5-2022

Machine Learning Based Aerodynamic Shape Optimization

Noe Martinez Jr.
The University of Texas Rio Grande Valley

MACHINE LEARNING BASED AERODYNAMIC SHAPE OPTIMIZATION

A Thesis

by

NOE MARTINEZ, JR.

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Mechanical Engineering

The University of Texas Rio Grande Valley

May 2022

MACHINE LEARNING BASED AERODYNAMIC SHAPE OPTIMIZATION

A Thesis
by
NOE MARTINEZ, JR.

COMMITTEE MEMBERS

Dr. Isaac Choutapalli
Chair of Committee

Dr. Robert Freeman
Committee Member

Dr. Horacio Vasquez
Committee Member

Dr. Stephen Crown
Committee Member

Dr. Arturo Fuentes
Committee Member

Dr. Constantine Tarawneh
Committee Member

Dr. Wenjie Dong
Committee Member

May 2022

ABSTRACT

Martinez Jr, Noe, Machine Learning Based Aerodynamic Shape Optimization. Master of Science in Engineering (MSE), May, 2022, 111 pp., 17 tables, 54 figures, references, 25 titles.

The coefficient of pressure distribution for various 2D airfoil geometries were found using source – vortex panel methods. The data obtained in these simulations was used in multiple machine learning models which would predict the airfoil geometry from a given coefficient of pressure distribution. The neural networks employed were fully connected feedforward networks with Levenberg – Marquardt backpropagation and one model employed Bayesian Regularization. A novel tool for optimizing airfoil shape for a given coefficient of pressure distribution was created which performed well during testing. These models serve as the first step in minimizing the conflict between aerodynamic and stealth design for both low-speed and high-speed aircraft.

DEDICATION

I dedicate this work to God, my father Noe Martinez, Sr., my mother Norma Linda G. Martinez, my sisters, Crystal and Stephanie, and my love, Katarina Almanza. All of you were always there to support me through this journey and believed in me every step of the way. Without all of you, I could not have come this far and will forever be grateful.

ACKNOWLEDGMENTS

I would first like to acknowledge Dr. Isaac Choutapalli, my committee chair and advisor, who gave me this wonderful opportunity and guided me along in the process. I also thank all the other professors on my thesis committee who helped to guide this thesis in the right direction and provide constructive criticism when needed. Finally, I would also like to thank the researchers within the Aerodynamics and Propulsion Laboratory who gave me comradery and support along the way.

TABLE OF CONTENTS

Page

ABSTRACT iii

DEDICATION iv

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vi

LIST OF TABLES x

LIST OF FIGURES xi

CHAPTER I. INTRODUCTION 1

Introduction 1

Background 1

Flow Optimization 1

Airfoil Optimization 3

Techniques 4

Neural Networks 5

Literature Review 6

Types of Airfoil Optimization 6

Data Acquisition and Generation....9
Neural Networks....10
Summary and Discussion....16
Motivation for Thesis....16
Organization of Thesis....17
CHAPTER II. APPROACH AND METHODOLOGY....18
Introduction....18
Procedure....18
Data Acquisition....20
Preprocessing....21
Data Reformatting....21
MATLAB Implementation....22
Panel Methods....26
Source Panel Methods....26
Source - Vortex Panel Methods....30
MATLAB Implementation....35
Neural Network Procurement....39
Gauss – Newton Method....40
Levenberg-Marquardt....42
Bayesian Regularization....43

MATLAB Implementation 46
Post Processing 49
Savitzky – Golay Filtering 49
CHAPTER III. NEURAL NETWORK TRAINING AND TESTING 52
Introduction 52
Training 53
Testing 56
CHAPTER IV. RESULTS AND DISCUSSION 59
Introduction 59
Model 1 61
Training 61
Testing 65
Model 2 67
Training 67
Testing 71
Model 3 73
Training 73
Testing 77
Model 4 79
Training 79

Testing....83

Discussion....85

CHAPTER V. CONCLUSION AND FUTURE WORK....87

Prediction Results....89

Future Work....91

REFERENCES....92

APPENDIX A....94

APPENDIX B....100

BIOGRAPHICAL SKETCH....111

LIST OF TABLES

Page

Table 1: Sample coordinates of airfoil AH85l120 22

Table 2:Subest of testing MSE for Model 1 62

Table 3:Subset of raw and smoothed Airfoil MSE for Model 1 63

Table 4: Raw and smoothed MSE values for testing Model 1 65

Table 5: Subest of testing MSE for Model 2 68

Table 6: Subset of raw and smoothed Airfoil MSE for Model 2 69

Table 7: Raw and smoothed MSE values for testing Model 2 71

Table 8: Subest of testing MSE for Model 3 74

Table 9: Subset of raw and smoothed Airfoil MSE for Model 3 75

Table 10: Raw and smoothed MSE values for testing Model 3 77

Table 11: Subest of testing MSE for Model 4 80

Table 12: Subset of raw and smoothed Airfoil MSE for Model 4 81

Table 13: Raw and smoothed MSE values for testing Model 4 83

Table 14: Smoothed output of training airfoils 85

Table 15: Smoothed output of the 10 benchmark airfoils 86

Table 16: Smoothed training airfoil MSE values for all models 95

Table 17: Individual network MSE values for training and testing datasets 98

LIST OF FIGURES

Page

Figure 1: Pressure distribution over an airfoil 2

Figure 2: Aerodynamic forces acting on an airfoil 2

Figure 3: Airfoil drag minimization of NACA 0012. Image via Hicks, 1974 4

Figure 4: Simple neural network diagram 5

Figure 5: Inverse and direct numerical design methods 8

Figure 6: GAN architecture for generating aged faces. Image via Achour, 2020 11

Figure 7: GAN architecture for generating airfoils. Image via Achour 2020 12

Figure 8: Bezier-GAN architecture. Image via Chen, 2019 13

Figure 9: CNN architecture to produce an airfoil geometry. Image via Sekar, 2019 14

Figure 10: Genetic algorithm flow chart. Image via Akram, 2018 15

Figure 11: Flowchart for procedure 19

Figure 12: a) Plot of airfoil B540ols b) Plot of airfoil AH85l120 21

Figure 13: Original and polynomial fit of airfoil RAE100 25

Figure 14: Source panel method coordinate system 26

Figure 15: (a) A discretized airfoil surface into panels (b) Tangent and normal unit vectors at point i and j 27

Figure 16: Vortex panel method coordinate system 31

Figure 17: Kutta condition 34

Figure 18: Coefficient of pressure distribution at AoA=4 38
Figure 19:Coefficient of pressure distribution for various angles of attack.....38
Figure 20: Data formatting for model 1 and 2.....47
Figure 21: Data formatting for model 3 and 4.....47
Figure 22: General neural network architecture 48
Figure 23: Savitzky-Golay filter diagram 50
Figure 24: a) Raw neural network output b) Smoothed neural network output 51
Figure 25:Neural network data utilization.....55
Figure 26:Raw neural network output of airfoil AH85l120.....56
Figure 27:Raw neural network output of airfoil GOE775 57
Figure 28:Bar graph of test MSE for Model 1.....62
Figure 29:a) Raw Model 1 output of airfoil NACA0010 b) Filtered Model 1 output of airfoil NACA0010.....64
Figure 30:a) Raw Model 1 output of airfoil NACA63012a b) Filtered Model 1 output of airfoil NACA63012a.....66
Figure 31: Bar graph of test MSE for Model 2.....68
Figure 32: a) Raw Model 2 output of airfoil NACA0010 b) Filtered Model 2 output of airfoil NACA0010.....70
Figure 33: a) Raw Model 2 output of airfoil NACA63012a b) Filtered Model 2 output of airfoil NACA63012a.....72
Figure 34: Bar graph of test MSE for Model 3.....74
Figure 35: a) Raw Model 3 output of airfoil NACA0010 b) Filtered Model 3 output of airfoil NACA0010.....76

Figure 36: a) Raw Model 3 output of airfoil NACA63012a b) Filtered Model 3 output of airfoil NACA63012a 78
Figure 37: Bar graph of test MSE for Model 4 80
Figure 38: a) Raw Model 4 output of airfoil NACA0010 b) Filtered Model 4 output of airfoil NACA0010 82
Figure 39: a) Raw Model 4 output of airfoil NACA63012a b) Filtered Model 4 output of airfoil NACA63012a 84
Figure 40: Smoothed training airfoil output 101
Figure 41: Smoothed training airfoil output 102
Figure 42: Smoothed training airfoil output 103
Figure 43: Smoothed training airfoil output 104
Figure 44: Smoothed training airfoil output 105
Figure 45: Smoothed output for airfoil NACA63012A 105
Figure 46: Smoothed output for airfoil NACA63015a 106
Figure 47: Smoothed output for airfoil GOE459 106
Figure 48: Smoothed output for airfoil GOE460 107
Figure 49: Smoothed output for airfoil GOE775 107
Figure 50: Smoothed output for airfoil NACA001234 108
Figure 51: Smoothed output for airfoil NACAM3 108
Figure 52: Smoothed output for airfoil OAF139 109
Figure 53: Smoothed output for airfoil RAF27 109
Figure 54: Smoothed output for airfoil RAF30 110

CHAPTER I

INTRODUCTION

Introduction

Airfoil optimization is a field which comprises many different techniques that help in the design of an airfoil for a specific application. Research into these techniques is continually being done to advance the field and to aide designers for airfoil shape, wing shape and aerodynamic shape. Optimization may be used to increase operational angles of attack, reduce the coefficient of drag, or optimize the coefficient of lift to drag ratio, among many other characteristics.

This chapter will discuss the different techniques that are used in airfoil optimization and the motivation for this study. The state-of-the-art of airfoil optimization will be explored within the literature review as well.

Background

Flow Optimization

The flow of air or any fluid over a body is critical to understanding what aerodynamic characteristics it may exhibit. With airfoil optimization, shape can be modified by a variety of methods which will affect the overall flow around the body. Foundational to all of these characteristics is the velocity gradient, vorticity gradient and pressure gradient around the airfoil which are all tied to each other. Coefficient of pressure distribution, which is shown in

Figure 1, can give a snapshot of what the pressure distribution of the flow around the airfoil will look like and how that will affect the aerodynamic characteristics such as lift and drag, which is shown in Figure 2.

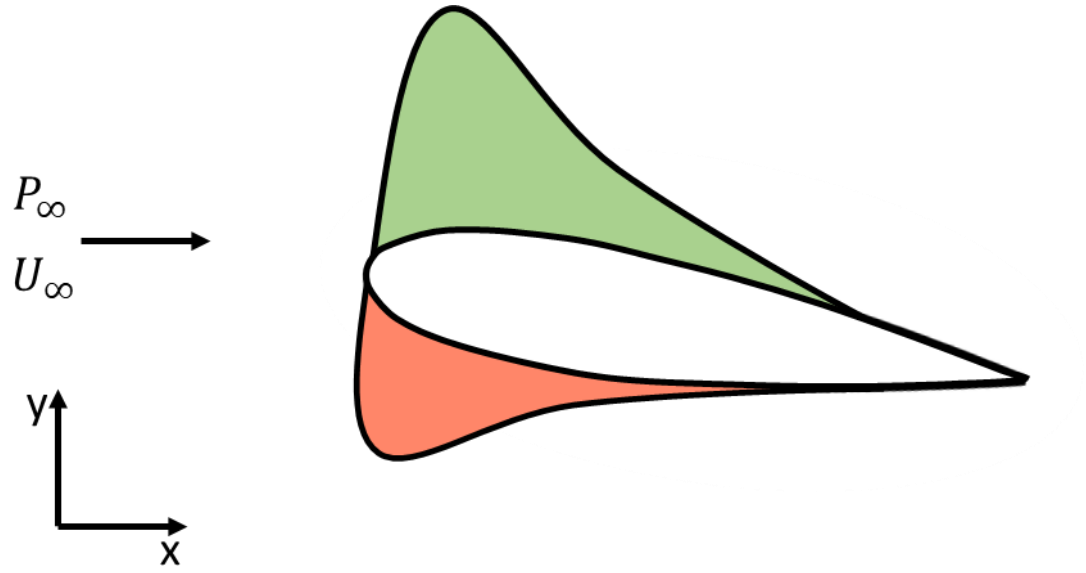


Figure 1: Pressure distribution over an airfoil

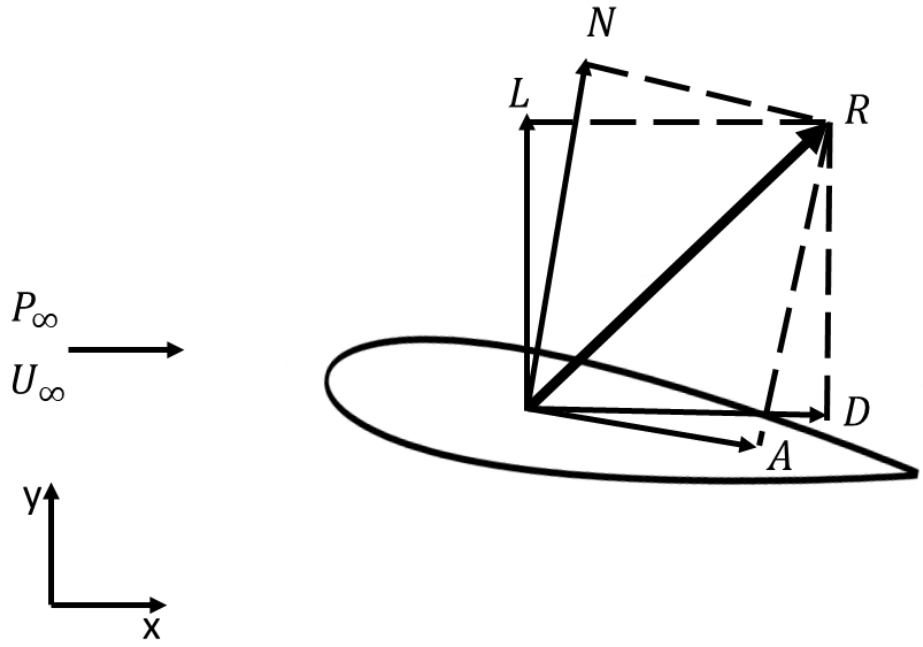


Figure 2: Aerodynamic forces acting on an airfoil

Mathematically, the coefficient of pressure is directly related to the coefficient of lift and the coefficient of drag which are important when calculating the total lift and drag force of a three-dimensional wing. Knowing what the lift and drag characteristics of an airfoil over different angles of attack is crucial when designing wings or other aerodynamic lifting surfaces. These forces can be seen in Figure 2 where lift is denoted by L, drag by D, normal force N, axial force A, and the resultant force R. Lift and drag are in the global coordinate system while normal and axial force are in terms of the airfoil coordinate system.

Different studies into airfoil shape optimization will have different opinions or attitudes on what should be optimized or modified and how that should be done. Some focus on coefficient of pressure, coefficient of lift, or the coefficient of lift to drag ratio. Depending on the application and the goal of the study, one may be more beneficial than another to investigate.

Airfoil Optimization

Starting in the 1960s there have been efforts to optimize the performance of an airfoil (Hague et al., 1968). Initially they were based on optimization programs which would change the geometry of a given airfoil to reach an optimization goal such as drag minimization or airfoil volume maximization at specific flow conditions relating to Mach number or Reynolds number which can be seen in Figure 3 (Hicks et al., 1974). Nowadays, the approach is much different thanks to the advances in machine learning techniques and computer processing power. Current studies rely on neural networks or evolutionary algorithms to either optimize a given airfoil for optimization criteria or to generate an airfoil for a desired aerodynamic characteristic (Akram et al., 2021). The main motivation for airfoil optimization or shape generation is to produce an airfoil which is best suited for a particular application. The optimum airfoil design for a fighter jet compared to a passenger jetliner will be very different because of their operational

constraints. Even between aircraft that are similar, airfoils will be optimized for specific characteristics; to create more lift or reduce drag or increase the lift to drag ratio. It is for these reasons that airfoil optimization can be an invaluable tool for aircraft designers and engineers when designing an aerodynamic vehicle and why it is needed in the aerodynamic design process.

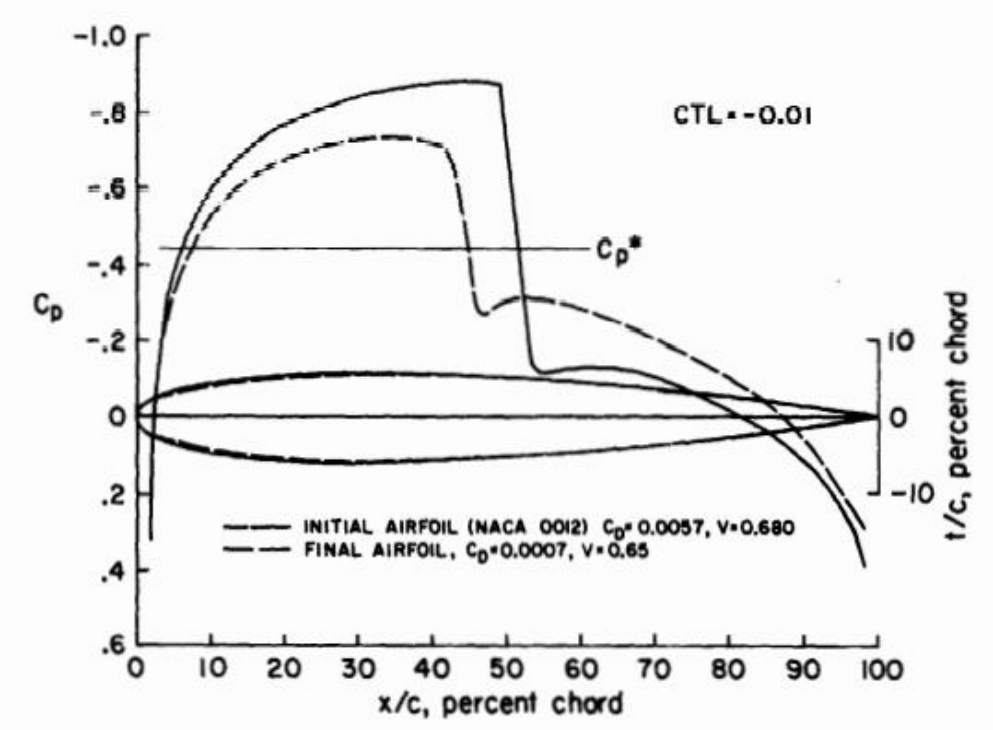


Figure 3: Airfoil drag minimization of NACA 0012. Image via Hicks, 1974

Techniques

Initially, studies utilized optimization algorithms and quickly the field moved to other approaches such as genetic algorithms. Currently, most studies use some type of neural network that may use empirical, simulation or image data or a combination of these as the input.

In the 1960s and 1970s optimization algorithms such as method of feasible directions were used in the beginning of this field's history (Hicks et al., 1974). Shortly thereafter there were algorithms based on sequential applications of a Taylor's series (Vanderplaats et al., 1979). Quasi-Newton methods were used in 1983 by Kennelly for a pseudo-inverse airfoil optimization

problem. Kharal and Saleem in 2012 used different types of neural networks to generate airfoils for a given coefficient of pressure distribution. This type of approach is now widespread and most published papers on this subject utilize some form of neural network or genetic algorithm for airfoil optimization.

Neural Networks

Neural networks are the basis of machine learning. This is a field that has emerged out of optimization algorithms to simulate the human brain and the interactions that individual neurons have to one another. A simple fully connected, feedforward neural network architecture can be seen in Figure 4 which contains an input layer with two inputs, a hidden layer with three neurons and a single output, output layer. Depending on the application of the neural network, they can be used for simple regression tasks or for generating artificial images of human faces.

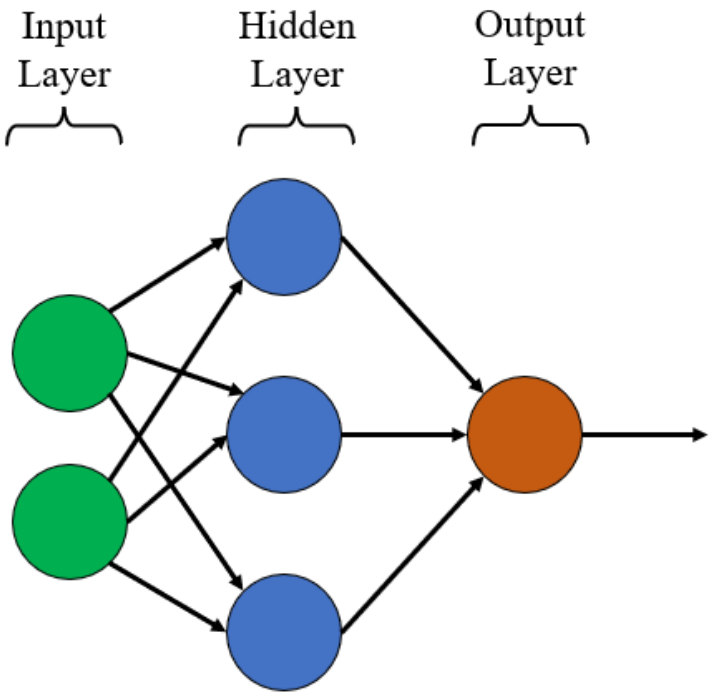


Figure 4: Simple neural network diagram

Some networks are better at taking large datasets of complex data while others are better suited for smaller tasks. For this reason, it is important that when setting out to use a neural network to correlate data, that the problem is well understood. The wide variety of networks as well as the problems they can tackle can be overwhelming, but when used correctly can produce great results.

Literature Review

The field of airfoil optimization is diverse in terms of the methods used and the objectives each study aims to fulfill. There is the inverse design which takes a desired aerodynamic characteristic such as coefficient of pressure distribution, coefficient of lift distribution or coefficient of drag distribution and by utilizing some model, generates an airfoil which would have those characteristics. The other design method would be to start with an airfoil shape and iteratively modify the geometry so that the airfoil has a better characteristic in specific flow conditions such as decreased drag at a certain Mach number or specific angle of attack. Another important aspect of airfoil optimization is what type of model or method will be used and what type of data will be used. Some studies utilize a mix of database airfoils and machine generated airfoils while others use either just one or the other. The data on these airfoils, whether it be on the coefficient of lift, drag, or pressure will be acquired through different techniques such as panel methods or other premade applications such as XFOIL or some CFD application. Finally, the method used to optimize the shape of the airfoil can range from numerical methods to machine learning based approaches which on their own are wide and diverse fields.

Types of Airfoil Optimization

Some airfoil optimization focuses on taking a known airfoil and iteratively modify the airfoil geometry to improve certain aerodynamic characteristics. This is what Della Vecchia et al.

called direct numerical design in 2014. Going back to 1974 where Hicks et al. utilized an optimization method based on a method of feasible directions, this approach has been in use since the beginning of airfoil optimization. In that study, multiple problems were tackled such as drag minimization at Mach number M=0.8, airfoil volume maximization at M=0.8, drag minimization at M=0.85, and drag minimization at M=1.3. Each of those cases either used NACA 0006, NACA 0012, or an arbitrary body as the initial airfoil shape. This study helped to popularize airfoil optimization utilizing numerical methods and spur new interest so that more studies were published on the topic. A few years later in 1979, Vanderplaats who was in the previous publication, published another study on airfoil optimization which already shows improvement to the previous approach by a factor of two by developing an optimization algorithm which was based on sequential application of a second-order Taylor's series approximation of the airfoil characteristics.

There is also the inverse design method which compiles the data from a collection of airfoils to generate an airfoil geometry which fulfils a specific aerodynamic characteristic. This would look like entering the desired pressure distribution and using a model to generate an airfoil geometry which would have that pressure distribution such as in (Kennelly, 1983). In that study, the subject of transonic airfoils was used while an optimization program was designed to compute the geometry of the airfoil more economically. The aim of the study was not necessarily to generate the airfoil but to improve the efficiency of the optimization program which was quasi-Newton based. This was called the pseudo-inverse problem since an understanding of what the pressure distribution would look like is needed or else the output won't be usable. Since the early days of airfoil optimization and shape generation, there have been many improvements, mainly in terms of what type of optimization solver is being used or in terms of the modern

approaches, what neural network architecture is being used. A comparison of the two main methods of airfoil optimization is shown in Figure 5. The diagram shows the example of inverse design by generating an airfoil shape from a coefficient of lift distribution by utilizing a neural network which was fed a large dataset of airfoils. An example of direct numerical method which takes an initial airfoil and through some iterative process generates an improved airfoil is also shown.

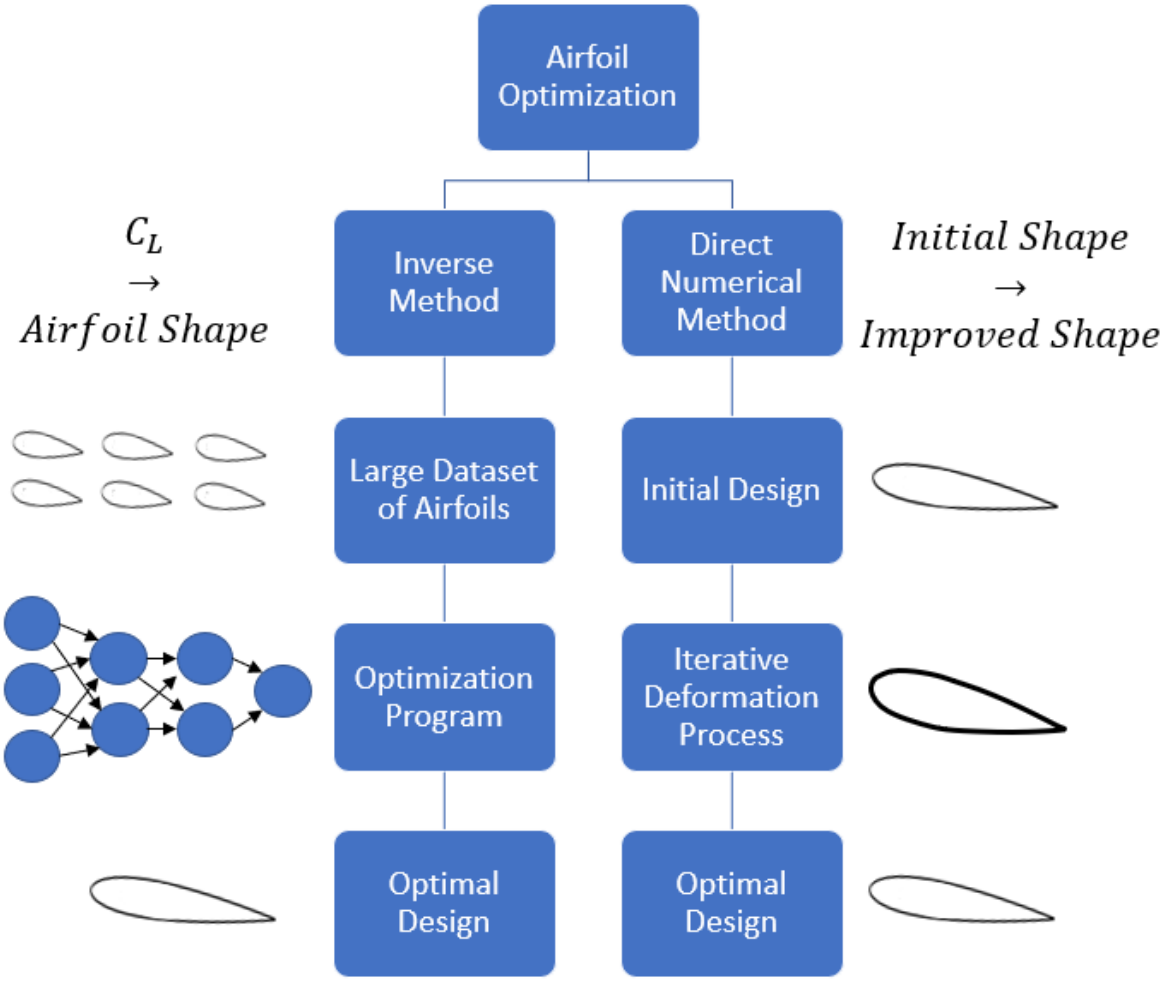


Figure 5: Inverse and direct numerical design methods

Data Acquisition and Generation

Once the type of airfoil optimization is chosen, the next step would be to acquire data which will greatly affect how the optimization package will perform overall. If there is bad data, then the model will not be accurate and depending on what the aim of the study is, certain data will have to be chosen or thrown out. The case of subsonic airfoil optimization is different from transonic airfoil optimization and there may be differences in terms of symmetrical airfoils compared to unsymmetric airfoils. All these considerations must be made to further hone the model and whether it is better to create a model that generalizes all airfoils or one which is designed for more specific cases.

Going back to the beginning of airfoil optimization, the data was obtained by using relaxation methods, which are similar to the panel methods used later on (Hicks et al., 1974; Kharal et al., 2012). Eventually data started to be obtained through CFD methods which became more economical as time went on (Kennelly, 1983). These methods would also be used for data validation, so that when the output of the optimization package was obtained, it was fed back into whatever solver was used to create the data. That was then cross validated and used to give the system an overall score on how well it converges on the correct result.

In terms of panel methods, a commonly used application which uses this method is XFOIL. It is possible to find the pressure distribution, coefficient of lift, drag and moment at different angles of attack as well as different flow conditions (Chen et al., 2019). There are also databases with wing and airfoil data which can be used as the basis of the optimization package (Sun et al., 2015). Depending on the application of the optimization package and the availability of data, certain approaches may be more beneficial than others given the circumstances.

Neural Networks

Nowadays, most approaches utilize some form of artificial neural network (ANN) to correlate the aerodynamic data with the airfoil geometry, this is for the inverse design. In terms of the direct numerical method, evolutionary and genetic algorithms, primarily, are used as the basis of the optimization package.

There are many different types of ANNs, such as the generative adversarial network (GAN), multi-layer perceptron (MLP), convolutional neural network (CNN), and many different adaptations of these networks which exist and may be used for airfoil optimization. GAN employs a method which takes an initial design which fulfils a specific criteria, and combines that input with a generator which adds a random amount of noise. This then becomes a newly generated set of data which is then fed to a discriminator and can give a probability for that data to fulfil the initial criteria. Through many iterations the generator is trained with the discriminator's output and in the end, a trained generator will be able to generate a dataset which can fulfill a specific criterion.

The classic example for this approach is realistic facial generation or artificial facial aging which is shown in Figure 6. A dataset of faces with labels which determine the age is used in conjunction with a generator, this generated data is then fed to a discriminator which will give a probability for the generated image to be real and what age it is. Once the generator is fully trained, a facial image and age can be input, and the aged output will be generated.

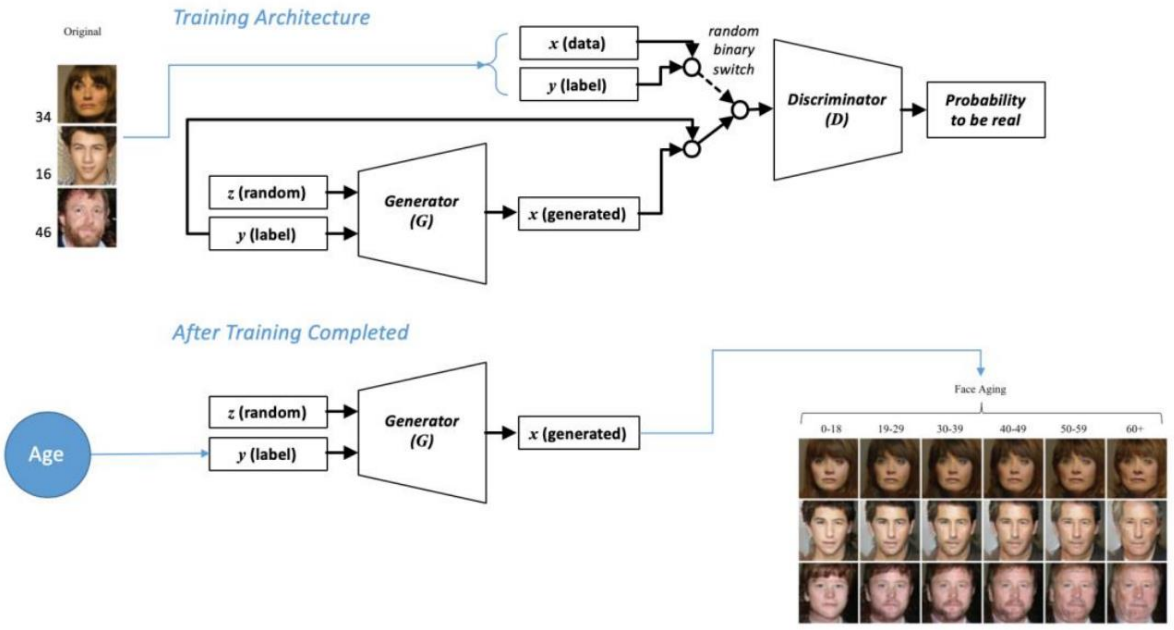


Figure 6: GAN architecture for generating aged faces. Image via Achour, 2020

In terms of airfoil optimization, airfoil shape and aerodynamic labels which denote the coefficient of lift for example can be fed in conjunction with a generator to a discriminator which may run a panel method application such as XFOIL to determine what the generated shape's label would be. The trained generator will now be able to generate an airfoil which can generate an airfoil which fulfills a criterion such as a coefficient of lift at a certain angle of attack. This type of method was used by Achour in 2020, where a modified GAN architecture named conditional GAN (CGAN) was developed and tested. The general architecture of the CGAN is shown in Figure 7 and the results showed that 75% of the generated airfoils successfully converged to have the correct lift to drag ratio and airfoil area.

