Master of Science Thesis

# Physics-Informed Neural Networks for Fluid Mechanics

**Dobbin Huang**

February 21, 2022

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

# Physics-Informed Neural Networks for Fluid Mechanics

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Dobbin Huang

February 21, 2022

Student number: 4031083
Project duration: 03-May-2021 – 10-Mar-2022

Thesis committee:
Dr. N.A.K. Doan, TU Delft, supervisor
Prof. R.P. Dwight, TU Delft, chair
Dr. Ir. E.-J. van Kampen, TU Delft, examiner

Faculty of Aerospace Engineering · Delft University of Technology

**Delft University of Technology**

# Abstract

Physics-informed machine learning is a novel approach to solving flow problems with physics-informed neural networks (PINNs), that combines physical knowledge and machine learning. This study aims to investigate the potential of the application of PINNs in fluid mechanics problems by solving two practical flow problems.

The first case considers the reconstruction of the full, time accurate flow fields with PINNs from partial data in the wake of an airfoil with periodic vortex shedding. The results have shown a decent success in flow reconstruction case, but has trouble maintaining a high accuracy at higher time values. This is likely due to the lack of a mechanism forcing a time marching approach to prevent information flowing in the opposite direction of the positive time axis.

The second case attempts to train PINNs on a steady flow problems with parametric NACA airfoils first to devise a strategy for training PINNs on a parameteric airfoil and show its feasibility. This is followed by using the found strategy to train PINNs for the more complex flow cases with PARSEC airfoils to use them as fast surrogate flow solvers, while still producing the full flow fields with a comparable accuracy to conventional CFD.

Overall, the flow fields inferred by the PINNs have shown a good qualitative match with the OpenFOAM validation data, even in the most complex case with a PARSEC airfoil, which makes them sufficiently accurate as a surrogate flow solver for a preliminary design optimization. These results show not only the technical feasibility of PINNs for fluid mechanics, but also a practical value with accelerating the computation time up to a factor of 3.7 when accounting for the training time and producing the OpenFOAM training data.

# Acknowledgments

In these unusual times of social distancing and remote working, the support from the people around me have been more important than ever. I would like to thank my supervisor, Dr. N.A.K. Doan, for granting me the opportunity to work on this project and guiding me during all stages. Finally, I am grateful for the unconditional support and encouragement from my parents and brother, as I could not have completed this work without them.

# Table of Contents

# List of Figures

# List of Tables

# Nomenclature

## Abbreviations

| | |
|---|---|
| BPA | Batch size per area; the number of sample points per unit area or length in 1 batch |
| BPA | Batches per epoch; the number of batches that a full data set (epoch) has been divided in to |
| LHS | Latin Hypercube Sampling |
| MAV | Micro Air Vehicle |
| PDE | Partial differential equation |
| PINN | Physics-informed neural network |
| SDF | Signed Distance Function |

## Greek Symbols

| | |
|---|---|
| $\alpha$ | Angle of attack |
| $\nu$ | Kinematic viscosity |
| $\rho$ | Density |
| $\tau$ | Wall shear stress |
| $\theta$ | Set of weights and biases of a neural network |

## Latin Symbols

| | |
|---|---|
| $\mathbf{b}_i$ | Trainable biases of the $i^{th}$ layer |
| $\mathbf{W}_i$ | Trainable weights of the $i^{th}$ layer |
| $\mathbf{x}_i$ | Input of the $i^{th}$ layer |
| $\phi_i$ | The $i^{th}$ layer of the network |
| $\mathcal{R}e$ | Reynolds number |
| $\sigma$ | Activation function |
| $c_a$ | Axial force coefficient |
| $c_d$ | Drag coefficient |
| $c_f$ | Skin friction coefficient |
| $c_l$ | Lift coefficient |
| $c_n$ | Normal force coefficient |

| | |
|---|---|
| $c_p$ | Pressure coefficient |
| $I$ | Objective function |
| $MSE$ | Total mean squared error loss of the PINN |
| $MSE_f$ | Mean squared error loss corresponding to the collocation points |
| $MSE_u$ | Mean squared error loss corresponding to the solution data |
| $N_u$ | Number of training samples |
| $p$ | Pressure |
| $t$ | Time |
| $u$ | Horizontal velocity component |
| $u_{pred}$ | The solution predicted by the PINN |
| $u_{true}$ | The true solution that $u_{pred}$ needs to match |
| $v$ | Vertical velocity component |
| $x$ | Horizontal spatial coordinate |
| $y$ | Vertical spatial coordinate |

# Chapter 1

# Introduction

The combination of deep learning and availability of big data in recent years have brought a revolution to solving problems in multiple fields and industries. Applications of deep learning can already be found in society, such as the digital assistant on a smartphone or self-driving electric cars. The fields of science and engineering have been less eager to adopt deep learning, because the massive amount of data that is required for training a deep neural network is often difficult and expensive for physical problems.

However, one of the recent developments in deep learning, physics-informed machine learning, appears to be suited for such problems. A physics-informed neural network, or PINN, can draw training data from the available physical knowledge on the problem of interest, while regular deep learning will require a large amount of labelled data on the solution to train a deep neural network to a sufficient degree to infer an accurate solution [22]. This thesis will focus on the application of PINNs in fluid mechanics and investigate the case of flow reconstruction of an fixed airfoil in transient flow and a parametric airfoil in steady flow in detail. The potential of PINNs in solving fluid mechanics problems will be explored with these two problems and the current capabilities and limits of PINNs are demonstrated to evaluate the merits of PINN use in fluid mechanics.

This report starts with a summary of the state-of-the-art of physics-informed machine learning and airfoil optimization, followed by the formulated research questions and objective. Next, the relevant methods and theories to this work will be explained in detail and the used software described. Following this, process and results of the airfoil flow reconstruction case using a PINN will be described. The next case of solving a parametric airfoil in steady flow with PINNs is divided in two parts: the first part will focus on verification and validation with a NACA 4-digit airfoil parameterization and the latter part will expand the degrees of freedom of the problem with a PARSEC airfoil representation and uses it for numerical design optimization. Finally, the report will end with the conclusions on the findings of the previous sections.

# Chapter 2

# Literature review

The literature review will start by summarizing the current state-of-the-art of PINNs. Based on this, the research goal an questions are formulated, followed by the relevant theory, methods and software tools used during this project.

## 2.1 State-of-the-art

Conventional computational fluid dynamics (CFD) has come a long way in solving flow problems governed by the Navier-Stokes equations. However, integrating CFD in the design process for problems with the complexity of industrial applications is often not seamless as meshes at that level are time consuming to generate and on top of that, every design change requires a new mesh and simulation [6]. In the recent years, great progress has been made in machine learning and enabled us to solve problems in a way we could not do before. A particular class of machine learning algorithms, deep learning, with the availability of big data has already transformed some fields in the industry. Examples are big data analytics [31], speech recognition and computer vision [10].

Deep neural networks, belonging to the family of deep learning architectures, can approximate any function if it is deep and large enough according to the universal approximation theorem [17]. In the field of fluid mechanics this can be used to directly solve non-linear problems, without simplifying the problem by making assumptions beforehand or applying linearization [36]. However, the predictive capabilities of the neural network is dependent on the quality and quantity of the available training data. Obtaining high quality data in the necessary amount to train a neural network by performing numerical and experimental simulations is expensive and is detrimental to the cost-effectiveness of using machine learning in practice. Another issue with a purely data-driven model is that it may lead to physically inconsistent predictions, while fitting the training data well.

A promising solution to this problem is the recently introduced physics-informed neural network (PINN) [36], which can use our knowledge on flow physics to aid the training of the neural network. This method creates an additional loss term by using the partial derivatives of the neural net with respect to the inputs to construct an additional network to enforce the governing Navier-Stokes equations on collocation points sampled from the solution domain. These partial derivatives are obtained at a minimal additional computational cost by using automatic differentiation [3]. Application of PINNs on benchmark problems yielded promising results for cases such as the Schrodinger equation and Korteweg-De Vries equation in [36] and flow past a cylinder in [6], but its value is mainly academic and has few practical uses so far.

Despite that, PINNs have the potential to bring machine learning to the fields of physics and engineering and become a new class of solver alongside the current methods as they show their merits in problems where conventional methods struggle. They are especially effective in solving ill-posed and inverse problems, which require data assimilation [22]. At this time, current numerical grid-based solvers do outperform PINNs in well-posed forward problems, when the training time is taken into account. However, a trained network is able to predict the solution in a mere fraction of a second. The reduction of the solution computation time to mere fractions of a second opens the door to many interesting applications that were not possible with conventional methods. An instantaneous solution feedback could be used in interactive design space exploration and numerical optimization. Another application is in real-time simulations where this near instantaneous feedback is necessary, such as real-time evaluation of aerodynamics for autonomous vehicles, robotic vision and control or digital twins.

Another advantage of PINNs is the ability to integrate the data from simulations and experiments with the information of the known physics seamlessly. This is useful when the problem has both imperfect data and model, and research has shown that even then meaningful results can be obtained [37]. As a PINN is a mesh-free solver, it can deal with complex domain geometries and even moving-domain problems without trouble [47]. PINNs also offer strong generalization when not a large amount of data is available by using the physics of the problem as constraints or for data augmentation. This allows the training of a PINN with only a small amount of data available. Examples of research on practical applications of PINNs can be found in different fields, such as quantum chemistry [34], material sciences [27] and geophysics [25].

The great potential PINNs show for solving problems also applies to problems in fluid mechanics, but examples of these in literature are mostly limited to academic benchmark problems. This work will take a step beyond that and investigates the application of PINNs in practical fluid mechanics problems and specifically, the problem of an airfoil design optimization and flow reconstruction. The airfoil design optimization problem will consider a parameterized airfoil geometry in a steady freestream flow, which will be optimized for a objective function that is dependent on the aerodynamic coefficients. The flow reconstruction problem attempts to reconstruct the full flow fields of a transient flow by training a PINN on measurement data that has a limited spatial coverage of the computational domain and using the trained PINN to infer the complete time-accurate flow fields.

Numerical airfoil design optimization is currently often done with the help of an adjoint reduced order model [26] or a kriging surrogate model [30] to reduce computation time. The flow reconstruction problem is commonly handled in a similar way, by linking sensor measurements to a reduced state using a reduced-order model [30]. These approaches reduce the full order model to a simpler model that is much cheaper to solve, but these simplifications lead to a loss of information making analyzing and interpreting the results more difficult.

If trained PINNs are used here instead as surrogate models, they will still be able to predict the full velocity and pressure fields, while also reducing the computation time of a function evaluation to less than a second. Training of a PINN may not be less computationally expensive currently than conventional CFD solvers when trying to obtain the solution for one specific problem with fixed parameters, but a big advantage is the speed of inferring multiple solutions using a trained network for a parametric problem. This means that when a PINN is trained to predict the solution of a flow problem with a parameterized geometry, any solution within the parameter ranges could be inferred in a fraction of a second without the need for generating new meshes for every new geometry. In the flow reconstruction case with a PINN, it will not be needed anymore to construct a reduced order model that is difficult to physically interpret. Instead the PINN will attempt to solve the flow reconstruction directly using the limited measurement data and boundary conditions.

## 2.2 Research Question, Aim/Objectives and Sub-goals

This work is an exploratory research in the application of PINNs in fluid mechanics. The maturity of PINN technology is still low, so there are still many questions left unanswered. The basic principles are observed and reported and some potential applications have been studied by academics. Examples of real-world use in relevant and practical problems are few. This work will investigate the possibilities of using PINNs to solve fluid mechanics problems and will try to demonstrate it with some specific examples and record it in detail.

The flow cases will be set up under the assumption of incompressible and laminar flow with Reynolds numbers of 5000 and below. We limit ourselves to these low Reynolds numbers because PINNs have so far been unable to deal with turbulent flows. As a first case, the flow reconstruction will be considered for an unsteady flow case of an airfoil at a high angle of attack with periodic vortex shedding in the wake. An attempt will be made to train a PINN to predict a time-accurate solution, which would reconstruct the flow in effect. Following this, parametric airfoil cases with NACA and PARSEC airfoil parameterization will be attempted in a steady flow, where the trained PINN will be able solve the full velocity and pressure fields of all the airfoil geometries within the parameter bounds. The practical use of this can be found in a fixed wing design of a Micro Air Vehicle (MAV). MAVs are a class of small miniaturized UAVs with a size of around 10 centimeters and fly at these very low Reynolds numbers where viscous forces dominate the flow. The maximum lift-drag ratio sees a large decrease for conventional airfoils below a Reynolds number of 10,000 [49] and the aerodynamic efficiency of similar sized insects has not been successfully replicated with man-made machines. PINNs could bring a more efficient way of exploring and designing airfoils

for such vehicles, while simultaneously allowing analysis of the full velocity and pressure fields to improve our understanding of the aerodynamics at these low Reynolds numbers.

### 2.2.1 Research Objectives

The main research objective of this thesis is:

> "To investigate the potential of physics-informed neural networks (PINN) as a viable method for solving relevant real-world fluid mechanics problems by performing PINN training and usage on the cases of a parameterized airfoil geometry in steady free stream flow and flow reconstruction of a fixed airfoil geometry in transient flow from limited measurements."

### 2.2.2 Research Questions

Several research question can be formulated and need to be solved in reaching the project goal. The research questions can be split between the ones related to the creation of the PINN and the usage.

- How is a PINN trained as a solver for practical flow problems and what are its limits?
  - How much data will be needed to train a PINN on the considered problems?
  - How does the distribution of the sampled data affect the quality of the PINN predictions?
  - What are the limitations on the type of flow problems that PINNs can solve?
- How well is a PINN able to function as a surrogate model for a flow problem?
  - What are the practical limitations of a PINN on the degrees of freedom and parameter ranges of a high dimensional parameterized problem?
  - How does it compare to conventional CFD methods in accuracy?
  - How effective is a PINN in predicting the transient solution in an unsteady problem?
  - How well can a trained PINN perform in an airfoil geometry design optimization in terms of speed?
  - What is an appropriate optimization method when using a PINN as the flow solver?

## 2.3 Theory and Methods

This section describes all the theory, methods and software that are used in this work to explore the applications of physics-informed neural networks in fluid mechanics.

### 2.3.1 Deep Learning

A feed-forward fully-connected neural network approximates the solution of a problem that is governed by a partial differential equation (PDE) and its boundary and initial conditions. To train the neural network, a loss function needs to be constructed in order to quantify how well the neural network approximates the solution of the PDE and constraints of the problem. Then, a mean-squared error loss function can be formulated as Equation 2.1

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u^i_{pred} - u^i_{true}|^2, \tag{2.1}$$

where $N_u$ is the number of training samples, $u_{true}$ is true solution given by the training data and $u_{pred}$ is the solution inferred by the neural network. Minimization of this loss function will produce a set of weights and biases $\theta$ that attempts to make the neural network an accurate predictor for the PDE solution. As the trained neural network is able to infer a solution for any input within the boundaries of the problem, it is a grid independent solution method of the PDE.

The structure of the neural network for a 2D flow problem is drawn in Figure 2.1. The network is a computational graph, where the inputs are propagated through the network in one direction till it reaches the output layer.

Here, the input layer has the spatial and temporal variables $x$, $y$ and $t$. The output layer has the velocity and pressure variables $u$, $v$, $p$. In between the input and output layer, are the hidden layers consisting of neurons with non-linear activation functions. Each neuron of the hidden layer in a fully-connected neural network receives an input from every neuron of the previous layer and gives and passes its out put to every neuron of the next layer. When the neurons of an hidden layer are grouped together, the equation for this is given by Equation 2.2.

$$\phi_i(\mathbf{x}_i) = \sigma(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i) \tag{2.2}$$

Here, $\phi_i$ is the $i^{th}$ layer of the network, $\mathbf{x}_i$ is the input of this layer, $\sigma$ is the activation function, $\mathbf{W}_i$ and $\mathbf{b}_i$ are the trainable weight and bias. The layers of the network with $n$ layers can be put together as a function composition as in Equation 2.3 to obtain the equation for the neural network [16].

$$u_{net}(\mathbf{x};\theta) = \mathbf{W}_n\{\phi_{n-1}\circ\phi_{n-2}\circ\cdots\circ\phi_1\circ\phi_E\}(\mathbf{x}) + \mathbf{b}_n, \tag{2.3}$$

where $\mathbf{x}$ in the network input, $\theta$ are the trainable parameters $\theta = \{\mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{W}_n, \mathbf{b}_n\}$ and $\phi_E$ is input encoding layer set to the identity function. Besides the trainable parameters of

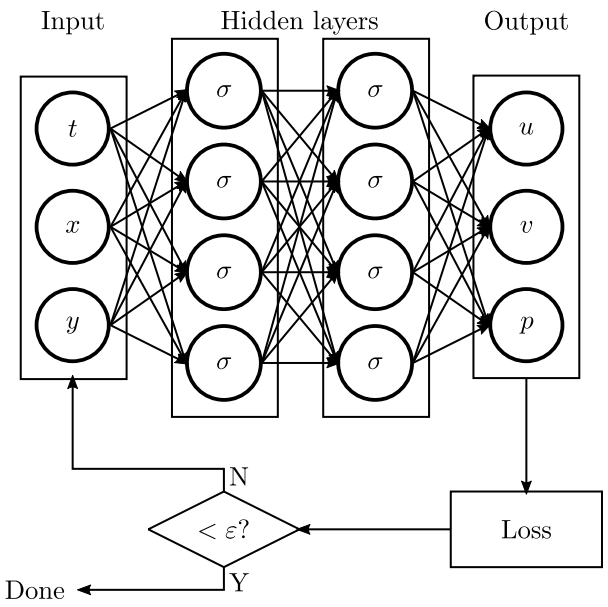



**Figure 2.1:** Structure of a fully-connected neural network.

the neural network, there are many other parameters that can be tuned outside of training and affects the network performance. The optimal values for these so-called hyperparameters are usually not easily learned with an optimization algorithm, but need manual tuning. Some important hyperparameters will be explained below.

