *PLOS ONE*, vol. 10, pp. 1–25, 11 2015.

[34] "Unsteady aerodynamics experiment phase vi." https://www.nrel.gov/docs/fy02osti/29955.pdf. [Accessed: June 13, 2023].

[35] W. Chen, K. Chiu, and M. D. Fuge, "Airfoil design parameterization and optimization using bézier generative adversarial networks," *AIAA journal*, vol. 58, no. 11, pp. 4723–4735, 2020.

[36] P. M. V. Rodrigues, "Efficient aerodynamic optimization of aircraft wings," Master's thesis, Instituto Superior Técnico Lisboa, 2018.

[37] Q. Wang, J. Chen, X. Pang, S. Li, and X. Guo, "A new direct design method for the medium thickness wind turbine airfoil," *Journal of Fluids and Structures*, vol. 43, pp. 287–301, 2013.

[38] "Advanced courses in machine learning : Gans." https://developers.google.com/machine-learning/gan.

[39] "Train vae." https://www.mathworks.com/help/deeplearning/ug/train-a-variational-autoencoder-vae-to-generate-images.html. [Accessed: May 10, 2023].

[40] "Define custom training loops, loss functions, and networks." https://www.mathworks.com/help/deeplearning/ug/define-custom-training-loops-loss-functions-and-networks.html. [Accessed: April 6, 2023].

[41] M. Ge, D. Tian, and Y. Deng, "Reynolds number effect on the optimization of a wind turbine blade for maximum aerodynamic efficiency," *Journal of Energy Engineering*, vol. 142, p. 04014056, 09 2014.

# Appendices

# *NREL phase VI Geometry*

## A.1 Blade Geometry

Table A.1: Blade chord and twist distributions

| Radial distance $r$ (m) | Chord length $c$ (m) | Twist angle (°) | Airfoil |
|---|---|---|---|
| 0.0 | Hub | Hub | Hub |
| 0.508 | 0.218 | 0.0 | cylindrical |
| 0.660 | 0.218 | 0.0 | cylindrical |
| 0.883 | 0.183 | 0.0 | cylindrical |
| 1.008 | 0.349 | 6.7 | S809 |
| 1.067 | 0.441 | 9.9 | S809 |
| 1.133 | 0.544 | 13.4 | S809 |
| 1.257 | 0.737 | 20.040 | S809 |
| 1.343 | 0.728 | 18.074 | S809 |
| 1.510 | 0.711 | 14.292 | S809 |
| 1.648 | 0.697 | 11.909 | S809 |
| 1.952 | 0.666 | 7.979 | S809 |
| 2.257 | 0.636 | 5.308 | S809 |
| 2.343 | 0.627 | 4.715 | S809 |
| 2.562 | 0.605 | 3.425 | S809 |
| 2.867 | 0.574 | 2.083 | S809 |
| 3.172 | 0.543 | 1.150 | S809 |
| 3.185 | 0.542 | 1.115 | S809 |
| 3.476 | 0.512 | 0.494 | S809 |
| 3.781 | 0.482 | -0.015 | S809 |
| 4.023 | 0.457 | -0.381 | S809 |
| 4.086 | 0.451 | -0.475 | S809 |
| 4.391 | 0.420 | -0.920 | S809 |
| 4.696 | 0.389 | -1.352 | S809 |
| 4.780 | 0.381 | -1.569 | S809 |
| 5.000 | 0.358 | -1.775 | S809 |
| 5.305 | 0.328 | -2.191 | S809 |
| 5.532 | 0.305 | -2.500 | S809 |

## A.2 Airfoil S809 coordinates

Table A.2: S809 profile coordinates

| Upper Surface | | Lower Surface | |
|---|---|---|---|
| **x/c** | **y/c** | **x/c** | **y/c** |
| 0.00037 | 0.00275 | 0.00140 | -0.00498 |
| 0.00575 | 0.01166 | 0.00933 | -0.01272 |
| 0.01626 | 0.02133 | 0.02321 | -0.02162 |
| 0.03158 | 0.03136 | 0.04223 | -0.03144 |
| 0.05147 | 0.04143 | 0.06579 | -0.04199 |
| 0.07568 | 0.05132 | 0.09325 | -0.05301 |
| 0.10390 | 0.06082 | 0.12397 | -0.06408 |
| 0.13580 | 0.06972 | 0.15752 | -0.07467 |
| 0.17103 | 0.07786 | 0.19362 | -0.08447 |
| 0.20920 | 0.08505 | 0.23175 | -0.09326 |
| 0.24987 | 0.09113 | 0.27129 | -0.10060 |
| 0.29259 | 0.09594 | 0.31188 | -0.10589 |
| 0.33689 | 0.09933 | 0.35328 | -0.10866 |
| 0.38223 | 0.10109 | 0.39541 | -0.10842 |
| 0.42809 | 0.10101 | 0.43832 | -0.10484 |
| 0.47384 | 0.09843 | 0.48234 | -0.09756 |
| 0.52005 | 0.09237 | 0.52837 | -0.08697 |
| 0.56801 | 0.08356 | 0.57663 | -0.07442 |
| 0.61747 | 0.07379 | 0.62649 | -0.06112 |
| 0.66718 | 0.06403 | 0.67710 | -0.04792 |
| 0.71606 | 0.05462 | 0.72752 | -0.03558 |
| 0.76314 | 0.04578 | 0.77668 | -0.02466 |
| 0.80756 | 0.03761 | 0.82348 | -0.01559 |
| 0.84854 | 0.03017 | 0.86677 | -0.00859 |
| 0.88537 | 0.02335 | 0.90545 | -0.00370 |
| 0.91763 | 0.01694 | 0.93852 | -0.00075 |
| 0.94523 | 0.01101 | 0.96509 | 0.00054 |
| 0.96799 | 0.00600 | 0.98446 | 0.00065 |
| 0.98528 | 0.00245 | 0.99612 | 0.00024 |
| 0.99623 | 0.00054 | 1.00000 | 0.00000 |
| 1.00000 | 0.00000 | 0.00000 | 0.00000 |