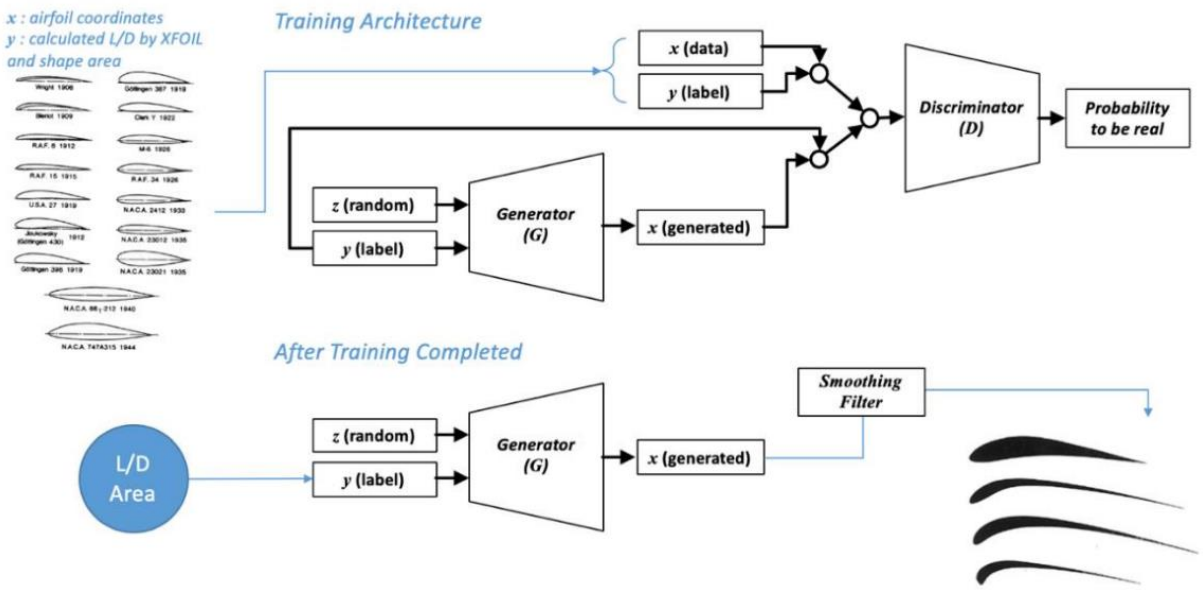


Figure 7: GAN architecture for generating airfoils. Image via Achour 2020

There also exists variations to the GAN architecture such as Bezier-GAN, and conditional Wasserstein GAN (CWGAN). In the case of Chen in 2019, a so-called Bezier-GAN was created to generate realistic airfoil shapes. Bezier curves were used to smoothen the generated shape before going into the discriminator. These generated airfoils were then fed into optimization methods and genetic algorithms to further improve the shape for the given aerodynamic criterion which can be seen in Figure 8. It was concluded that the Bezier-GAN can be a good starting point and gradient-based optimization methods could be used to further refine the generated shapes. This shows the efficiency that a GAN generated dataset can have to generate realistic airfoil geometries, but also shows how much noise and imperfections that a GAN can imbue on the output.

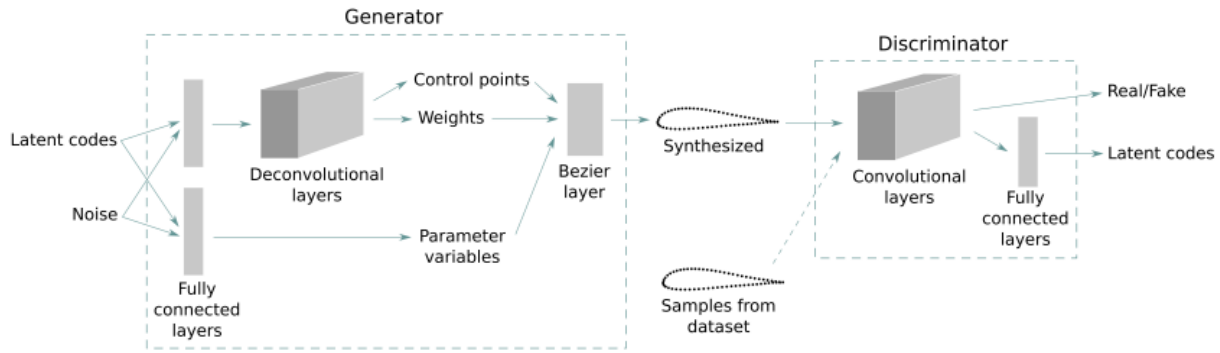


Figure 8: Bezier-GAN architecture. Image via Chen, 2019

For there to still be more processing and genetic algorithms as well as optimization methods to be used to smoothen the shape and further optimize, this shows that GAN networks and many other machine learning techniques will inherently have noise and undulations in the output data which should be mitigated and accounted for.

Another network is the deep convolutional network or deep CNN. Deep denotes that there are multiple hidden layers of neurons and the convolutional network means that there are multiple kernels or filters which are smaller than the previous layer. This means that the data that is input is convolved to a smaller number of data points each layer which is then pooled together into a new set of data. This can then be fed to a fully connected layer which is where the neural network comes in. The fully connected layers consist of neurons which are then fed forward into the output neurons. This method is usually used to identify images since the original image resolution can be large, it is convolved and pooled into a smaller image or representation of an image which can then be fed to the neural network architecture to be correlated. CNNs can also be used on data which is two-dimensional or three-dimensional which is very large since it would be hard for a regular neural network to be able to process the raw data. Convolution allows for higher-order characteristics to be determined. For airfoil optimization this can be very useful as seen in Sekar and Yilmaz in 2019 and 2018. For the first study, the input data were

images which were 144x144 pixels of the coefficient of pressure data. This was used to produce the airfoil shape as the output. The network architecture of this CNN is shown in Figure 9 where C signifies a convolutional layer, P indicates a pooling layer, FC denotes the fully connected neurons and y is the output layer of y-coordinates.

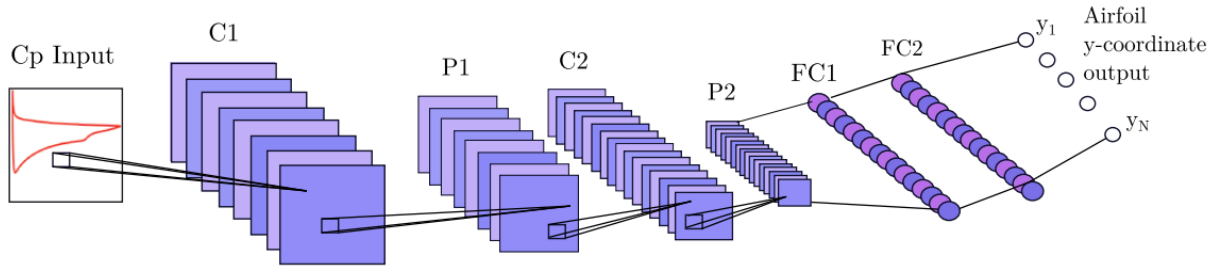


Figure 9: CNN architecture to produce an airfoil geometry. Image via Sekar, 2019

For direct numerical design, the approach of genetic and evolutionary algorithms has been widely used. Genetic algorithms work by inputting a "parent" airfoil which is a starting point of the optimization. This parent will then produce offspring which will have some mutations. Once enough offspring are created, the best will be selected with a fitness function as parents for a new generation and the process starts again. Over time this iterative approach will produce an optimal shape. Surrogate modeling is usually used in this method because it can lessen the computational time needed to evaluate the fitness of the offspring by using a surrogate instead of a CFD program or some other application which can be computationally intensive. One case in which a genetic algorithm was utilized was with Akram and Kim in 2018 and the construction of the genetic algorithm is shown in Figure 10. It was found that the drag coefficients for a subsonic and transonic airfoil decreased by 10% and 12%, respectively. The lift

to drag ratios also improved by 7.4% and 15.9%, respectively. This was all at 3 degrees angle of attack.

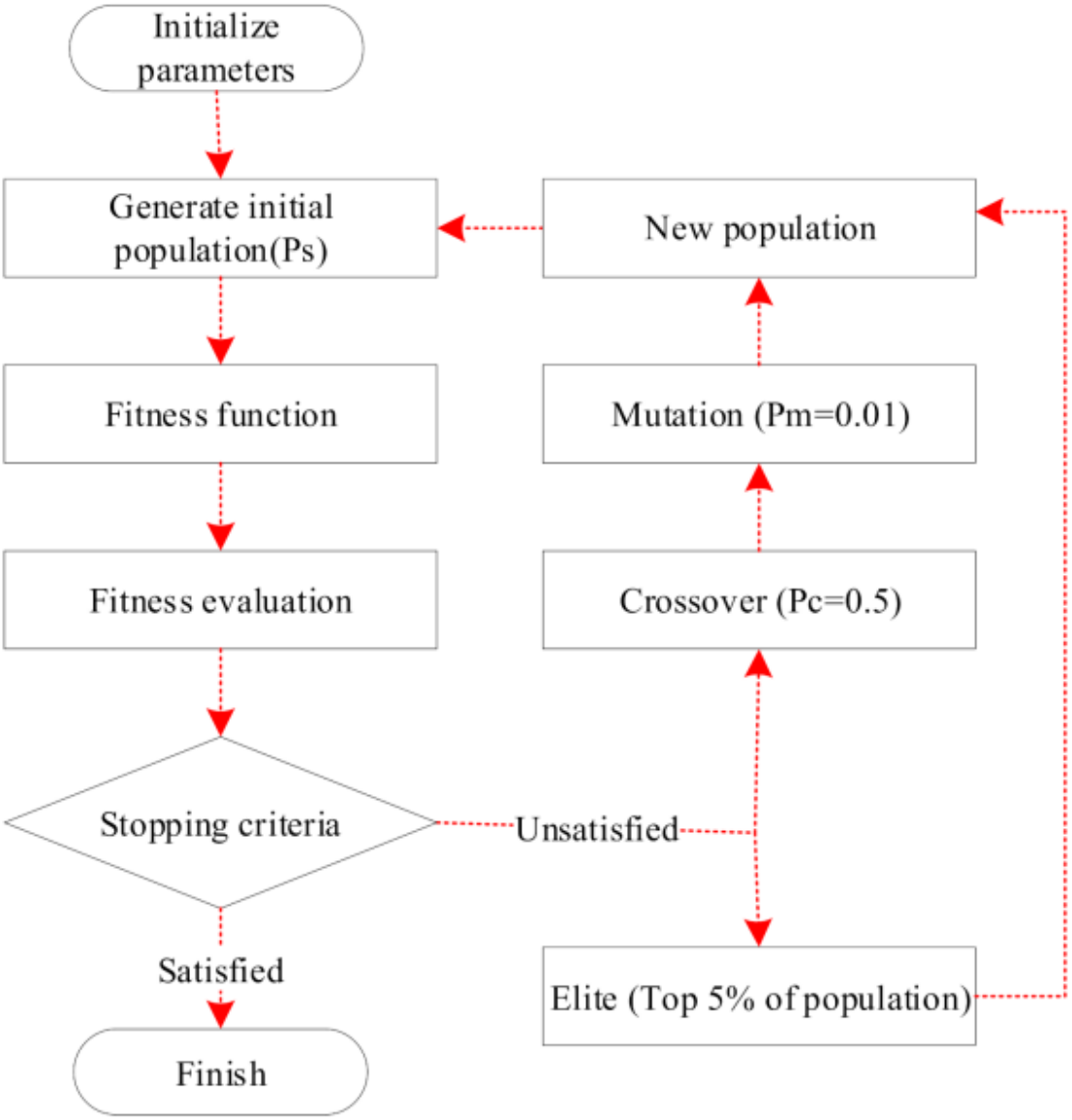


Figure 10: Genetic algorithm flow chart. Image via Akram, 2018

Summary and Discussion

Airfoil optimization is a wide field of methods and techniques that can iteratively modify an existing airfoil or utilize neural networks for the inverse design and generate an airfoil which meets some aerodynamic characteristic criterion. There are many ways to go about data acquisition such as using empirical data, databases, or creating unique data which can be used to train those neural networks. Even so, genetic algorithms can be utilized to further optimize and search for the best airfoil geometry for the targeted value. There are many different neural network architectures that exist and may be of worth to investigate for use in an airfoil optimization package. There is no perfect technique, though some studies have come close, but there are ways to mitigate the undulations and noise that is inherent with machine learning algorithms. Many post processing methods, airfoil parameterization techniques, and so on exist which may make the problem easier to tackle or allow the output data to be smoothed into something that is usable.

Motivation for Thesis

Airfoil optimization is an invaluable tool that engineers and aerospace designers can utilize when designing wings and other aerodynamic surfaces for lift generation. It allows the user to either optimize an existing airfoil shape or generate an optimal shape which is unique. In terms of optimization and efficiency, these are always desired when the desired aerodynamic characteristics are critical to the design. Thus, any pursuit to create a new tool or optimization package which may help in this process is valuable. There exists a need for more research into this topic so that more methods can be created to generate optimal designs.

In previous studies, the inverse design method includes taking either coordinate or parameterization data and correlating it to some aerodynamic characteristic or pressure

distribution. This study aims to find a direct correlation between the y-coordinates of a symmetric airfoil and the coefficient of pressure distribution at that point by utilizing machine learning methods.

Organization of Thesis

This section has introduced airfoil optimization and the techniques that may be used as well as the motivation for this thesis. The following section will go over the approach that this study takes to airfoil optimization. This includes data acquisition, preprocessing, neural network procurement and post processing. Each will be explained in detail and the subsequent chapters will detail the results from the study as well as the conclusions that can be made on this data and any potential direction changes that can be made for future work.

CHAPTER II

APPROACH AND METHODOLOGY

Introduction

To combine all the techniques and methods into an actual study on real airfoils, the approach must be described so that there is a procedure for going forward. Airfoil optimization and shape generation is a complex task when all these factors are considered. The problem has been defined, the inverse design method will be pursued, where coefficient of pressure data will be used to generate the optimal airfoil. First, all airfoil data and coefficient of pressure data must be acquired, preprocessed, and normalized to be used in a machine learning model. An optimal neural network or multiple networks will be determined so that it can correlate the data and finally the output will be postprocessed.

Procedure

The flow for this procedure is shown in Figure 11 which shows the major steps of the process to obtain a working optimization package for generating an airfoil. The airfoils chosen for this study come from the University of Illinois Urbana-Champagne (UIUC) database. This database has the x and y-coordinates for hundreds of airfoils that have been developed. The focus of this study was to use symmetric airfoils, this will lead to a more focused model as the final product which can generalize symmetric airfoils well. Symmetric airfoils have the advantage of the y-coordinates on the upper and lower surfaces being the negative of the other which will come in handy later in the process.

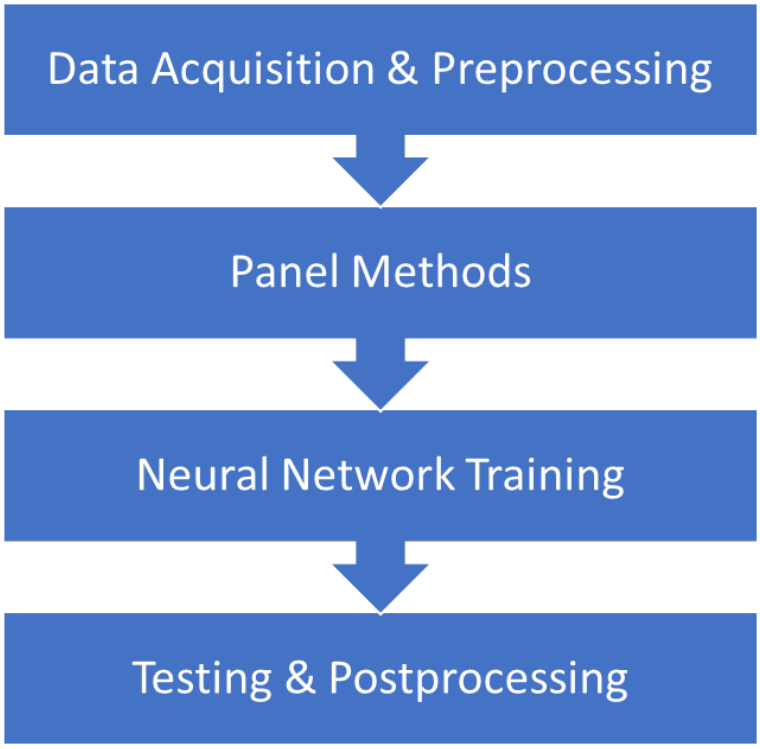
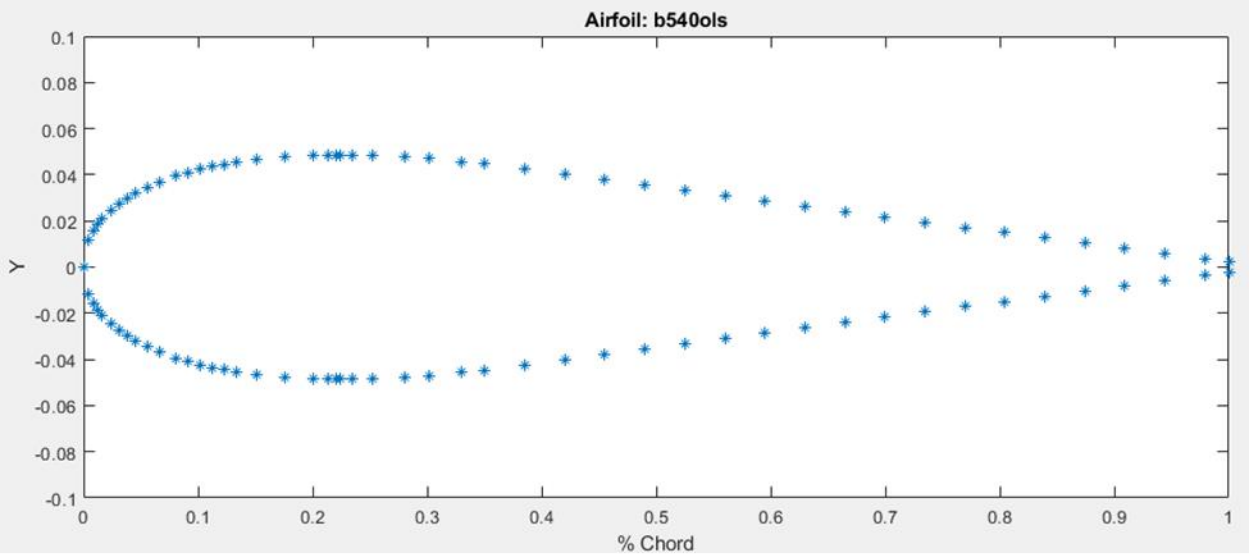


Figure 11: Flowchart for procedure

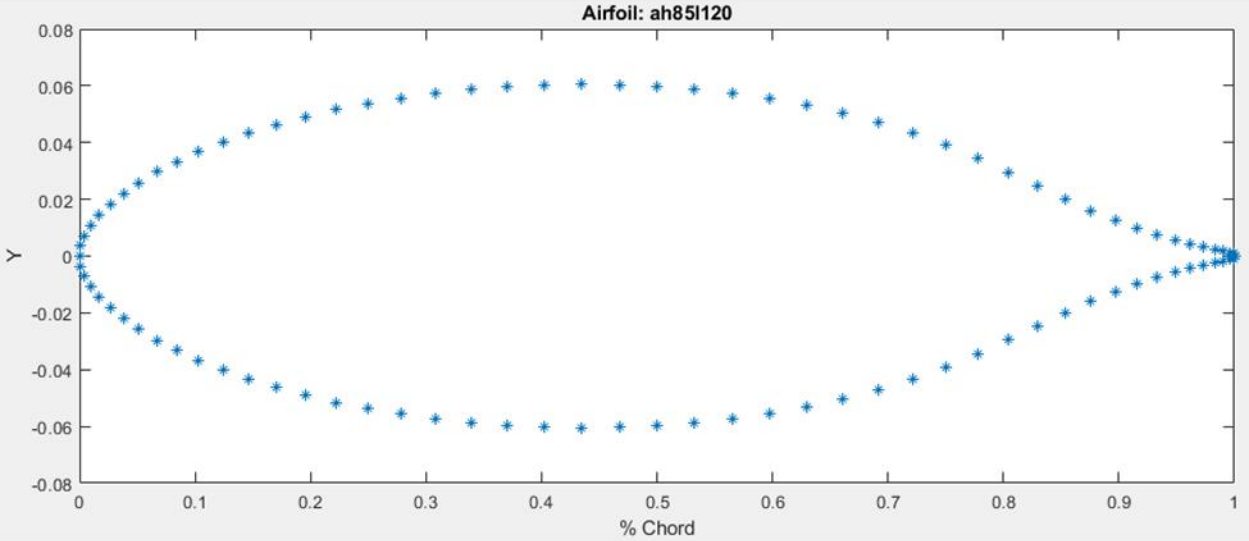
Once all coordinate data for the airfoils has been obtained the coordinates must be normalized since not all the data is formatted the same way and different airfoils have differing numbers of x and y-coordinates. These normalized coordinates will then be fed into a panel method program which finds the coefficient of pressure distribution for each airfoil at varying angles of attack. The coefficient of pressure data as well as the y-coordinates will be fed into a neural network as the input features and output features, respectively. Once the output of the neural network is obtained, the data must be smoothed and postprocessed for reasons that will become apparent later. In the following sections the different steps of the procedure will be explained in detail from the mathematical grounds as well as the implementation in MATLAB.

Data Acquisition

The first step when training any neural network is the data acquisition and the data must be carefully selected to ensure that there are no outliers that may affect the overall neural network model. Since this study is focused on symmetric airfoils, any unsymmetric or cambered airfoils must be thrown out of the set. All airfoils were obtained from the UIUC airfoil database by utilizing airfoiltools.com to categorize the airfoils and select only the symmetric airfoils. One problem that arose from this data was that not all of the x and y-coordinates of the airfoils had the same number of points that the airfoil was evaluated on. This can be seen visually in Figure 12 which shows two different airfoils plotted with their raw data output from the UIUC database. Airfoil B540ols has 97 coordinates while airfoil AH85l120 has 95 coordinates. What is noticeable is that the coordinate spacing is different in both where one emphasizes the leading-edge shape with many coordinates and the other emphasizes cosine spacing at the leading and trailing edges. Without a camber line or thickness curve it is impossible to know exactly where the coordinates of the airfoils will occur without performing some type of regression on the coordinate data. Because the data has to be normalized to be used in neural network, each airfoil's y-coordinates must be evaluated on the same x-coordinates. 101 x-coordinates were chosen with cosine spacing, with 50 points on the upper surface and 50 points on the lower surface as well as one point at the trailing edge where x is equal to 1 and y is equal to 0. This is to satisfy the Kutta condition which will be discussed in a later section.



(a)



(b)

Figure 12: a) Plot of airfoil B540ols b) Plot of airfoil AH85l120

Preprocessing

Data Reformatting

Some studies utilized PARSEC or other types of airfoil parameterization, but in this study a script was developed that iterated through different orders of polynomials from first to fiftieth order polynomials to fit the airfoil curve. This method searched for the best polynomial which

would fit the airfoil geometry the best in terms of mean squared error (MSE). The output was then visually inspected so that the shape matched well. There were some outliers when using this technique which would produce an optimal MSE but would have unnecessary undulations in the polynomial curve which would create bad coefficient of pressure results for the next step in the procedure. The reason why this method was chosen instead of some interpolation method was that because 101 x-coordinates were chosen for the airfoil geometry to be evaluated on, the airfoil geometries with a low number of coordinates were not able to be interpolated on. An interpolation method would be too crude to accurately predict the correct shape of the airfoil.

MATLAB Implementation

First, each symmetric airfoil from the UIUC database had to be downloaded. The symmetric airfoils were determined and the .dat files which contained the x and y-coordinates were downloaded through airfoiltools.com, which utilizes data from the UIUC database,. Since each were formatted differently and had a different number of data points the MATLAB script had to account for this. Some files contained unneeded numbers or letters at the beginning of the coordinate data such as in airfoil AH85l120 which can be seen in Table 1.

Table 1: Sample coordinates of airfoil AH85l120

x	y
AH	85-L-120/17
1	0
0.99893	0.0008
0.99572	0.00112
0.99039	0.00167
0.98296	0.00229
0.97347	0.00314
0.96194	0.00422
0.94844	0.00564

In the first row of data there are letters and numbers that are not needed, thankfully MATLAB does not read this row when using the *readmatrix()* function because it contains letters, but in other cases the first data point will be a number which is not needed. To accommodate for this any number greater than 1 will not be permitted to stay in the matrix so if the relation $x_1 > 1$, is satisfied, that data point is deleted. Another problem that arises is that the first term in the x column is equal to zero. For this study it is required that the x-coordinates start at 1 so that they have equal formatting. If the case $x_1 = 0$, is satisfied, then the upper half of the coordinates will be flipped upside down using the *flipud()* function. The reason why only half of the matrix is flipped is because when this occurs the data in the lower half starts with 0, which is desired. Another formatting requirement is that the coordinates start at the lower side of the airfoil meaning that the y-coordinates must be negative at the beginning of the data. To avoid this, if the relation, $y_1 > 0$, is fulfilled, the entire matrix is flipped, using the *flipud()* function again. After all of this, sometimes the middle two datapoints are the same where $x = 0$ and $y = 0$ which is not needed. To avoid this, if $x_{N/2+1} = 0$ is true, then the data point is eliminated where N is the number of datapoints. The final issue is that there will sometimes be a NaN value located within the matrix which will be catastrophic when sending the data points to be fitted with a polynomial and sent to the panel methods. To avoid this, if any x-coordinate is equal to NaN its row is deleted, which is carried out by the *isnan()* function.

After this reformatting occurs, the data is now ready to be fitted with a polynomial curve. To reach the best curve for the geometry data, a small optimization script was created to optimize the MSE of the regressed polynomial curve. First, the *polyfit()* function is used to fit the coordinate data to a polynomial with varying degrees from 1 to 50, after this, the polynomial is evaluated on the same x-coordinates as the initial data so that the MSE can be found from the

newly evaluated y-coordinates. Once all of the MSE values are found, the lowest value is chosen and its respective degree is used to fit a polynomial on the geometry data. Each newly fit airfoil is plotted on the 101 x-coordinates, and after a visual inspection of each airfoil, the newly created coordinates were fed into a panel method function which would find the coefficient of pressure over the airfoil for varying angles of attack. The beauty of working with symmetrical airfoils is that it is only necessary to create a polynomial fit for one side of the airfoil because the absolute value of the upper and lower y-coordinates are equal to each other. The only step necessary after gaining a polynomial fit is to duplicate the array and make it negative. The two arrays can then be concatenated and the normalized y-coordinates are found. This process can be seen in Figure 13 where the first plot shows the original airfoil data graphed over the polynomial fit evaluated at the original x-coordinates. The next plot shows the original data plotted over the polynomial evaluated at the normalized x-coordinates used for each airfoil. The final plot shows both sides of the airfoil after concatenating the two arrays which are negatives of each other.

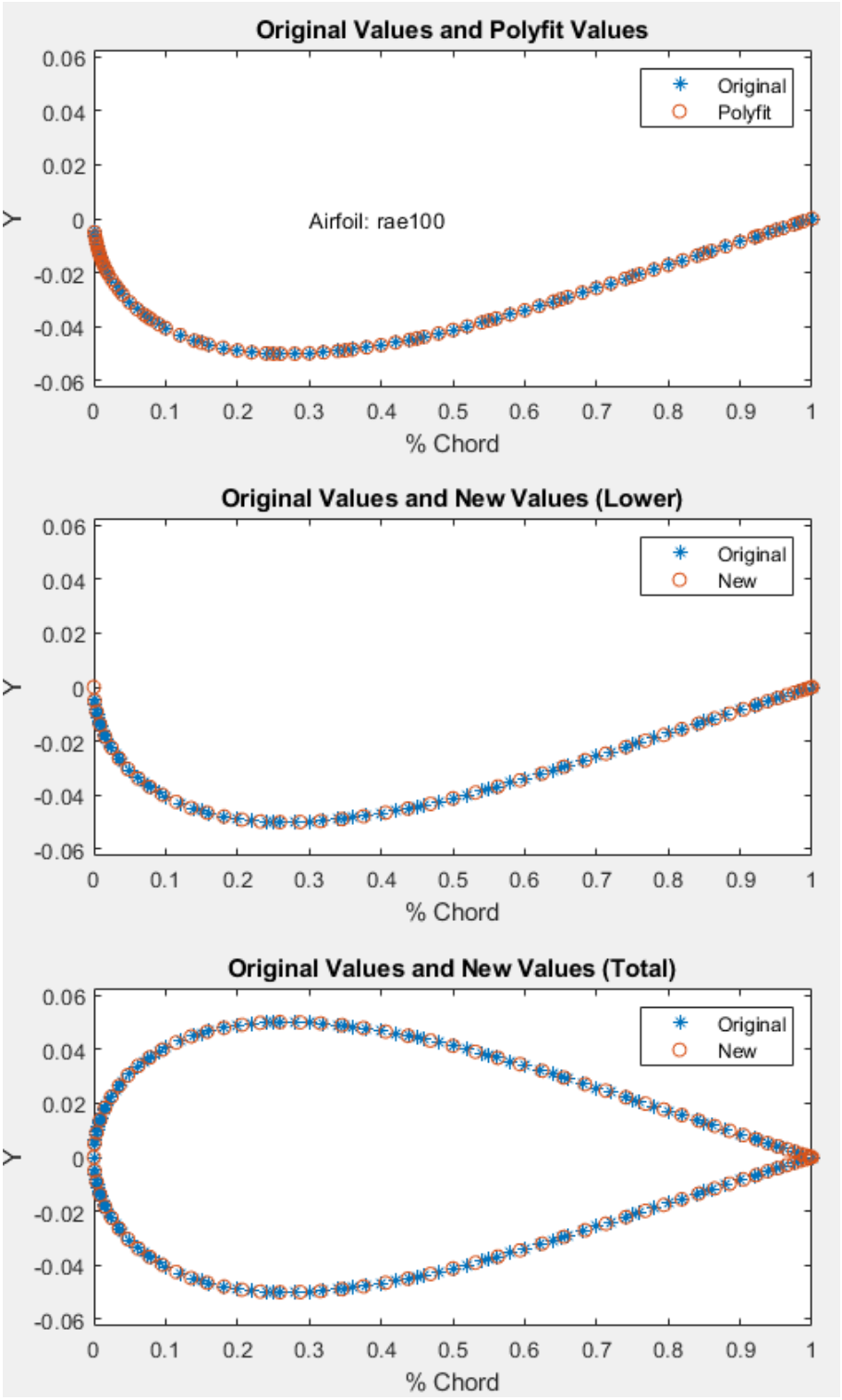


Figure 13: Original and polynomial fit of airfoil RAE100

Panel Methods

Panel methods are a technique that were developed in the 1960s by Hess and Smith to numerically find the aerodynamic characteristics of 2D shapes (Houghton *et al.*, 2013). It achieves this by discretizing the geometry into so-called “panels”, to find the individual tangential velocities at those panels. This velocity gradient can then be used to find coefficient of pressure and furthermore, the coefficient of lift and drag. There are two different type of panel methods; lower-order and higher-order. The panel methods developed by Hess and which are used in this study are the lower-order panel methods which operate in linear, inviscid, irrotational flow conditions. This section will go over the different panel methods that contribute to the selected source-vortex panel method. In this study, it is ultimately used to find the coefficient of pressure of a 2D airfoil.