The network size of a neural network decides the capacity of the network. A larger network will be able to approximate more complex systems and larger solution spaces. The capacity of a neural network can be modified by changing the amount of neurons per hidden layer (width) and the number of hidden layers (depth). There is no clear method currently to estimate the required network width and depth beforehand and it needs to be determined experimentally. However, previous research has shown that increasing the depth is a more effective way of increasing the network capacity compared to increasing the width in general [11].

The activation function acts on the input at every neuron in the network. It is required for the activation function to be non-linear in order to effectively create a network to solve complex non-linear problems. Historically, the tanh ($\tanh(x) = (e^x - e^{-x})(e^x + e^{-x})^{-1}$) and sigmoid ($\sigma(x) = (1 + e^{-x})^{-1}$) functions were popular choices for their infinitely differentiable property, which makes them suitable for gradient-based optimization methods. Later on, it has been mostly replaced in the deep learning community by the Rectified Linear Unit (ReLU) (ReLU$= \max\{0, x\}$), because it makes networks easier to train despite its lower order of continuity [15]. Figure 2.2 shows the curves of the aforementioned activation functions.

The Swish activation function has been gaining popularity lately as it offers consistent improvements over the ReLU function in a number of challenging benchmark problems [38]. This function is defined as $x \cdot \sigma(\beta x)$, where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function and $\beta$ is a constant or trainable parameter. Depending on the value of $\beta$, the Swish function becomes the scaled linear function $f(x) = \frac{x}{2}$ at $\beta = 0$ and becomes the ReLU function as

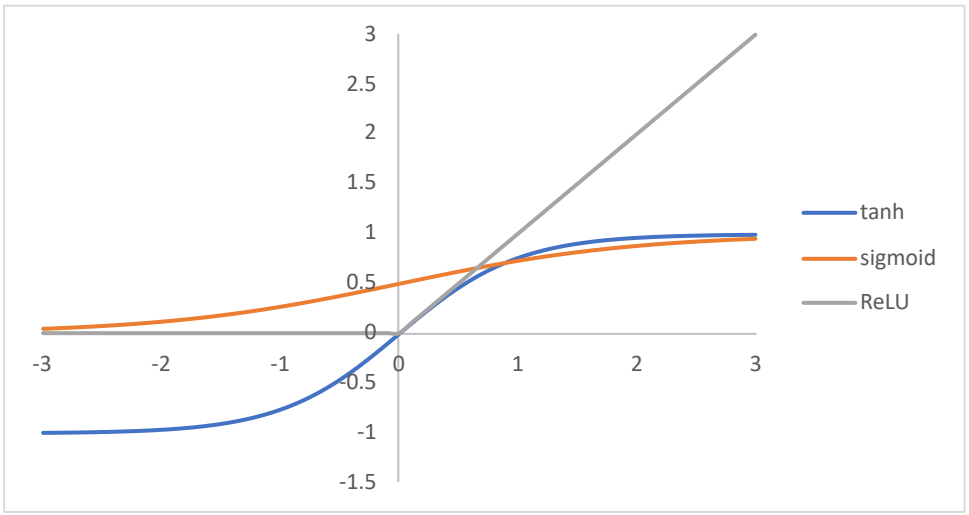



**Figure 2.2:** Graph of the tanh, sigmoid and ReLU activation functions.

$\beta \to \infty$. Thus, the Swish function can be viewed as a function that interpolates between the linear function and the ReLU function as can be seen from Figure 2.3, which shows the curve for different values of $\beta$.

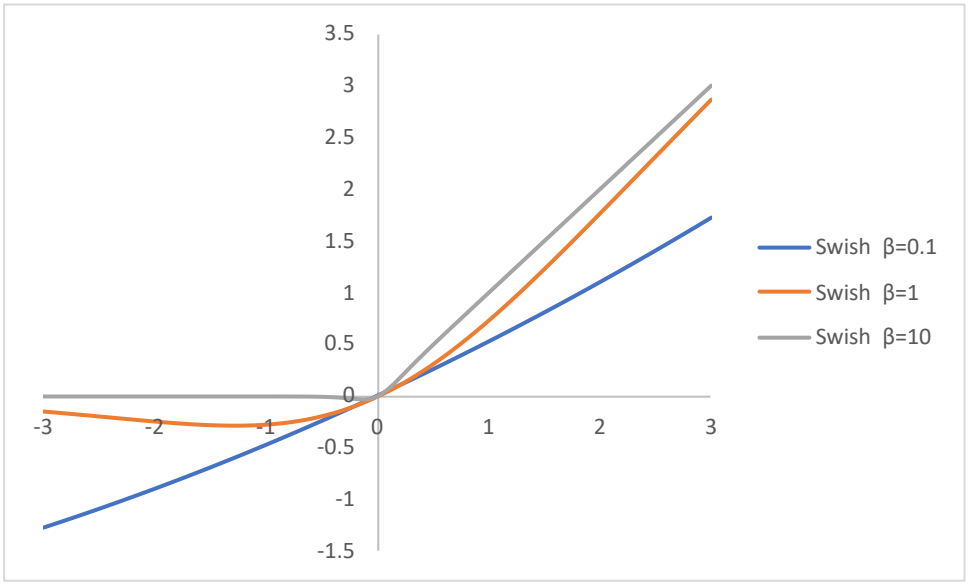



**Figure 2.3:** The Swish activation function for different values of $\beta$.

The trainable parameters of a neural network needs to be found in order to solve the problem. For this, training data is needed, which is a set solution samples for different inputs. A neural network can be trained with the training data using a numerical optimization algorithm to find the values of the trainable parameters that result in a prediction that is close to the validation samples. Amongst the many available iterative methods, which are extensively covered in detail in books such as [21], the Adam optimizer is one of the most popular for deep learning applications and has become the default choice over the years. This is a first-

order gradient based optimizer for stochastic objective functions, which can handle problems with large amounts of data and parameters in a computationally efficient way and compares favorably to other stochastic optimizers with little tuning of its hyperparameters required [23].

An important hyperparameter related to the optimizer is the learning rate or step size. The value of this parameter decides how fast the network learns and its choice is a trade-off between learning speed and accuracy: if the step size is too large, it may overshoot the minimum between iterations and prevent it from finding an accurate enough minimum.

Machine learning deals often with very large data sets, which may exceed the available memory on the GPU. For this reason the training data cannot be propagated all at once through the network every iteration, but instead mini-batches are used. The total training data set will be divided up into these mini-batches and the network is trained on those one at a time. This approach sits in between the full-batch and the stochastic approach. Besides the necessity of mini-batches in the case of physical memory limits, it offers faster network training compared to the full-batch method in general [5]. As the weights are updated every iteration, the mini-batch method updates its weights more frequently compared to the full-batch method resulting in faster training. However, smaller mini-batches have the downside of less accurate gradients and less stable training. When the whole data set has been propagated through the network in this piece-wise manner, an epoch is reached. The epoch size is the number of full passes of the training data set. Full passes need to be repeated till the loss value has reached the convergence criteria.

### 2.3.2 Physics-Informed Machine Learning

A regular deep neural network will need a large amount of data for training to find the trainable weights and biases to obtain a network that is able to make accurate predictions of the solution of a PDE. Such a large amount of data is difficult to obtain for fluid mechanics problems, because the amount of publicly available experimental and simulation data remains limited. This makes a data-driven machine learning approach a very resource intensive method when the training data needs to be gathered by yourself in the required quantities and often not a practical method in this context. However, in problems involving physics such as fluid mechanics problems, there is the vast amount of knowledge available on the physics governing the problem. A PINN is able to leverage this knowledge to partially or entirely use this as a substitute for the training data. Physical problems can be categorized based on the available data and known physics as shown in Table 2.1.

**Table 2.1:** The 3 different categories of physical problems.

| | Small data | Some data | Big data |
|---|---|---|---|
| Available data | low | medium | high |
| Known physics | high | medium | low |

Physics-informed machine learning can seamlessly integrate data from different sources with the governing physical equations resulting in a trained PINN that can predict a solution that

respects the underlying physics of the problem. For fluid mechanics problems a trained PINN can predict the velocity and pressure fields of a flow problem, thus acting as a surrogate model for conventional CFD that predicts the solution at orders of magnitude lower cost. Also more complex flow problems as a multi-physics conjugate heat transfer problem with a parameterized heat sink can be solved with PINNs [16], making it an efficient approach to solving fluid mechanics problems when the computation time of the multitude of CFD simulations exceed the training time of a PINN on the parameterized problem. In such cases it has been demonstrated that computations can be accelerated up to 135,000x compared to OpenFOAM [16].

Besides accelerating the solution of problems that can be solved by conventional means, PINNs also feature the ability to solve problems that are difficult or impossible to solve before. Such potential applications include the ill-posed problem of finding unknown parameters from flow visualization without boundary conditions [37], 3D flow reconstruction from 2D data of the flow [6] and determining the 3D velocity and pressure fields from 3D temperature measurements [7].

### Structure of a PINN

A physics-informed neural network is obtained when an additional part is added to compute the residuals of the governing PDE. The PDE is not only a function of the output, but also their derivatives with respect to the inputs, which can be computed at a low cost with automatic differentiation. These residuals are then used to construct an additional loss term that contributes to the total loss of the network. When the total loss is minimized during training, the PINN will not only try to match its prediction to the training data of the solution, but also tries to satisfy the governing equation on the interior collocation points where the residuals are evaluated. So, the residual network provides extra training information on the solution domain and also steers the network away from non-physical solutions during training that do not satisfy the governing equations. In the case of incompressible flow, the Navier-Stokes equations take the form of Equation 2.4.

$$u_x + v_y = 0 \tag{2.4a}$$
$$u_t + (uu_x + vu_y) = -p_x + \nu(u_{xx} + u_{yy}) \tag{2.4b}$$
$$v_t + (uv_x + vv_y) = -p_y + \nu(v_{xx} + v_{yy}) \tag{2.4c}$$

Then, Equation 2.5 defines the equations for the residual network.

$$f_1 := u_x + v_y \tag{2.5a}$$
$$f_2 := u_t + (uu_x + vu_y) + p_x - \nu(u_{xx} + u_{yy}) \tag{2.5b}$$
$$f_3 := v_t + (uv_x + vv_y) + p_y - \nu(v_{xx} + v_{yy}) \tag{2.5c}$$

The derivatives for Equation 2.5 can be obtained by automatic differentiation, which a method of computing derivatives by repeatedly using the chain rule and features a low computational cost for deep neural networks [3]. This network will then be able to compute the residuals on the collocation points $N_f$ to set up the additional loss term in Equation 2.6.

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_u^i, x_u^i, y_u^i)|^2 \tag{2.6}$$

The final loss term is then the sum of the losses of Equation 2.1 and 2.6 as written in Equation 2.7.

$$MSE = MSE_u + MSE_f \tag{2.7}$$

In the case of surrogate modeling with a PINN without the use of simulation and experimental data, all training information will need to come from the available physical knowledge of the problem. Information about the solution is only present on the initial condition and boundary conditions. These are uniformly sampled to create one part of the training data. No information on the solution will be available in the interior of the solution space, so the training of the PINN will have to rely on the collocation points in the interior as the true value of the residuals of the governing equations are zero and will form the other part of the training data. This method of training a physics-informed neural network without external data on the solution has shown to be working on academic benchmark problems [36].

Figure 2.4 shows the structure of a PINN algorithm of a 2D flow problem, governed by the Navier-Stokes equations.

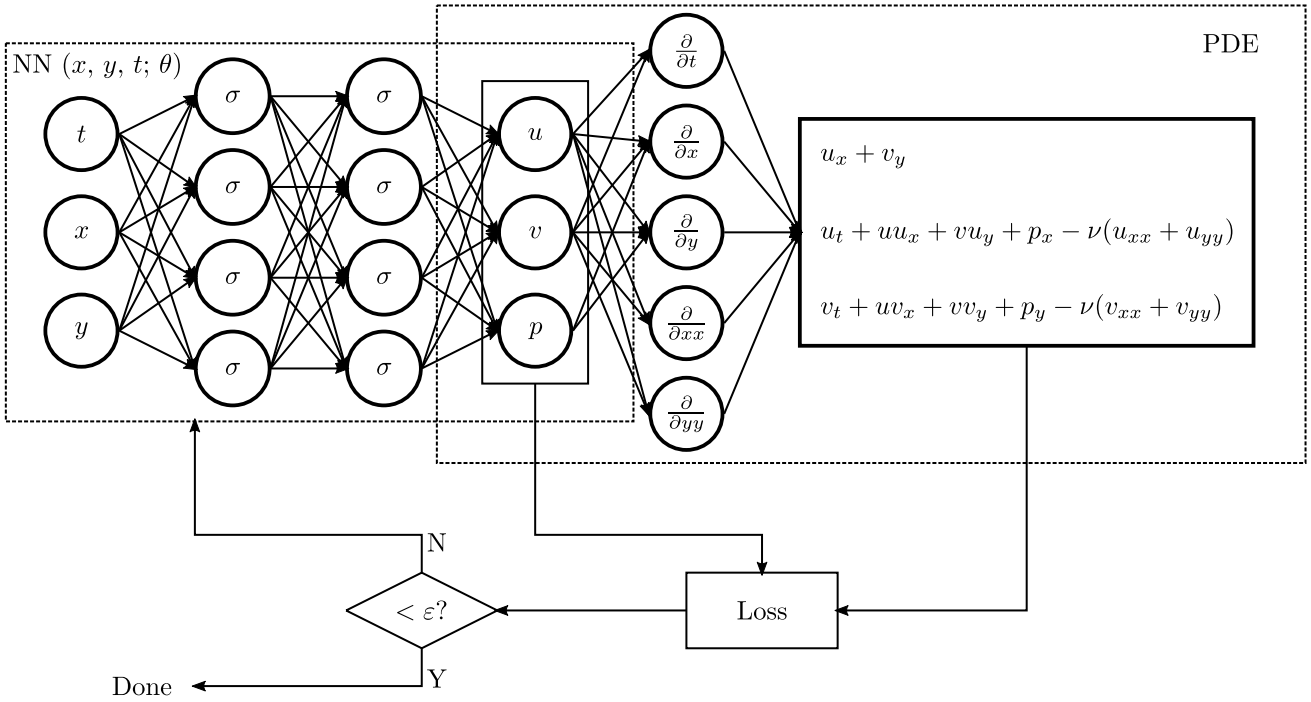



**Figure 2.4:** Fully connected PINN with the Navier-Stokes residuals for the residual network.

On the left side is the neural network with $x$, $y$, and $t$ inputs and the velocity and pressure fields $u$, $v$ and $p$ as outputs. In between this input and output layer, the hidden layers are present consisting of the non-linear activation functions $\sigma$. Connected to the output layer, is the residual network where the required partial derivatives are computed for the Navier-Stokes residual equations. Both the output layer and the residual network will have a loss with the total loss being computed with Equation 2.7. While this error is larger than the set convergence criteria the training will continue.

### 2.3.3 Airfoil Design Optimization

One of the investigated practical application of a trained PINN is using it for an airfoil design optimization problem.

Numerical optimization of a design problem is performed by framing the problem as a minimization problem. This minimization problem will have an objective function $I$, where its minimization will result in desirable characteristics of the airfoil design and is a function of the design parameters. Equation 2.8 is the standard form of an optimization function under box constraints [43].

$$\begin{aligned} \min_{x} \quad & I(x) \\ \text{s.t.} \quad & l_i \leq x_i \leq u_i \end{aligned} \tag{2.8}$$

In an airfoil optimization problem, the objective function will be an function of the aerodynamic coefficients, which in turn are functions of the airfoil design parameters. The aerodynamic coefficients for an airfoil can be computed by integrating the pressure and friction around the airfoil [2].

Equation 2.9 and Equation 2.10 are the formulas for the pressure and friction coefficients which are integrated using Equation 2.11 and Equation 2.12 to obtain the normal and axial force coefficients.

$$c_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.9}$$

$$c_f = \frac{\tau}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.10}$$

$$c_n = \frac{1}{c}\left[\int_0^c (c_{p,l} - c_{p,u})dx + \int_0^c (c_{f,u}\frac{dy_u}{d_x} + c_{f,l}\frac{dy_l}{dx})dx\right] \tag{2.11}$$

$$c_a = \frac{1}{c}\left[\int_0^c (c_{p,u}\frac{dy_u}{dx} - c_{p,l}\frac{dy_l}{dx})dx + \int_0^c (c_{f,u} + c_{f,l})dx\right] \tag{2.12}$$

The lift and drag coefficients can be computed from the force coefficients and the angle of attack $\alpha$ with Equations 2.13 and 2.14.

$$c_l = c_n \cos\alpha - c_a \sin\alpha \tag{2.13}$$

$$c_d = c_n \sin\alpha - c_a \cos\alpha \tag{2.14}$$

The design parameters are defined by a parametric representation of the airfoil, such as the NACA 4-digit airfoils. The airfoil geometry of this family of airfoils are defined by 3 parameters.

The numerical optimization itself can be performed with a numerical optimization algorithm that attempts to find the values for the design parameters that minimize the objective function. During this thesis, the L-BFGS-B implementation of SciPy will be used for airfoil optimization, which is a variation of the iterative Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for solving non-linear optimization problems with the ability to prescribe simple box constraints on its design parameters [4].

Numerical optimization of an airfoil requires a multitude of function evaluations of the objective function because of the iterative nature of the optimization method. If a high fidelity solution is required of the aerodynamic coefficients, CFD will be needed to resolve the Navier-Stokes equations to obtain the velocity and pressure fields to compute the lift and drag coefficients. These are computationally expensive to perform when hundreds of iterations are required, so in practice the problem will be simplified or a surrogate model will be used.

### NACA 4-digit airfoils

The NACA 4-digit family of airfoils is well known and extensively tested since its initial development. Many test results are available such as [19], which makes these airfoils suited for validating the PINN results. The geometries of this family of airfoils are defined by three intuitive parameters, which are directly related to its geometry. These are summarized in Table 2.2.

**Table 2.2:** Summary of the NACA 4-series geometrical parameters.

| parameter | description |
|---|---|
| $m$ | maximum camber as a percentage of the the chord length |
| $p$ | location of the maximum camber as tenths of the chord |
| $t$ | maximum thickness as percentage of the chord |

The chord thickness is given by Equation 2.15.

$$y_t = 5t\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \tag{2.15}$$

The camber line is described by Equation 2.16.

$$y_c = \begin{cases} \dfrac{m}{p^2}(2px - x^2) & 0 \leq x \leq p \\ \dfrac{m}{(1-p)^2}[(1-2p) + 2px - x^2] & p \leq x \leq 1 \end{cases} \tag{2.16}$$

Then, Equation 2.17 the coordinates of the airfoil upper surface $(x_U, y_U)$ and lower surface $(x_L, y_L)$.

$$x_U = x - y_t \sin\theta, \qquad y_U = y_c + y_t \cos\theta \tag{2.17a}$$

$$x_L = x + y_t \sin\theta, \qquad y_L = y_c - y_t \cos\theta \tag{2.17b}$$

Here, $\theta$ is given by Equation 2.18.

$$\theta = \arctan \frac{dy_c}{dx}, \qquad \frac{dy_c}{dx} = \begin{cases} \frac{2m}{p^2}(p-x) & 0 \leq x \leq p \\ \frac{2m}{(1-p)^2}(p-x) & p \leq x \leq 1 \end{cases} \tag{2.18}$$

### PARSEC airfoil

While the equations of the NACA 4-digit series describe airfoil geometries with parameters, they are all existing airfoils and will not be of much use for airfoil design optimization as the result will be certainly an existing one with well documented characteristics. Thus, a different parameterization method is needed with more degrees of freedom that can describe a larger variety of airfoil shapes. The PARSEC method from [40] is a good candidate for this. A comparison between airfoil parameterization schemes has been studied in [42], where the PARSEC method scores well in most of their rating criteria. The main advantage is the intuitiveness of the PARSEC method, where its eleven parameters are describing the main geometrical features of the airfoil directly. This makes it suitable for aerodynamic optimization because it enables physical understanding of the effects caused by the changes in the geometric parameters, unlike methods with abstract parameters that have no clear meaning. The PARSEC parameters are summarized in Table 2.3 and shown in Figure 2.5.

**Table 2.3:** Summary of the eleven geometrical PARSEC parameters.

| PARSEC parameter | Geometry parameter | Description |
|---|---|---|
| $p_1$ | $r_{le}$ | leading edge radius |
| $p_2$ | $x_{up}$ | upper crest horizontal position |
| $p_3$ | $z_{up}$ | upper crest vertical position |
| $p_4$ | $z_{xx,up}$ | upper curvature at crest position |
| $p_5$ | $x_{lo}$ | lower crest horizontal position |
| $p_6$ | $z_{lo}$ | lower crest vertical position |
| $p_7$ | $z_{xx,lo}$ | lower curvature at crest position |
| $p_8$ | $z_{te}$ | trailing edge vertical position |
| $p_9$ | $\Delta z_{te}$ | trailing edge thickness |
| $p_{10}$ | $\alpha_{te}$ | trailing edge direction |
| $p_{11}$ | $\beta_{te}$ | trailing edge wedge angle |

The vertical airfoil coordinates are described by the polynomial in Equation 2.19.

$$z(x) = \sum_{n=1}^{6} a_n x^{n-\frac{1}{2}} \tag{2.19}$$

The coefficients $a_n$ for the upper side of the airfoil are found by solving the system of equations in Equation 2.20, using the 11 PARSEC parameters [46].

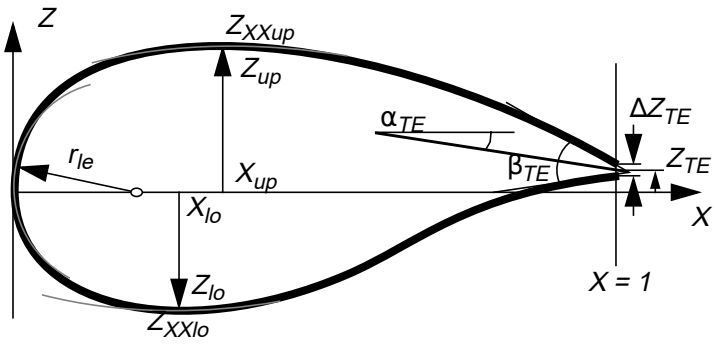



**Figure 2.5:** PARSEC parameters on an airfoil (adapted from [40]).

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ p_2^{\frac{1}{2}} & p_2^{\frac{3}{2}} & p_2^{\frac{5}{2}} & p_2^{\frac{7}{2}} & p_2^{\frac{9}{2}} & p_2^{\frac{11}{2}} \\ \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & \frac{7}{2} & \frac{9}{2} & \frac{11}{2} \\ \frac{1}{2}p_2^{-\frac{1}{2}} & \frac{3}{2}p_2^{\frac{1}{2}} & \frac{5}{2}p_2^{\frac{3}{2}} & \frac{7}{2}p_2^{\frac{5}{2}} & \frac{9}{2}p_2^{\frac{7}{2}} & \frac{11}{2}p_2^{\frac{9}{2}} \\ -\frac{1}{4}p_2^{-\frac{3}{2}} & \frac{3}{4}p_2^{-\frac{1}{2}} & \frac{15}{4}p_2^{\frac{1}{2}} & \frac{35}{4}p_2^{\frac{3}{2}} & \frac{63}{4}p_2^{\frac{5}{2}} & \frac{99}{4}p_2^{\frac{7}{2}} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} p_8 + p_9/2 \\ p_3 \\ \tan(p_{10} - p_{11}/2) \\ 0 \\ p_4 \\ \sqrt{2p_1} \end{pmatrix} \tag{2.20}$$

The coefficients of the lower side are obtained in a similar way using Equation 2.21.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ p_5^{\frac{1}{2}} & p_5^{\frac{3}{2}} & p_5^{\frac{5}{2}} & p_5^{\frac{7}{2}} & p_5^{\frac{9}{2}} & p_5^{\frac{11}{2}} \\ \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & \frac{7}{2} & \frac{9}{2} & \frac{11}{2} \\ \frac{1}{2}p_5^{-\frac{1}{2}} & \frac{3}{2}p_5^{\frac{1}{2}} & \frac{5}{2}p_5^{\frac{3}{2}} & \frac{7}{2}p_5^{\frac{5}{2}} & \frac{9}{2}p_5^{\frac{7}{2}} & \frac{11}{2}p_5^{\frac{9}{2}} \\ -\frac{1}{4}p_5^{-\frac{3}{2}} & \frac{3}{4}p_5^{-\frac{1}{2}} & \frac{15}{4}p_5^{\frac{1}{2}} & \frac{35}{4}p_5^{\frac{3}{2}} & \frac{63}{4}p_5^{\frac{5}{2}} & \frac{99}{4}p_5^{\frac{7}{2}} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} p_8 - p_9/2 \\ p_6 \\ \tan(p_{10} + p_{11}/2) \\ 0 \\ p_7 \\ -\sqrt{2p_1} \end{pmatrix} \tag{2.21}$$

### 2.3.4 Software

The success of deep learning in recent years have led to the rapid development of open-source deep learning software tools. Two well known machine learning libraries are Tensorflow [1] and PyTorch [33]. Use of such software tools requires sufficient computing power as training deep neural networks is a time consuming process. GPU computing is available for most of the popular software tools and have shown advantages in performance compared to CPU based computing for deep learning [39]. NVIDIA SimNet [16] will be used as a software tool to create and train the PINNs. This is a PINNs framework with a TensorFlow backend that enables users to rapidly create, train and use physics-informed neural networks. This choice has been made as it is actively developed and contains more features than competing projects as SciANN [14] and DeepXDE [28] because their use has only been demonstrated on simple 1D and 2D benchmark problems.

SimNet is distributed as a Docker image, so it is run in a Docker container on a Ubuntu workstation system with a RTX8000 GPU. OpenFOAM is used to perform conventional CFD simulations to produce training and validation data for the PINN solver. Meshes have been generated with gmsh. The simulations are run on a Intel(R) Xeon(R) CPU E5-2660v3 with 20 threads in a cluster. Visualization of the flow field results are performed with Paraview.

# Chapter 3

# Flow reconstruction

The first case to be investigated is the flow reconstruction of the transient flow solution of a airfoil in free flow from limited measurement data. The airfoil is at an angle of attack of 25 degrees in an incompressible flow with a chord-based Reynolds number of $\mathcal{R}e = 200$ to induce vortex shedding as in [30]. First an OpenFOAM simulation of this case will be performed to generate a data set to use for reference. A part of this data will be taken in order to emulate measurement data to use in PINN training. Finally, the predictions of the trained PINN will be compared to the OpenFOAM data to evaluate its performance.

## 3.1 OpenFOAM simulation

First, a mesh needs to be created of the computational domain. The mesh is created with gmsh and is unstructured for the most part. It counts 48006 nodes and 133206 elements and is shown in Figure 3.1.

This case is modeled in a non-dimensional way by choosing a value of 1 for the freestream velocity $u_\infty$ and the airfoil chord length $L$. Then, the chord-based Reynolds number will only depend on the viscosity $\nu$ as $\mathcal{R}e = \frac{u_\infty L}{\nu} = \frac{1}{\nu}$.

The simulation is performed with the IcoFoam solver at a Reynolds number of $\mathcal{R}e = 200$ by setting the viscosity to $\nu = 0.005$. The rectangular computational domain of the OpenFOAM simulation measures $[-10, 20] \times [-10, 10]$. The airfoil is the optimized PARSEC geometry from the last chapter with unit length 1. Inlet velocity is $u = 1$ and $v = 0$, top and bottom boundary conditions are set to slip, pressure outlet is fixed at $p = 0$ and airfoil has the no-slip condition. Time range is $t = (0, 35)$ with a $\Delta t = 0.0008$. Data is recorded for every 50 time steps, so the recorded data has a time step of $\Delta t = 0.004$ with 75 snapshots in total. A snapshot of the u velocity is shown at $t = 30$ in Figure 3.2 where periodic vortex shedding can be observed.

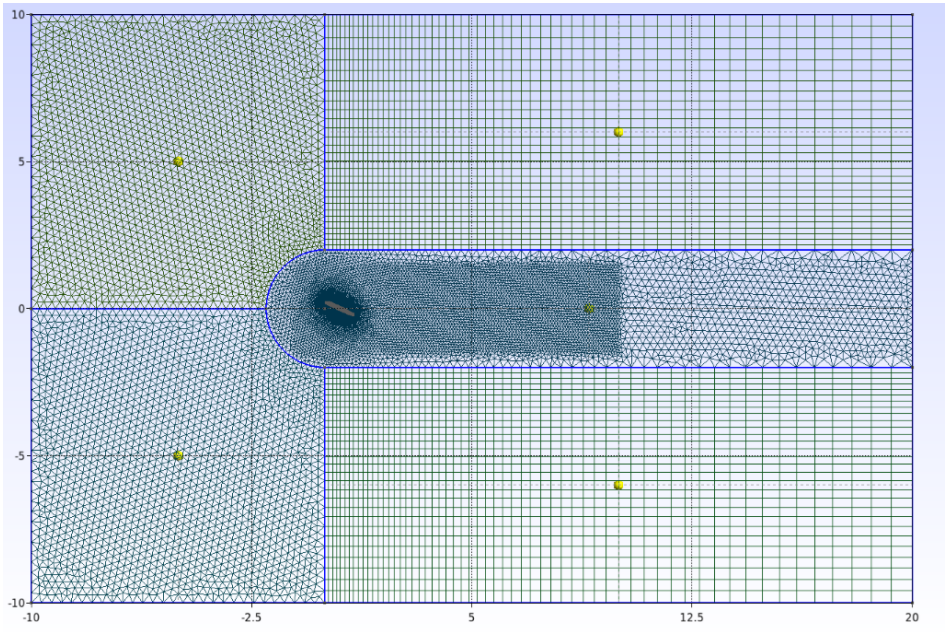



**Figure 3.1:** Mesh for the transient OpenFOAM simulation of vortex shedding of an airfoil.

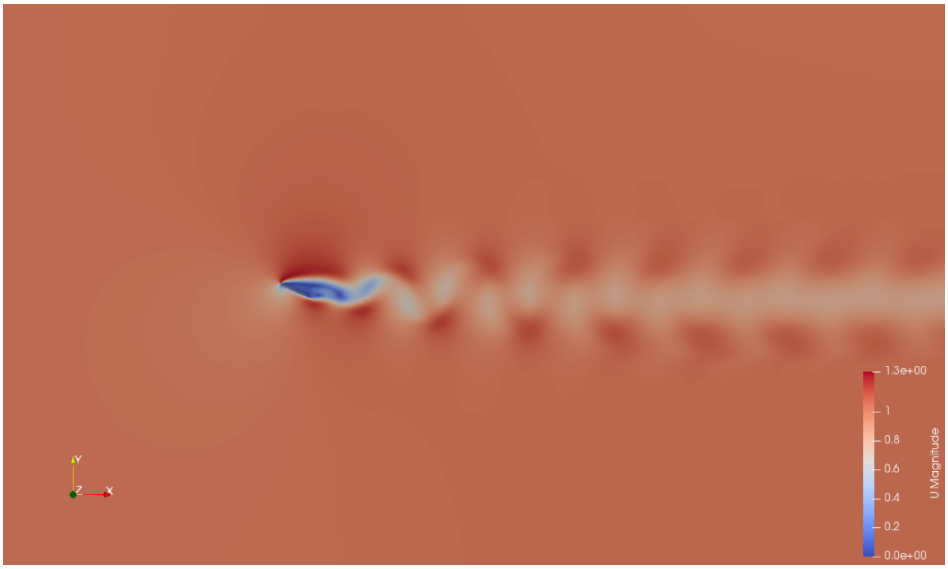



**Figure 3.2:** The $u$-component of the velocity field at $t = 30$.

The vortex shedding frequency can be observed from Figure 3.3 which shows the lift coefficient fluctuating with time at approximately 0.35 hz.