Source Panel Methods

The source panel method is used for non-lifting flows over a 2D geometry. To model the potential flow around the body, sources can be distributed over the body surface. An airfoil which is modeled as a source sheet is shown in Figure 14 and describes the coordinate system.

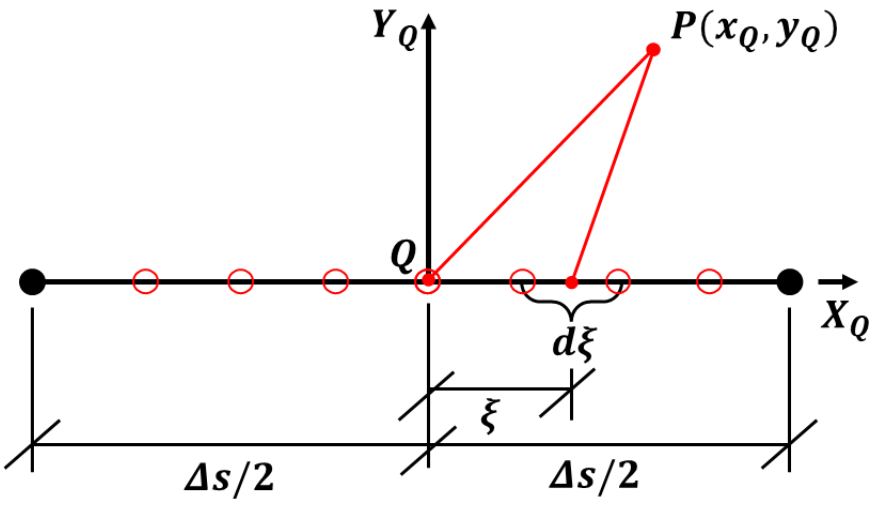


Figure 14: Source panel method coordinate system

When the body is placed in a uniform flow of velocity *U*, the total velocity potential of the flow can be found by superimposing a disturbed flow around the body, and the following relation is found

$$\Phi = Ux + \phi \tag{1}$$

where ϕ is the disturbance potential. Given a coordinate system as shown in Figure 14, where the source strength per unit length is σ_Q and the velocity potential, P, due to sources along the length ds_Q, have the following relation

$$\phi_{PQ} = \sigma_Q ln R_{PQ} ds_Q \tag{2}$$

By integrating this equation over the surface of the body, the velocity potential due to the sources distributed on the surface becomes

$$\Phi_P = Ux + \oint \sigma_Q ln R_{PQ} ds_Q \tag{3}$$

Figure 15 shows how the airfoil is discretized into individual panels where a) shows the numbering convention for the panels and b) shows the tangential and normal unit vectors at the i^{th} and j^{th} panel.

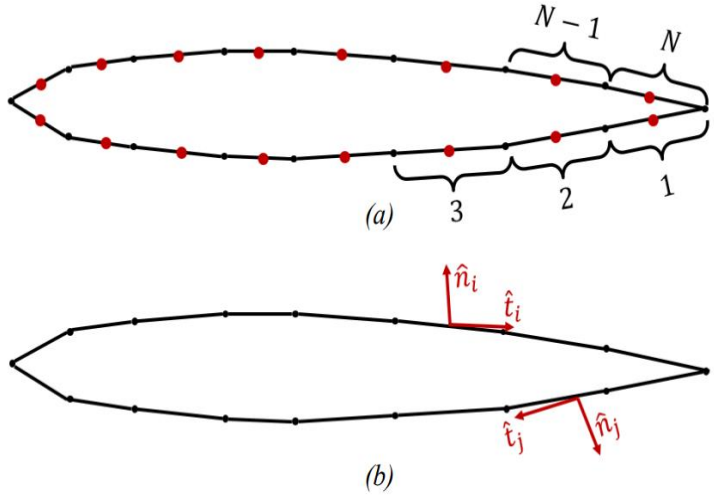


Figure 15: (a) A discretized airfoil surface into panels (b) Tangent and normal unit vectors at point i and j

At the midpoint of each panel there are points called collocation points which is where the distributed sources induce a velocity. The sources distributed over panel *j* induce a velocity, $\mathbf{v}_{ij}$, at the collocation point of panel *i*. The normal and tangential velocity components can be written as

$$\mathbf{v}_{ij} \cdot \hat{n}_i \quad and \quad \mathbf{v}_{ij} \cdot \hat{t}_i \tag{4}$$

respectively, where $\hat{n}_i$ is the unit normal vector and $\hat{t}_i$ is the unit tangential vector. These equations are supposed to be proportional to the source strengths on panel *j* so the following relation between the velocity components, the source strengths, and the normal and tangential velocities is

$$\mathbf{v}_{ij} \cdot \hat{n}_i = \sigma_j N_{ij} \quad and \quad \mathbf{v}_{ij} \cdot \hat{t}_i = \sigma_j T_{ij} \tag{5}$$

where N_{ij} and T_{ij} are the normal and tangential influence coefficients, respectively. These two values are used to represent the normal and tangential velocities induced at the collocation point of panel *i* due to the panel *j*. To find these coefficients, the dot products of the velocity and the unit vectors can be used as shown

$$N_{ij} = \mathbf{v}_{PQ} \cdot \hat{n}_i \quad and \quad T_{ij} = \mathbf{v}_{PQ} \cdot \hat{t}_i \tag{6}$$

where $\mathbf{v}_{PQ}$ is the velocity induced at point *P* by sources of unit strength distributed over the panel with a collocation point *Q*. The velocity can be broken down into components such as the following

$$\mathbf{v}_{PQ} = v_{x_Q}\hat{t}_j + v_{y_Q}\hat{n}_j \tag{7}$$

The disturbance potential can be used to obtain the velocity components at point *P* and the disturbance potential over the entire panel can be described as

$$\phi_{PQ} = \int_{-\Delta s/2}^{\Delta s/2} ln\sqrt{\left(x_Q - \xi\right)^2 + y_Q^2}\, d\xi \tag{8}$$

Now to find the velocity components, the derivative of the disturbance potential with respect to the axes of the coordinate system will be

$$v_{x_Q} = \frac{\delta \phi_{PQ}}{\delta x_Q} = \int_{-\Delta s/2}^{\Delta s/2} \frac{x_Q - \xi}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi = -\frac{1}{2} ln \left[\frac{\left(x_Q + \Delta s/2\right)^2 + y_Q^2}{\left(x_Q - \Delta s/2\right)^2 + y_Q^2}\right] \tag{9}$$

$$v_{y_Q} = \frac{\delta \phi_{PQ}}{\delta y_Q} = \int_{-\Delta s/2}^{\Delta s/2} \frac{y_Q}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi = -\left[\tan^{-1}\left(\frac{x_Q + \frac{\Delta s}{2}}{y_Q}\right) - \tan^{-1}\left(\frac{x_Q - \frac{\Delta s}{2}}{y_Q}\right)\right] \tag{10}$$

The influence coefficients can then be represented using these velocity components as shown

$$\mathrm{N}_{ij} = \mathbf{v}_{PQ} \cdot \hat{n}_i = v_{x_Q} \hat{n}_i \cdot \hat{t}_j + v_{y_Q} \hat{n}_i \cdot \hat{n}_j \tag{11}$$

$$\mathrm{T}_{ij} = \mathbf{v}_{PQ} \cdot \hat{t}_i = v_{x_Q} \hat{t}_i \cdot \hat{t}_j + v_{y_Q} \hat{t}_i \cdot \hat{n}_j \tag{12}$$

To obtain the normal and tangential velocities at collocation point *i*, the respective induced velocities due to each panel and the freestream velocity at collocation point *i* must be summed. The sum of the total tangential velocity can also be found likewise. This relationship would be the following

$$v_{n_i} = \sum_{j=1}^{N} \sigma_j N_{ij} + \vec{U} \cdot \hat{n}_i \tag{13}$$

$$v_{s_i} = \sum_{j=1}^{N} \sigma_j T_{ij} + \vec{U} \cdot \hat{t}_i \tag{14}$$

One boundary condition that is needed to solve the influence coefficients is that the sum of the velocities normal to each panel must be equal to zero, or $v_{n_i} = 0$, therefore the normal velocity in Equation 13 can be written as

$$\sum_{j=1}^{N} \sigma_j N_{ij} = -U \cdot \hat{n}_i \ (i = 1,2, \dots, N) \tag{15}$$

Equation 15 is now a system of linear algebraic equation with N unknowns where the strengths of the sources can be solved for by using the normal influence coefficients and the freestream velocity multiplied by the normal unit vector at each panel

$$\boldsymbol{N}\sigma = \boldsymbol{b} \tag{16}$$

where $\boldsymbol{N}$ is the normal influence coefficient matrix, σ is the vector of source strengths, and $\boldsymbol{b}$ are the right hand side of the system which is the negative of the freestream velocity multiplied by the vector of unit normal vectors. Once the source strengths are found and the tangential influence coefficients, T_{ij}, are calculated, they can be plugged back into Equation 14 for the tangential velocities at each panel. Using Bernoulli's equation, the coefficient of pressure distribution can be calculated over the surface of the body as shown

$$C_{p_i} = 1 - \left(\frac{v_{s_i}}{U}\right)^2 \tag{17}$$

Source - Vortex Panel Methods

Vortex panel methods allow lifting flows to be studied by using vortices that are distributed along each panel. This allows circulation to occur and be calculated which gives rise to the creation of lift. Much like the source method where sources are distributed over each panel with a unique strength, the vortex method has vortices distributed which have a uniform strength per unit length which is unique for each panel.

For this method, there must be different boundary conditions to create a unique solution. One condition that can be used from the source method is flow tangency which requires that the normal velocity of the flow at each panel must equal zero. A new boundary condition that can be introduced is the Kutta condition which requires that the velocity at the upper and lower panel

that touch the trailing edge must approach the same value. Now that this condition has been introduced there are now N+1 knowns which are the N number of normal velocities and the Kutta condition, but there are only N number of unknown vortex strengths. A way to deal with this is to distribute sources and vortices and allow the source strengths to vary from panel to panel while allowing there to be only one value for vortex strength, thus creating N+1 unknowns. Our system of linear algebraic equations can now be solved just like in the source panel method.

Take the following coordinate system in Figure 16 for reference where an airfoil is modeled as a vortex sheet.

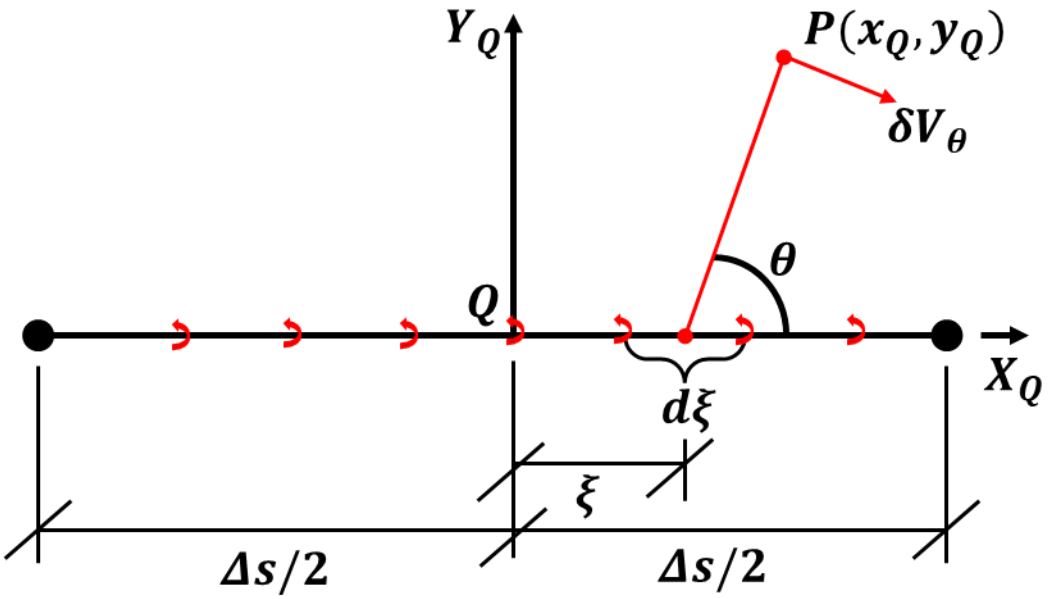


Figure 16: Vortex panel method coordinate system

Recall from the source panel method that $N_{ij}$and T_{ij} are $N\ x\ N$ matrices. An $N\ x\ N+1$ matrix will be required when solving the system of linear algebraic equations, so the following relationship is created

$$N_{i,N+1} = \sum_{j=1}^{N} N'_{ij} \tag{18}$$

$$T_{i,N+1} = \sum_{j=1}^{N} T'_{ij} \tag{19}$$

where N'_{ij} and T'_{ij} are the influence coefficients for the vortex panel method. These are found similar to the influence coefficients from the source panel method and by using the coordinate system in Figure 17 the vortex influence coefficients can be described as

$$N'_{ij} = \mathbf{V}_{PQ} \cdot \hat{n}_i \text{ and } T'_{ij} = \mathbf{V}_{PQ} \cdot \hat{t}_i \tag{20}$$

where $\mathbf{V}_{PQ}$ is the velocity induced at point *P* by sources of unit strength at collocation point *Q* of a panel. This velocity can be described by its components, as in the source methods, as follows

$$\boldsymbol{V}_{PQ} = V_{x_Q}\hat{t}_j + V_{y_Q}\hat{n}_j \tag{21}$$

The vorticity distribution on the length $d\xi$ which is pictured in Figure 17 can be used to find the velocity due to these vortices such as

$$\delta V_\theta = \frac{\gamma}{R} d\xi \tag{22}$$

As can be seen in Figure 16, vorticity is plotted in polar coordinates so it is necessary to translate these polar coordinates to cartesian coordinates by using trigonometric functions. This is also the reason for the R variable in Equation 22 which is the distance of *P* to *Q* and can be calculated with the following equation

$$R = \sqrt{\left(x_Q - \xi\right)^2 + y_Q^2} \tag{23}$$

Now to translate the coordinates into the x_Q and y_Q directions the following is used

$$\delta V_{x_Q} = \delta V_\theta \sin\theta = \frac{\gamma y_Q}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi \tag{24}$$

$$\delta V_{y_Q} = -\delta V_\theta \cos\theta = \frac{\gamma (x_Q - \xi)}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi \tag{25}$$

The velocity components can then be integrated along the panels with respect to S which is the panel length

$$V_{y_Q} = \gamma \int_{\Delta s/2}^{\Delta s/2} \frac{x_Q - \xi}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi = \frac{\gamma}{2} ln \left[\frac{\left(x_Q + \Delta s/2\right)^2 + y_Q^2}{\left(x_Q - \Delta s/2\right)^2 + y_Q^2}\right] \tag{26}$$

$$V_{x_Q} = \gamma \int_{\Delta s/2}^{\Delta s/2} \frac{y_Q}{\left(x_Q - \xi\right)^2 + y_Q^2} d\xi = -\gamma \left[\tan^{-1}\left(\frac{x_Q + \frac{\Delta s}{2}}{y_Q}\right) - \tan^{-1}\left(\frac{x_Q - \frac{\Delta s}{2}}{y_Q}\right)\right] \tag{27}$$

the vortex strength can be set to 1, γ=1, so that the influence coefficients can be written as

$$\mathrm{N'}_{ij} = \mathbf{V}_{PQ} \cdot \hat{n}_i = V_{x_Q} \hat{n}_i \cdot \hat{t}_j + V_{y_Q} \hat{n}_i \cdot \hat{n}_j \tag{28}$$

$$\mathrm{T'}_{ij} = \mathbf{V}_{PQ} \cdot \hat{t}_i = V_{x_Q} \hat{t}_i \cdot \hat{t}_j + V_{y_Q} \hat{t}_i \cdot \hat{n}_j \tag{29}$$

The relation between the velocities in the vortex panel method and the source panel method is as follows

$$V_{x_Q} = v_{y_Q} \text{ and } V_{y_Q} = -v_{x_Q} \tag{30}$$

The vortex influence coefficients can then be written in terms of the source influence coefficients thusly

$$\mathrm{N'}_{ij} = v_{y_Q} \hat{n}_i \cdot \hat{t}_j - v_{x_Q} \hat{n}_i \cdot \hat{n}_j \tag{31}$$

$$\mathrm{T'}_{ij} = v_{y_Q} \hat{t}_i \cdot \hat{t}_j - v_{x_Q} \hat{t}_i \cdot \hat{n}_j \tag{32}$$

previously in Equation 18 and 19, the influence coefficients of the N+1 column were given and they now can be written as

$$N_{i,N+1} = \sum_{j=1}^{N} (v_{y_Q} \hat{n}_i \cdot \hat{t}_j - v_{x_Q} \hat{n}_i \cdot \hat{n}_j) \tag{33}$$

$$T_{i,N+1} = \sum_{j=1}^{N} (v_{y_Q} \hat{t}_i \cdot \hat{t}_j - v_{x_Q} \hat{t}_i \cdot \hat{n}_j) \tag{34}$$

The normal and tangential velocity components will then respectively become

$$V_{n_i} = \sum_{j=1}^{N} \sigma_j N_{ij} + \gamma N_{i,N+1} + \vec{U} \cdot \hat{n}_i \tag{35}$$

$$V_{S_i} = \sum_{j=1}^{N} \sigma_j T_{ij} + \gamma T_{i,N+1} + \vec{U} \cdot \hat{t}_i \tag{36}$$

Remember from source panel method that the total normal velocity to the surface at each collocation point *i* must equal zero, and the normal velocity will become

$$\sum_{j=1}^{N} \sigma_j N_{ij} + \gamma N_{i,N+1} = -\vec{U} \cdot \hat{n}_i \tag{37}$$

To satisfy the Kutta condition, the direction of the tangent unit vectors must be taken into account, which is seen in Figure 17, and will be in opposite directions at the trailing edge panels.

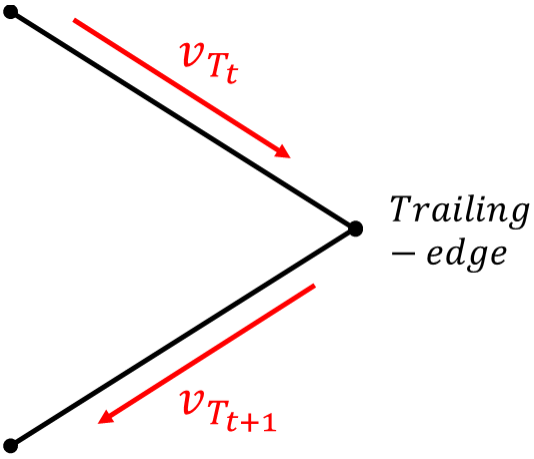


Figure 17: Kutta condition

The tangential velocities at the trailing edge must be equal to each other so the following relation is defined

$$\sum_{j=1}^{N} \sigma_j T_{t,j} + \gamma T_{t,N+1} + \vec{U} \cdot \hat{t}_t = -\left(\sum_{j=1}^{N} \sigma_j T_{t+1,j} + \gamma T_{t+1,N+1} + \vec{U} \cdot \hat{t}_{t+1} \right) \tag{38}$$

It can be simplified to

$$\sum_{j=1}^{N} \sigma_j (T_{t,j} + T_{t+1,j}) + \gamma (T_{t,N+1} + T_{t+1,N+1}) = -\vec{U} \cdot (\hat{t}_t + \hat{t}_{t+1}) \tag{39}$$

As seen before in the source panel methods this is a system of linear algebraic equations and can be written as follows

$$\boldsymbol{M}a = \boldsymbol{b} \tag{40}$$

where $\boldsymbol{M}$ is the normal influence coefficient matrix, a is the vector of source strengths and single vortex strength, and $\boldsymbol{b}$ is the vector of the freestream velocity multiplied by the normal and tangential unit vectors.

After calculating T_{ij} and solving for a to find the source and vortex strengths, the tangential velocity can be found with Equation 36. The coefficient of pressure can be written as

$$C_{p_i} = 1 - \left(\frac{V_{S_i}}{U}\right)^2 \tag{41}$$

MATLAB Implementation

Most studies utilize XFOIL to generate the required data, but there was an in-house code created at the UTRGV Aerodynamic Propulsion Laboratory (APL) which utilizes source-vortex panel method to find the coefficient of pressure distribution over an airfoil with linear, inviscid, irrotational flow.

The MATLAB implementation of these panel methods included creating for loops to iteratively calculate the normal and tangential influence coefficient matrices. The matrices can be calculated by using Equations 11, 12, 33 and 34 which define the original coefficients from the source method as well as the final column which was found in the source – vortex method.

$$\mathrm{N}_{ij} = \mathbf{v}_{PQ} \cdot \hat{n}_i = v_{x_Q} \hat{n}_i \cdot \hat{t}_j + v_{y_Q} \hat{n}_i \cdot \hat{n}_j \tag{11}$$

$$\mathrm{T}_{ij} = \mathbf{v}_{PQ} \cdot \hat{t}_i = v_{x_Q} \hat{t}_i \cdot \hat{t}_j + v_{y_Q} \hat{t}_i \cdot \hat{n}_j \tag{12}$$

$$N_{i,N+1} = \sum_{j=1}^{N} (v_{y_Q} \hat{n}_i \cdot \hat{t}_j - v_{x_Q} \hat{n}_i \cdot \hat{n}_j) \tag{33}$$

$$T_{i,N+1} = \sum_{j=1}^{N} (v_{y_Q} \hat{t}_i \cdot \hat{t}_j - v_{x_Q} \hat{t}_i \cdot \hat{n}_j) \tag{34}$$

The total number of iterations that are needed for *i* and *j* is equal to the number of panels which is equal to N. The only case where this calculation must be intervened is when $i = j$ which would be the normal and tangential velocity at a panel due to source and vortex at that panel. In that case, the sources would be

$$v_{x_Q} = 0 \text{ and } v_{x_Q} = \pi$$

The influence coefficients at $i = j$ would then be

$$\mathrm{N}_{ij} = v_{x_Q} \hat{n}_i \cdot \hat{t}_j + v_{y_Q} \hat{n}_i \cdot \hat{n}_j = \pi$$

$$\mathrm{T}_{ij} = v_{x_Q} \hat{t}_i \cdot \hat{t}_j + v_{y_Q} \hat{t}_i \cdot \hat{n}_j = 0$$

For the N+1 column they would become

$$\mathrm{N'}_{ij} = v_{y_Q} \hat{n}_i \cdot \hat{t}_j - v_{x_Q} \hat{n}_i \cdot \hat{n}_j = 0$$

$$\mathrm{T'}_{ij} = v_{y_Q} \hat{t}_i \cdot \hat{t}_j - v_{x_Q} \hat{t}_i \cdot \hat{n}_j = -\pi$$

Finally, at the N+1 row the influence coefficients would become

$$N_{N+1,j} = T_{t,j} T_{t+1,j} \tag{42}$$

$$T_{N+1,j} = 0 \tag{43}$$

Remember that the problem is a system of linear algebraic equations which take the form of

$$\boldsymbol{M}a = \boldsymbol{b} \tag{40}$$

The left hand side of the equations would be

$$\boldsymbol{M} = \begin{bmatrix} N_{i,j} & N_{i,j+1} & \cdots & N_{i,N} & N_{i,N+1} \\ N_{i+1,j} & N_{i+1,j+1} & & N_{i+1,N} & N_{i+1,N+1} \\ & \vdots & \vdots & & \vdots \\ N_{N,j} & N_{N,j+1} & & N_{N,N} & N_{N,N+1} \\ N_{N+1,j} & N_{N+1,j+1} & \cdots & N_{N+1,N} & N_{N+1,N+1} \end{bmatrix} \tag{44}$$

The right hand side would take the form

$$\boldsymbol{b} = \begin{pmatrix} -\vec{U} \cdot \hat{n}_i \\ -\vec{U} \cdot \hat{n}_{i+1} \\ \vdots \\ -\vec{U} \cdot \hat{n}_N \\ -\vec{U} \cdot (\hat{t}_t + \hat{t}_{t+1}) \end{pmatrix} \tag{45}$$

Solving this in MATLAB requires the use of the operator "\" which will find the strengths of the sources and vortices as

$$\boldsymbol{a} = \begin{pmatrix} \sigma_i \\ \sigma_{i+1} \\ \vdots \\ \sigma_N \\ \gamma \end{pmatrix} \tag{46}$$

These strengths can then be input into Equation 36

$$V_{S_i} = \sum_{j=1}^{N} \sigma_j T_{ij} + \gamma T_{i,N+1} + \vec{U} \cdot \hat{t}_i \tag{36}$$

Finally, the coefficient of pressure distribution can be found and saved

$$C_{p_i} = 1 - \left(\frac{V_{S_i}}{U}\right)^2 \tag{37}$$

For this case the coefficient of pressure is needed over multiple angles of attack which range from -6° to +30° in two-degree increments. Each angle of attack will then be iterated upon and saved to a Microsoft Excel file. The coefficient of pressure data for airfoil RAE100 for one angle as well as multiple angles is shown in Figures 18 and 19, respectively. This data is then used to train the neural network which is detailed in the next section.

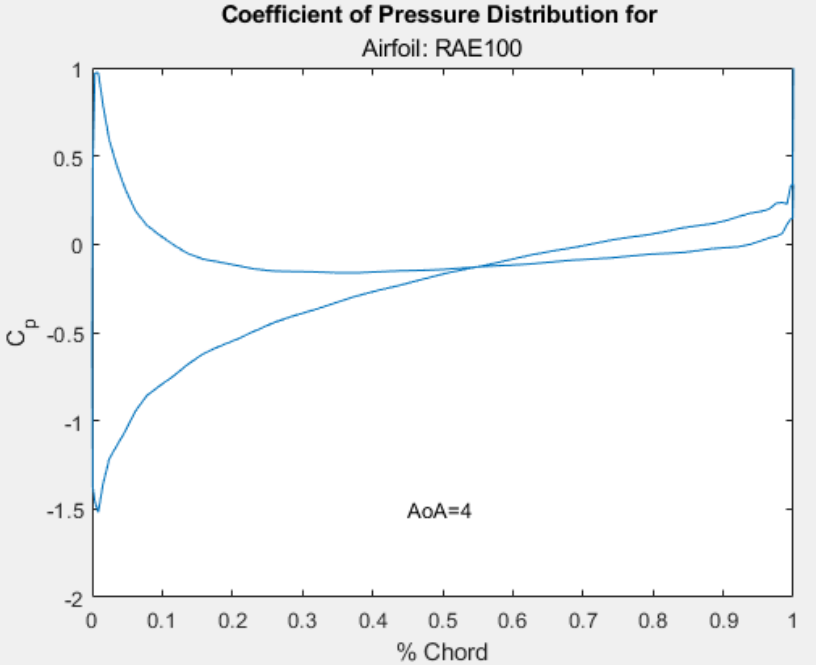


Figure 18: Coefficient of pressure distribution at AoA=4

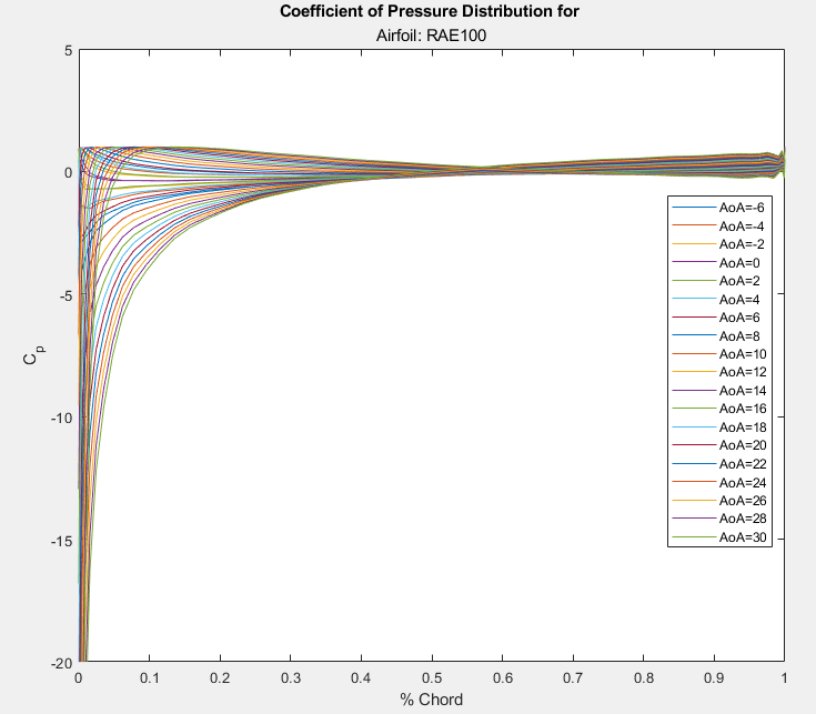


Figure 19:Coefficient of pressure distribution for various angles of attack

Neural Network Procurement

The neural network is the heart of this study which will perform the heavy-lifting in correlating the airfoil shape to the coefficient of pressure distribution. Neural networks were designed to recreate the connections that neurons have in the human brain. It does this by creating layers of neurons. One layer of neurons will communicate to the next layer but will not communicate between neurons in the same layer, this can be seen back in Figure 4.

Depending on what the input is to an individual neuron, the input will elicit a response from an activation function within the neuron which decides whether that input will continue to the next layer or not. This relationship between neurons of one layer to the next acts just as the brain does where there are individual neurons which fire data through synapses to the next neuron (Bishop, 1995). Depending on what neuron fires and what the firing represents, the neural network will have a unique output. The neural network, mainly the weights and biases, are modified after inputting the data and acquiring the output. The weights and biases are modified so that the output of the network will more closely match the desired output. After multiple iterations of this which are called epochs, the neural network will reach its objective which may be to minimize the mean squared error or some other parameter and the neural network can be saved to be used on new data that the user would like to know the response to.

Neural networks were borne out of the need for intelligent systems all the way back in the 1940s. It wasn't until the 1990s that there was a resurgence in interest in the topic and new methods were developed again (Picton, 2000). Many neural networks have origins in optimization functions which have been modified to fit within the neural network architecture one such approach is the Gauss – Newton method which gave rise to Levenberg – Marquardt

algorithm. This algorithm could then be generalized by using Bayesian regularization. These algorithms will be discussed in the following sections.

Gauss – Newton Method

The Gauss – Newton Method is an optimization algorithm which tries to solve a nonlinear least squares problem. To begin there is a response variable y which is governed by some predictor variables or covariates which are x and β

$$y = f(x;\beta) + \epsilon, f(x) - not\ linear \tag{47}$$

where ϵ is noise added to the equation. To optimize this function, the β values must be iterated through and to do this correctly there must be a residual function which finds the difference between the desired value and the predicted value

$$r_i = y_i - f(x_i;\beta) \tag{48}$$

these residual values are then used in a loss function which is similar to mean squared error which sum all of the residuals together for a value that is called loss

$$L = \sum_i r_i^2 \tag{49}$$

The objective of the process is to minimize L so that the predictive model fits the desired data well. Because the function is nonlinear there is no easy way to find the values for β. The Newton method or Newton – Raphson method must be applied which is an iterative method. The method allows the roots of the function to be found.

Let a function $f(x_t)$ be the target function which depends on a variable x at time step t. To improve the values at time step $t + 1$, and find the roots of the function the following can be used

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)} \tag{50}$$

Further, to find the minimum and maximum of the function the next derivatives of the functions can be used

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)} \tag{51}$$

This can be simplified to the following

$$x_{t+1} = x_t - (\nabla_x^2 f)^{-1} \nabla_x f \tag{52}$$

This term $\nabla_x^2 f$ is the second derivative of the function, otherwise known as the Hessian matrix. The Hessian matrix can be difficult, computationally intensive, and time consuming to calculate which gives rise to finding an approximation for it so that computational time can be minimized.