The data will be limited to a time range of 3 in total from $t = 30.00$ to $t = 33$ to have a full period of vortex shedding. At $t = 30.00$ the full velocity and pressure fields will be used as an initial condition for the PINN. The time steps $t = 30.04$ to $t = 33.00$ with $\Delta t = 0.04$ will only have the pressure measurements on the airfoil surface and velocity and pressure data in a square behind the airfoil.

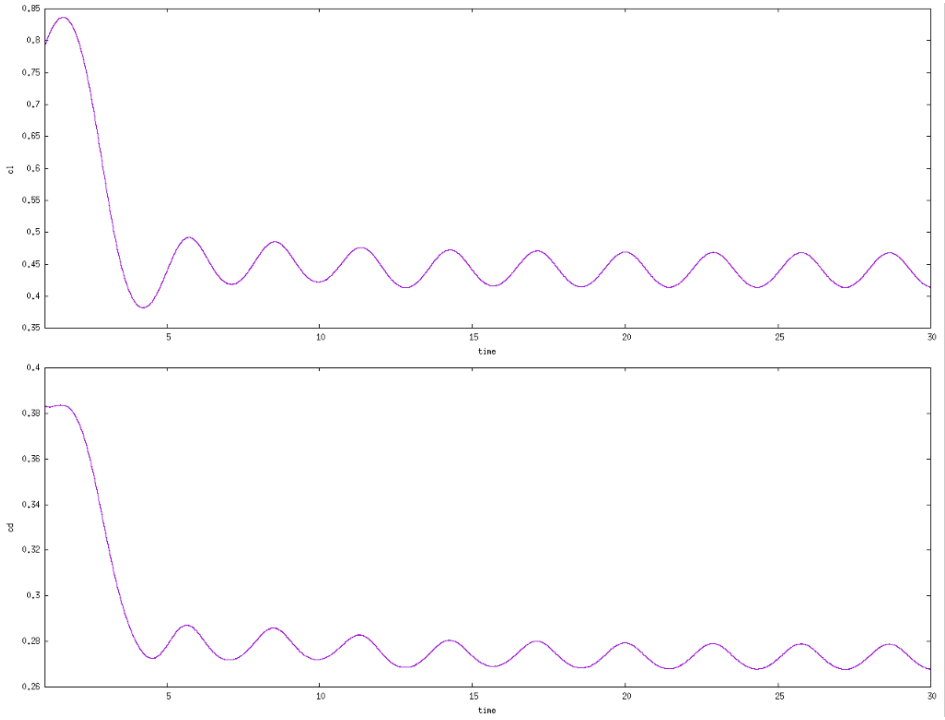



**Figure 3.3:** Lift and drag coefficient as a function of time.

## 3.2 Flow Reconstruction with PINNs

All required preparations are completed with OpenFOAM data ready to start training the PINN for the flow reconstruction case. The PINN will be created and trained with SimNet, NVIDIA's framework for physics-informed machine learning.

### 3.2.1 PINN set-up

The PINN for this problem is a fully-connected network with the Swish activation function and has 9 layers with 128 neurons each. Adam optimizer parameters are left at the SimNet defaults $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 1 \cdot 10^{-8}$. The learning rate starts at 0.001 with an exponential decay rate of 0.95 every 2000 iterations. Training is performed for 150k iterations.

The computational domain is $[-1, 6] \times [-2, 2]$ in space and $[30, 33]$ in time. Collocation points where the Navier-Stokes residuals are evaluated are generated beforehand by random sampling the computational domain in space and time. This is done with a sampling density of a batch size per area (BPA) of 64, meaning 64 points per unit area per batch. There is an increased sampling density of 2048 points per unit area in a $[-0.2, 1.2] \times [-0.7, 0.7]$ square around the airfoil to capture the sharp gradients. The no-slip condition is enforced by sampling the airfoil boundary as with a sample density of 128 samples per unit length. These points are added to the training data on the solution with their velocity components set to zero. The

importance of the collocation points are weighted by area, so the increase in sampling density around the airfoil does not lead to a higher importance locally, but remains constant across the computational domain. This is helpful in avoiding the trivial solution in the densely sampled area, which would slow down the training in the beginning. These collocation points are divided in 1000 batches per epoch (BPE), so the full data set is propagated through the network 150 times during training. Figure 3.4 shows the spatial domain of the PINN with the dotted square indicating the area with increased sampling density and the dashed square the measurement window where the time-dependent OpenFOAM data is available.

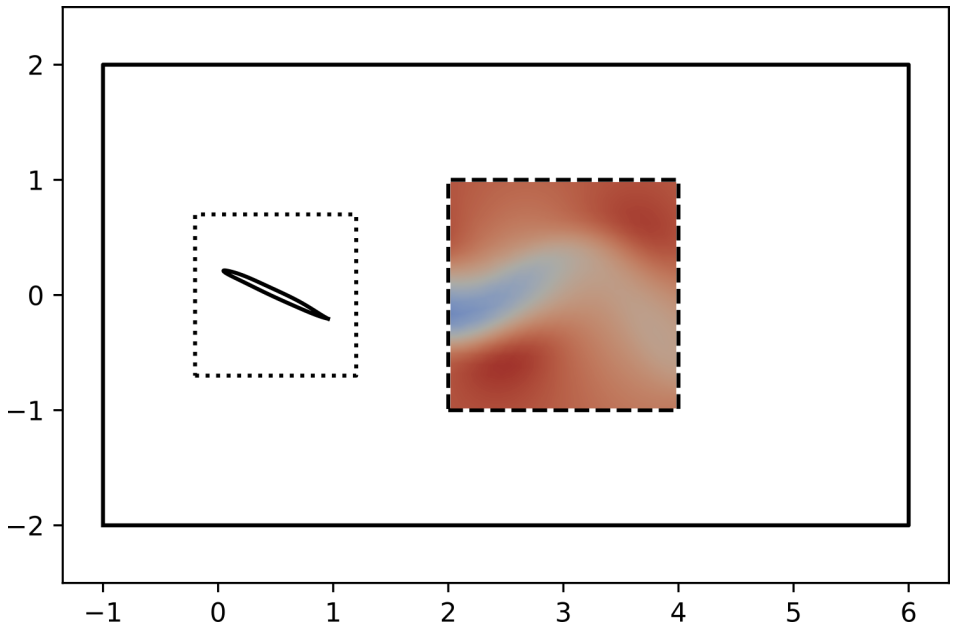



**Figure 3.4:** Computational domain of the PINN. The black rectangle shows the boundaries of the computational domain where flow reconstruction will be attempted. An increased sampling density of the collocation points is used in the area close to the airfoil, marked by the dotted square. Training data from the OpenFOAM simulation is only available in the form of pressure data on the airfoil perimeter and the flow field inside the square behind the airfoil.

The OpenFOAM data used for training consists of 3 parts: the initial condition with the full solution at $t = 30.00$, transient pressure data on the airfoil surface from $t = 30.04$ to $t = 33.00$ and the transient solution in a simulated measurement window downstream of the airfoil of the size $[2, 4] \times [-1, 1]$ from $t = 30.04$ to $t = 33.00$. The transient data has a constant spacing of $\Delta t = 0.04$, resulting in 75 snapshots in time. The OpenFOAM data is divided into 10 batches and its weighted with a additional weight factor of 100 for the loss calculation. This was factor was experimentally determined to be the most suitable for this specific case as both higher and lower weights led to poor predictions of the PINN, where the vortices in the wake are mostly lost and steady state solution is approached.

## 3.3 Flow Reconstruction Results

Two PINNs are trained for this case with different amounts of data available to investigate the accuracy of the resulting PINNs depending on the amount of training data. The first PINN will have full velocity an pressure fields $u$, $v$ and $p$ in the measurement window, while the second will only have the pressure field $p$.

### 3.3.1 PINN results with full data in measurement window

The first PINN has the full OpenFOAM data available in the measurement window with the important settings summarized in Table 3.1. The loss has been plotted in Figure 3.5 with a

**Table 3.1:** Summary of the parameters of the first PINN.

| Parameters | Setting |
|---|---|
| PINN size | 9 layers of 128 neurons |
| Computational domain | $x = [-1, 6]$, $y = [-4, 4]$, $t = [30, 33]$ |
| Boundary conditions | no-slip airfoil (u=v=0) |
| PINN Sampling density | lr: BPA=64, BPE=1000 |
| | hr: BPA=2048, BPE=1000 |
| | airfoil: BPA=128, BPE=1000 |
| Training iterations | 150,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 200$ (by setting $\nu = 0.005$) |
| OpenFOAM training data | IC: full solution at $t = 30.00$ |
| | airfoil: $p$ for $t = [30.04, 33.00]$ |
| | window: $u$, $v$, $p$ for $t = [30.04, 33.00]$ |
| OpenFOAM data number of batches | IC: BPE=10, weight=100 |
| | airfoil: BPE=10, weight=100 |
| | window: BPE=10, weight=100 |

logarithmic vertical axis. Figure 3.5 shows the loss eventually settles below a value of 1, but peaks often over the course of the training, implying a difficult optimization landscape.

The hyperparameters of the Adam optimizer are left at their default settings as mentioned before. The learning rate varies during training as shown in Figure 3.6 due to the learning rate decay.

The comparison between the PINN prediction and the OpenFOAM results are made at three snapshots in time at $t = 30.04$, $t = 31.52$ and $t = 33.00$. Figure 3.7, Figure 3.8 and Figure 3.9 show the $u$-velocity at the three snapshots with the measurement window marked in white. Pressure and $v$-velocity are found in Appendix A. The PINN predictions shows a decent similarity with the OpenFOAM simulation. Especially in the window downstream of the airfoil, where the full solution data was provided during training, the results are a close match. The largest differences are observed in the area between the airfoil and window,

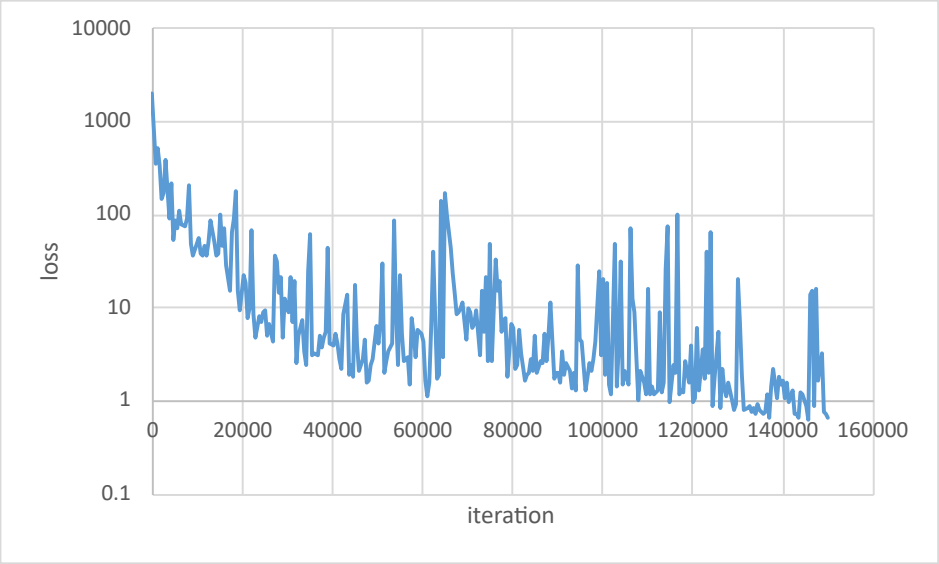



**Figure 3.5:** Loss graph of the PINN training with $u$, $v$ and $p$ data in the measurement window.

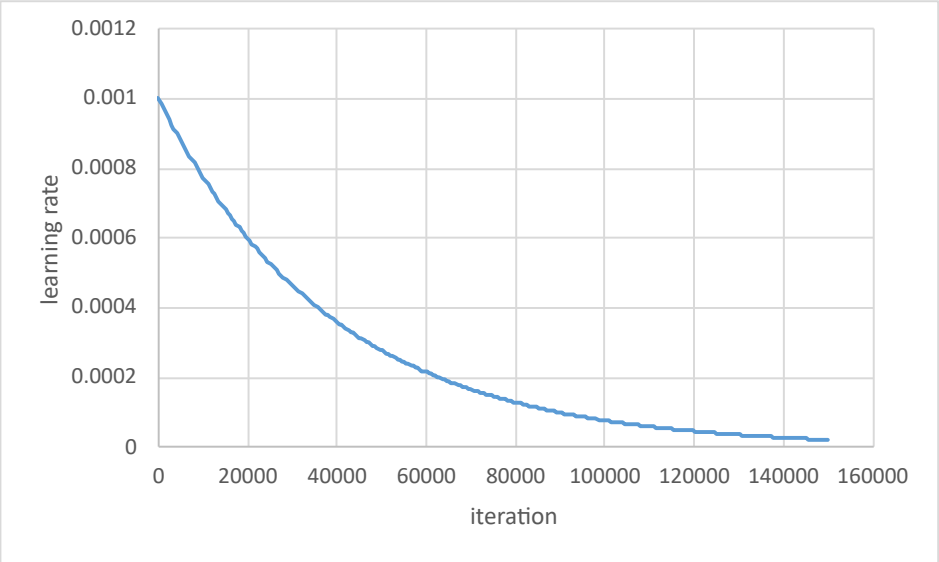
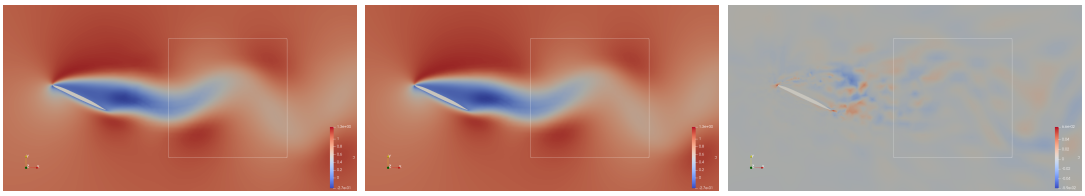


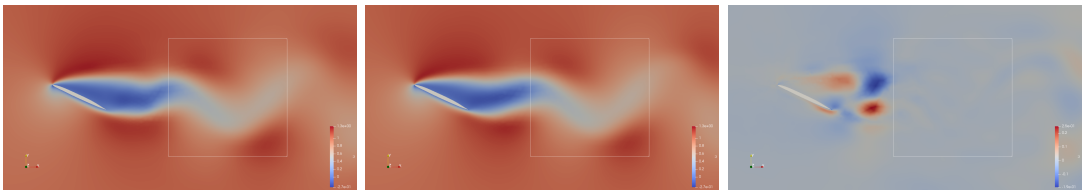

**Figure 3.6:** Learning rate graph of the PINN training with u,v and p data in the measurement window.

meaning the PINN has trouble with making an accurate prediction in that area. On the time axis, a better prediction can be observed closer to the lower bound. The likely reason for this is the presence of the initial condition resulting in more training data nearby. Overall, the PINN produces a decent prediction with the shape of the wake and the phase intact.



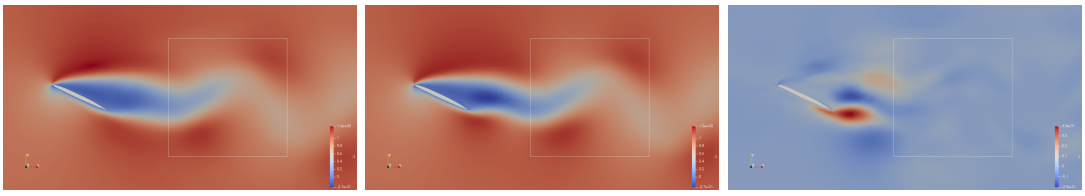
**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.7:** $u$-velocity comparison at $t = 30.04$ for the PINN trained on the full solution in the measurement window.



**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.8:** $u$-velocity comparison at $t = 31.52$ for the PINN trained on the full solution in the measurement window.



**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.9:** $u$-velocity comparison at $t = 33.00$ for the PINN trained on the full solution in the measurement window.

### 3.3.2 PINN results with only pressure data in measurement window

The second PINN is trained with reduced training data to observe its effect on the PINN accuracy. In this case the training data in the window only has the pressure available compared to the full solution for the previous PINN. Other than the reduction of the training data, the set-up is equal to the previous PINN as shown in Table 3.2. The loss graph of this PINN's training is shown in Figure 3.10, where similar peaking can be observed as in Figure 3.5. The loss settles around a value of 1, which is higher to the previous PINN, suggesting a worse accuracy. The identical hyperparameters for the optimizer results in the same learning rate decay as shown in Figure 3.6.