The first derivative or gradient of the loss function can be written and simplified as

$$\nabla_{\beta_j} L = \sum_i 2\, r_i \frac{\partial r_i}{\partial \beta_j} = -2 \sum_i r_i \frac{\partial f_i}{\partial \beta_j} = -2 \cdot J^T \cdot r \tag{53}$$

where J is the Jacobian matrix and T denotes the transpose of the matrix. Remember that the second derivative must be found, so the second derivative of the function can be written as

$$\nabla^2_{\beta_j \beta_k} L = -2 \sum_i \left(\frac{\partial f_i}{\partial \beta_k} \cdot \frac{\partial f_i}{\partial \beta_j} + r_i \frac{\partial^2 f_i}{\partial \beta_j \partial \beta_k} \right) \approx 2 \sum_i \frac{\partial f_i}{\partial \beta_k} \cdot \frac{\partial f_i}{\partial \beta_j} \tag{54}$$

which can be simplified as

$$\nabla^2_{\beta_j \beta_k} L \approx 2 \sum_i \frac{\partial f_i}{\partial \beta_k} \cdot \frac{\partial f_i}{\partial \beta_j} = 2 \cdot J^T \cdot J \tag{55}$$

The approximation does not include the second term of the derivative where the residuals are multiplied by the second derivative of the function. The Jacobian matrix is much easier to compute than the Hessian, thus minimizing computational time.

Going back to Equation 47 which iterates to find the best covariables, it can be rewritten with the new approximation for the second derivative of the loss function as

$$\beta_{t+1} = \beta_t - \left(\nabla_\beta^2 L\right)^{-1} \nabla_\beta L \approx \beta_t - (2 \cdot J_t^T \cdot J_t)^{-1}(-2 \cdot J_t^T \cdot r_t) \tag{56}$$

which simplifies into

$$\beta_{t+1} = \beta_t + (J_t^T \cdot J_t)^{-1} J_t^T \cdot r_t \tag{57}$$

This is the basis of the Gauss – Newton method which allows for the Hessian Matrix to be approximated and iterates through values to find the optimal β values for the function.

Levenberg-Marquardt

The Levenberg – Marquardt algorithm extends the findings from the Gauss Newton method and creates a new term which allows for the method to have variable step size when going through iterations. Gradient descent is another method which tries to find the minimum of the function by subtracting by the gradient which makes the value descend each iteration, hence the name. In the Gauss – Newton method there is no determined step size so the minimum will be found after a random search which may be ascending or descending. A new error function is created and used so that the new method looks like the following

$$\beta_{t+1} = \beta_t + (J_t^T \cdot J_t + \mu I)^{-1} J_t^T \cdot r_t \tag{58}$$

where μ determines the step size and I is the identity matrix. When the μ value goes towards infinity, it is gradient descent, when it goes towards zero it is Newton's method. Newton's method is faster and more accurate near an error minimum, so it is favorable to decrease this value after every iteration. Therefore, it is decreased after every iteration which results in a reduction in the performance function every step and will generate a better model.

Bayesian Regularization

Bayesian regularization allows for the predictive model to generalize better (Forsee, 1997). It does this by changing the performance function. Whereas the performance function would be equal to loss or L, the function is now defined as

$$F = \beta L_D + \alpha L_W \tag{59}$$

where F is the performance function, α and β are performance function parameters, L_D is the loss or sum of squared errors and L_W is the sum of squares of the network weights. Since the performance function now includes the network weights it now will normalize all of the weights. This results in a network that generalizes much better than a barebones Levenberg – Marquardt algorithm. The function parameters α and β are arbitrary but if $\alpha \ll \beta$ then the algorithm will drive the errors lower which is closer to the original Levenberg – Marquardt algorithm, whereas if $\alpha \gg \beta$ the network will emphasize weight size reduction which will result in a smoother network response.

Now the obstacle to creating an optimal network will be in setting the correct values for the performance function parameters. If the network weights are considered random variables, the density function for the weights can be restructured according to Bayes' rule

$$P(\mathbf{w}|D, \alpha, \beta, M) = \frac{P(D|\mathbf{w}, \beta, M)\, P(\mathbf{w}|\alpha, M)}{P(D|\alpha, \beta, M)} \tag{60}$$

Where D is the dataset, M is the neural network, **w** is the vector of network weights and is organized as follows

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}} \tag{61}$$

the posterior probability is the probability that **w** will occur given D is true, the likelihood is the likelihood of D given a fixed **w**, the prior density is the previous knowledge of the weights

before any data is collected. Finally, the evidence is the normalization factor which guarantees the total probability is 1.

If the noise in the training set is Gaussian and the prior distribution for the weights is Gaussian, the likelihood and prior probabilities can be rewritten as

$$P(D|\mathbf{W},\beta,M) = \frac{1}{Z_D(\beta)}\exp\left(-\beta E_D\right) \tag{62}$$

$$P(\mathbf{w}|\alpha,M) = \frac{1}{Z_W(\alpha)}\exp\left(-\alpha E_W\right) \tag{63}$$

where $Z_D(\beta) = \left(\frac{\pi}{\beta}\right)^{\frac{n}{2}}$ and $Z_W(\alpha) = \left(\frac{\pi}{\alpha}\right)^{\frac{N}{2}}$ and n is the number of datapoints, and N is the number of network parameters. Finally, the posterior becomes

$$P(\mathbf{w}|D,\alpha,\beta,M) = \frac{1}{Z_F(\alpha,\beta)}\exp\left(-F(\mathbf{w})\right) \tag{64}$$

The goal of this Bayesian framework is to maximize the posterior probability which is equivalent to minimizing the performance function in Equation 59. All of this is to find the density functions for the weights. Now Bayes' rule can be applied to optimize the performance function parameters α and β. The posterior becomes

$$P(\alpha,\beta|D,M) = \frac{P(D|\alpha,\beta,M)P(\alpha,\beta|M)}{P(D|M)} \tag{65}$$

since the likelihood function is the same as the normalization factor in the previous equations and is shown in Equation 62 in its Gaussian form, this allows for it to be substituted into the current relation as

$$P(D|\alpha,\beta,M) = \frac{P(D|\boldsymbol{w},\beta,M)\ P(\boldsymbol{w}|\alpha,M)}{P(\boldsymbol{w}|D,\alpha,\beta,M)} \tag{66}$$

Given Equations 62, 63, and 64, this can be rewritten as

$$P(D|\alpha,\beta,M) = \frac{Z_F(\alpha,\beta)}{Z_D(\beta)Z_W(\alpha)} \tag{67}$$

The only term that is not known is $Z_F(\alpha,\beta)$, which can be estimated by Taylor series expansion. The performance function $F(\mathbf{w})$ has a quadratic shape and can expanded around the minimum point of the posterior density $\mathbf{w}^{\mathrm{MP}}$, where the gradient is zero. Solving for the normalizing constant creates

$$Z_F \approx (2\pi)^{\frac{N}{2}}(\det((\mathbf{H}^{\mathrm{MP}})^{-1}))^{\frac{1}{2}} \exp\left(-F(\mathbf{w}^{\mathrm{MP}})\right) \tag{68}$$

Where $\mathbf{H}$ is the Hessian matrix and is equal to $\beta\nabla^2 E_D + \alpha\nabla^2 E_W$. To solve for optimal α and β at the minimum point, the derivative of Equation 62 and 63 is set to zero, which creates the following relations

$$\alpha^{\mathrm{MP}} = \frac{\gamma}{2E_W(\mathbf{w}^{\mathrm{MP}})} \; and \; \beta^{\mathrm{MP}} = \frac{n-\gamma}{2E_D(\mathbf{w}^{\mathrm{MP}})} \tag{69}$$

Where γ, the effective number of parameters, is as follows

$$\gamma = N - 2\alpha^{\mathrm{MP}}\mathrm{tr}(\mathbf{H}^{\mathrm{MP}})^{-1} \tag{70}$$

This number can range from 0 to N and is a measure of how many parameters in the neural network are effectively used in reducing the error function.

At the beginning of training the performance function parameters are set to $\alpha = 0$ and $\beta = 1$. After the first training step the performance function parameters will be updated from their original values. Once one step of the Levenberg – Marquardt algorithm is used to minimize the Bayesian performance function, Equation 59, the effective number of parameters in Equation 70 is found. This uses the Gauss – Newton approximation of the Hessian matrix, which utilizes the Jacobian matrix seen in Equation 57 and becomes

$$\mathbf{H} = \nabla^2 F(\mathbf{w}) \approx 2\beta \boldsymbol{J}^T \boldsymbol{J} + 2\alpha \boldsymbol{I}_N \tag{71}$$

The new estimates for the performance function parameters are then found by using Equation 69 and the process is iterated until the values converge.

The neural network used for this study was a two-layer feedforward network with sigmoid hidden neurons and a single linear output neuron with Levenberg – Marquardt backpropagation and in one model, Bayesian regularization was utilized for the performance function. For 19 feature input networks, there were 15 neurons in the hidden layer and for networks with 38 input features, there were 20 neurons in the hidden layer. This was found heuristically for which number of neurons gave the best MSE as well as regression correlation or R-squared values.

MATLAB Implementation

MATLAB has some tools which help in designing a neural network. There are even built-in functions which allow for neural network training which include *trainlm()* and *trainbr()* which allow the user to define a neural network with Levenberg – Marquardt and Bayesian Regularization backpropagation, respectively. Once all of the data was acquired and saved into Excel files, a script to read all of these files and concatenate all of the arrays was created which consolidated all of the data. There were 101 points of coefficient of pressure data and 101 points of y-coordinates at 19 different angles of attack, which makes for a large dataset which may be ineffective when entering into a neural network architecture since there are so many input features and output features. This approach set out to find a direct correlation between the coefficient of pressure at a specific x-coordinate to the y-coordinate at that point. To train the networks that means cutting all the coefficient of pressure data into the first point, second point and so on, this also includes the y-coordinates. There were multiple models created, the first of which sought to predict the upper geometry from its respective coefficient of pressure

contribution. The second model took the lower geometry and its contribution in the same way as the first model. The third and fourth model were used to predict the upper and lower geometry from the upper and lower contributions together. The way the data was formatted for use in the neural networks is shown in Figures 20 and 21. Figure 20 shows how the data was used for the

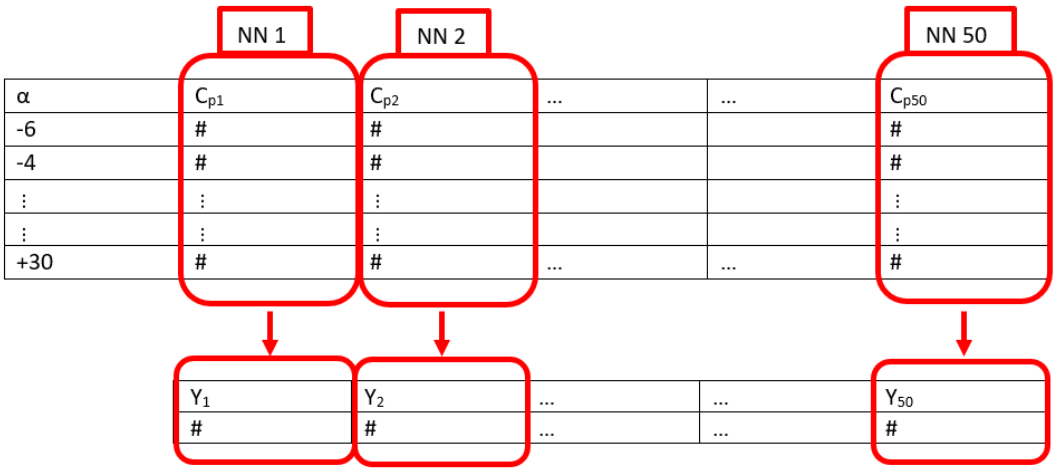


Figure 20: Data formatting for model 1 and 2

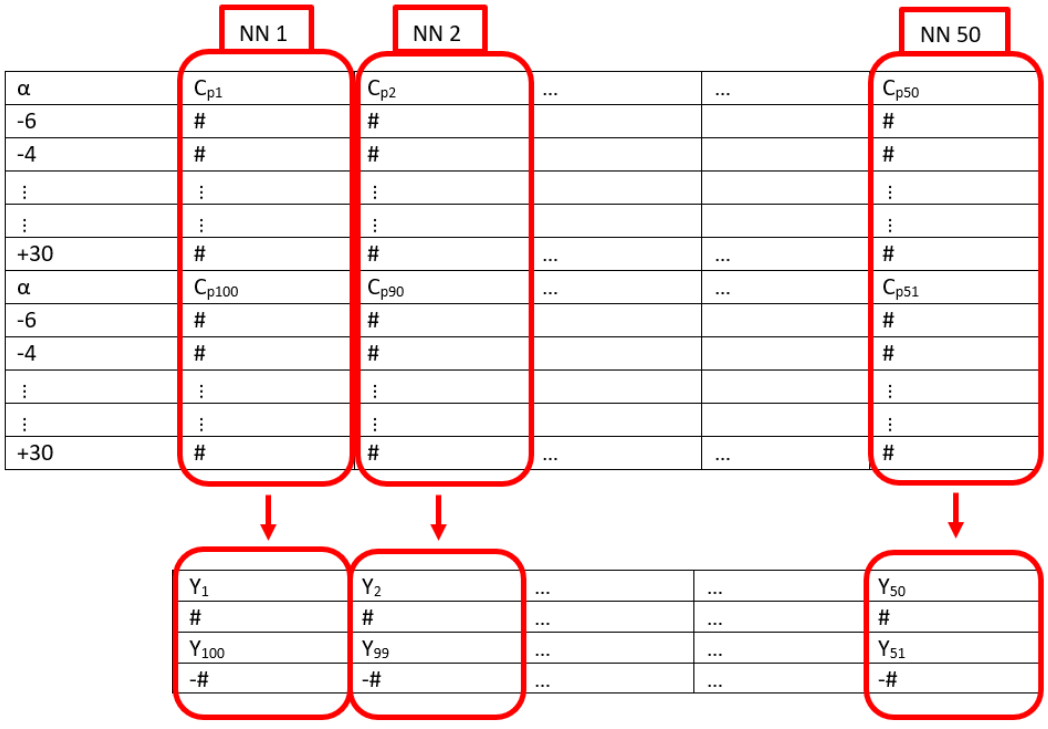


Figure 21: Data formatting for model 3 and 4

first two models while Figure 21 is for the third and fourth models. A unique neural network was created for each x-coordinate. The coefficient of pressure data across all angles of attack were used as input features and the single y-coordinate was the output feature. For the third and fourth model, there was no need for the network to predict two y-coordinates as they are the absolute value of each other. Much like in the preprocessing the outputted y-coordinates can be concatenated to an array which is the negative of itself. The main architecture of these networks is shown in Figure 22 which shows the input layer with *i* number of inputs and a hidden layer of *j* number of neurons which have sigmoid activation functions. There are also weights denoted as *w* and biases signified as *b*. This is then connected to a single output neuron which has a linear activation function.

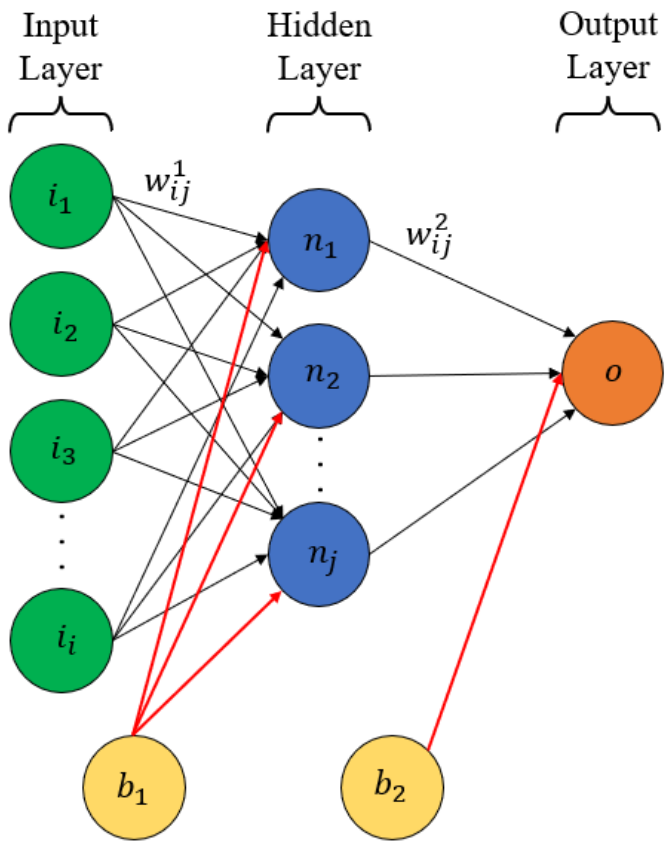


Figure 22: General neural network architecture

The data which is output is not smooth like how an airfoil should be. In the next section the use of a signal filter to smoothen the data is explained

Post Processing

As will be seen in the next chapter because of the way the neural network is used, the total response from multiple networks will have discontinuities and undulations in the data. The way that these neural networks were implemented were that at a single x-coordinate, the coefficient of pressure data for that point across multiple angles of attack from -6° to +30° in two-degree increments, was the input. The output or target for the training data was the y-coordinate of the airfoil shape at that x-coordinate. So, there were 19 input features and 1 output feature for each network which predicted the y-coordinate from the first two models. This means that 50 networks were created for one model to predict the y-coordinates. The other two models which had 38 input features and one output also contained 50 neural networks each.

Because the networks have no communication between each other, there is no way for the resulting airfoil shape to be smooth. To smooth the resulting airfoil geometry, a signal filter was used which has a built-in function in MATLAB. The next section will go over how this filter works and how it was utilized.

Savitzky – Golay Filtering

Savitzky – Golay filters allow for the signal to be smoothed by using a quadratic polynomial which is fitted to each window, the size of which is user-defined. For a given set of data with size $2M + 1$ number of points centered on n_0, a quadratic polynomial with N number of coefficients will be used to fit those points. This can be seen in Figure 23 where $M = 3$ and $N = 3$. Then another quadratic polynomial will be used to fit points centered around n_1 and so

on. All these polynomials will then be convoluted similar to something like a moving average but with polynomial approximations instead.

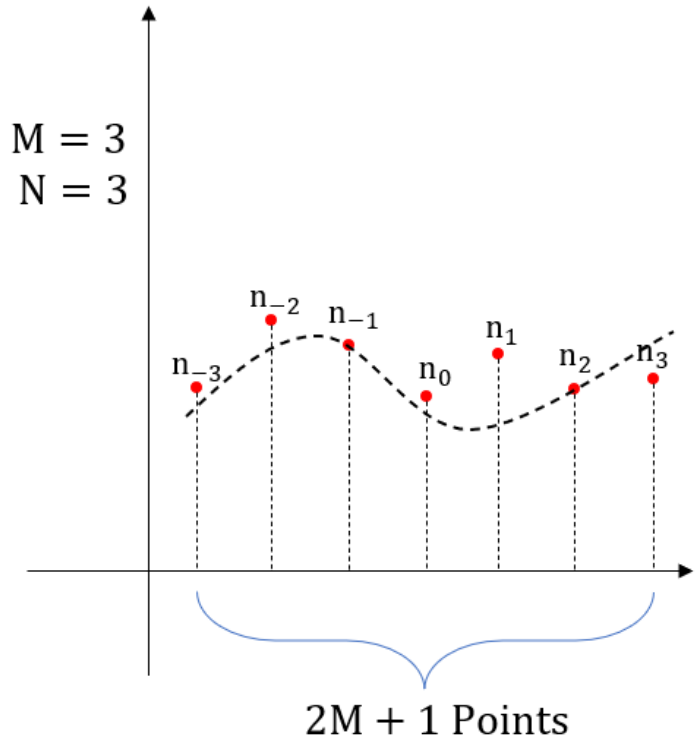
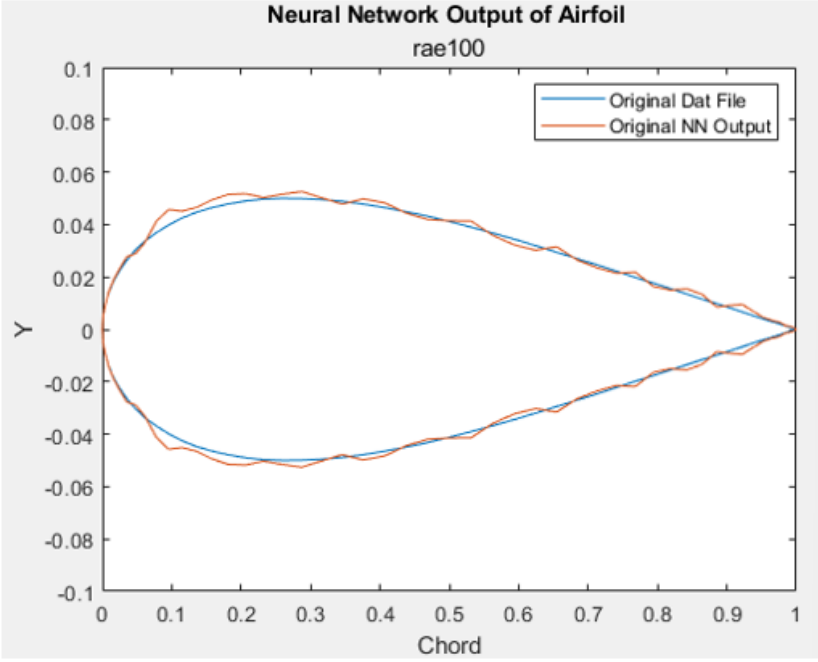


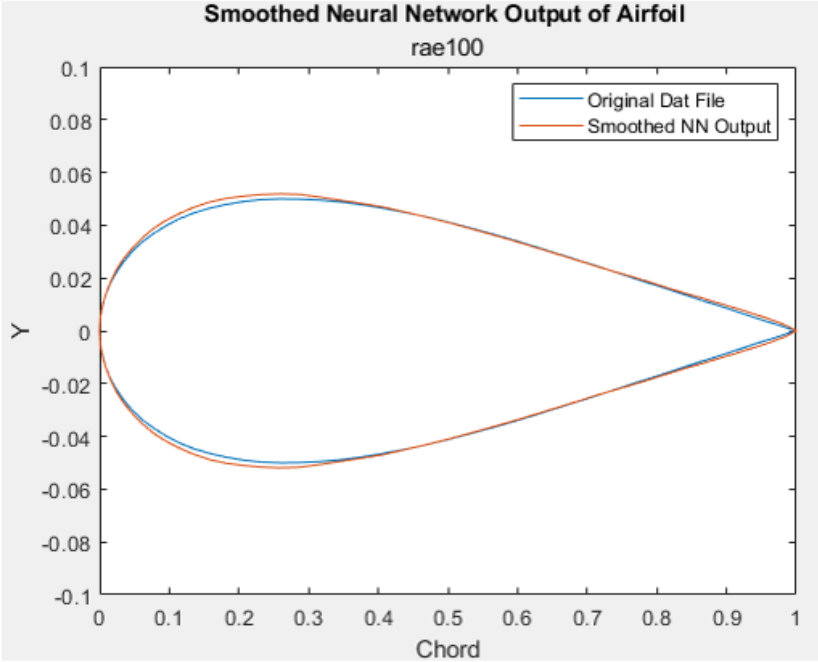
Figure 23: Savitzky-Golay filter diagram

Utilizing a signal filter for this use is not uncommon, in fact it was used in one study to smoothen the generated output from a GAN architecture before being input to the discriminator (Achour, 2018). A Savitzky – Golay filter was used and was effective in smoothing the shape of the randomly generated airfoils. The effectiveness can be seen in Figure 24 where in a) the raw neural network is shown overlayed onto the original .dat file data. In b) the smoothed output is shown and it shows good matching to the original data. This was implemented in MATLAB by using the *smoothdata()* function which allows for the user to choose the smoothing method and the size of the window. In this case a smoothing factor of 0.01 was used which is quite small and the degree of the quadratic polynomials in the filter was 2. Considering that the neural networks

have no communication between each other, this shows that there is a good correlation between the coefficient of pressure and the y-coordinates of a symmetrical airfoil.



(a)



(b)

Figure 24: a) Raw neural network output b) Smoothed neural network output

CHAPTER III

NEURAL NETWORK TRAINING AND TESTING

Introduction

In the previous chapter neural networks and the chosen architecture for this study was explained. There were four models which were created and from now on they can be called Model 1, Model 2, Model 3, and Model 4. Model 1 takes the coefficient of pressure distribution from the upper side of the airfoil surface and predicts the upper surface geometry. Model 2 uses the same process as the first but instead it is predicting the geometry from the lower component of the coefficient of pressure distribution. Model 3 and 4 both use the upper and lower coefficient of pressure distribution at a given point and predict the y-coordinate at that point. For each of these models there were 50 individual neural networks which predicted the y-coordinate of the airfoil geometry at a specific x-coordinate location from the coefficient of pressure at that location. The training data needed for each of these networks included data from -6° to +30° angle of attack in two-degree increments. This means that the individual neural networks for Model 1 and Model 2 both had 19 feature inputs and 1 feature output. For Model 3 and Model 4 there were 38 feature inputs and 1 feature output. The way that the data can be visualized is in Figures 20 and 21. Each of the neural networks utilized the Levenberg – Marquardt algorithm and only Model 4 had Bayesian regularization. Each neural networks' neuron activation functions were sigmoid, and the output neurons were linear. Model 1 and Model 2 had 15 hidden neurons and Model 4 and Model 5 had 20 hidden neurons.

As was discussed in the previous chapter the first step to training these models was to find x and y-coordinate data on symmetric airfoils. Each of the airfoils were downloaded from the UIUC database but because of formatting inconsistencies between data files, a script had to be made which would reformat each file. Various steps were taken to reformat and finally a polynomial regression was performed on the airfoils so that they could each be evaluated on the same x-coordinates which had cosine spacing. This is the way that the data had to be normalized. This new normalized data was then fed into a panel method script which found the coefficient of pressure distribution at multiple angles of attack and was saved to a Microsoft Excel file. These files were then loaded back into MATLAB to be consolidated and finally cut into the necessary datasets to train the neural networks. Once 50 neural networks were trained for each model, a Savitzky – Golay filter was utilized on their output. The training and testing of the neural networks will be discussed in the following sections which give more insight into how the data was utilized.

Training

There were 132 symmetric airfoils found on the UIUC database which was aided by using airfoiltools.com. The website allowed for a user to view only the symmetric airfoils contained within the database. Once all of the coordinate data files were downloaded, the polynomial fitting script was applied. After this fitting, there were only 89 airfoil whose geometries were not altered by the fitting. Although they had low MSE values, these polynomial regressions had erroneous undulations and discontinuities in the shape. Each airfoil which was evaluated on the normalized x-coordinates was plotted and visually inspected for situations like the one mentioned. This is to take out any unneeded data which would not train the models well.

There were 89 airfoils whose polynomial fitting was usable and these were then input to the panel methods. Now there were 101 points of coefficient of pressure data as well as the same number of y-coordinates. The reason why the models only required 50 neural networks to be used is that the middle point or the 51st datapoint where $x = 0$ and $y = 0$, can be neglected since this datapoint is the same across all airfoils. This also allows the data to be split evenly in half so that it can be used in Model 3 and Model 4 and how its shown in Figure 20. The x-coordinate data which is the same for all airfoils after the regression is made, starts at $x = 1$ for the first datapoint, then descends to $x = 0$ for the 51st datapoint, then ascends back up to $x = 1$ for the final or 101st datapoint. The y-coordinate data starts at the lower surface and after the 51st point contains the upper surface.

In terms of training, for the first three models, 70% of the data or 63 airfoils were utilized for training, 15% or 13 airfoils were utilized for validation and the final 15% was used for testing. What this means is that 70% of the data was fed into the neural network to train the model. The validation dataset allows the network to evaluate the loss of the validation dataset and if it becomes too low will interrupt the process to further fine tune the parameters but the network will not learn from this data. The testing data will then be used to evaluate the trained network and give an MSE for the predicted values. The data was cut up into their respective datasets randomly so that there would not be a bias towards one subset of data. The 63 airfoils used for training will be different between each individual neural network as well as the validation and testing datasets. An example of how the data was utilized into the different datasets is shown in Figure 25. For Model 4, the data was utilized differently where there was no validation set used. this is because in a regular neural network validation is needed to fine tune the weights and biases of the neural network layers and activation functions. In a network which

has Bayesian regularization, this is not needed since the performance function already accounts for this by normalizing the weights and biases.

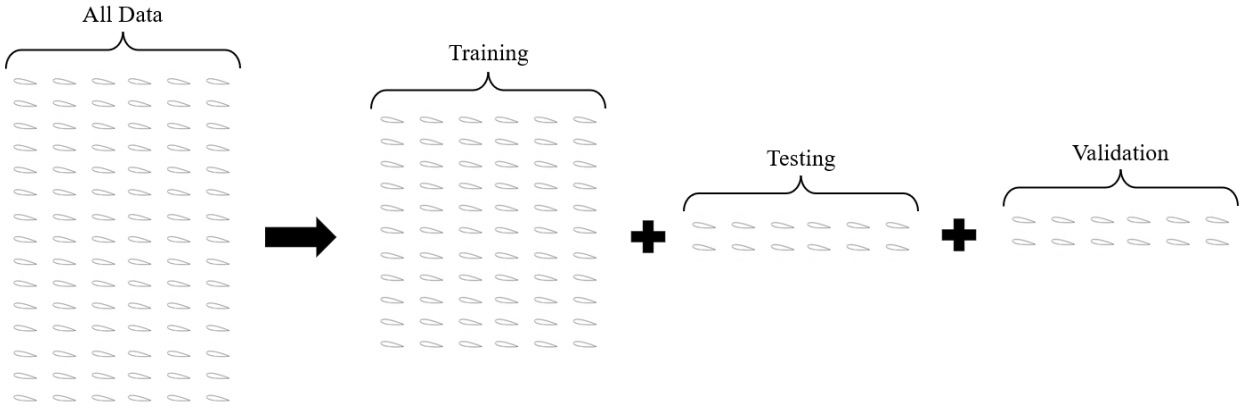


Figure 25:Neural network data utilization

The output of these neural networks is shown in Figure 26 where the top left graph shows the output of Model 1, the top right graph shows the output of Model 2, Model 3 is in the lower left graph and Model 4 is in the lower right graph. This is the raw output of the neural network and shows some inconsistencies which would not be found on a normal airfoil. This shows the need for the Savitzky – Golay filter mentioned in the previous chapter. Overall the shape of the airfoil is near what the original data is. Also, this is an airfoil whose data was used in the training process. Its coefficient of pressure distribution was input into the trained neural networks and its output is shown. This will not show us whether the data is overfit, but in the next section this question will be answered.

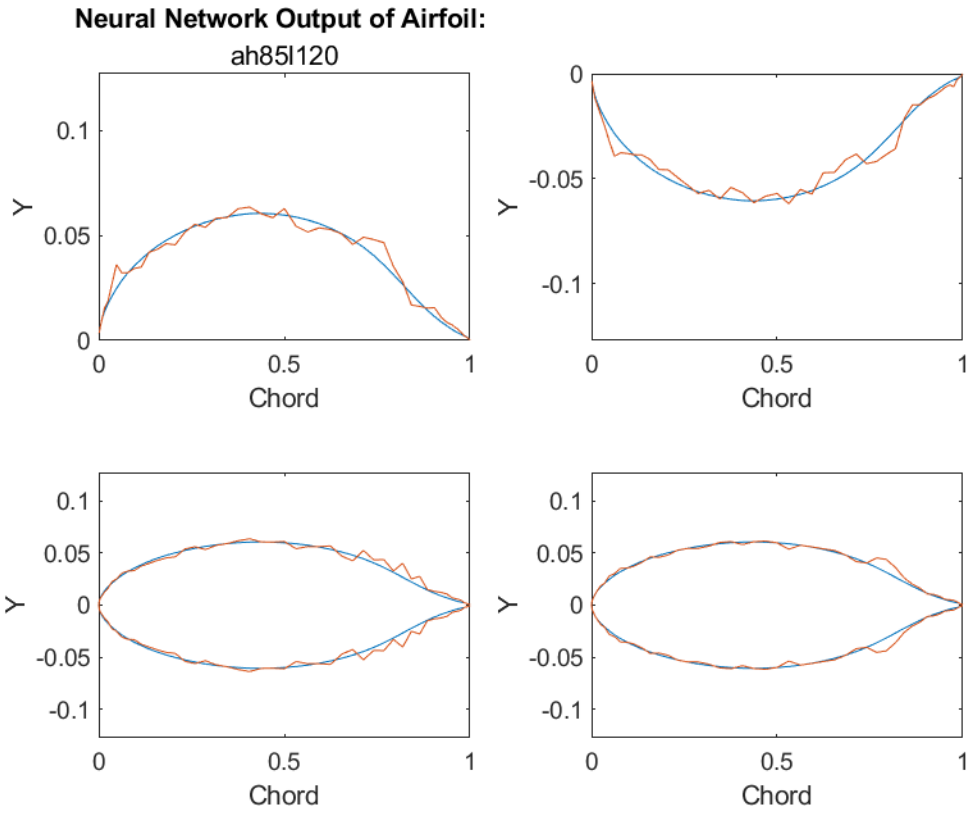


Figure 26:Raw neural network output of airfoil AH85l120

Testing

To actually test the models without using data the networks have already seen, it is essential to find symmetric airfoils which haven't been used for training, validation, or testing. this led to re-evaluating some of the airfoils whose polynomial fit could not be found. In some cases the polynomial would fit because the trailing side of the airfoil was linear, leading to a badly fit polynomial. The highest order the polynomial was limited to was 50 so this may have also had an effect on whether the airfoil could be properly fit. Regardless, there were ten airfoils whose shape could be evaluated on the proper x-coordinates after a new script was developed to account for these outliers. The coefficient of pressure for these airfoils was found by using the panel method script and their data was input into the trained models so that the airfoil shape

could be predicted. The details of these results will be discussed in the next chapter but the raw output of the neural networks for one of these airfoils is shown in Figure 27.

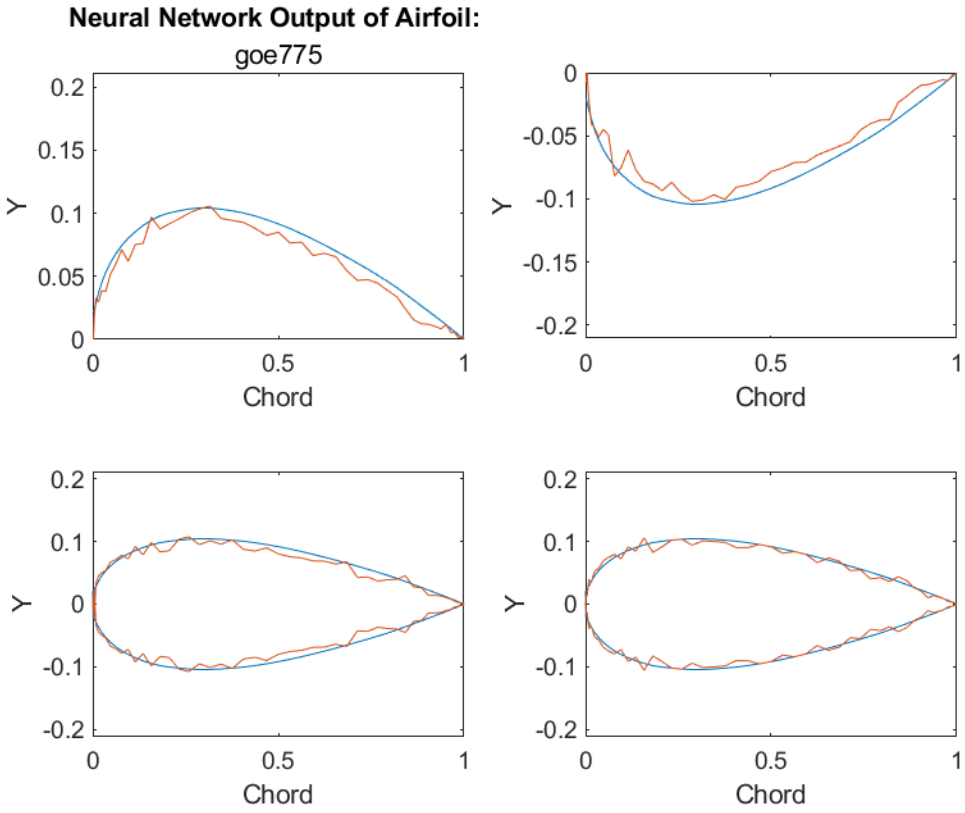


Figure 27:Raw neural network output of airfoil GOE775

One purpose of this chapter was to show the difference between what will be called training and testing. Training the neural network involves creating a training, validation, and testing dataset. Each of these datasets were randomized for each neural network so that there was no bias on the final trained networks. When testing is referenced, this is unrelated to the testing dataset which is used in the training of the networks. It is an actual test of the models by using airfoils that haven't been seen by the networks since every airfoil that was used to train was at some point used in the various datasets. In a regular application for neural networks, this type of

testing would not be needed since whatever the output is, is the final output. In this case there are multiple neural networks whose outputs are being used in parallel to create a composite output. Because the datasets are randomized for each network, this extra step is needed since all of the data has somehow touched and biased the networks to some degree.

The other aim of this chapter is to detail what exactly the input and output of the neural networks were and how they were used. There were 89 usable airfoils which were randomly sectioned into the training, validation, and testing datasets with a 70/15/15 split for the first three models and had a 85/15 split for the training and testing datasets for the fourth model. The networks had either 19 or 38 input features and one output which was the y-coordinate at that point and each of the models had 50 neural networks whose output was then filtered to produce an airfoil geometry. The next chapter will detail the results of all of this and discuss what the implications of these findings are.

CHAPTER IV

RESULTS AND DISCUSSION

Introduction

This chapter will discuss the outputs of the different models used to predict airfoil geometry from coefficient of pressure distribution including the raw output and the smoothed output. The details such as the MSE of the trained networks will also be discussed. Recall in previous chapters that there were initially 132 airfoils whose geometries were defined as x and y-coordinates in a .dat file. These airfoil geometries were then fitted with a polynomial curve which was found iteratively with a MATLAB script. This led to only 89 airfoils which had an adequate polynomial fit. These 89 airfoils were then fed into a panel method script which was automated to find the coefficient of pressure distribution across 19 different angles of attack from -6° to +30° in two-degree increments. Once all of this data was obtained, it was consolidated and allocated into their respective datasets to train the neural networks. The neural networks utilized Levenberg – Marquardt backpropagation while only Model 4 utilized Bayesian regularization. Model 1 and Model 2 both used 19 feature inputs with 1 output feature networks. Model 3 and Model 4 both had 38 feature inputs with 1 output feature networks. Each model contained 50 separate neural networks, each of which could predict one y-coordinate for a given coefficient of pressure distribution at a specific x-coordinate. For Model 1 and Model 2 the hidden layer contained 15 neurons which had sigmoid activation functions. Model 3 and Model 4 had 20 neurons in the hidden layer which also had sigmoid activation functions. The training

consisted of utilizing the data so that 70% was used for training, 15% for validation, and 15% for testing. Model 4 utilized the data as 85% for training and 15% for testing. When testing is used in this sense it means that there was data which was set aside from the 89 airfoils. After a network was trained and met an MSE which had converged on a minimum, this number could be recorded and evaluated to give the user a good estimate as to how well the network performs on data which was used for training.

After all these networks were trained, another script was created which would take a given coefficient of pressure distribution and apportion the data accordingly to be input into the various networks. The predicted values were obtained which is the raw neural network output. This raw output was then fed through a Savitzky – Golay signal filter which could smooth the overall shape of the predicted airfoil geometry. To properly test the networks with data which they had not seen before, symmetrical airfoils had to be input into the models to find an MSE and determine how well the overall model performs. Since the training, validation, and testing datasets were all allocated randomly for each network, the networks have inevitably been touched and thus biased by every airfoil. Thankfully, there were symmetric airfoils which were discarded after the polynomial fitting process which could be fitted by interpolation. Ten airfoils which were not part of the 89 that were used to train the networks were found which could be evaluated on the desired x-coordinates. This process led to shapes which were visually close to the original airfoil shapes. These 10 airfoils were fed into the panel method script to find the coefficient of pressure distribution at various angles of attack. This data was then fed into the script which would input the data to the neural networks to obtain a prediction which was then filtered using the Savitzky – Golay filter. This is what will be referred to as testing in this chapter.

Model 1

Model 1 used 50 – 19 input feature, single output feature neural networks which utilized Levenberg – Marquardt backpropagation with 15 sigmoid neurons in the hidden layer and one linear output neuron. This model was used to predict the airfoil geometry from the upper surface contribution of the coefficient of pressure distribution.

Training

The training of Model 1 included taking 63 of the 89 airfoils or 70% and utilizing the data to train the 50 neural networks. 13 airfoils or 15% of the 89 airfoils were used in validation and another 15% were used in testing to obtain a testing MSE. Table 2 shows a subset of the testing MSE values for the networks which were trained. The testing MSE is quite low which means that there is a good correlation between the input and output data which should be able to perform well on new coefficient of pressure data.

A bar graph of the values for testing MSE can be seen in Figure 28. This figure contains the MSE for each of the 50 networks used. For this model there are high MSE values for the first tenth of the chord as well as for halfway to three fourths of the chord. The reason for this is that these airfoils are highly variable and some will taper at the leading edge and the trailing edge quite differently. The variability of these airfoils can be seen in the Appendix B where there are images of the airfoils and their smoothed neural network output.

Table 2:Subest of testing MSE for Model 1

Network	Model 1
1	1.98E-07
2	3.54E-07
3	7.21E-07
4	1.03E-06
5	1.48E-06
6	2.68E-05
7	2.43E-06
8	3.29E-05
9	5.76E-06
10	2.39E-05
11	3.10E-05
12	1.87E-04
13	9.37E-06
14	6.30E-05
15	7.75E-05
16	9.50E-05
17	2.93E-04
18	1.90E-05
19	3.56E-05
20	2.00E-05

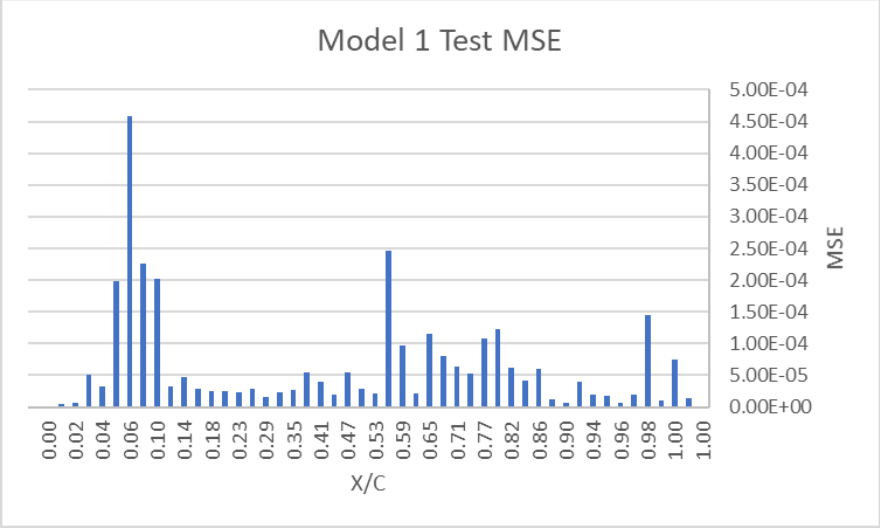


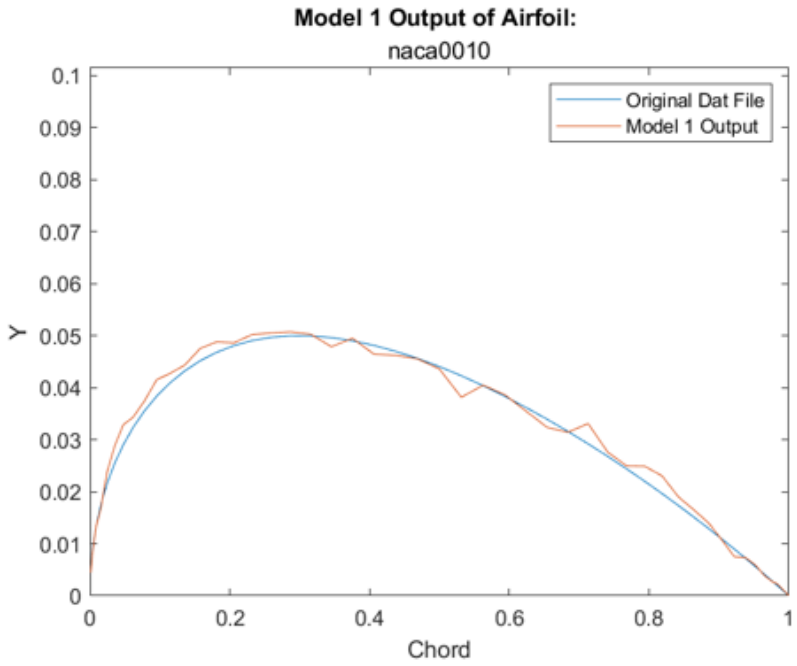
Figure 28:Bar graph of test MSE for Model 1

Table 3 shows the MSE of airfoils which were used in the training of these models when they were reinput into the neural network model. Instead of looking at each neural network's MSE, now the table is showing the total MSE between the entire original airfoil geometry and the predicted airfoil geometry. Table 3 only contains a subset of these airfoils for space considerations. These same airfoils will be used as a benchmark for each model in the following sections.

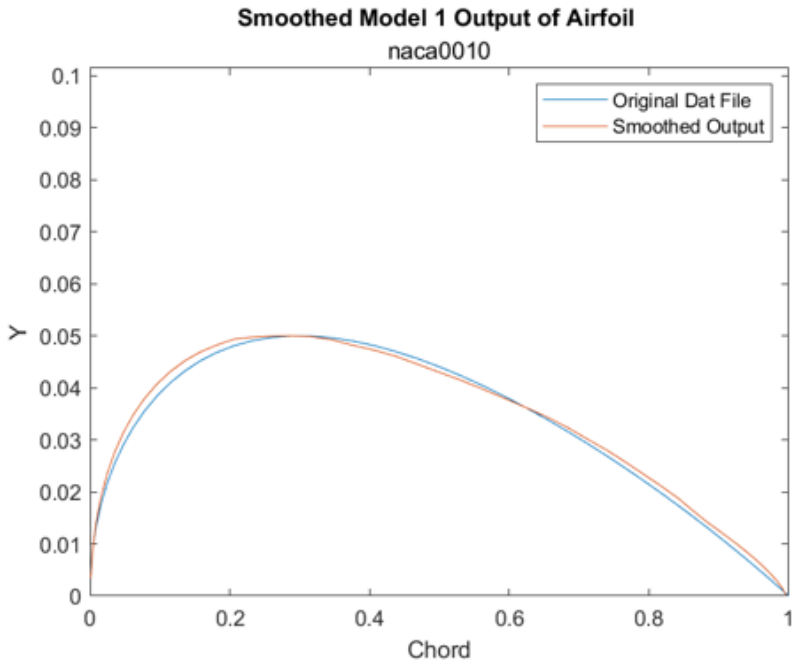
Table 3:Subset of raw and smoothed airfoil MSE for Model 1

Airfoil	Raw	Smoothed
naca0006	1.92E-04	1.82E-04
naca0008	3.16E-04	3.07E-04
naca0010	4.94E-04	4.86E-04
naca0012	7.17E-04	7.08E-04
naca0015	1.11E-03	1.10E-03
naca0018	1.57E-03	1.56E-03
naca0021	2.19E-03	2.18E-03
naca0024	3.04E-03	3.02E-03

Although the MSE only sees a marginal increase in error reduction, Figure 29 allows the unfiltered and filtered outputs to be seen against the original airfoil data. The signal filter does an excellent job in smoothing the shape to have more realistic characteristics which would perform more like a real-life airfoil as well as fits the overall geometry of the original airfoil closely.



(a)



(b)

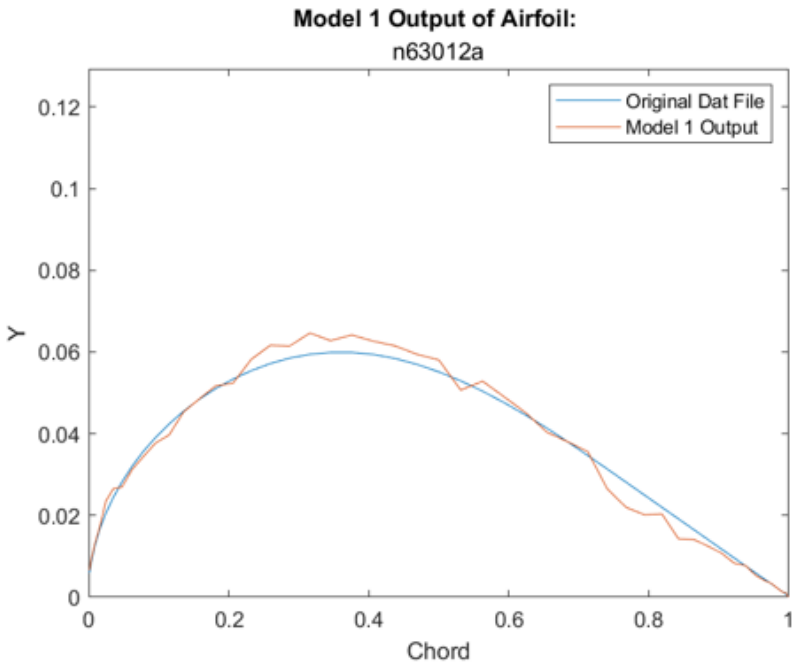
Figure 29:a) Raw Model 1 output of airfoil NACA0010 b) Filtered Model 1 output of airfoil NACA0010

Testing

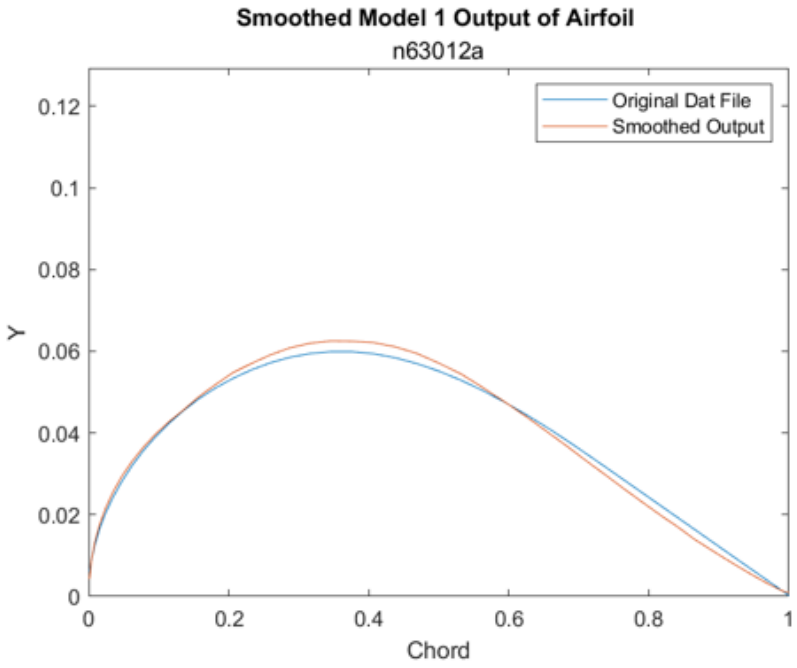
For the testing portion, as has been explained before, ten airfoils which the 50 networks within the model had never encountered, were used to realistically test the overall model performance. The testing MSE values for these 10 airfoils is seen in Table 4 and a sample of their actual output is shown in Figure 30 where a) shows the raw output and (b) shows the smoothed output. Aside from the visual likeness, Table 4 shows that these tested airfoils have comparable MSE values as to the original airfoils used to train the model which shows that this model is generalized well and not overfit. More graphs can be seen in Appendix B but overall the tested airfoils performed well especially after the signal filter is applied. In the case where the upper surface geometry is predicted by the coefficient of pressure at that point, this model shows that there is at least a small direct correlation between the two datapoints in symmetric airfoils.

Table 4: Raw and smoothed MSE values for testing Model 1

Airfoil	Raw	Smoothed
n63012a	6.01E-04	6.00E-04
n63015a	9.47E-04	9.50E-04
goe459	7.09E-04	6.96E-04
goe460	1.86E-03	1.85E-03
goe775	2.20E-03	2.17E-03
naca001234	3.43E-04	3.30E-04
nacam3	5.80E-04	5.66E-04
oaf139	1.71E-03	1.71E-03
raf27	4.53E-04	4.50E-04
raf30	8.23E-04	8.19E-04



(a)



(b)

Figure 30:a) Raw Model 1 output of airfoil NACA63012a b) Filtered Model 1 output of airfoil NACA63012a

Model 2

Model 2 consisted of 50 neural networks which were trained with 19 input features containing the coefficient of pressure distribution throughout different angles of attack for a specific x-coordinate to predict 1 y-coordinate output feature. These networks utilized Levenberg – Marquardt backpropagation with 15 sigmoid neurons in the hidden layer and one linear output neuron. This model was used to predict the airfoil geometry from the lower surface contribution to the coefficient of pressure distribution.

Training

The same procedure that was used to train Model 1 was used in this case for Model 2. The training, validation and testing datasets were in a 70/15/15 split which resulted in 63, 13 and 13 airfoils being allocated, respectively. The testing dataset was used to find a test MSE to estimate the effectiveness of the model. There were 50 different neural networks trained each of which would predict a separate y-coordinate at a specific x-coordinate. In Table 5 the MSE for testing during the training process of the networks is shown. Each network had an MSE but Table 5 only shows 20 different values, the table can be expanded to what is seen in the Appendix A and Table 17 which shows all 50 networks' MSE value for testing. The MSE is relatively low which shows that there may be a correlation when predicting the airfoil shape from coefficient of pressure distribution at that point. Figure 31 shows a bar graph of all 50 networks which allows the data to be visualized. Similar to the previous model, the MSE starts to increase around halfway through the chord until three quarters of the chord. Regardless, the overwhelming theme of the MSE shows that most are quite low and there are only a few outlier networks where the MSE is high.

Table 5: Subest of testing MSE for Model 2

Network	Model 2
1	1.45E-07
2	1.36E-05
3	7.50E-05
4	1.06E-05
5	1.44E-04
6	1.87E-05
7	7.03E-06
8	1.71E-05
9	2.04E-05
10	4.05E-05
11	6.64E-06
12	1.21E-05
13	6.08E-05
14	4.19E-05
15	6.20E-05
16	1.22E-04
17	1.07E-04
18	5.31E-05
19	6.28E-05
20	7.95E-05

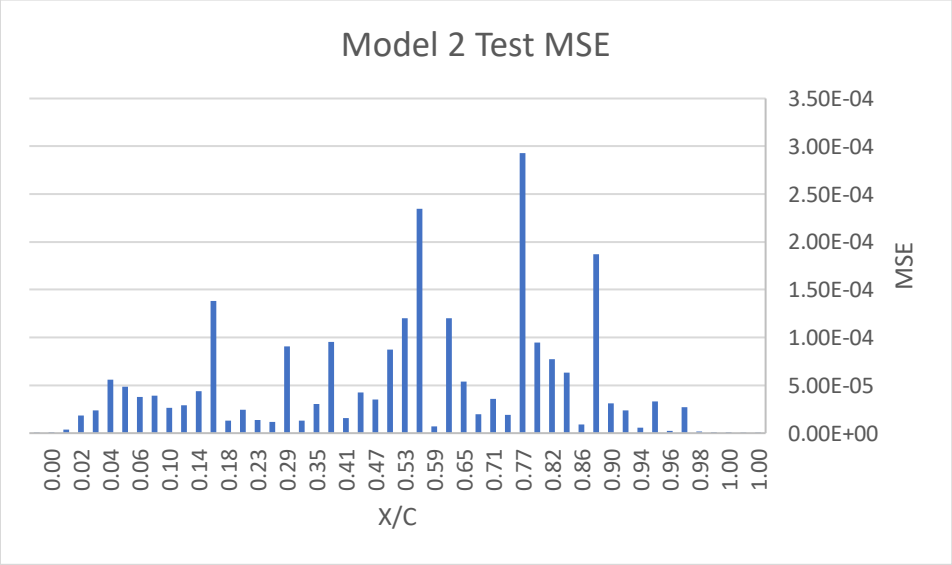


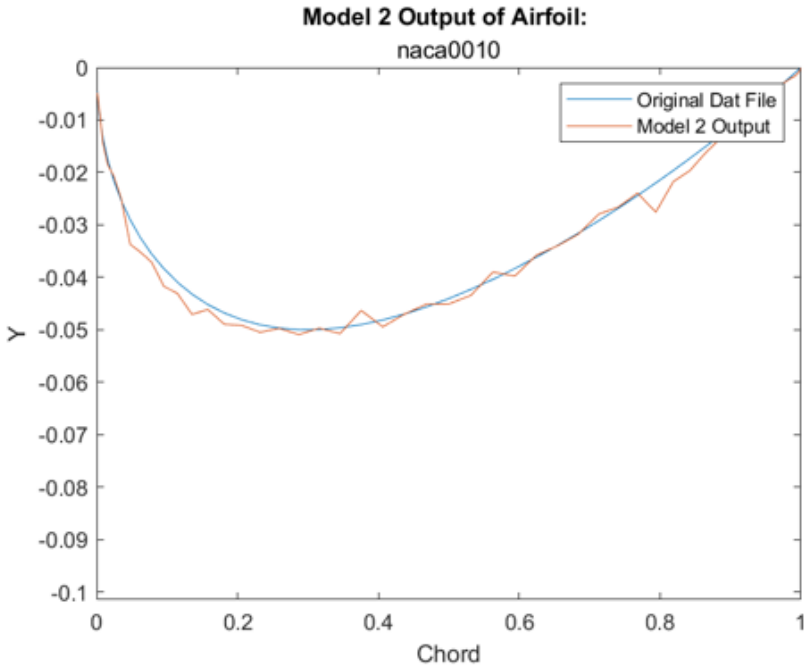
Figure 31: Bar graph of test MSE for Model 2

The same benchmark airfoils from the previous section are used to find an MSE for airfoils whose data was used in the training process. Their coefficient of pressure distribution was fed back into the networks, or Model 2, and the MSE was acquired from the raw Model output as well as the filtered output. This data can be seen in Table 6 where both datasets are shown. The MSE of the original airfoil to the predicted airfoil shape is similar to what was found in Model 1. While the coefficient of pressure distribution for the upper and lower surfaces are unique, both have similar trends which are inversions of each other. This must be the reason why both models perform similarly and have a similar amount of error and MSE.

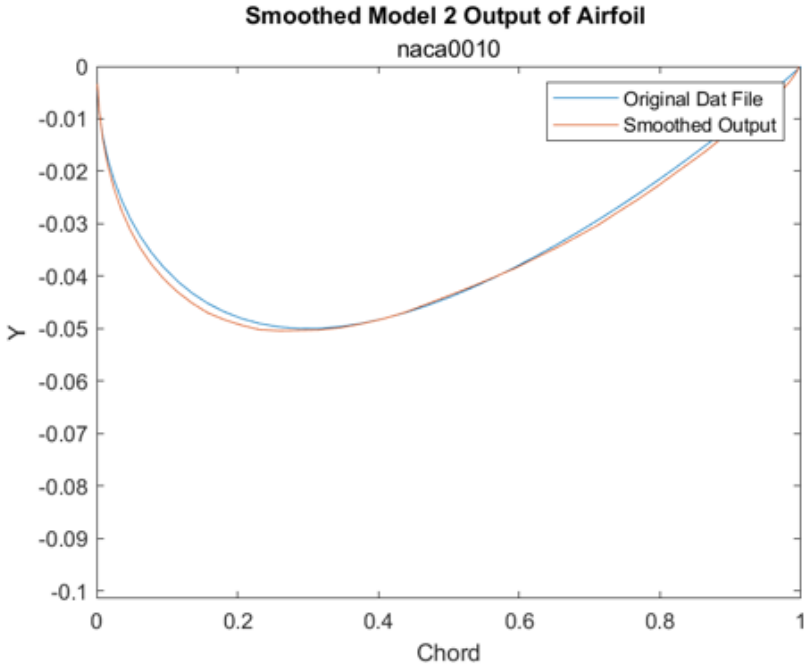
Table 6: Subset of raw and smoothed airfoil MSE for Model 2

Airfoil	Raw	Smoothed
naca0006	2.78E-04	2.80E-04
naca0008	3.83E-04	3.86E-04
naca0010	5.16E-04	5.18E-04
naca0012	6.71E-04	6.71E-04
naca0015	9.29E-04	9.30E-04
naca0018	1.22E-03	1.22E-03
naca0021	1.57E-03	1.57E-03
naca0024	2.10E-03	2.07E-03

The output of Model 2 for NACA0010 was plotted in Figure 32 which shows the unfiltered and filtered output. Visually, the smoothed output is very close to the original airfoil geometry. Mathematically, the airfoil has the third best MSE from the subset of airfoils that are being used as a benchmark. Without testing, these results look promising that an unknown airfoil geometry may be predicted with high accuracy.



(a)



(b)

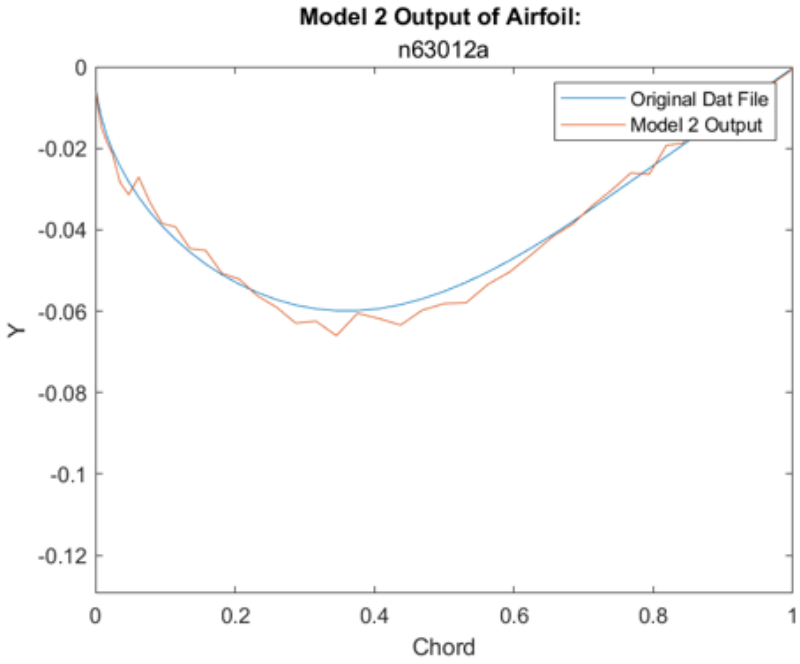
Figure 32: a) Raw Model 2 output of airfoil NACA0010 b) Filtered Model 2 output of airfoil NACA0010

Testing

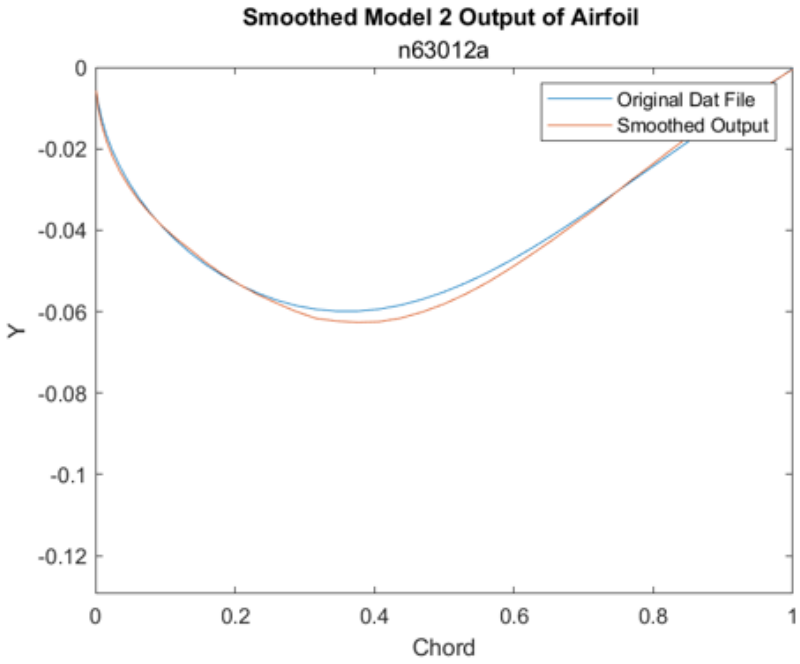
Testing Model 2 consisted of using the 10 airfoils being used as a benchmark which the model has not encountered before. The unfiltered and filtered output is shown in Table 7 which shows some of the airfoils had a higher MSE when filtered. The filter smooths the data and creates a more realistic shape but as can be seen in the second graph of Figure 33, the bulk of the error has just shifted to one portion of the airfoil which can be happening to the other airfoils. Although the shape is smoothed and may match better visually, the error may be shifted to other regions which now make up for the improved accuracy elsewhere to bring the MSE back up. This is an interesting case where while lower MSE is desired, the smoothed output is still chosen as the representative geometry because of its smoothness. Both Model 1 and Model 2 have proven that there can be a direct correlation between the coefficient of pressure experienced on an airfoil and the y-coordinate at that point. This gives credence to the approach and allows for further investigation to occur which can include the data from both the upper and lower surfaces.