In a similar fashion, the Figure 3.11, Figure 3.12 and Figure 3.13 show the PINN prediction results compared to the OpenFOAM data. This PINN shows a similar performance as the previous at the first snapshot at $t = 30.04$ in Figure 3.11. However, the accuracy drops for the following snapshots in Figure 3.12 and Figure 3.13 when moving along the positive direction of the time axis. The velocity gradients are less sharp in the wake and the extremes appear

**Table 3.2:** Summary of the parameters of the second PINN.

| **Parameters** | **Setting** |
|---|---|
| PINN size | 9 layers of 128 neurons |
| Computational domain | $x = [-1, 6]$, $y = [-4, 4]$, $t = [30, 33]$ |
| Boundary conditions | no-slip airfoil (u=v=0) |
| PINN Sampling density | lr: BPA=64, BPE=1000 |
| | hr: BPA=2048, BPE=1000 |
| | airfoil: BPA=128, BPE=1000 |
| Training iterations | 150,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 200$ (by setting $\nu = 0.005$) |
| OpenFOAM training data | IC: full solution at $t = 30.00$ |
| | airfoil: $p$ for $t = [30.04, 33.00]$ |
| | window: $p$ for $t = [30.04, 33.00]$ |
| OpenFOAM data number of batches | IC: BPE=10, weight=100 |
| | airfoil: BPE=10, weight=100 |
| | window: BPE=10, weight=100 |

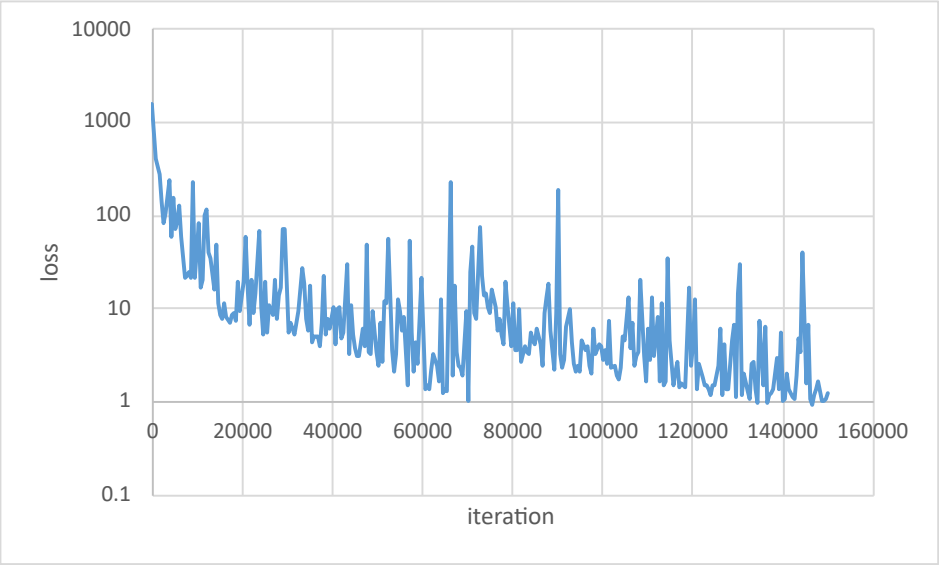
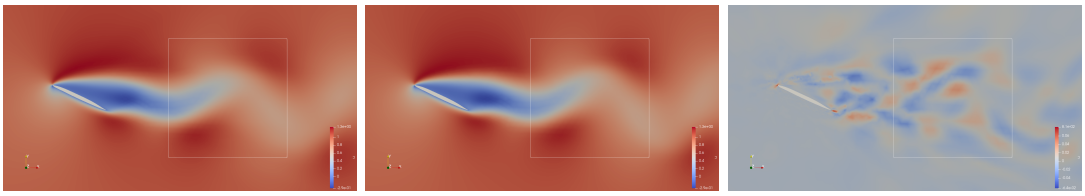



**Figure 3.10:** Loss graph of the PINN training with only $p$ data in the measurement window.

smoothed. The periodic vortex shedding structure in the flow is gradually lost higher time values and the PINN tends to predict as solution closer to the steady-state solution than the transient.

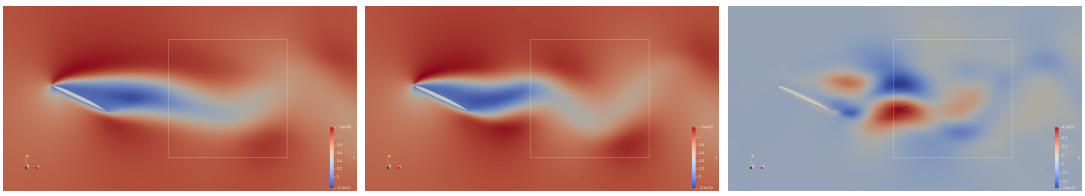
### 3.3.3 Conclusion

Comparing the accuracy of the two PINNs, the first, trained with full solution data in the measurement window, shows decent predictive capabilities. The predictions match the Open-



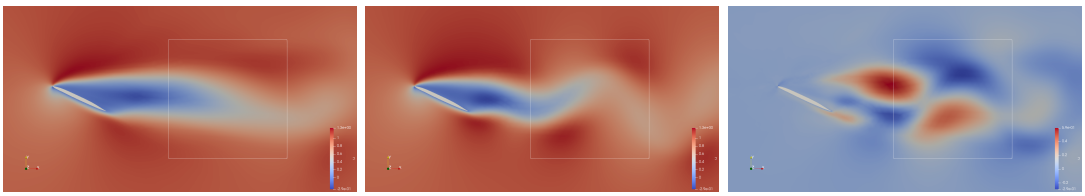
**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.11:** $u$-velocity comparison at $t = 30.04$ for the PINN trained on pressure in the measurement window.



**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.12:** $u$-velocity comparison at $t = 31.52$ for the PINN trained on pressure in the measurement window.



**(a)** $u$-velocity PINN **(b)** $u$-velocity OpenFOAM **(c)** $u$-velocity difference

**Figure 3.13:** $u$-velocity comparison at $t = 33.00$ for the PINN trained on pressure in the measurement window.

FOAM data qualitatively with clear periodic vortex shedding in the wake of the airfoil. The same cannot be said about the second PINN, which has reduced training data in the window behind the airfoil to only the pressure. Predictions close to the initial condition are close to the OpenFOAM results, but get increasingly worse at higher values of time. Thus the available data is insufficient to train an accurate PINN with this set-up.

This also reveals a weakness of how this PINN set-up handles time and its consequences. In this set-up, the PINN is trained for the whole time range simultaneously, while for a flow problem time has a dissipative effect and temporal information should only flow along the positive time axis. For a transient flow problem, later states are dependent on the earlier states. When during training, a prediction of a later state is corrected before that of an earlier state, that correction together with other later states may become wrong when the prediction of an earlier state is changed to fit the training data and governing equations better. This makes the training of the PINN quite inefficient and is the likely cause of the peaks in the loss value over the training process in Figure 3.5 and Figure 3.10. Also, this makes the global optimum more difficult to find and more likely to get stuck in a local optimum.

A clue to solving this can be found in the high accuracy of the predictions near the initial condition. This suggests that a PINN with a small time range is likely to give an accurate prediction. A larger time range can be tackled by discretizing the computational domain along the time axis, thus dividing the problem in many smaller problems which can be solved by individual PINNs. These are trained sequentially starting at the lower time limit, where the previous trained PINN will predict an initial condition for the next PINN. This would emulate a time-marching method and would provide a mechanism to make information only propagate in one direction and all the trained PINNs together would cover the entire time range. Similar approaches are described in [32] and [48], but research in such a discretized approach with PINNs is limited and implementation complex, thus deemed out of the scope of this project.

An alternative method to emulating time-marching is temporal weighting of the losses. This increases the weights of the losses at lower time values to prioritize them over the lower weighted losses at higher time values. However, a lower accuracy may still be observed at higher time values at the end, because their importance is also reduced with the lower loss weights, resulting in a lower contribution to the total loss.

As the PINN struggles to maintain a high accuracy in the latter part of the time range for an unsteady problem, it is unlikely for a PINN to predict a time accurate-solution for more complex cases with a parametric geometry. For this reason, the following cases in the next chapters will only consider steady flow.

# Chapter 4

# PINNs on parametric NACA 4-series airfoils in steady flow

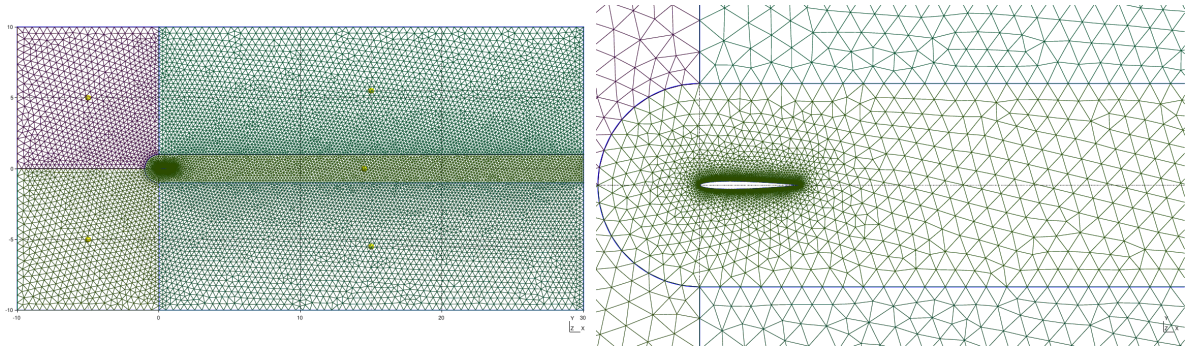
The previous chapter has shown that it is difficult for a PINN to deal with time in a continuous way with the current set-up. This chapter will move on to the case of a parametric NACA 4-series airfoil that is a regular well-posed forward problem, where the boundary conditions are all defined. The complexity of the problem increases with the addition of the three NACA parameters, which add extra dimensions to the flow case. The three NACA parameters (maximum camber $m_{\text{NACA}}$, maximum camber position $p_{\text{NACA}}$ and thickness $t_{\text{NACA}}$) can be added to the input layer of the PINN if they are not set to a fixed value. It is unlikely that a PINN in a similar set-up will succeed in solving this problem in a transient setting, since only moderate success was achieved in the previous chapter. Thus, starting from this chapter, the time dependency of the flow will be removed and the steady-state flow case will be solved instead of the transient flow case. While it is possible to train a PINN with only physical information on a flow case with a fixed geometry, some preliminary test have shown difficulties in reaching a satisfactory level of accuracy for parametric cases. So, some additional data from OpenFOAM simulations will supplement the training data to improve the convergence and accuracy of the PINN in addition to the available physical information. The OpenFOAM results for PINN training and validation will be assumed to be the physical truth, which is shown later to be valid with a comparison to results from literature.

So, this chapter describes the training of two PINNs on parametric NACA airfoils in a steady, incompressible and viscous flow at a Reynolds number of $\mathcal{R}e = 600$ and performing the required OpenFOAM simulations to obtain training and validation data. The flow condition is selected to match the NACA 0008 airfoil at $\alpha = 0$ and $\mathcal{R}e = 600$ of [29]. Starting with the training and validation data for the PINNs, meshes need to be created for various airfoil geometries to run the OpenFOAM simulations on. Then, a PINN will be trained on a range of symmetric NACA airfoils, which only has the thickness parameter added as an additional input parameter. The remaining geometric parameters $m_{\text{NACA}}$ and $p_{\text{NACA}}$ have a fixed value of 0 for symmetric NACA airfoils. The accuracy of this trained PINN is compared to the

OpenFOAM validation data and reference data from literature to make sure that the OpenFOAM simulations provide sufficiently accurate data to train and validate the PINNs. Next, the full parametric case of a cambered NACA airfoil will be treated, where the PINN will have three additional geometrical parameters spanning a 3-dimensional airfoil design space. Lastly, the chapter concludes with the observations and findings on the PINN performance on the NACA airfoil cases.

## 4.1 OpenFOAM simulations

Two cases will be investigated for the NACA 4-series airfoils: the symmetric and cambered airfoils. The PINN on the symmetric NACA airfoils will be trained on the results of NACA 0008 and 0012 airfoils and validated on the NACA 0006, 0010 and 0014 airfoils. The simulation on these airfoil cases are done on coarse, unstructured meshes. Figure 4.1 shows the NACA 0008 mesh, which has 31,712 nodes, 66,092 elements. The boundary conditions are set-up

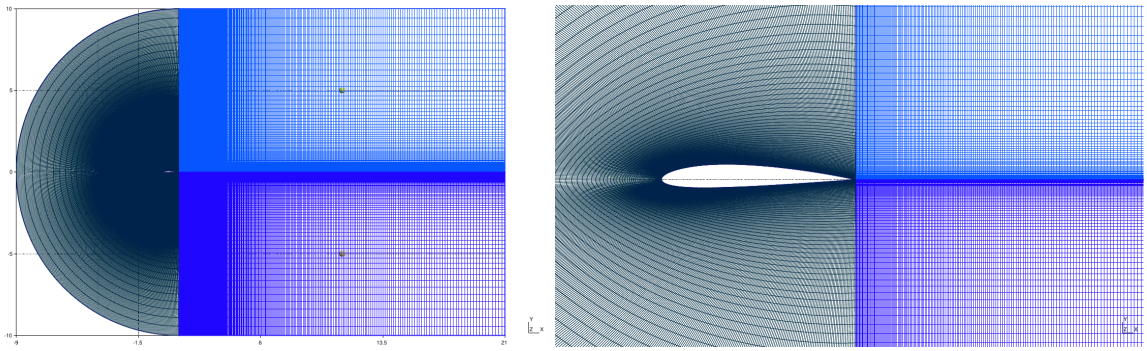


**(a)** Full view **(b)** Close view

**Figure 4.1:** Unstructured mesh of the NACA 0008 airfoil.

like in the previous chapter with $u = 1$ and $v = 0$ at the inlet on the left side, slip condition on the top and bottom boundary and $p = 0$ at the outlet on the right side. Viscosity is set to $\nu = 0.0016666667$ to obtain $\mathcal{R}e = 600$. SimpleFOAM is used as a solver for the steady flow problem.

In a similar fashion, the simulations of the cambered NACA airfoils are performed. NACA 1311, 3311, 1511, 3511, 1313, 3313, 1513 and 3513 form the training set and NACA 2412, 2413 will be used for validation. These simulations are performed on a higher resolution structured meshes. Figure 4.2 shows the mesh of the NACA 2412 airfoil with 259,199 nodes and 265,000 elements.



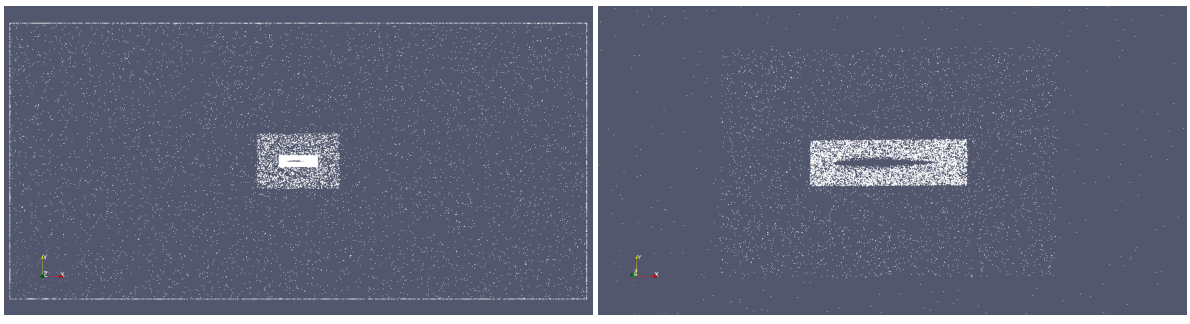
**(a)** Full view **(b)** Close view

**Figure 4.2:** Structured mesh of the NACA 2412 airfoil.

## 4.2 PINN set-up

The PINN set-up is similar to the previous chapter except for a few necessary changes to match the current flow case. Regarding architecture, it is still a fully-connected neural network with Swish activation functions. Adam optimizer settings are once again left at their default values and the same exponential learning rate decay.

The input layer will have the time parameter omitted and a number of geometrical NACA parameters take its place. In the computational domain, the spatial domain size increased to $x = [-10, 11]$ and $y = [-5, 5]$. There are 2 rectangular areas around the airfoil where the sampling density of the collocation points are increased to capture the boundary layer. The larger rectangle measures $x = [-1, 2]$ and $y = [-1, 1]$ and the smaller rectangle measures $x = [-0.2, 1.2]$ and $y = [-0.2, 0.2]$. In contrast to the case of the last chapter, this case has properly defined boundary conditions for the PINN. The inlet, top and bottom boundary will have their velocity defined as $u = 1$ and $v = 0$ and the outlet has its pressure fixed at $p = 0$. Figure 4.3 shows the boundary sample points and the interior collocation points of the symmetrical NACA 00XX case. Unlike the transient case, the OpenFOAM training sets



**(a)** Full PINN spatial domain **(b)** Close-up of the refinement zones in the airfoil area

**Figure 4.3:** Boundary and collocation points in one batch for the NACA 00XX case.

here are only weighted by the total area without any additional weight factor.

## 4.3 Symmetrical NACA 00XX airfoils

The symmetrical NACA airfoil case has only the thickness $t_{\text{NACA}}$ as an additional geometrical parameter in the input layer. It has a range of $t_{\text{NACA}} = [5, 15]$ as shown in Figure 4.4 and OpenFOAM data on NACA 0008 ($t_{\text{NACA}} = 8$) and 0012 ($t_{\text{NACA}} = 12$) are marked with red squares and will be used for training at a BPE of 10. The green circles indicate the validation samples NACA 0006, 0010 and 0014.

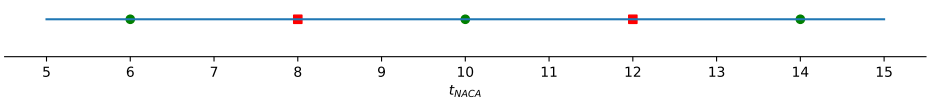



**Figure 4.4:** Design space of the NACA 00XX case.

The geometric parameter is implemented in a discrete manner allowing only integer values, resulting in 11 slices for the 11 symmetrical NACA airfoils for the set thickness range. Table 4.1 list the most important PINN parameters settings, where lr, hr and hr2 refers to the basic sampling rate, high-res sampling rate in the larger rectangle around the airfoil and the high-res sampling rate in the second, smaller rectangle around the airfoil respectively.

**Table 4.1:** Summary of the parameters of the PINN on the NACA 00XX case.

| **Parameters** | **Setting** |
|---|---|
| PINN size | 12 layers of 64 neurons |
| Computational domain | $x = [-10, 11]$, $y = [-5, 5]$ |
| Geometrical parameters | $t_{\text{NACA}} = [5, 15]$ |
| Design space samples | all NACA airfoils within design space boundaries (11) |
| Boundary conditions | inlet, top, bottom: $u = 1$, $v = 0$<br>outlet: $p = 0$<br>no-slip airfoil: $u = v = 0$ |
| PINN sampling density boundaries | inlet, top, bottom, outlet: BPA=64, BPE=1000<br>airfoil: BPA=128, BPE=1000 |
| PINN sampling density interior | lr: BPA=16, BPE=1000<br>hr: BPA=512, BPE=1000<br>hr2: BPA=16384, BPE=1000 |
| Training iterations | 150,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 600$ (by setting $\nu = 0.0016666667$) |
| OpenFOAM training data | $u$, $v$, $p$ for NACA 0008 and 0012 |
| OpenFOAM data number of batches | NACA 0008 and 0012: BPE=10 |

### 4.3.1 PINN training

The PINN has been trained for 150k iterations with the loss converging to a value of 0.01586. Figure 4.5 shows the loss development during training and is smoother compared to the Figure 3.5 and Figure 3.10 of the transient cases in chapter 3, especially in the latter 40k iterations. This loss is composed from various sources, which can be split between the losses

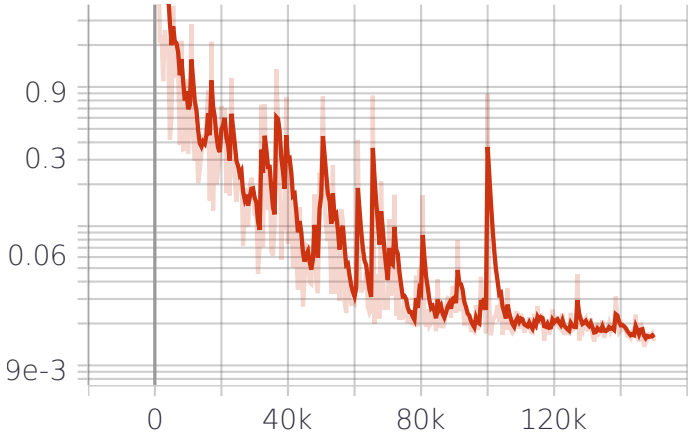
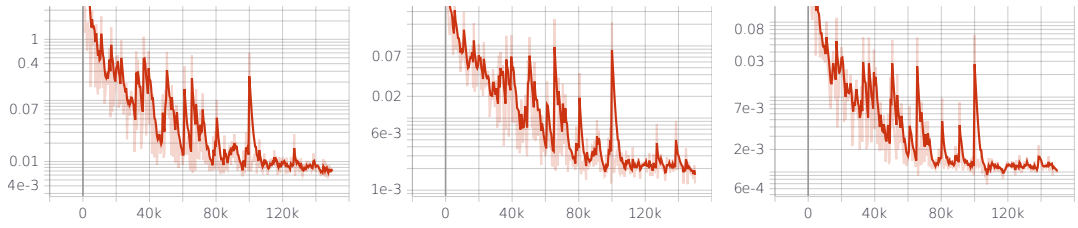



**Figure 4.5:** Total L2 loss of the PINN on the NACA 00XX case.

due to the difference in the predicted solution and training data and the losses due to the non-zero Navier-Stokes residuals. The prediction losses, shown in Figure 4.6 for $u$,$v$ and $p$ are a measure of how well the prediction matches the training data. This training data set consists of all the data on the solution that is used for training. These are the boundary conditions where data is sampled from and the additional solution data from OpenFOAM simulations. Figure 4.6a shows a larger loss for the $u$-velocity at the end of training compared to the $v$-velocity in Figure 4.6b and $p$ in Figure 4.6c. A likely reason for this, is the horizontal direction of the flow, resulting in larger range of gradient values for $u$. This makes the $u$-velocity field more complex compared to the $v$-velocity field and pressure field, thus more difficult to predict. For this reason, the focus will be on the $u$-velocity when evaluating the PINN performance in the next section and pressure and $v$-velocity will be found in Appendix B. The Navier-

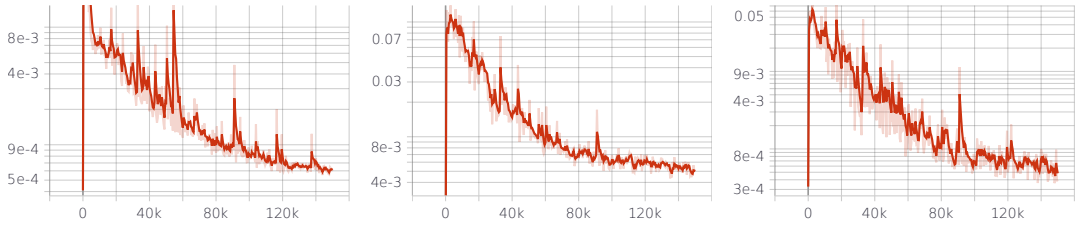



**(a)** Loss velocity $u$ **(b)** Loss velocity $v$ **(c)** Loss pressure $p$

**Figure 4.6:** L2 loss of prediction for the NACA 00XX case.

Stokes residual losses in Figure 4.7 are indicative of how well the PINN predictions satisfy the continuity and momentum equations. It can be observed that these are minimized during training and converging at a low values at the later iterations. The sharp rise of the loss value at the start is caused by the initial values of the trainable PINN parameters, as they happen

to result in a prediction that satisfy the continuity and momentum equations well. Naturally, the prediction itself shows a large deviation from the training data as can be observed from the high losses in Figure 4.6 at the start, since the PINN training is just starting. The $x$-momentum loss in Figure 4.7b ends up at a significantly higher value compared to the $y$-momentum and continuity loss for likely a similar reason as the higher $u$-velocity loss.

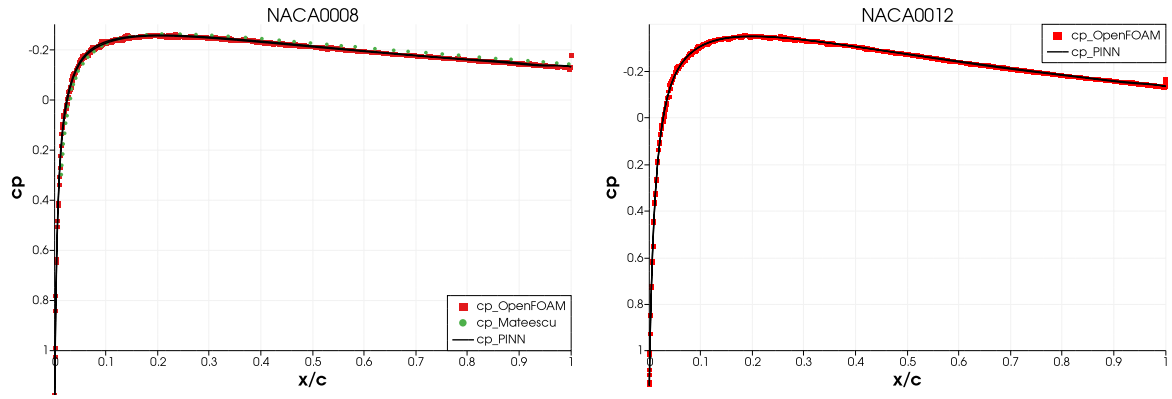



**(a)** Loss continuity **(b)** Loss momentum $x$ **(c)** Loss momentum $y$

**Figure 4.7:** L2 loss of Navier-Stokes residuals for the NACA 00XX case.

### 4.3.2 PINN results

The trained PINN is able to predict the full velocity an pressure fields for any of the symmetrical NACA airoils with a thickness between 5% and 15%. In this 1D design space, the airfoils can be divided in to 2 categories: the airfoils with OpenFOAM training data and those without. A PINN is more likely to learn to predict a good solution for airfoils if there is more data available to train it on, so it is expected that the NACA 0008 and 0012 should have accurate PINN predictions. The pressure distributions of Figure 4.8 confirm this.



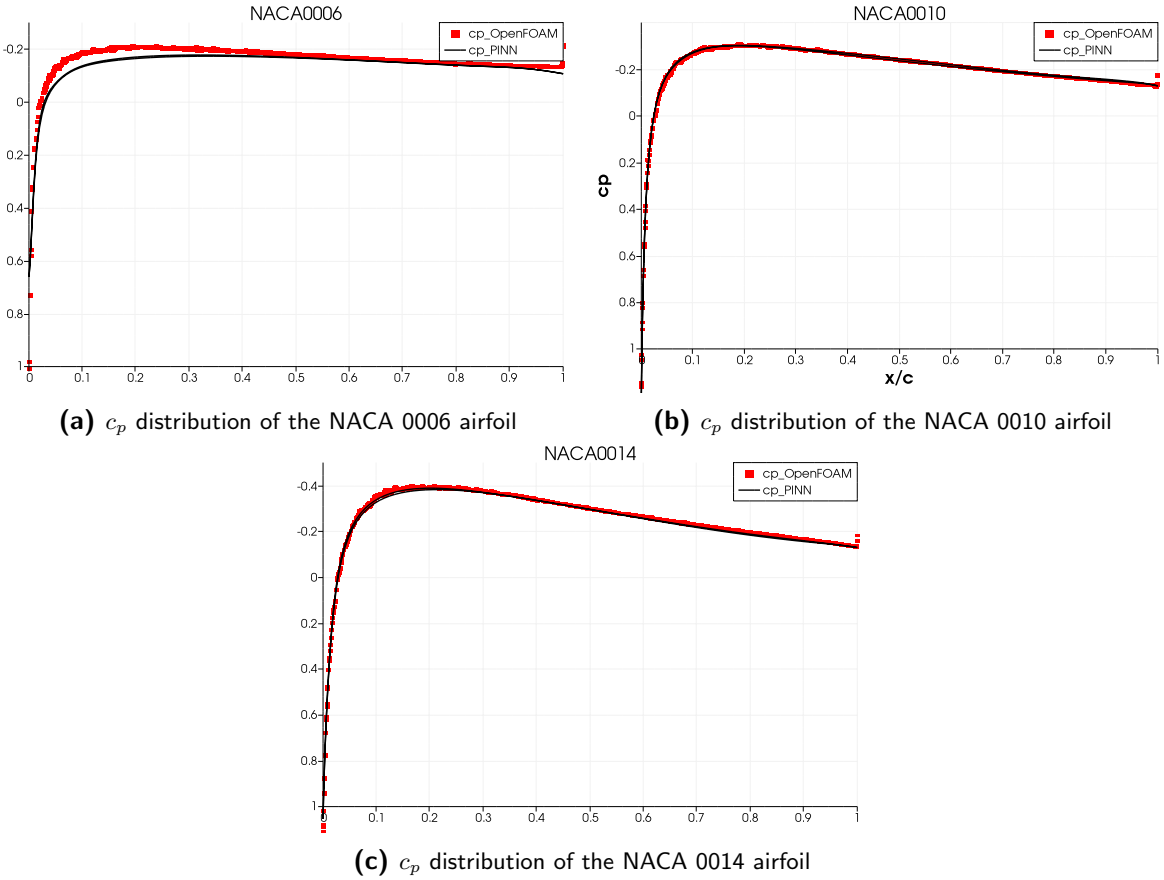

**(a)** $c_p$ distribution of the NACA 0008 airfoil **(b)** $c_p$ distribution of the NACA 0012 airfoil

**Figure 4.8:** Pressure distribution of the symmetrical NACA airfoils, which have OpenFOAM training data.

Figure 4.8a shows a comparison between the pressure distribution predicted by the PINN, the OpenFOAM simulation and the reference values from [29]. The data from [29] and the OpenFOAM simulation match closely, thus confirming that the OpenFOAM simulations are

a valid method to obtain accurate results to train and validate the PINNs. Figure 4.8b shows a good match between the OpenFOAM training data and the PINN prediction in a similar way.

Next, are the pressure plots of Figure 4.9 for the NACA 0006, 0010 and 0014 airfoils for which no OpenFOAM training data was used. For these airfoils, the PINN did not have access to the right solution during training, so these will be suited to evaluate the PINN performance on.



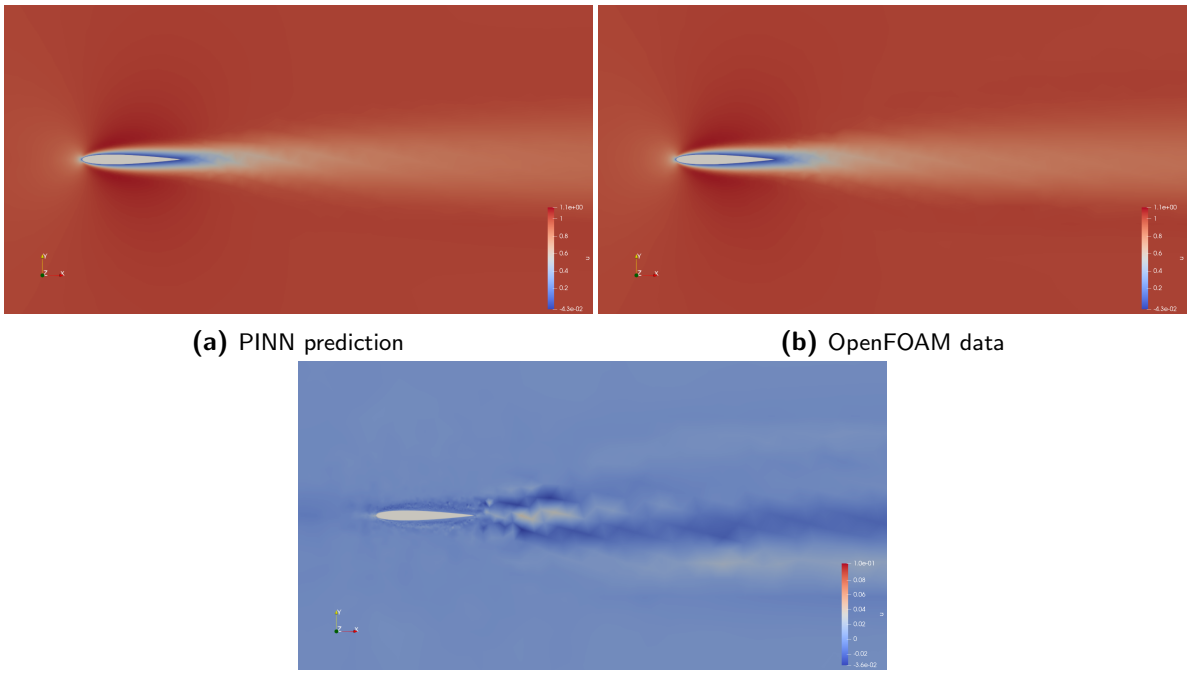

**(a)** $c_p$ distribution of the NACA 0006 airfoil

**(b)** $c_p$ distribution of the NACA 0010 airfoil

**(c)** $c_p$ distribution of the NACA 0014 airfoil

**Figure 4.9:** Pressure distribution of the symmetrical NACA airfoils, which have no OpenFOAM training data.

The PINN appears to show an accurate prediction for NACA 0010 and NACA 0014 in Figure 4.9b and Figure 4.9c, but the results for NACA 0004 airfoil show a larger difference with the OpenFOAM validation data in Figure 4.9a. Figure 4.10 shows the $u$-velocity fields of the NACA 0010 airfoil. While there is no OpenFOAM data on this airfoil present during training, it sits between the NACA 0008 and 0012 airfoils where OpenFOAM data is available. So effectively, the PINN can interpolate between these 2 points in the 1D design space resulting in an accurate prediction as the difference between Figure 4.10a and Figure 4.10b can be hardly observed. Only by directly visualizing the difference in Figure 4.10c some minor inconsistencies can be found in the wake of the airfoil.