Table 7: Raw and smoothed MSE values for testing Model 2

Airfoil	Raw	Smoothed
n63012a	7.60E-04	7.55E-04
n63015a	1.01E-03	1.01E-03
goe459	7.90E-04	7.93E-04
goe460	1.66E-03	1.66E-03
goe775	1.53E-03	1.54E-03
naca001234	8.66E-04	8.45E-04
nacam3	7.11E-04	7.04E-04
oaf139	5.58E-04	5.68E-04
raf27	5.13E-04	5.12E-04
raf30	7.44E-04	7.40E-04



(a)



(b)

Figure 33: a) Raw Model 2 output of airfoil NACA63012a b) Filtered Model 2 output of airfoil NACA63012a

Model 3

Model 3 is different from the first two because the goal is to predict the airfoil geometry from both the upper and lower coefficient of pressure contributions. This will allow the model to have more input features which may help to improve accuracy of the networks. This model uses 50 neural networks which utilize Levenberg – Marquardt backpropagation with 20 sigmoid neurons in the hidden layer and one linear output neuron. The networks are trained on an input which has 38 features, the coefficient of pressure at 19 various angles of attack on the upper and lower surface of the airfoil. Each network also has 1 output feature.

Training

Training was done in the same way as in the previous models, by allocating the training, validation, and testing datasets in a 70/15/15 split. The only difference for this model is that there were 38 input features instead of 19 because the upper and lower surface pressure coefficient distributions were combined. This combining should allow the networks to train better and have lower MSE since the input will be more defined now. In Table 8, the MSE for the first 20 networks for this model are shown which have lower MSE than the previous models. Figure 34 is a bar graph of each neural network's test MSE. As will be seen in Appendix A, Table 17, the overall trend in MSE from the previous models to the current one is that this one consistently has lower testing MSE. There are a couple of networks whose MSE are lower compared to the networks in Model 3 but the tendency is for them to be lower. This confirms the hypothesis that the error should be reduced when defining the input further and will allow the model to predict an even more accurate airfoil shape when the coefficient of pressure distribution is inputted.

Table 8: Subest of testing MSE for Model 3

Network	Model 3
1	5.88E-07
2	9.89E-07
3	4.52E-06
4	6.95E-06
5	4.81E-06
6	9.75E-05
7	1.27E-06
8	3.94E-05
9	7.02E-06
10	1.57E-05
11	1.97E-05
12	6.24E-05
13	1.18E-04
14	1.35E-03
15	1.42E-05
16	3.96E-04
17	1.50E-05
18	1.07E-04
19	5.50E-05
20	1.84E-05

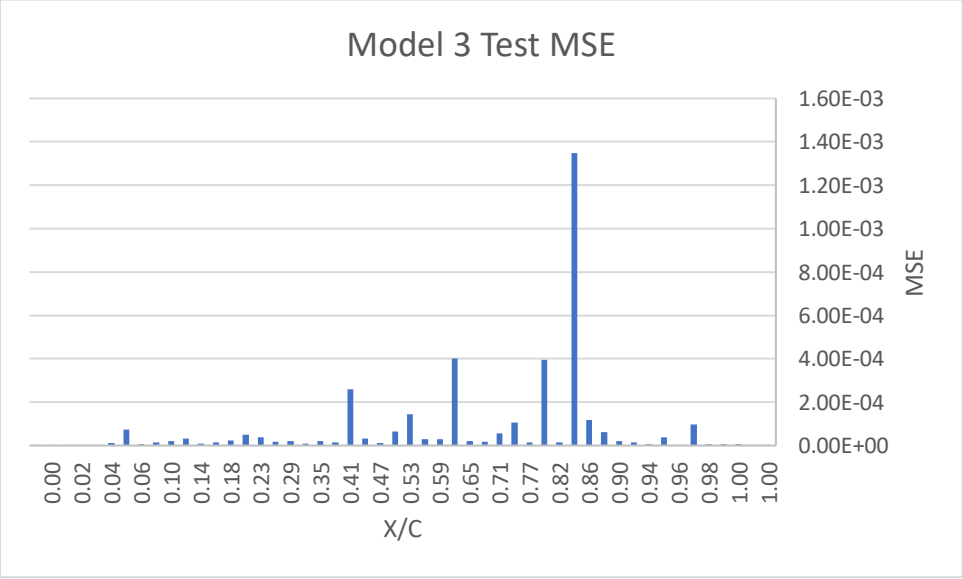


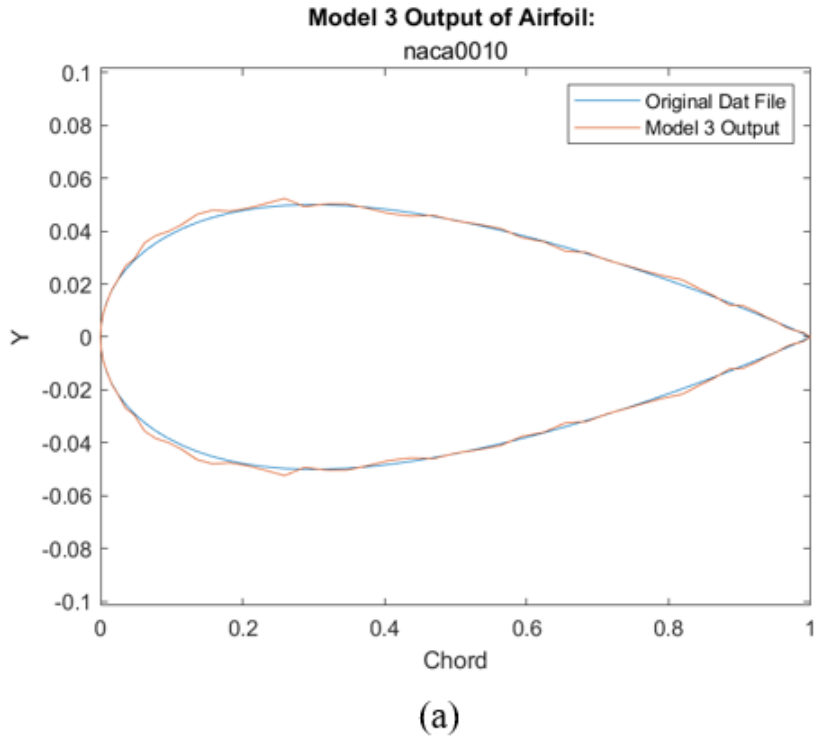
Figure 34: Bar graph of test MSE for Model 3

Figure 34 shows that the majority of the networks have an MSE lower than $2x10^{-5}$ but there are some outliers one of which that nearly reaches $1.4x10^{-3}$ which is quite high. Although that particular network has a high MSE, the output for the benchmarked airfoils still comes out better than the previous models and the filter also will help in smoothing any discontinuities or erroneous data that is within the predicted geometry. This is shown in Table 9 where the raw and smoothed MSE of the airfoils are compared side by side.

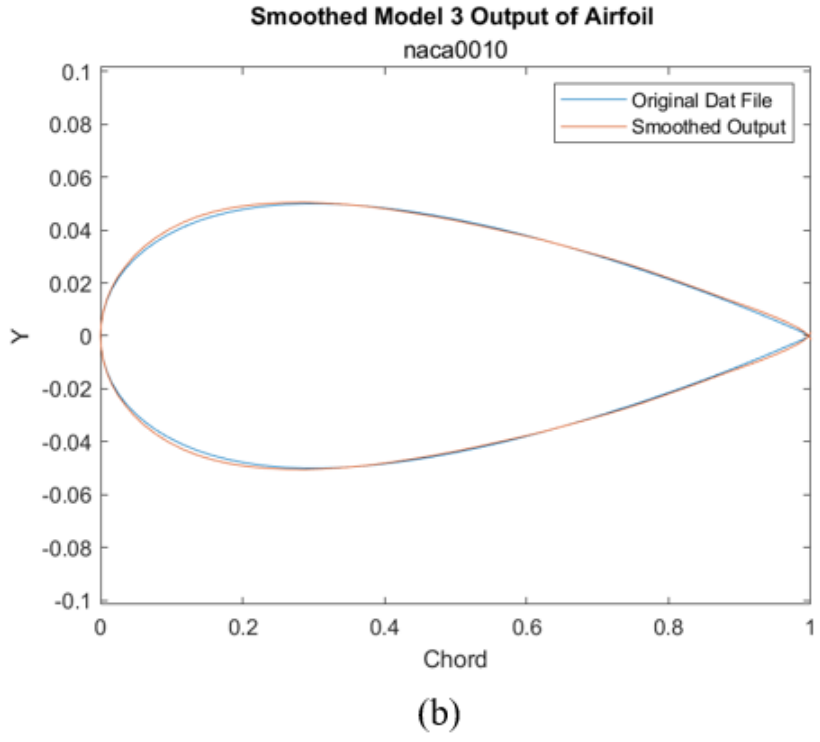
Table 9: Subset of raw and smoothed Airfoil MSE for Model 3

Airfoil	Raw	Smoothed
naca0006	8.33E-06	8.33E-06
naca0008	3.28E-06	2.71E-06
naca0010	1.39E-06	8.50E-07
naca0012	1.63E-06	7.92E-07
naca0015	7.88E-06	6.36E-06
naca0018	2.28E-05	2.28E-05
naca0021	3.94E-05	3.94E-05
naca0024	5.10E-05	5.10E-05

The output of the unfiltered and filtered model response is also plotted in Figure 35 where a) is the unfiltered and b) is the filtered Model 3 output. If this model has a low MSE for the testing benchmark airfoils, this can show that the networks are not overfitted and are able to predict airfoil geometry from previously unencountered coefficient of pressure distribution data. The next section will show the results of these benchmarks.



(a)



(b)

Figure 35: a) Raw Model 3 output of airfoil NACA0010 b) Filtered Model 3 output of airfoil NACA0010

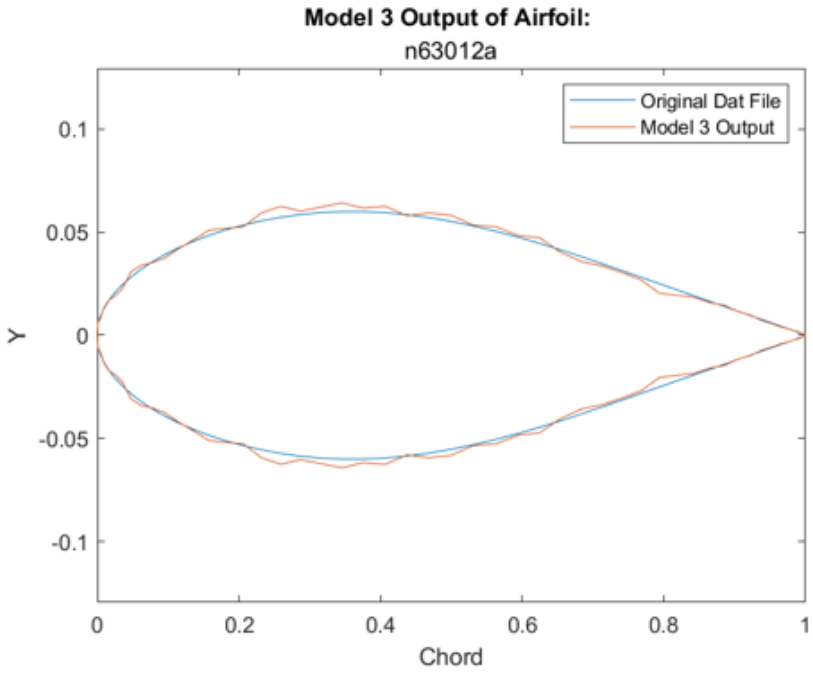
Testing

Model 3 follows the same procedure that was used in the first two models for benchmarking the model and testing it with airfoils whose pressure distribution the model has not previously encountered. The MSE data can be seen in Table 10 where the unfiltered and filtered output are shown. In most cases the MSE is improved after filtering and in one case, for airfoil GOE775, the MSE virtually stays the same. Again, this may be a side effect of the filter improving the smoothness to create a more realistic airfoil and improving error at certain regions of the output but also displacing more error in another region which leads to the entire geometry having a similar or unimproved MSE value. The other case may be that the raw output of the model has too many discontinuities which disallows the filter from adequately smoothing the data.

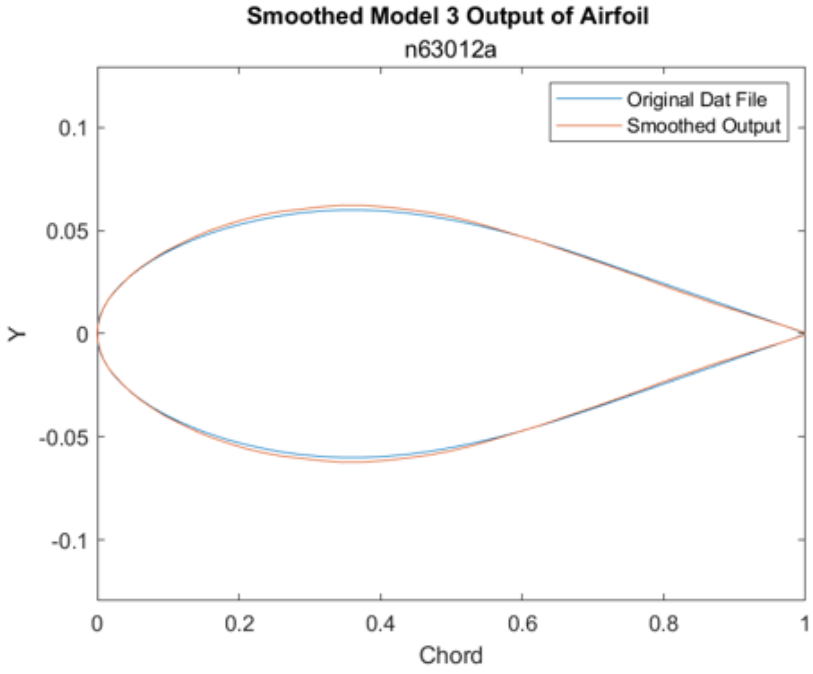
Table 10: Raw and smoothed MSE values for testing Model 3

Airfoil	Raw	Smoothed
n63012a	3.61E-06	1.20E-06
n63015a	7.04E-06	4.74E-06
goe459	1.77E-05	3.48E-06
goe460	7.67E-05	7.67E-05
goe775	1.05E-04	1.05E-04
naca001234	1.42E-05	4.32E-06
nacam3	1.22E-05	4.52E-06
oaf139	2.32E-05	1.78E-05
raf27	9.41E-06	3.21E-06
raf30	1.55E-05	3.47E-06

These benchmarks can be visualized in Figure 36 where a) shows the unfiltered Model 3 output and b) shows the filtered output. For this airfoil, NACA63012a, the MSE is the best of any of the previous models in terms of raw output as well as filtered output which shows that this model can fit unencountered data well and can give an accurate geometry for a given coefficient of pressure distribution. This proves that the model MSE will improve with more input features



(a)



(b)

Figure 36: a) Raw Model 3 output of airfoil NACA63012a b) Filtered Model 3 output of airfoil NACA63012a

to define the airfoil's coefficient of pressure distribution. Potentially, Bayesian regularization can improve this MSE since the objective of this is to generalize the model more to prevent overfitting.

Model 4

Model 4 was trained in the exact same way that Model 3 was, in fact they used identical datasets for training. There were 38 input features of coefficient of pressure distribution and 1 output feature for the y-coordinate. The neural network architecture is the same as the previous model which has 20 hidden sigmoid neurons and 1 linear output neuron which utilize Levenberg – Marquardt backpropagation, the only difference is that the backpropagation utilizes Bayesian regularization which allows the weights and biases of the network to be normalized so that the model can generalize well. While training may be more computationally intensive, the potential error reduction and generalization to data outside of what is being used to train can outweigh the cost.

Training

Training for this model is different from the previous models. Because this model utilizes Bayesian regularization, there is no need for validation which aids in fine-tuning the activation weights and biases; these values are already iteratively improved by the performance function. Instead of utilizing 70% for training, 85% was used for training 0% was used for validation and 15% was used for testing the model. Table 11 shows a subset of the network testing MSE values for Model 4. Some of these MSE values are even better than the previous model which is an empirical example of the great generalization Bayesian regularization can create. Figure 37

Table 11: Subest of testing MSE for Model 4

Network	Model 4
1	3.43E-07
2	3.38E-08
3	1.07E-05
4	1.21E-03
5	4.61E-04
6	4.47E-06
7	7.12E-07
8	4.26E-06
9	7.63E-06
10	1.03E-05
11	8.21E-06
12	1.07E-05
13	3.31E-04
14	2.70E-05
15	3.56E-05
16	3.11E-05
17	1.18E-04
18	4.77E-05
19	3.42E-05
20	4.44E-04

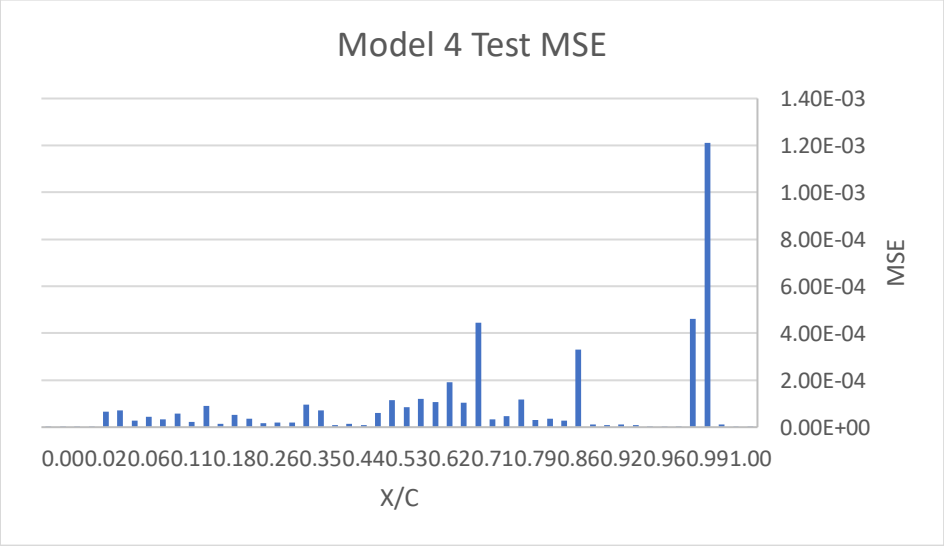


Figure 37: Bar graph of test MSE for Model 4

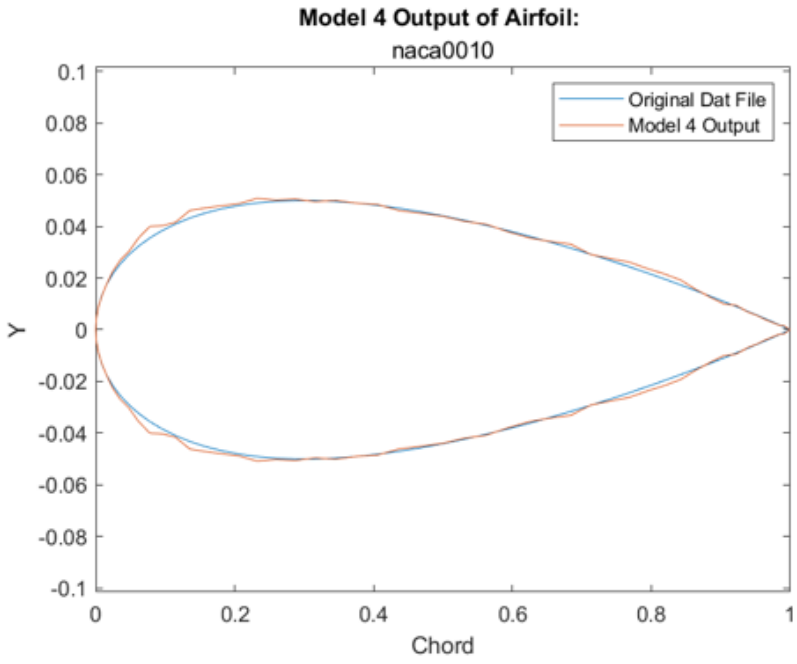
visualizes the network MSE values into a bar graph and although there is one outlier which is a high $1.20x10^{-3}$, virtually all values stay below $2.00x10^{-4}$ and many are far below that threshold. A full set of this data can be seen in Appendix A, Table 17 to be further studied.

When reinputting the benchmark airfoils which were used to train the model back into Model 4, the output had generally low error. These MSE values are some of the lowest that have been seen of any of the previous networks. The values are further improved by the filter's smoothing. In terms of predicting the airfoil geometry accurately, this is the most accurate and can be visualized in Figure 38 which plots the unfiltered and filtered Model 4 output.

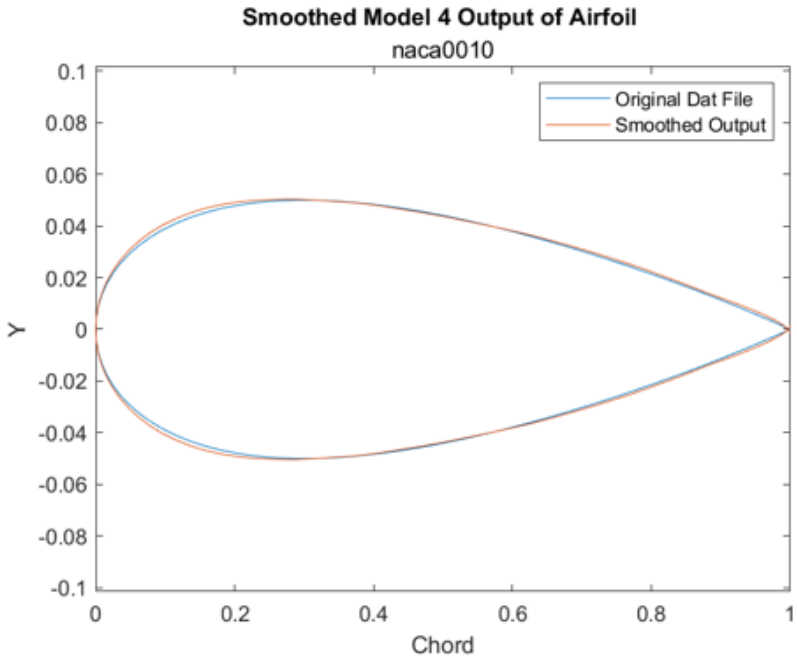
This proves that Bayesian regularization can generalize well on this training data and can provide an accurate prediction for airfoil geometry from coefficient of pressure distribution. The next section will show what prediction the model will give when inputting data which the model has not previously encountered.

Table 12: Subset of raw and smoothed Airfoil MSE for Model 4

Airfoil	Raw	Smoothed
naca0006	5.96E-06	5.96E-06
naca0008	3.03E-06	2.65E-06
naca0010	1.54E-06	1.12E-06
naca0012	1.09E-06	4.47E-07
naca0015	4.06E-06	3.43E-06
naca0018	1.30E-05	1.10E-05
naca0021	2.10E-05	2.10E-05
naca0024	3.03E-05	3.03E-05



(a)



(b)

Figure 38: a) Raw Model 4 output of airfoil NACA0010 b) Filtered Model 4 output of airfoil NACA0010

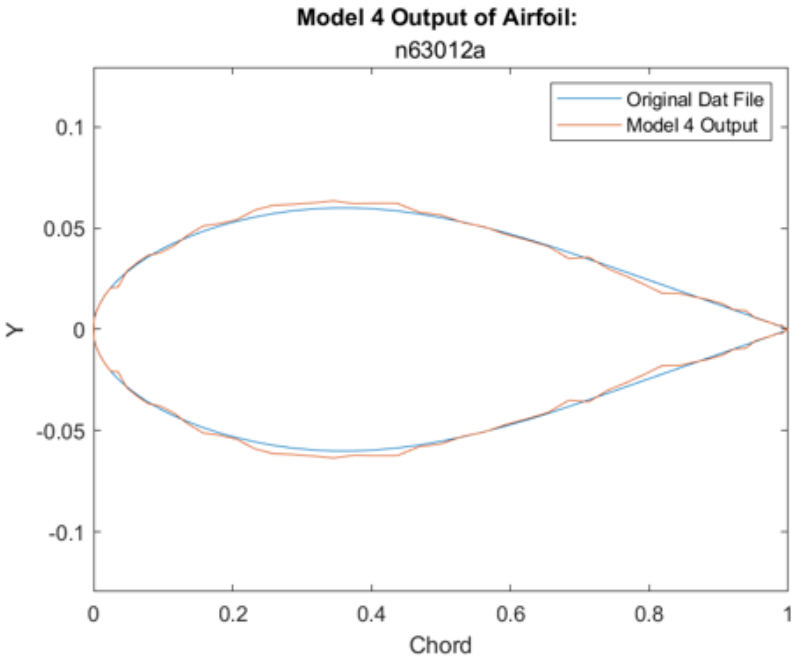
Testing

The same 10 airfoils which were used for benchmarking the previous models were used and the MSE of the predicted airfoil geometry to the original geometry is shown in Table 13 which contains the raw and smoothed output of Model 4. Some of these values are also lower than the previous models' MSE values. Some are higher or similar to the previous model which may indicate that these two models are near the optimal values that this machine learning method with this dataset can produce.

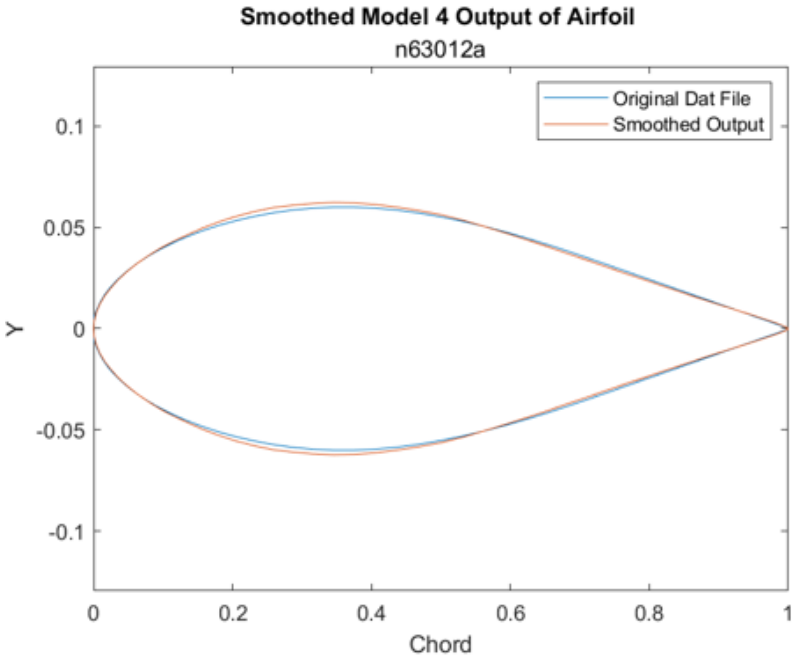
Table 13: Raw and smoothed MSE values for testing Model 4

Airfoil	Raw	Smoothed
n63012a	3.33E-06	1.36E-06
n63015a	1.28E-05	7.35E-06
goe459	1.73E-05	3.43E-06
goe460	4.34E-05	1.34E-05
goe775	3.78E-05	1.06E-05
naca001234	2.41E-05	1.54E-05
nacam3	1.22E-05	3.58E-06
oaf139	2.11E-05	1.59E-05
raf27	1.14E-05	3.88E-06
raf30	1.54E-05	5.34E-06

The output is plotted in Figure 39 which shows the unfiltered and filtered output of airfoil NACA63012a. Visually, the airfoil is very similar to the original airfoil and shows that this method is valid in producing an airfoil geometry from the coefficient of pressure distribution. While the error decrease is marginal when compared to the previous model, this model may be able to generalize better when encountering datasets which were not used to train the model.



(a)



(b)

Figure 39: a) Raw Model 4 output of airfoil NACA63012a b) Filtered Model 4 output of airfoil NACA63012a

Discussion

The models which were created in this study were able to predict airfoil geometry from coefficient of pressure distribution using multiple methods with varying degrees of accuracy. Model 1 and Model 2 were similar in approach and in accuracy, which acted more as a proof of concept that the coefficient of pressure distribution at a specific x-location can be directly related to the y-coordinate at that location for a symmetric airfoil. Model 3 utilized more input features to define the coefficient of pressure distribution and improved the accuracy between the previous models. Model 4 marginally improved the accuracy of Model 3 by utilizing Bayesian regularization which should help in generalizing the model for unknown data. As far as reinputting the airfoils utilized in the training process into the completed model, the MSE of the smoothed output is shown in Table 14.

Table 14: Smoothed output of training airfoils

Airfoil	Model 1	Model 2	Model 3	Model 4
naca0006	1.82E-04	2.80E-04	6.53E-06	4.93E-06
naca0008	3.07E-04	3.86E-04	2.64E-06	2.31E-06
naca0010	4.86E-04	5.18E-04	7.73E-07	9.25E-07
naca0012	7.08E-04	6.71E-04	7.92E-07	4.47E-07
naca0015	1.10E-03	9.30E-04	6.29E-06	3.41E-06
naca0018	1.56E-03	1.22E-03	1.86E-05	1.07E-05
naca0021	2.18E-03	1.57E-03	3.06E-05	1.31E-05
naca0024	3.02E-03	2.07E-03	3.19E-05	1.03E-05

For the benchmark which was created to properly test the networks with never-before-seen data, the MSE improved similarly. The unsmoothed data of each network increased and consequently the smoothed data which is shown in Table 15 improved as well. This shows the ability of feedforward network and a highly featured input to predict the airfoil geometry from the coefficient of pressure distribution of a relatively small dataset of symmetric airfoils.

Table 15: Smoothed output of the 10 benchmark airfoils

Airfoil	Model 1	Model 2	Model 3	Model 4
n63012a	6.00E-04	7.55E-04	1.20E-06	1.36E-06
n63015a	9.50E-04	1.01E-03	4.74E-06	7.35E-06
goe459	6.96E-04	7.93E-04	3.48E-06	3.43E-06
goe460	1.85E-03	1.66E-03	7.67E-05	1.34E-05
goe775	2.17E-03	1.54E-03	1.05E-04	1.06E-05
naca001234	3.30E-04	8.45E-04	4.32E-06	1.54E-05
nacam3	5.66E-04	7.04E-04	4.52E-06	3.58E-06
oaf139	1.71E-03	5.68E-04	1.78E-05	1.59E-05
raf27	4.50E-04	5.12E-04	3.21E-06	3.88E-06
raf30	8.19E-04	7.40E-04	3.47E-06	5.34E-06

CHAPTER V

CONCLUSION AND FUTURE WORK

The purpose of this study was to create a machine learning model which would predict airfoil geometry from a given coefficient of pressure distribution. What this study tried to accomplish which was unique is find a direct connection between the coefficient of pressure at various angles of attack at a specific location and the airfoil geometry or y-coordinate at that point. 50 neural networks were used to train each model and each model was able to predict the airfoil geometry with varying degrees of success.

The procedure to create these models included data acquisition which involved downloading the .dat files of symmetric airfoils from the UIUC database where 132 airfoils were acquired. The airfoils were normalized by being evaluated on the same x-coordinates for every shape. This geometry was then input into a panel method script which found the coefficient of pressure distribution across multiple angles of attack from -6° to +30° in two-degree increments. This data was then consolidated and allocated into their respective datasets to train the 50 individual networks for each model. Once the networks within the models were trained, a Savitzky – Golay filter was used to smooth the output of the model to create a more realistic airfoil shape and improve the MSE of the output.

Model 1 and Model 2 utilized the upper and the lower coefficient of pressure distribution, respectively. Both had 19 input features as well as one output feature and the networks consisted of 15 hidden sigmoid neurons and 1 linear output neuron. Finally, each utilized Levenberg – Marquardt backpropagation. For training, the training, validation, and testing datasets included 70%, 15%, and 15% of the training data, respectively. They both were able to predict the airfoil geometry with adequate test MSE during the training process. The 10 airfoils which were used as a benchmark also returned similar MSE values which showed a fair amount of generalization.

Model 3 and Model 4 both used the upper and lower coefficient of pressure distributions together. The 50 networks which were trained in each model had 38 input features with 1 output feature and had 20 hidden sigmoid neurons and 1 linear output neuron. Model 3 utilized Levenberg – Marquardt backpropagation while Model 4 employed Bayesian regularization in the backpropagation which modifies the performance function and allows the network to generalize better. Model 3 split the data into training, validation and testing in the same way Model 1 and Model 2 did which was a 70/15/15 split, respectively. Model 4 allocated the training, validation and testing datasets into 85%, 0%, and 15% of the data since validation is not needed to change the biases because the Bayesian regularization performance function already does this. Both had similar MSE values when testing airfoils which were used to train the networks and on the 10 airfoils which were chosen to properly test the models. The use of the filter also helped to lower MSE and give a realistic shape of an airfoil. The next couple of sections will go over the results from the training and testing processes for the models and discusses the implications from the findings as well as discusses potential future work into this topic.

Prediction Results

The training data from the neural networks showed that there were low values for testing MSE. These values can indicate whether the models are overfit when comparing with the training MSE and can also be further evaluated by testing using the 10 unencountered airfoils. A comparison between the training and testing MSEs can be found in Appendix A, Table 17. The training MSE is sometimes lower than the testing MSE for the different models and networks. To improve the models, it is essential that the testing MSE is lower than the training MSE if this approach is used again. The MSE values of the networks within Model 1 ranged from $1.7x10^{-7}$ to $1.15x10^{\wedge-4}$ for training performance and from $1.45x10^{-7}$ to $4.58x10^{-4}$ for testing performance. When inputting coefficient of pressure data of airfoils used to train the model back into Model 1, the MSE for the entire airfoil ranged from $7.50x10^{-5}$ to $7.57x10^{-3}$. When inputting the 10 airfoils which had not been encountered by the model, the MSE values for the entire airfoil geometry ranged from $3.30x10^{-4}$ to $2.17x10^{-3}$. For Model 2 the network training performance MSE values ranged from $1.83x10^{-7}$ to $1.35x10^{-4}$ and the testing performance MSE values ranged from $1.57x10^{-7}$ to $2.93x10^{-4}$. When inputting the airfoils which were used to train the networks, the MSE between the original airfoils and the predicted airfoils ranged between $1.17x10^{-4}$ to $6.29x10^{-3}$ and the 10 unencountered airfoils ranged from $5.12x10^{-4}$ to $1.66x10^{-3}$. Model 3's training performance MSE ranged from $2.74x10^{-8}$ to $1.02x10^{-4}$ and the testing performance ranged from $5.40x10^{-7}$ to $1.35x10^{-3}$. The airfoil MSE for trained airfoils ranged from $3.82x10^{-7}$ to $8.33x10^{-4}$ and the 10 unencountered airfoils ranged from $1.20x10^{-6}$ to 1.05^{-4}. Finally, Model 4's training performance MSE ranged from $1.63x10^{-8}$ to $3.20x10^{-5}$ and the testing performance ranged from $3.38x10^{-8}$ to $1.21x10^{-3}$. The trained airfoil MSE values ranged from $1.93x10^{-7}$ to $5.66x10^{-4}$ and the unencountered airfoils MSE ranged from $1.36x10^{-6}$ to $1.59x10^{-5}$.

All these values were the smoothed outputs for the airfoil MSEs and can be seen in Appendix A, Table 16 for easier reading but at this level, generally the MSE was lowered after each model. While the MSE values of the trained airfoils is usually lower, the unencountered airfoils had similar MSE and for Model 3 and Model 4, the MSE values were starting to converge which can indicate the lowest values possible with this method and size of dataset.

One aspect of this study which may be a detriment, would be the size of the dataset. This study only utilized 89 total airfoils for training the networks which comprised the training, validation, and testing datasets. Testing using airfoils which the networks had previously not seen only included 10 airfoils. For optimization methods that utilize neural networks or machine learning, studies usually include data from hundreds to thousands of airfoils so that the model can generalize well and reach an adequate error minimum. Only symmetric airfoils were used for this study which is why the dataset was so small, but if more were utilized, the models may have been better trained. Regardless of this fact, the MSE of the outputted airfoils is very low considering the constraint to symmetric airfoils and the study has proven at least some connection between the coefficient of pressure and the airfoil shape at a given point. This approach may be used and will probably perform well when containing more data such as unsymmetric or cambered airfoils. Finally, since the output of these models was also filtered through a signal filter, it smooths the shape whether or not those shape characteristics may aid in contributing to the correct coefficient of pressure distribution. One case which should be accounted for, is when a coefficient of pressure distribution is given and the raw model output contains a bump or tubercle which may allow the shape to reach the given coefficient of pressure distribution. The output would be filtered and smoothed which may smooth over the useful bump in the shape which may alter the optimal shape.

Future Work

Since the connection between the coefficient of pressure at a given x-location and the airfoil shape at that location has been found with a small dataset, this approach can be extended to cambered airfoils. Another approach which would be possible when containing a more diverse dataset, is including more input features such as Mach number or Reynold's number or some other flow parameter which would aid an airfoil or wing designer in the design process. To circumvent using a signal filter to smooth the raw output of the model, a larger dataset may also aid in contributing to a smoother shape and including beneficial bumps or tubercles. Another option would be to include the coefficient of pressure distribution from the two adjacent x-locations of a single point and to predict those three y-coordinates. The final values for all of these y-coordinate values can be averaged and this may help in smoothing the airfoil shape and simulate some form of communication between models. Many other approaches exist which may be beneficial to symmetric airfoil design, including GAN and CNN based machine learning models. These approaches would be a completely different direction in terms of approach but have been seeing a wide acceptance and validation with recent studies.

REFERENCES

Achour, G., Sung, W. J., Pinon-Fischer, O. J., & Mavris, D. N. (2020). Development of a conditional generative adversarial network for airfoil shape optimization. *AIAA Scitech 2020 Forum*.

AirfoilTools.com. *Airfoil database search (symmetrical)*. Symmetrical Airfoil database search. (2022). Retrieved December 8, 2021, from http://airfoiltools.com/search/index?m%5BmaxCamber%5D=0&m%5Bsort%5D=5

Akram, M. T., & Kim, M.-H. (2021). CFD analysis and shape optimization of airfoils using class shape transformation and genetic algorithm—part I. *Applied Sciences*, 11(9), 3791.

Anderson, J. D. (2010). *Fundamentals of aerodynamics* (5th ed.). McGraw-Hill.

Arias-Montano, A., Coello, C. A. C., & Mezura-Montes, E. (2012). Multi-objective airfoil shape optimization using a multiple-surrogate approach. In *2012 IEEE Congress on Evolutionary Computation*, 1-8.

Bishop, C. M. (2005). *Neural Networks for Pattern Recognition*. Oxford University Press.

Chen, W., Chiu, K., & Fuge, M. (2019). Aerodynamic design optimization and shape exploration using generative adversarial networks. *AIAA Scitech 2019 Forum*.

Della Vecchia, P., Daniele, E., & D'Amato, E. (2014). An airfoil shape optimization technique coupling PARSEC parameterization and evolutionary algorithm. *Aerospace Science and Technology*, *32*(1), 103-110.

Dulikravich, G. (1995). Shape inverse design and optimization for three-dimensional aerodynamics. In *33rd Aerospace Sciences Meeting and Exhibit*, 695.

Foresee, F., & Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. *Proceedings of International Conference on Neural Networks*, *3*, 1930–1935.

Hicks, R. M., Murman, E. M., & Vanderplaats, G. N. (1974). *An assessment of airfoil design by numerical optimization* (No. A-5506).

Houghton, E. L., Carpenter, P. W., Valentine, D. T., & Collicott, S. H. (2013). *Aerodynamics for Engineering Students* (6th ed.). Butterworth-Heinemann.

Kennelly, Jr, R. (1983). Improved method for transonic airfoil design-by-optimization. In *Applied Aerodynamics Conference*, 1864.

Kharal, A., & Saleem, A. (2012). Neural networks based airfoil generation for a given C_p using Bezier-PARSEC parameterization. *Aerospace Science and Technology*, 23(1), 330-344.

MacKay, D. J. (1992). Bayesian interpolation. *Neural Computation, 4*, 415–447.

Mukesh, R., Lingadurai, K., Selvakumar, U. (2013). Airfoil shape optimization using non-traditional optimization technique and its validation. *Journal of King Saud University – Engineering Sciences,* 26(2), 191-197.

Orfandis, S. J. (2010). *Introduction to signal processing*. Prentice Hall.

Picton, P. (2000). *Neural networks*. Palgrave Macmillan.

Sekar, V., Zhang, M., Shu, C., & Khoo, B. C. (2019). Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal, 57*(3), 993-1003.

Sun, G., Sun, Y., & Wang, S. (2015). Artificial neural network based inverse design: Airfoils and wings. *Aerospace Science and Technology*, *42*, 415-428.

Vanderplaats, G. N. (1979). Efficient algorithm for numerical optimization. *Journal of Aircraft*, *16*(12), 842-847.

Yilmaz, E., & German, B. (2018). A deep learning approach to an airfoil inverse design problem. In *2018 multidisciplinary analysis and optimization conference*, 3420.

Yonekura, K., Miyamoto, N., & Suzuki, K. (2021). Inverse airfoil design method for generating varieties of smooth airfoils using conditional WGAN-gp. *arXiv preprint arXiv:2110.00212*.

Zhang, Y., Sung, W. J., & Mavris, D. N. (2018). Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference*, 1903.

Zhang, Y., Yan, C., & Chen, H. (2020). An inverse design method for airfoils based on pressure gradient distribution. *Energies*, *13*(13), 3400.

APPENDIX A

APPENDIX A

Table 16: Smoothed training airfoil MSE values for all models

Airfoil	Model 1	Model 2	Model 3	Model 4
ah85l120	2.96E-04	9.94E-04	5.58E-06	4.22E-06
b540ols	7.90E-04	2.33E-04	5.83E-06	5.61E-06
bqm34	5.03E-04	5.45E-04	1.02E-06	8.32E-07
e168	1.11E-03	4.89E-04	6.60E-06	4.61E-06
e169	1.47E-03	5.64E-04	9.40E-06	4.25E-06
e171	7.64E-04	7.24E-04	2.96E-06	3.26E-06
e297	4.78E-04	7.47E-04	6.95E-07	2.89E-05
e472	1.10E-03	3.84E-04	1.26E-05	7.41E-06
e473	1.92E-03	5.48E-04	2.39E-05	1.73E-05
e474	1.31E-03	5.96E-04	1.10E-05	5.13E-06
e475	1.25E-03	7.98E-04	1.50E-05	7.92E-06
e479	1.96E-03	6.77E-04	9.04E-06	1.86E-06
e520	1.09E-03	1.40E-03	3.50E-05	3.44E-05
e521	7.58E-04	1.24E-03	6.38E-06	1.38E-06
e836	4.81E-04	1.07E-03	1.34E-06	1.93E-07
e837	1.04E-03	1.33E-03	1.37E-06	1.10E-06
e838	1.07E-03	2.02E-03	1.29E-05	3.61E-06
e862	6.77E-03	2.15E-03	1.27E-04	1.91E-05
e863	7.57E-03	2.59E-03	1.95E-04	3.83E-05
eh0009	4.32E-04	4.71E-04	5.36E-06	3.91E-06
fx71089a	5.26E-04	4.09E-04	1.70E-06	9.24E-07
fx71120	9.34E-04	6.00E-04	3.76E-06	2.67E-06
fx711520	1.26E-03	1.12E-03	4.51E-06	6.82E-06
fx711525	1.31E-03	1.19E-03	5.24E-05	4.62E-05
fx711530	1.25E-03	9.63E-04	3.40E-06	2.90E-06
fx71l150	1.25E-03	9.63E-04	3.40E-06	2.90E-06
fx76100	5.32E-04	4.98E-04	1.35E-06	1.39E-06
fx76120	7.86E-04	6.32E-04	2.34E-06	1.92E-06
fx77080	3.59E-04	3.54E-04	1.04E-06	8.33E-07
fx79l100	5.88E-04	4.91E-04	4.96E-06	3.47E-06
fx79l120	8.40E-04	6.25E-04	3.05E-06	2.95E-06

Table 16, cont.

fxl142k	1.28E-03	6.60E-04	1.05E-05	1.08E-05
fxlv152	1.20E-03	9.70E-04	4.38E-06	5.28E-06
griffith30symsuction	1.70E-03	6.29E-03	8.33E-04	2.37E-05
hq010	4.80E-04	6.49E-04	4.97E-06	2.66E-06
hq07	2.32E-04	4.44E-04	1.02E-05	6.44E-06
hq09	3.15E-04	4.95E-04	7.52E-06	4.11E-06
ht05	1.42E-04	1.45E-04	3.29E-05	2.92E-05
ht08	1.81E-04	1.54E-04	9.17E-06	5.57E-06
ht12	2.18E-04	1.17E-04	9.50E-06	2.06E-06
j5012	5.57E-04	7.75E-04	7.01E-07	6.11E-07
joukowsk0009	6.37E-04	2.68E-04	1.77E-05	9.04E-05
joukowsk0015	1.69E-03	6.20E-04	5.59E-06	7.25E-06
joukowsk	1.08E-03	5.44E-04	4.58E-06	7.72E-06
ls013	6.13E-04	9.96E-04	5.64E-06	4.02E-06
lwk79100	7.46E-04	4.04E-04	5.76E-06	8.47E-06
lwk80080	1.94E-04	5.96E-04	7.32E-06	5.17E-06
lwk80100	2.98E-04	8.03E-04	5.49E-06	2.68E-06
lwk80120k25	4.34E-04	1.02E-03	5.24E-06	3.45E-06
lwk80150k25	6.69E-04	1.39E-03	1.30E-05	1.07E-05
n0009sm	3.88E-04	4.32E-04	3.82E-07	4.12E-07
n0011sc	2.80E-04	8.97E-04	1.21E-05	1.31E-05
n0012	7.10E-04	6.27E-04	3.28E-06	2.46E-06
naca0006	1.82E-04	2.80E-04	8.33E-06	5.96E-06
naca0008	3.07E-04	3.86E-04	2.71E-06	2.65E-06
naca0010	4.86E-04	5.18E-04	8.50E-07	1.12E-06
naca001264	3.65E-04	8.59E-04	7.07E-06	5.75E-06
naca0012	7.08E-04	6.71E-04	7.92E-07	4.47E-07
naca0012h	1.06E-03	5.49E-04	1.49E-04	5.66E-04
naca0015	1.10E-03	9.30E-04	6.36E-06	3.43E-06
naca0018	1.56E-03	1.22E-03	2.28E-05	1.10E-05
naca0021	2.18E-03	1.57E-03	3.94E-05	2.10E-05
naca0024	3.02E-03	2.07E-03	5.10E-05	3.03E-05
naca16009	7.99E-05	7.23E-04	1.75E-06	4.68E-07
naca633018	1.58E-03	1.36E-03	7.84E-06	5.83E-06
naca66-018	6.37E-04	2.02E-03	1.72E-05	1.41E-05
prandtldtip	5.50E-04	4.03E-04	1.00E-06	1.80E-06
rae100	6.60E-04	4.17E-04	1.48E-06	1.26E-06
rae101	5.45E-04	4.75E-04	3.70E-06	3.34E-06
rae102	4.34E-04	5.99E-04	3.17E-06	1.99E-06

Table 16, cont.

rae103	3.12E-04	8.26E-04	7.45E-06	6.14E-06
rae104	2.46E-04	8.04E-04	4.64E-06	2.66E-06
s1010	1.90E-04	3.54E-04	9.47E-06	5.83E-06
s1012	5.06E-04	1.05E-03	9.26E-06	5.70E-06
s1014	7.26E-04	1.75E-03	2.17E-05	2.89E-05
s1016	8.91E-04	1.83E-03	2.69E-05	2.61E-05
s1046	1.75E-03	9.74E-04	4.97E-06	1.29E-06
s1048	1.19E-03	7.15E-04	4.70E-06	2.26E-06
s8035	1.09E-03	8.24E-04	2.16E-06	1.87E-06
s9026	6.98E-04	3.24E-04	1.51E-06	1.72E-06
s9027	4.41E-04	3.15E-04	3.03E-06	2.57E-06
s9032	5.54E-04	3.64E-04	1.92E-06	2.11E-06
s9033	4.49E-04	2.31E-04	3.24E-06	2.46E-06
sc20010	3.65E-04	6.42E-04	3.87E-06	2.60E-06
sc20012	5.37E-04	8.37E-04	9.44E-06	8.38E-06
sd8020	5.41E-04	5.06E-04	7.60E-07	5.62E-07
ultimate	6.01E-04	7.45E-04	9.25E-06	5.96E-06
us1000root	2.16E-03	1.04E-03	1.86E-05	9.73E-06
ys900	7.50E-05	1.02E-03	4.58E-05	1.12E-05

Table 17: Individual network MSE values for training and testing datasets

	Model 1		Model 2		Model 3		Model 4	
Net-work	Train	Test	Train	Test	Train	Test	Train	Test
1	1.70E-07	1.98E-07	3.35E-07	1.45E-07	1.75E-07	5.88E-07	2.13E-07	3.43E-07
2	4.53E-07	3.54E-07	1.32E-06	1.36E-05	1.66E-06	9.89E-07	8.64E-08	3.38E-08
3	1.79E-06	7.21E-07	6.60E-07	7.50E-05	1.46E-07	4.52E-06	8.02E-08	1.07E-05
4	3.07E-06	1.03E-06	5.05E-06	1.06E-05	2.35E-06	6.95E-06	8.96E-07	1.21E-03
5	3.59E-06	1.48E-06	6.04E-06	1.44E-04	2.60E-06	4.81E-06	2.16E-06	4.61E-04
6	5.91E-06	2.68E-05	2.87E-06	1.87E-05	1.29E-06	9.75E-05	8.90E-07	4.47E-06
7	4.22E-06	2.43E-06	3.12E-06	7.03E-06	1.93E-06	1.27E-06	1.16E-06	7.12E-07
8	9.11E-06	3.29E-05	2.20E-06	1.71E-05	3.12E-06	3.94E-05	3.82E-07	4.26E-06
9	1.21E-05	5.76E-06	1.50E-05	2.04E-05	6.04E-06	7.02E-06	9.54E-07	7.63E-06
10	1.28E-05	2.39E-05	1.85E-05	4.05E-05	2.68E-06	1.57E-05	2.00E-05	1.03E-05
11	1.98E-05	3.10E-05	1.55E-05	6.64E-06	4.96E-06	1.97E-05	3.31E-06	8.21E-06
12	3.36E-05	1.87E-04	1.54E-05	1.21E-05	3.42E-06	6.24E-05	4.49E-06	1.07E-05
13	2.66E-05	9.37E-06	2.62E-05	6.08E-05	1.17E-05	1.18E-04	4.89E-06	3.31E-04
14	5.31E-05	6.30E-05	3.93E-05	4.19E-05	5.16E-06	1.35E-03	6.81E-06	2.70E-05
15	4.27E-05	7.75E-05	4.98E-05	6.20E-05	4.25E-05	1.42E-05	3.20E-05	3.56E-05
16	8.89E-05	9.50E-05	5.27E-05	1.22E-04	7.23E-06	3.96E-04	2.54E-05	3.11E-05
17	5.83E-05	2.93E-04	2.11E-05	1.07E-04	6.66E-05	1.50E-05	2.39E-05	1.18E-04
18	6.05E-05	1.90E-05	3.34E-05	5.31E-05	3.75E-05	1.07E-04	2.37E-06	4.77E-05
19	3.00E-05	3.56E-05	1.03E-04	6.28E-05	1.02E-04	5.50E-05	1.77E-05	3.42E-05
20	4.72E-05	2.00E-05	5.98E-05	7.95E-05	1.81E-05	1.84E-05	6.86E-06	4.44E-04
21	3.71E-05	1.15E-04	3.03E-05	5.36E-05	1.40E-05	2.03E-05	6.98E-06	1.04E-04
22	5.52E-05	2.09E-05	6.39E-05	1.20E-04	6.76E-05	4.01E-04	1.53E-06	1.91E-04
23	4.26E-05	9.62E-05	4.71E-05	7.20E-06	2.76E-05	2.93E-05	2.90E-06	1.07E-04
24	1.17E-05	2.45E-04	5.73E-05	2.35E-04	2.66E-05	2.96E-05	2.25E-06	1.19E-04
25	6.27E-05	2.06E-05	4.47E-05	1.20E-04	1.02E-05	1.43E-04	4.38E-06	8.54E-05
26	2.41E-05	2.81E-05	4.08E-05	8.73E-05	1.18E-05	6.38E-05	3.72E-06	1.16E-04
27	1.98E-05	5.36E-05	2.20E-05	3.49E-05	1.34E-05	1.23E-05	2.46E-06	5.99E-05
28	1.62E-05	1.93E-05	2.32E-05	4.26E-05	8.88E-06	3.22E-05	2.62E-05	9.32E-06
29	2.05E-05	3.94E-05	1.45E-05	1.56E-05	8.49E-06	2.58E-04	2.04E-05	1.44E-05
30	3.00E-05	5.52E-05	1.53E-05	9.54E-05	6.54E-06	1.42E-05	1.41E-05	9.36E-06
31	2.31E-05	2.68E-05	1.70E-05	3.05E-05	1.52E-05	2.16E-05	1.12E-05	7.17E-05
32	1.20E-05	2.39E-05	1.25E-05	1.31E-05	1.17E-05	9.60E-06	1.28E-05	9.62E-05
33	1.17E-05	1.52E-05	1.05E-05	9.08E-05	2.12E-05	2.02E-05	1.19E-05	1.85E-05
34	3.70E-05	2.86E-05	1.05E-05	1.16E-05	2.04E-05	1.77E-05	8.77E-06	1.92E-05
35	1.69E-05	2.37E-05	2.26E-05	1.37E-05	7.92E-06	3.76E-05	9.30E-06	1.61E-05
36	2.77E-05	2.48E-05	1.40E-05	2.46E-05	2.15E-05	5.13E-05	8.20E-06	3.53E-05

Table 17, cont.

37	2.93E-05	2.59E-05	1.80E-05	1.28E-05	7.19E-06	2.38E-05	1.12E-05	5.30E-05
38	4.85E-05	2.94E-05	1.71E-05	1.38E-04	1.37E-05	1.37E-05	9.29E-06	1.33E-05
39	3.55E-05	4.74E-05	3.06E-05	4.41E-05	1.09E-05	7.29E-06	9.76E-06	9.09E-05
40	2.96E-05	3.32E-05	4.35E-05	2.90E-05	1.35E-05	3.32E-05	1.23E-05	2.15E-05
41	4.66E-05	2.02E-04	2.58E-05	2.66E-05	1.18E-05	1.99E-05	1.30E-05	5.67E-05
42	9.19E-05	2.27E-04	6.14E-05	3.95E-05	1.07E-05	1.31E-05	9.47E-06	3.46E-05
43	1.01E-04	4.58E-04	3.81E-05	3.81E-05	1.78E-05	6.38E-06	4.86E-06	4.47E-05
44	1.15E-04	1.99E-04	0.000135	4.88E-05	3.06E-06	7.45E-05	3.43E-06	2.87E-05
45	3.06E-05	3.32E-05	9.18E-05	5.62E-05	5.10E-06	1.09E-05	2.23E-06	7.05E-05
46	2.38E-05	5.08E-05	3.81E-05	2.37E-05	5.59E-06	4.04E-06	1.31E-06	6.68E-05
47	7.44E-06	6.78E-06	2.02E-05	1.82E-05	1.13E-06	1.37E-06	9.11E-08	1.55E-06
48	3.59E-06	5.06E-06	6.61E-06	3.50E-06	2.93E-07	1.75E-06	7.99E-08	2.22E-06
49	6.69E-07	1.22E-06	1.48E-06	1.12E-06	2.74E-08	2.38E-06	3.95E-08	3.21E-07
50	3.38E-07	1.95E-07	1.83E-07	1.57E-07	2.02E-07	5.40E-07	1.63E-08	5.98E-08

APPENDIX B

APPENDIX B

Figures 40 through 44 will show the smoothed output of Model 1 on the upper left plot, Model 2 in the upper right plot, Model 3 in the lower left plot, and Model 4 in the lower right plot for the airfoils used for training. The original .dat file coordinates are shown in blue while the smoothed output is in red. Figures 45 through 54 show the smoothed Model outputs for the 10 previously unencountered airfoils. It follows the same convention as the previous figures with Model 1 in the top left corner, Model 2 in the top right corner, Model 3 in the lower left corner, and Model 4 in the lower right corner.