**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

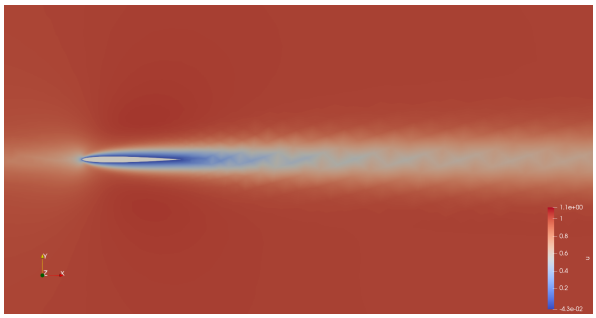
**Figure 4.10:** $u$-velocity field of the NACA 0010 airfoil.

The PINN prediction on the NACA 0006 is clearly less accurate in Figure 4.11 as the pressure plot in Figure 4.9a already suggested. Notably, the PINN underestimates the velocity in front of the airfoil, behind the airfoil and in the area around the airfoil. Unlike the NACA 0010 airfoil, the NACA 0006 airfoil only has the NACA 0008 OpenFOAM training data to the right with no other OpenFOAM data to the left. This makes it more difficult for the PINN to learn how this airfoil relates to the others as it needs learn by extrapolating the relation to the left of the NACA 0008 airfoil.

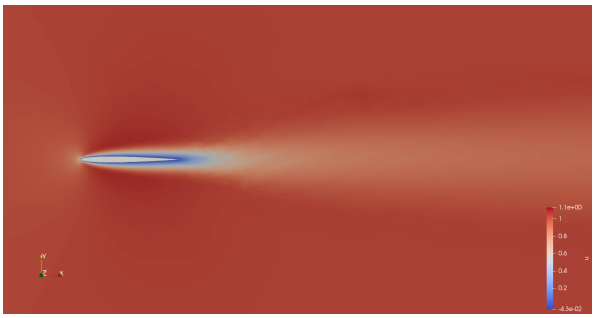
Similarly, the relation of the NACA 0014 airfoil also needs to be learned by extrapolating from the NACA 0012 OpenFOAM data. The accuracy of the prediction around the airfoil is better here as can be seen from Figure 4.9c and Figure 4.12, but the wake remains problematic.

The reason why the NACA 0014 prediction is more accurate than for the NACA 0006 airfoil, despite the seemingly similar conditions, is how much these airfoils actually differ from the nearest airfoil with OpenFOAM data. Both have a difference of 2%, however the change from 12% to 14% thickness is an relative increase of 17% while the decrease in thickness from 8% to 6% is a relative decrease of 25%. So, the NACA 0012 and 0014 airfoils are more similar to each other than the NACA 0006 and 0008 airfoils, resulting in better a better prediction for the 0014 airfoil. At the end of the day, they are both not very accurate. The PINN appears to have a harder time to learn by extrapolation than interpolation, so it would be beneficial to have OpenFOAM data on the boundaries of the design space to enclose as much of the design space as possible with the OpenFOAM training samples.

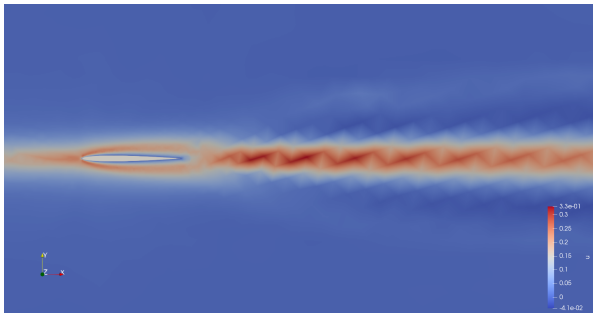
On a different note, the mesh for the symmetrical NACA airfoils on which the OpenFOAM



**(a)** PINN prediction

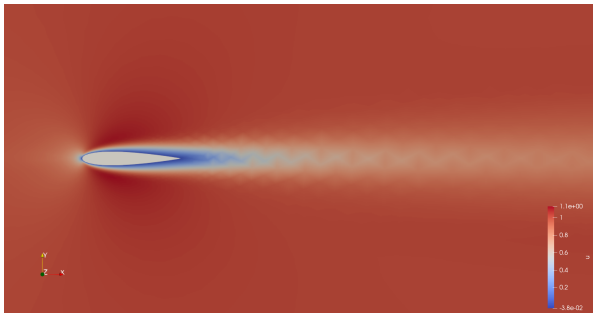


**(b)** OpenFOAM data

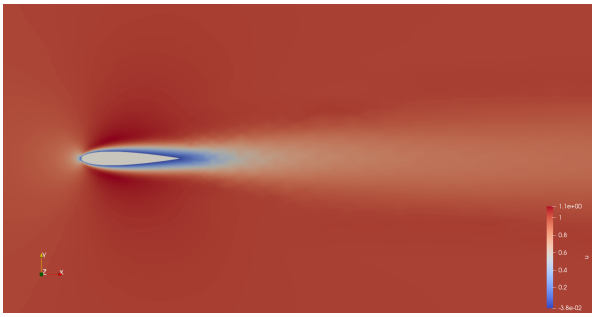


**(c)** difference

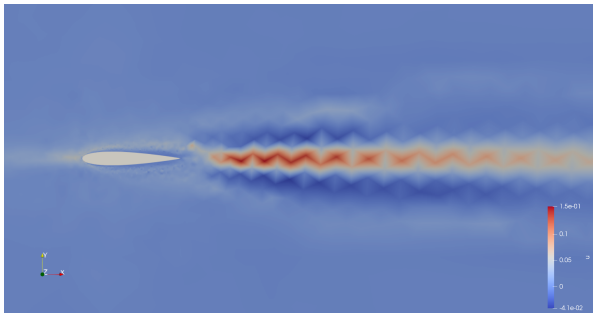
**Figure 4.11:** $u$-velocity field of the NACA 0006 airfoil.



**(a)** PINN prediction



**(b)** OpenFOAM data

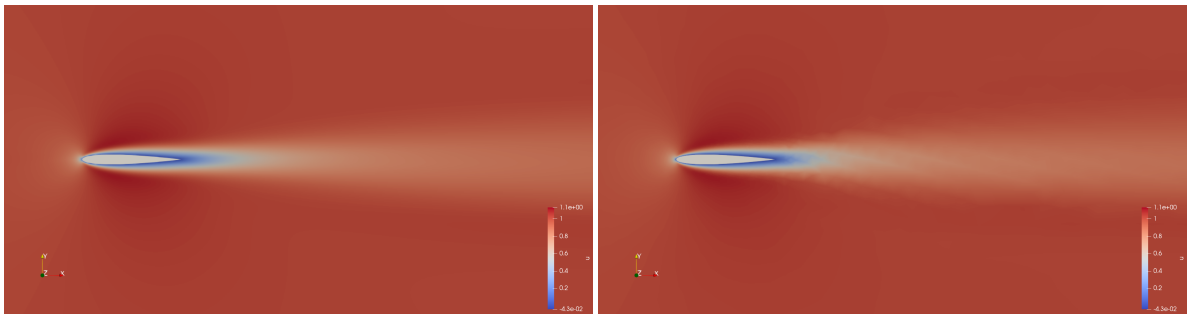


**(c)** difference

**Figure 4.12:** $u$-velocity field of the NACA 0014 airfoil.

simulations are performed are rather coarse and as a consequence some detail is lost in the wake and does not appear smooth as can be seen from Figure 4.10, Figure 4.11 and Figure 4.12.

The PINN prediction there has been performed on the same grid as the OpenFOAM simulation to make a direct comparison, however the PINN is a meshless solver that is not bound to a predefined grid. It is possible for the trained PINN to infer a solution on a finer grid, which in essence fills in some gaps of the OpenFOAM results and increases its resolution in a way that satisfies the governing equations. Figure 4.13 shows a comparison of a PINN inference on a finer grid for the NACA 0010 airfoil with the OpenFOAM simulation results on the coarse mesh.



**(a)** PINN on fine grid **(b)** OpenFOAM on coarse grid

**Figure 4.13:** Inference of the $u$-velocity of the NACA0010 airfoil on a fine grid compared to the OpenFOAM simulation.

In the end, the trained PINN is able to predict accurate solutions for the airfoils that lie in between the two OpenFOAM training samples for NACA 0008 and NACA 0012 with more detail than the original OpenFOAM simulations. However, the PINN struggles with correctly learning the relations between the airfoil geometry and flow fields through extrapolation resulting in poor accuracy for airfoils thinner than NACA 0008 and airfoils thicker than NACA 0012.

## 4.4 Cambered NACA XXXX airfoils

The cambered airfoil case of the NACA XXXX airfoils has all three NACA parameters as additional input parameters added to the PINN's input layer. This results in a parametric problem with three geometrical parameters and a 3D design space as shown in Figure 4.14.

The OpenFOAM data samples are supplied for the corners of the design space, to cover the entire volume so the geometrical relations can be learned from interpolating between the OpenFOAM samples. Validation airfoils are chosen on points in the design space where no OpenFOAM training data is used. In Figure 4.14 the OpenFOAM training samples NACA 1311, 3311, 1511, 3511, 1313, 3313, 1513 and 3513 are marked with red squares and the validation samples NACA 2412 and 2413 are marked with green circles.

For this case, the PINN set-up requires a couple of changes to adapt it to a 3D design space. The PINN size has been increased to learn a more complex problem and the collocation point

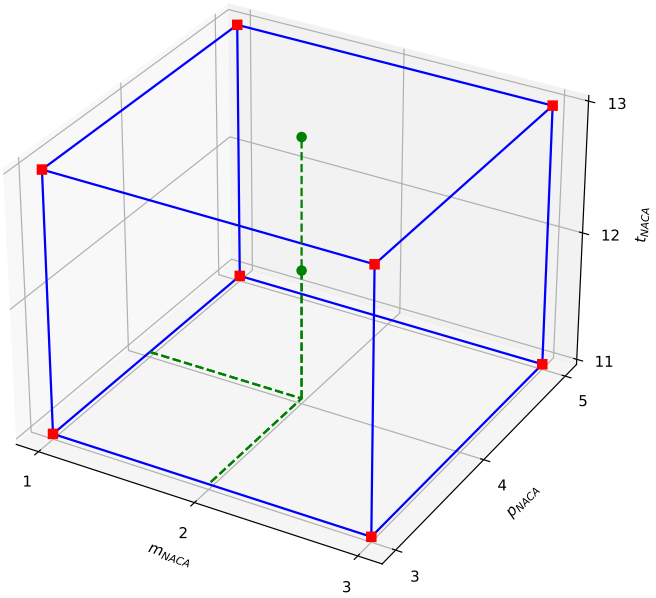



**Figure 4.14:** Design space of the NACA XXXX case.

sampling density has been reduced to limit the memory and computational burden. Table 4.2 lists the PINN settings.

**Table 4.2:** Summary of the parameters of the PINN on the NACA XXXX case.

| **Parameters** | **Setting** |
|---|---|
| PINN size | 14 layers of 96 neurons |
| Computational domain | $x = [-10, 11]$, $y = [-5, 5]$ |
| Geometrical parameters | $m_{\text{NACA}} = [1, 3]$, $p_{\text{NACA}} = [3, 5]$, $t_{\text{NACA}} = [11, 13]$ |
| Design space samples | all NACA airfoils within design space boundaries (27) |
| Boundary conditions | inlet, top, bottom: $u = 1$, $v = 0$<br>outlet: $p = 0$<br>no-slip airfoil: $u = v = 0$ |
| PINN sampling density boundaries | inlet, top, bottom, outlet: BPA=64, BPE=1000<br>airfoil: BPA=128, BPE=1000 |
| PINN sampling density interior | lr: BPA=16, BPE=100<br>hr: BPA=256, BPE=100<br>hr2: BPA=1024, BPE=100 |
| Training iterations | 150,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 600$ (by setting $\nu = 0.0016666667$) |
| OpenFOAM training data | $u$, $v$, $p$ for NACA 1311, 3311, 1511, 3511, 1313, 3313, 1513 and 3513 |
| OpenFOAM data number of batches | BPE=10 |

The spatial domain for each airfoil geometry remains the same as for the symmetrical case.

### 4.4.1 PINN training

The training of the PINN on the cambered airfoils has not shown any surprises. Figure 4.15 shows the total loss graph during training. The loss converges to a value of $6.21 \times 10^{-3}$ and

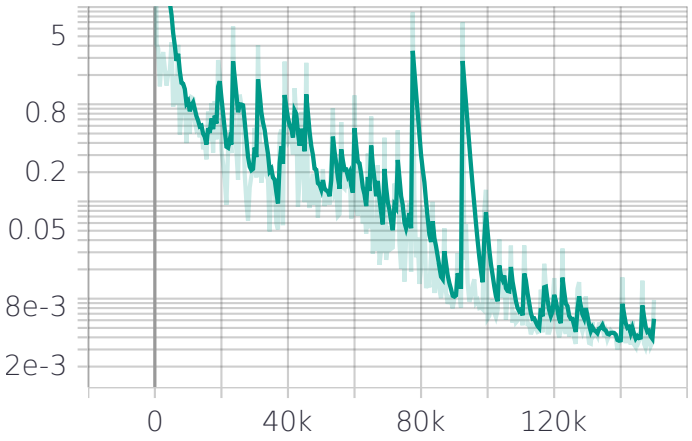
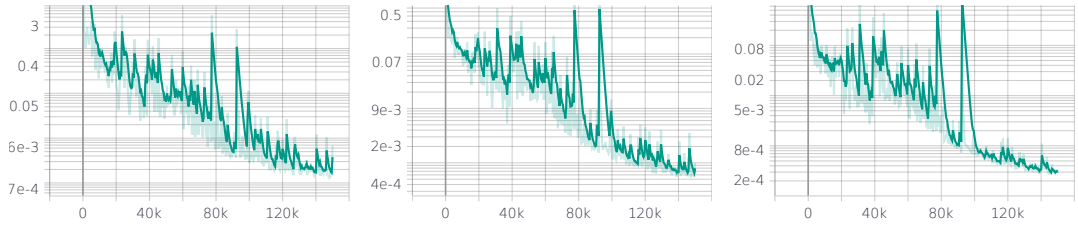



**Figure 4.15:** Total L2 loss of the PINN on the NACA XXXX case.

finishes training successfully.

The loss components due to differences in the predicted solution are shown in Figure 4.16



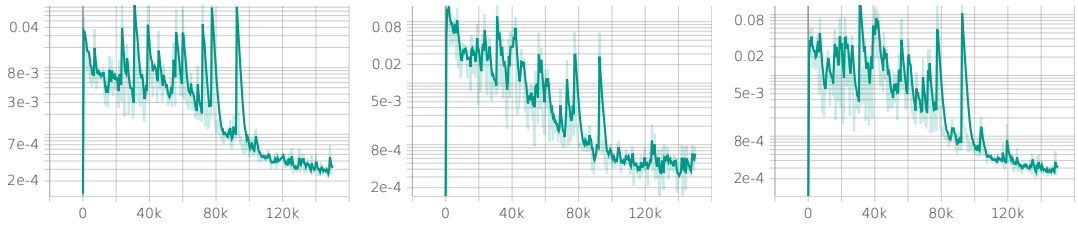

**(a)** Loss velocity $u$ **(b)** Loss velocity $v$ **(c)** Loss velocity $p$

**Figure 4.16:** L2 loss of prediction for the NACA XXXX case.

They all converge to values below $8 \times 10^{-4}$ indicating a good agreement between the training data and the PINN predictions. The $u$-velocity loss is consistently higher than the others as was also observed for the symmetrical NACA airfoils case.