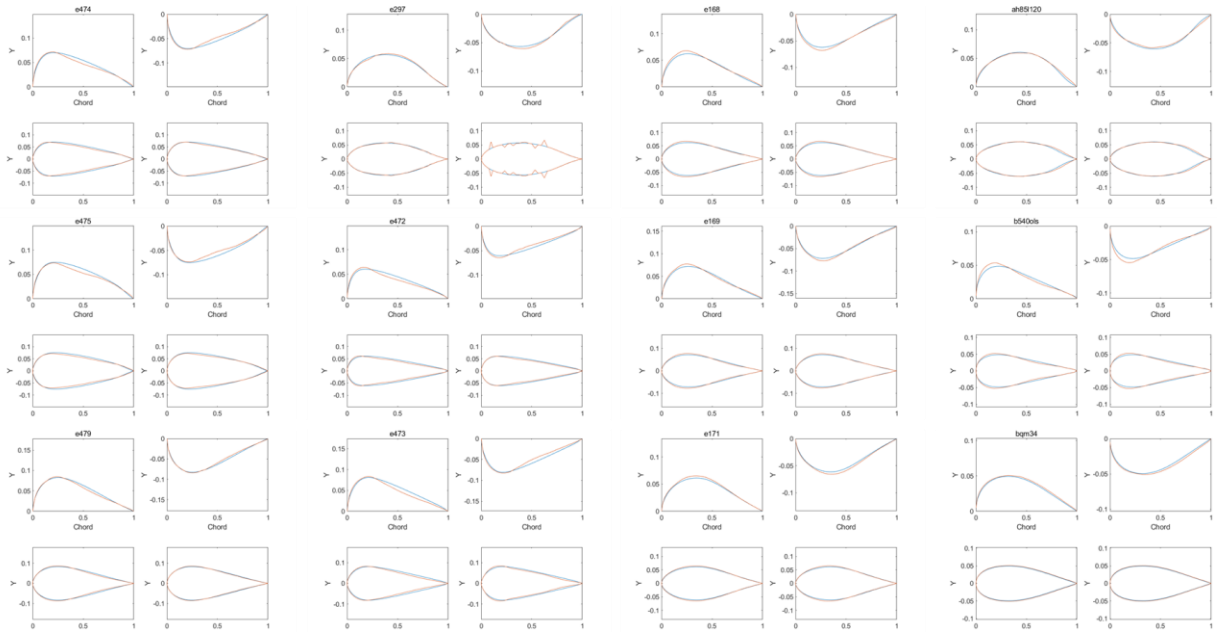


Figure 40: Smoothed training airfoil output

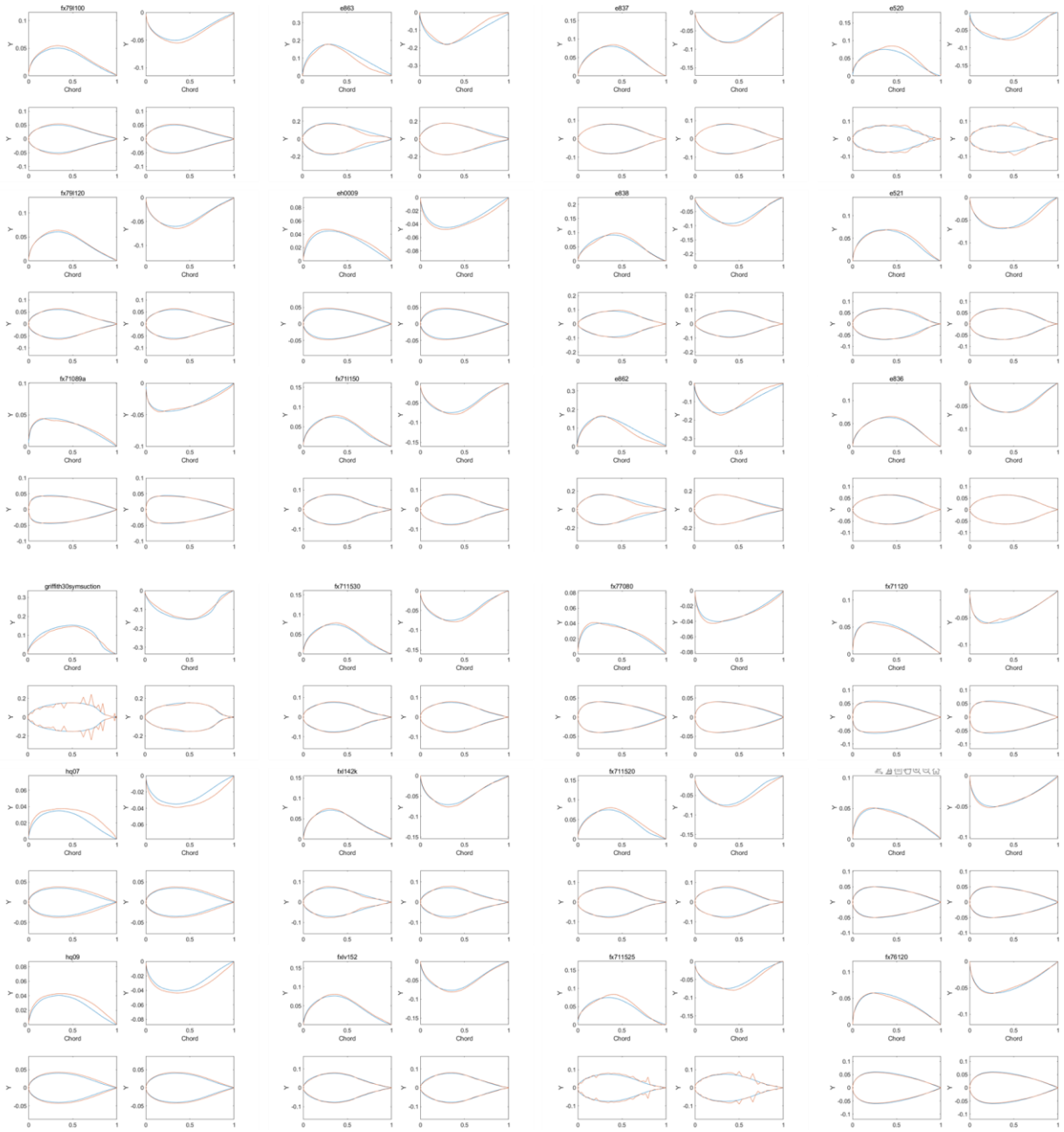


Figure 41: Smoothed training airfoil output

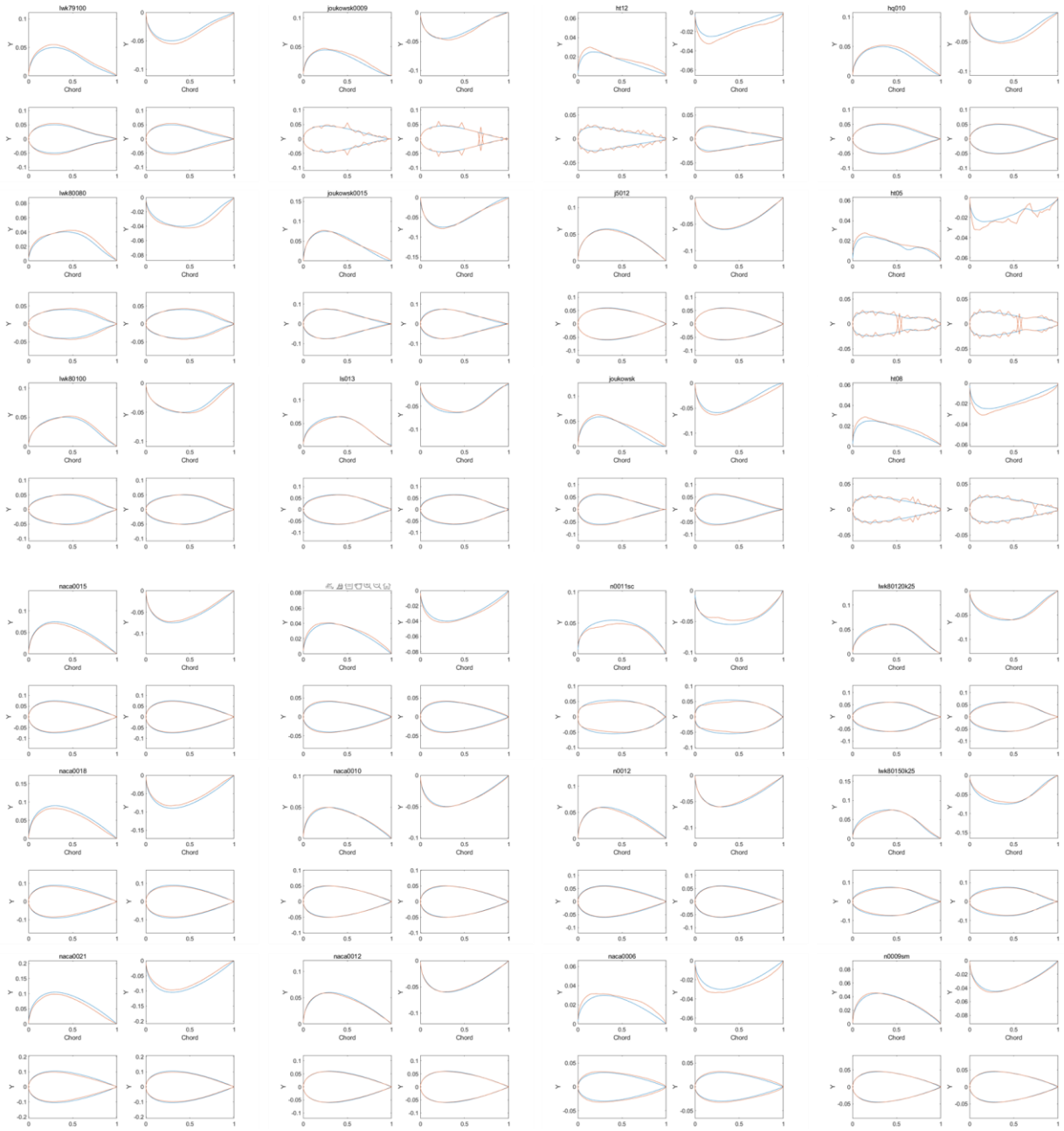


Figure 42: Smoothed training airfoil output

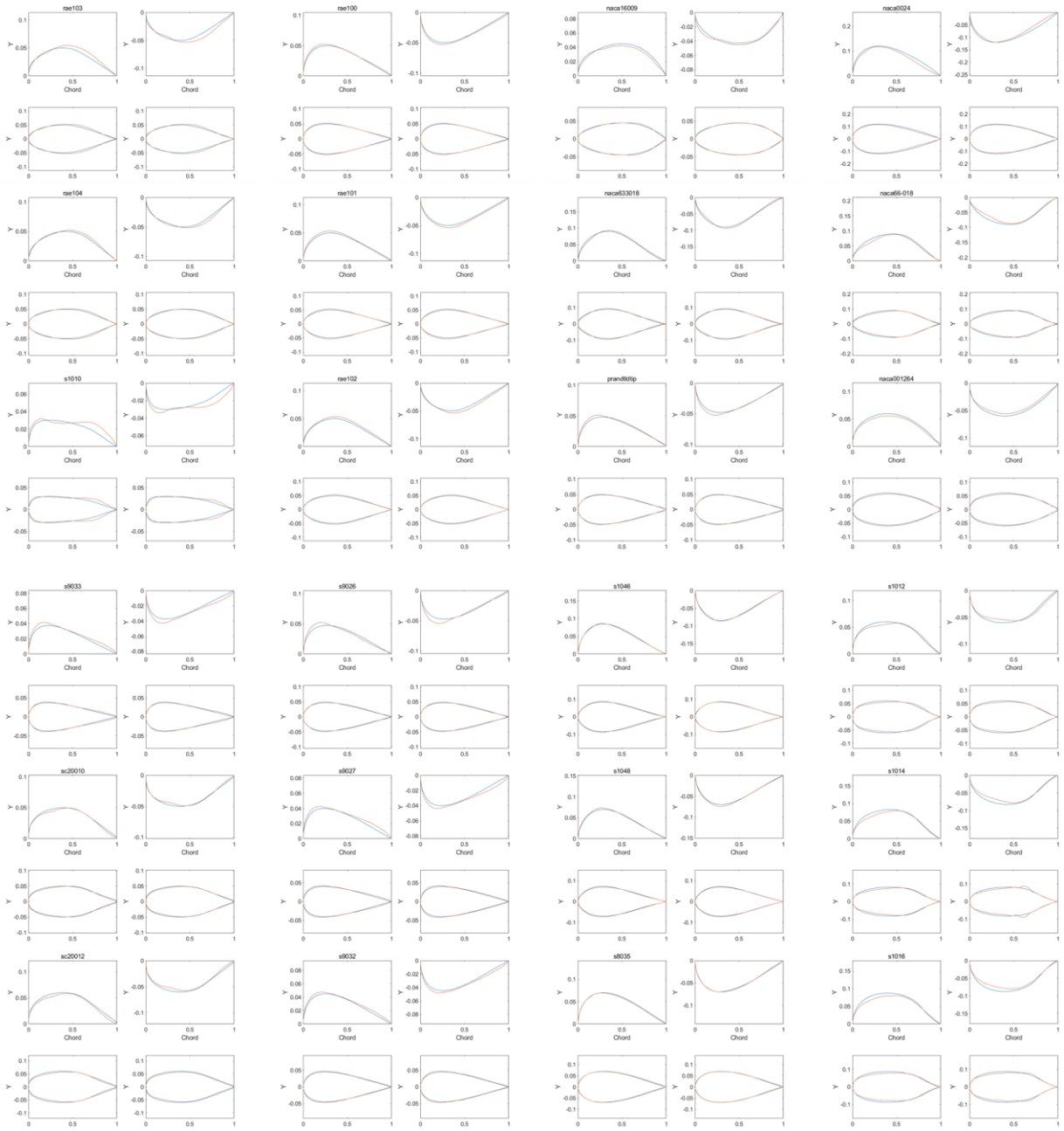


Figure 43: Smoothed training airfoil output

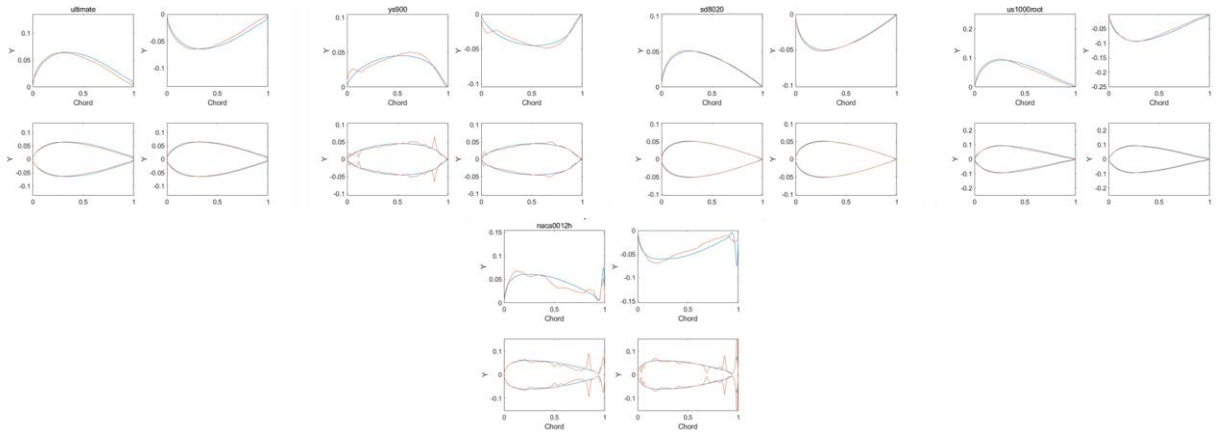


Figure 44: Smoothed training airfoil output

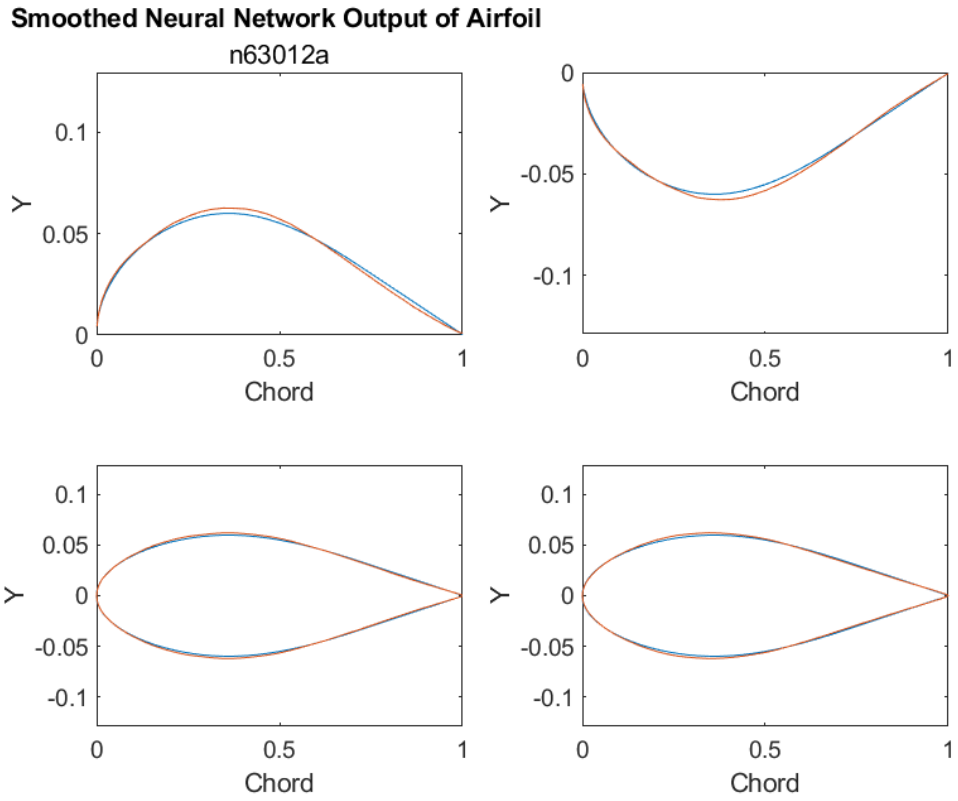


Figure 45: Smoothed output for airfoil NACA63012A

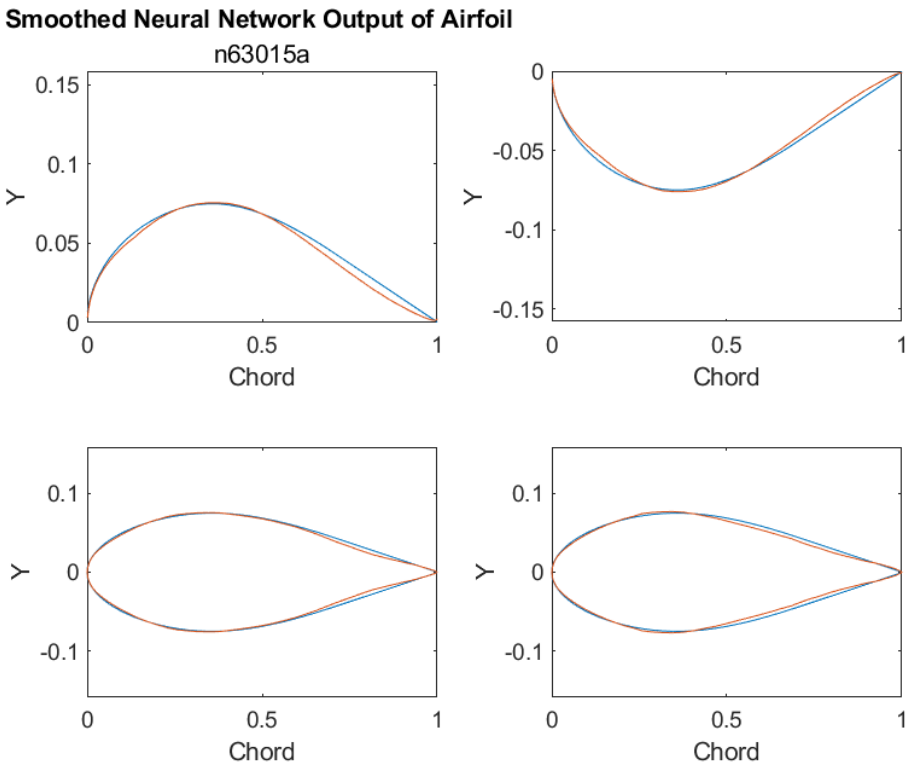


Figure 46: Smoothed output for airfoil NACA63015a

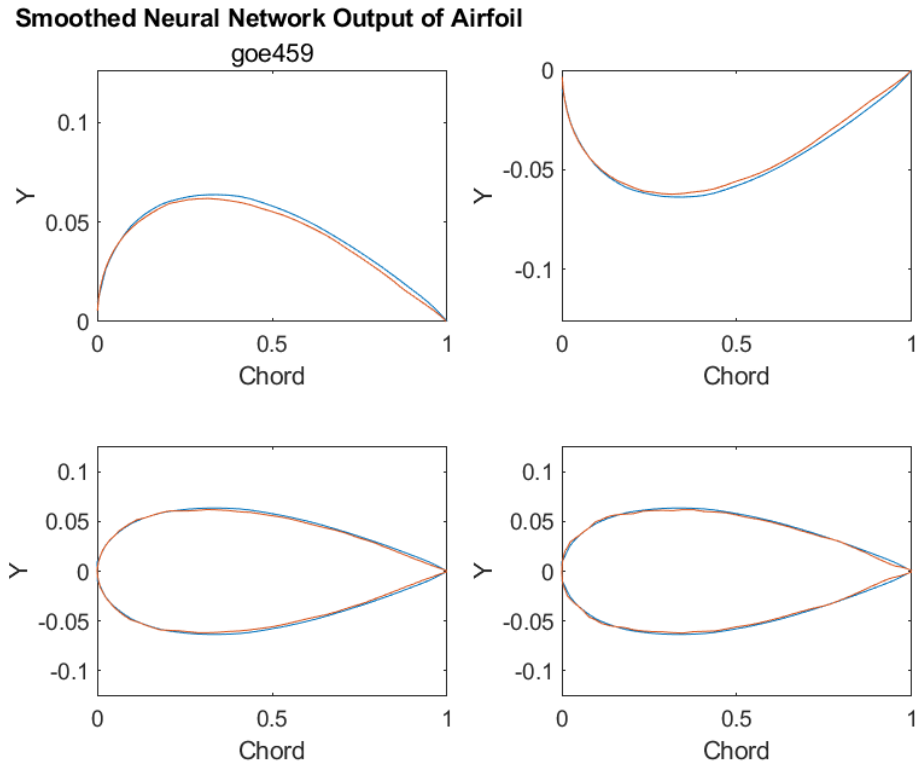


Figure 47: Smoothed output for airfoil GOE459

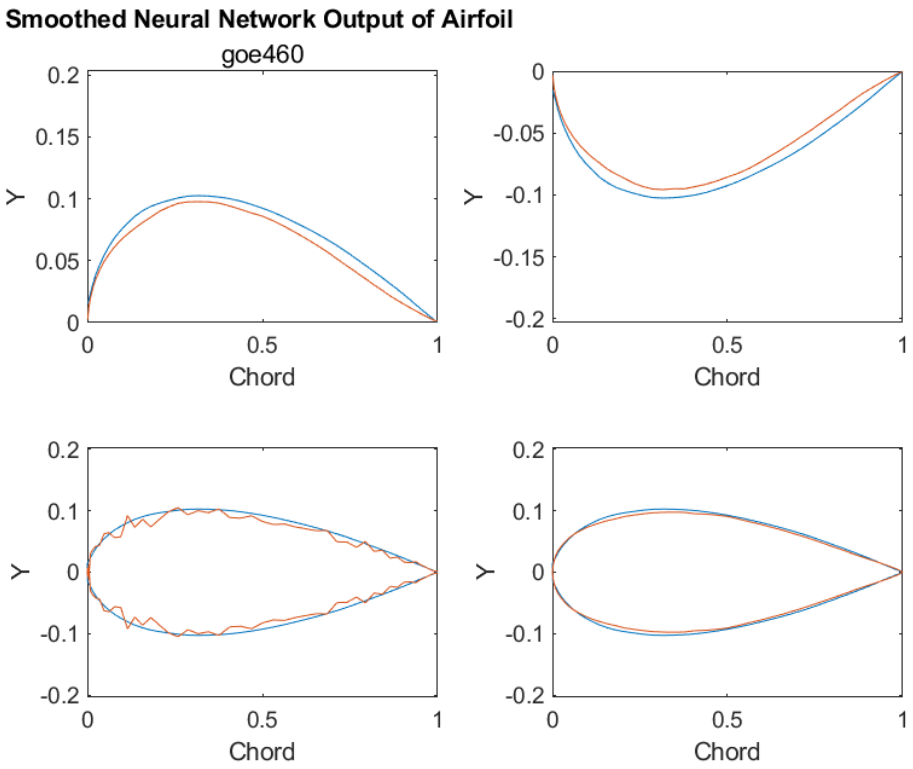


Figure 48: Smoothed output for airfoil GOE460

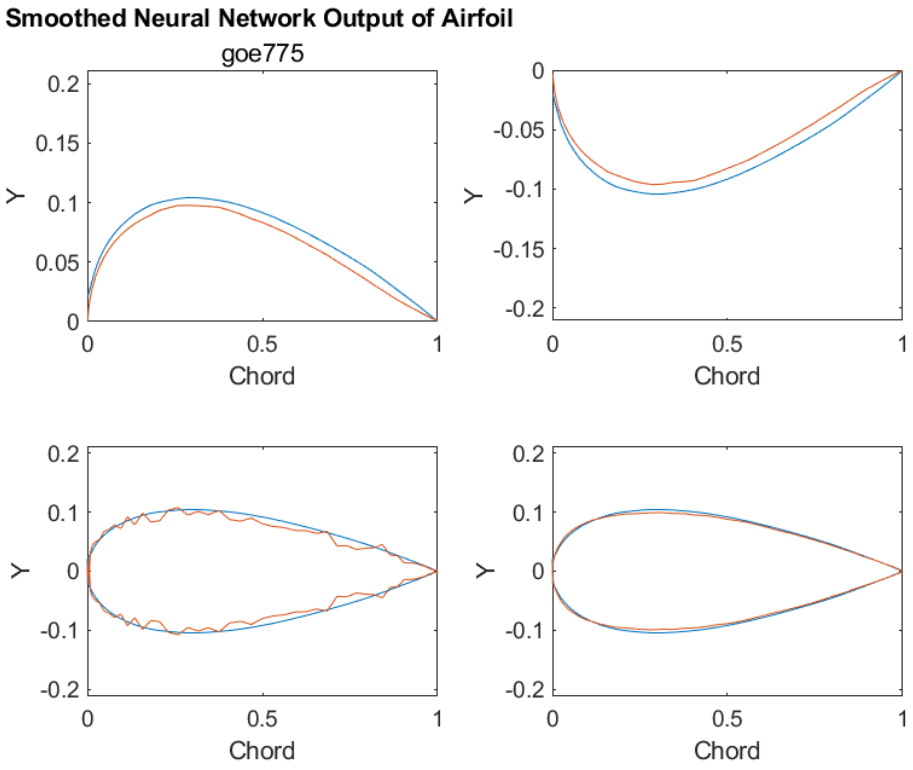


Figure 49: Smoothed output for airfoil GOE775

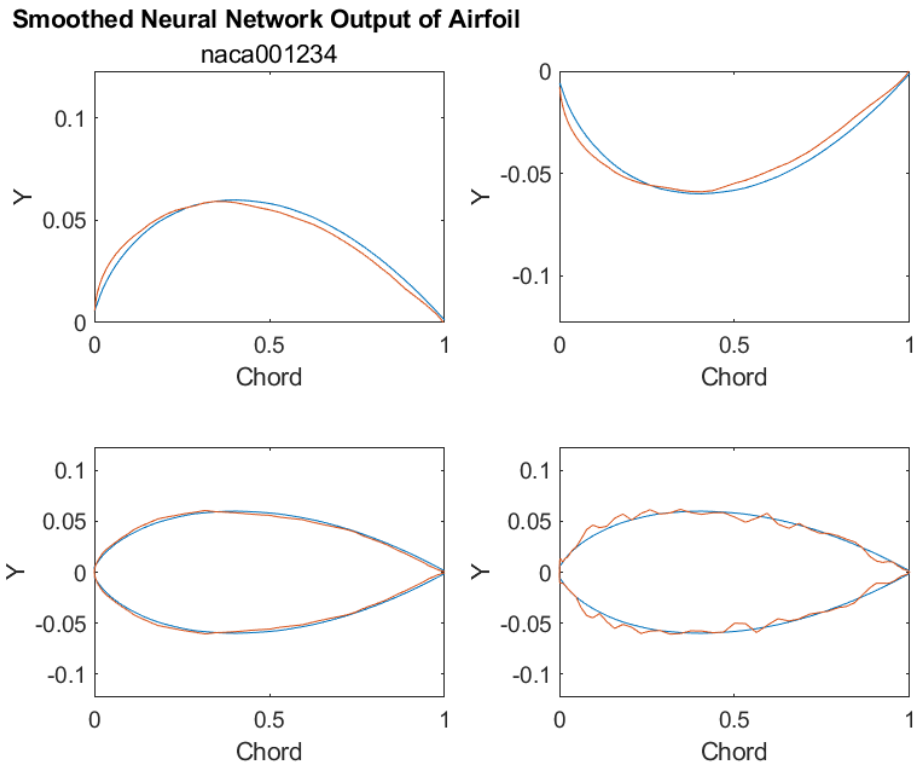


Figure 50: Smoothed output for airfoil NACA001234

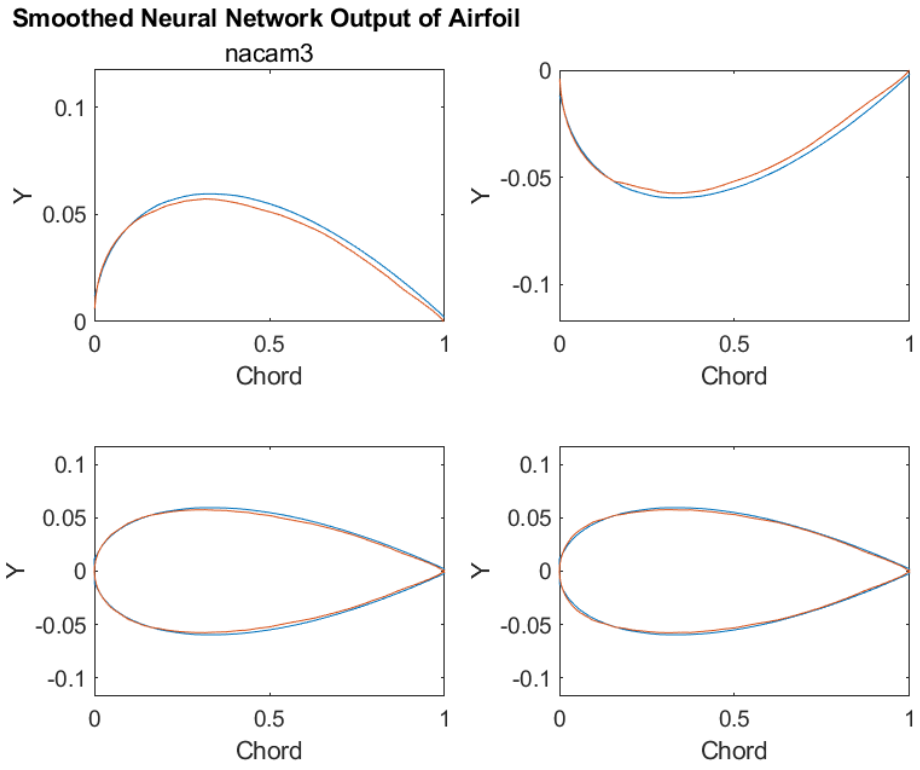


Figure 51: Smoothed output for airfoil NACAM3

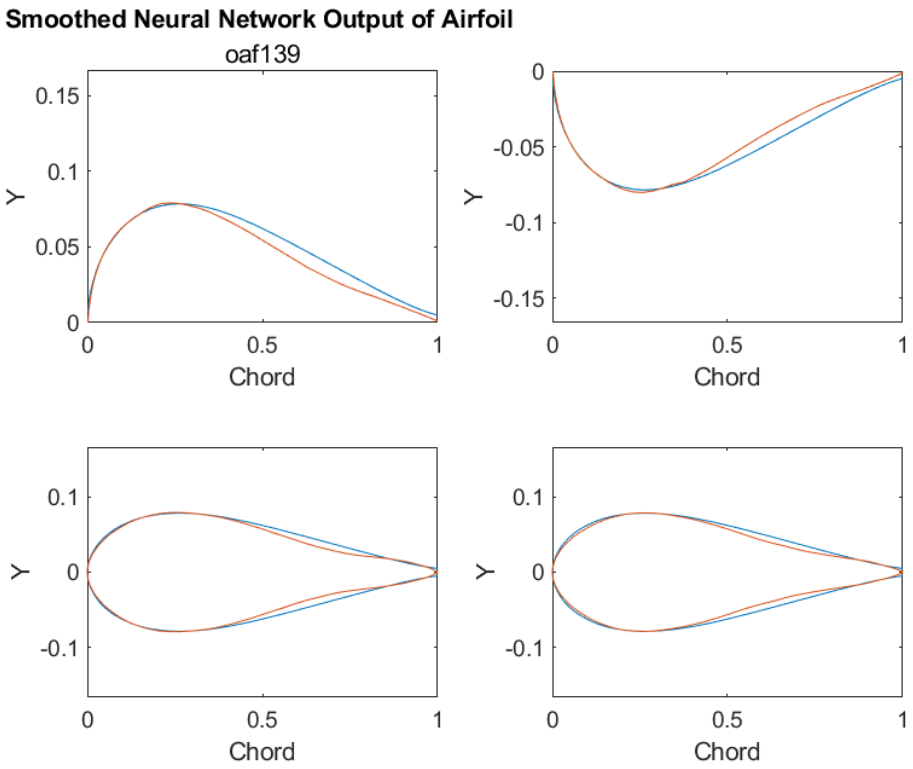


Figure 52: Smoothed output for airfoil OAF139

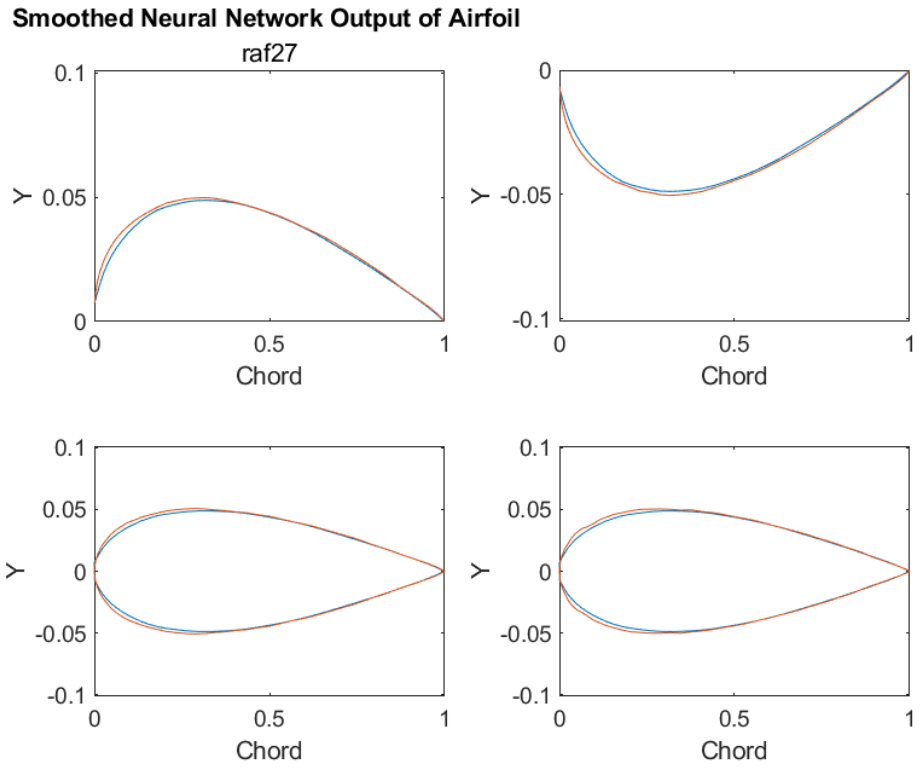


Figure 53: Smoothed output for airfoil RAF27

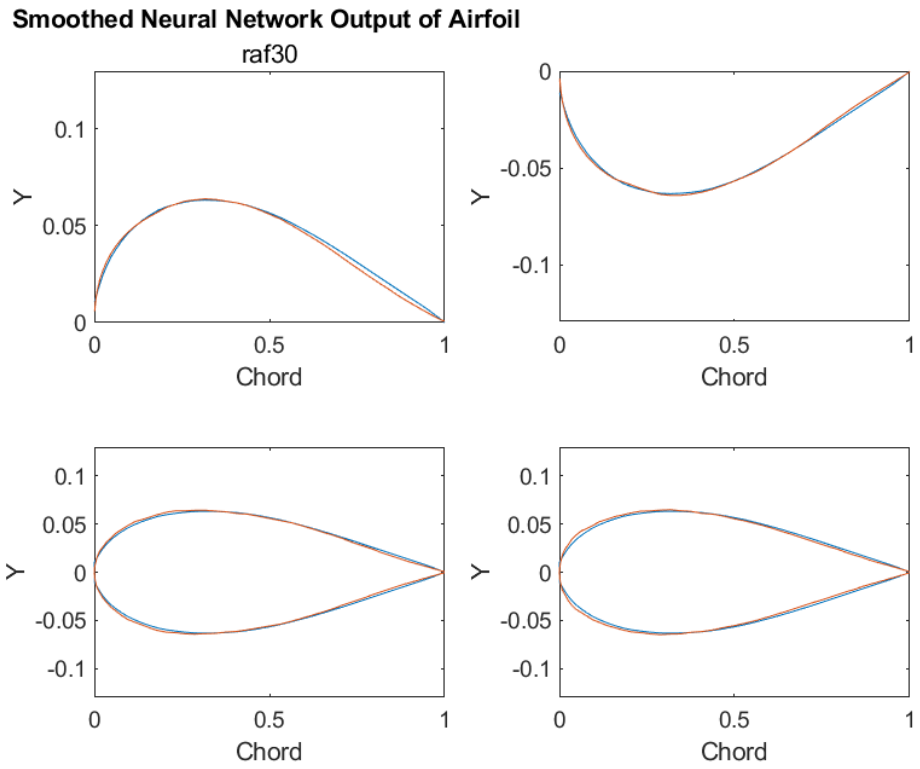


Figure 54: Smoothed output for airfoil RAF30

BIOGRAPHICAL SKETCH

Noe Martinez, Jr., was born and raised in Harlingen, Texas on May 27, 1997. He attended Harlingen High School South and graduated in 2015 while participating in engineering preparatory courses throughout his time there. In August 2015 he enrolled at Texas A&M Kingsville where he was involved in multiple student organizations such as the society of Automotive Engineers, the American Society of Mechanical Engineers, and the XA chapter where he participated in multiple extracurricular projects such as BAJA and the GreenPower USA electric vehicle competition. He completed his Bachelor of Science in Mechanical Engineering with minors in Aerospace Engineering and Mathematics in May of 2019. In June of 2020 he decided to pursue a Master of Science in Mechanical Engineering at the University of Texas Rio Grande Valley where he joined the Aerodynamics and Propulsion Laboratory as a graduate research assistant under the direction of Dr. Isaac Choutapalli. He spent the next two years working on coursework, his thesis, and helping to lead the Measurements and Instrumentation lab at UTRGV, which extended his knowledge and passion for engineering subjects and on May 14, 2022, he graduated with his Master of Science in Engineering in Mechanical Engineering.

Personal Email: noemar527@gmail.com