The continuity and momentum equations are satisfied well with none of the losses exceeding $8 \times 10^{-4}$ in Figure 4.17.

The loss in $x$-momentum is again the highest among the three.




**(a)** Loss continuity **(b)** Loss momentum $x$ **(c)** Loss momentum $y$

**Figure 4.17:** L2 loss of Navier-Stokes residuals for the NACA XXXX case.

### 4.4.2 PINN prediction results

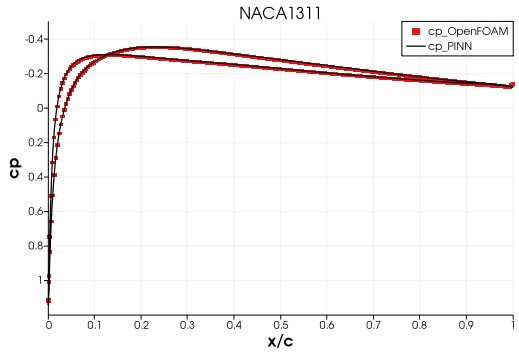
Last section's training results seem hopeful that a PINN, capable of accurately predicting the solution for any NACA airfoil within the parameter constraints, has been successfully trained. First, the pressure distributions from PINN inference will be compared the OpenFOAM results and followed by a comparison on the full $u$-velocity fields. The pressure distributions of the eight corner airfoils are shown to be matching closely in Figure 4.18. This hints at a high accuracy of the PINN in predicting the full pressure and velocity fields of these airfoils as expected for the airfoils where solution data was available during training.

The NACA 2412 and 2413 airfoil do not coincide with any of the OpenFOAM training samples. Figure 4.14 shows the NACA 2412 airfoil siting in the center of the design space, while the NACA 2413 airfoil lies on a rectangular plane formed by four OpenFOAM samples. Also for these two airfoils, the PINN predict accurate pressure distributions in Figure 4.19.

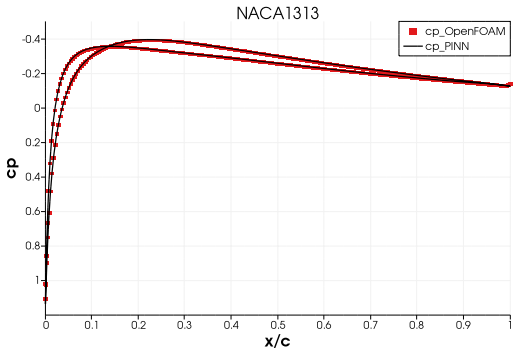
These promising results for the PINN accuracy will be confirmed, starting with NACA 3513 as a representative for the airfoil geometries where simulation data was available. Figure 4.20 shows that the PINN prediction closely resembles the OpenFOAM results to the extent that differences are hard to observe in a side-by-side comparison. When the difference between the two fields is plotted as in Figure 4.20c, the PINN appears to overestimate the velocity on upper side of the airfoil slightly while underestimating it slightly on the lower side. The same difference is seen behind the airfoil in the wake. The differences do not exceed $3 \times 10^{-2}$ results a maximum relative difference of 3% when compared tot the freestream velocity of $u_\infty = 1$.

The NACA 2412 should be more challenging for the PINN as no OpenFOAM data on this airfoil geometry was present during training. Figure 4.21 shows a comparable PINN performance here as for the NACA 3513 airfoil, where the PINN prediction seems to mirror the validation data. Similarly, Figure 4.21c show no differences larger than 3%, relative to the freestream velocity.

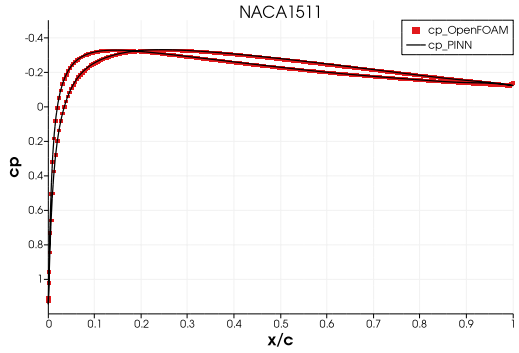
The NACA 2413 is also an airfoil geometry without OpenFOAM training data. Figure 4.22 shows closely matching $u$-velocity fields again. The plotted difference observed in Figure 4.22c is higher here compared to the two previous airfoil geometries, resulting in a relative difference reaching a maximum of 5.2%.



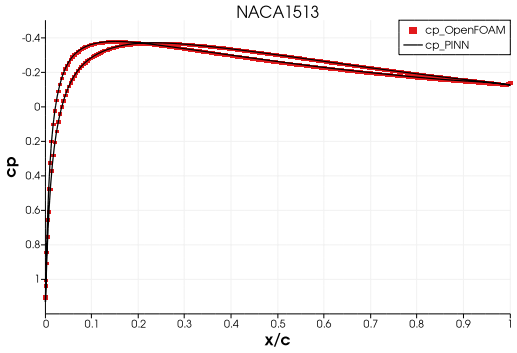

**(a)** $c_p$ distribution of the NACA 1311 airfoil



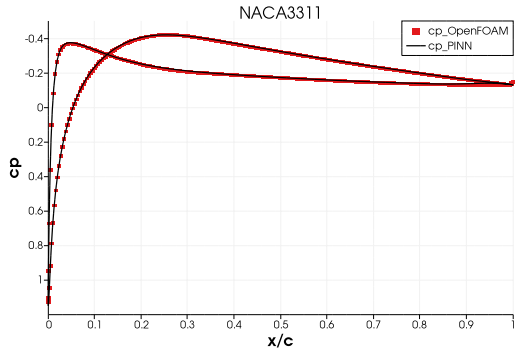

**(b)** $c_p$ distribution of the NACA 1313 airfoil



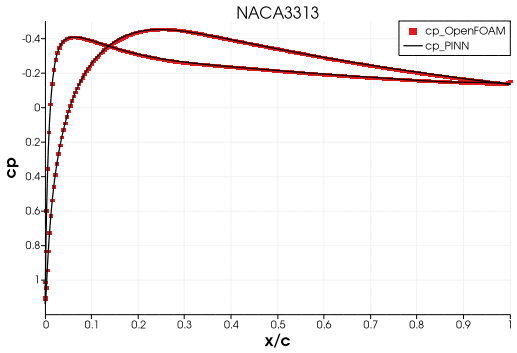

**(c)** $c_p$ distribution of the NACA 1511 airfoil



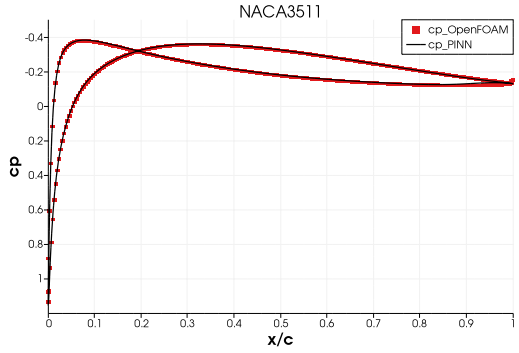

**(d)** $c_p$ distribution of the NACA 1513 airfoil



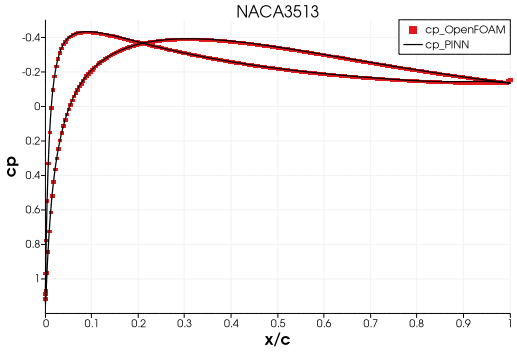

**(e)** $c_p$ distribution of the NACA 3311 airfoil




**(f)** $c_p$ distribution of the NACA 3313 airfoil

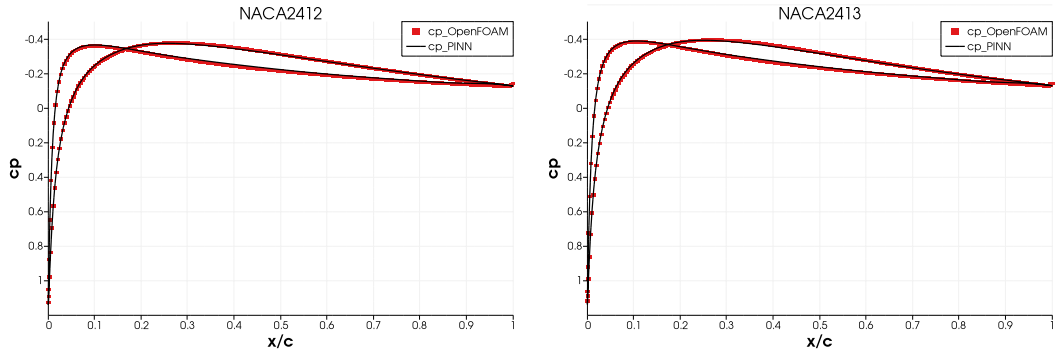



**(g)** $c_p$ distribution of the NACA 3511 airfoil



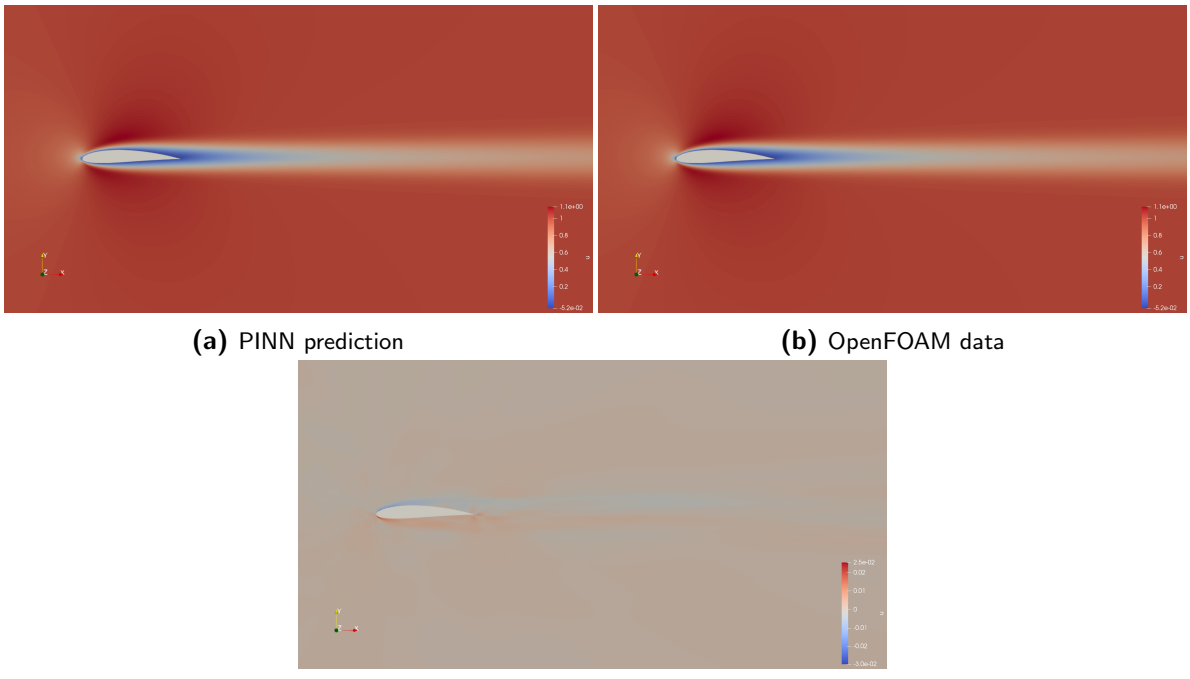

**(h)** $c_p$ distribution of the NACA 3513 airfoil

**Figure 4.18:** Pressure distribution of the cambered NACA airfoils, which have OpenFOAM training data.




**(a)** $c_p$ distribution of the NACA 2412 airfoil **(b)** $c_p$ distribution of the NACA 2413 airfoil

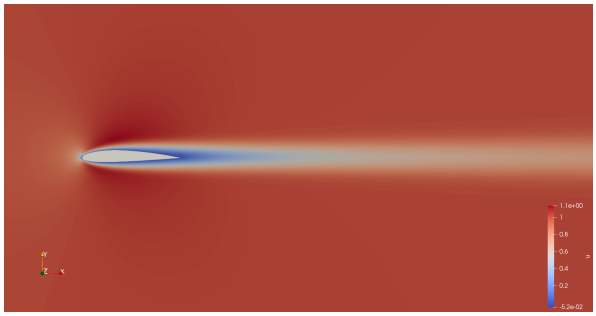
**Figure 4.19:** Pressure distribution of the cambered NACA airfoils, which have no OpenFOAM training data.



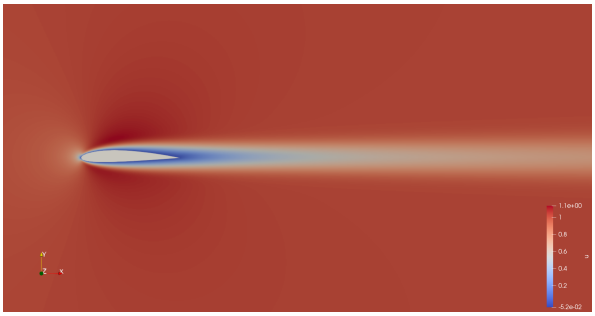
**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure 4.20:** $u$-velocity field of the NACA 3513 airfoil.



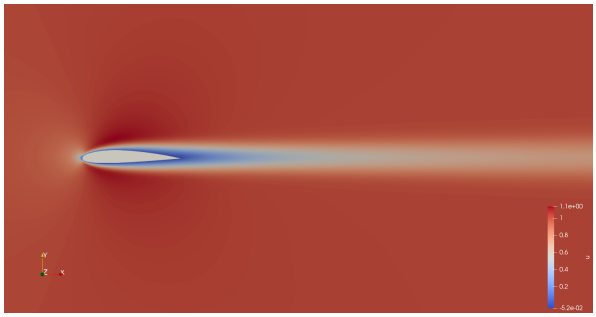
**(a)** PINN prediction



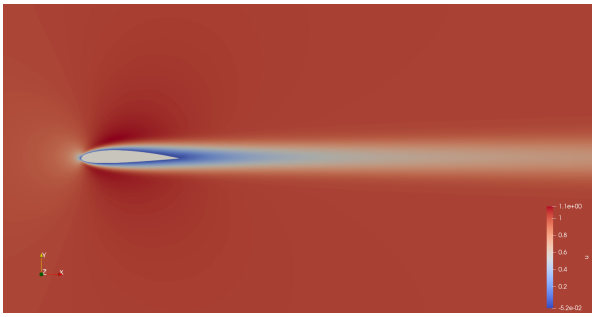
**(b)** OpenFOAM data

**(c)** difference

**Figure 4.21:** $u$-velocity field of the NACA 2412 airfoil.



**(a)** PINN prediction



**(b)** OpenFOAM data

**(c)** difference

**Figure 4.22:** $u$-velocity field of the NACA 2413 airfoil.

## 4.5 Conclusion

Two PINNs have been trained in this chapter on the flow problems for a range of symmetrical NACA airfoils and cambered NACA airfoils. Both physical information and OpenFOAM results on a limited number of design space samples are used to train the PINNs. In a comparison with [29], the OpenFOAM results have been shown to be sufficiently accurate to assume it is the physical truth for PINN training and validation. The key lesson learned from the symmetrical NACA airfoil case, is the importance of where the design space is sampled. It has been observed that the PINN is much more effective in learning the relation between the geometries in a parametric design space by interpolation than extrapolation. Thus, ensuring the boundaries are well sampled results in a higher accuracy throughout the design space for the trained PINN. Also, the trained PINN was able to infer a more detailed solution for the flow fields compared to the original OpenFOAM data sets for the symmetrical NACA airfoils that were produced on coarse meshes, because a PINN is a meshless solution method that can infer a solution on any grid. As long as the collocation points are sampled densely enough to capture all flow phenomena of interest, this can be used to fill in gaps or increase the resolution of existing data sets while making sure the continuity and momentum equations are still satisfied.

The lesson on design space sampling was used in setting up the PINN for the more complex case of the parametric cambered NACA airfoil, where the trained PINN need to predict the velocity and pressure fields of any of the cambered airfoils defined by the three NACA parameters within the parameter boundaries. So, OpenFOAM samples were generated for the eight corners for the 3D design space, enclosing the entire design space so interpolation can be used during training to learn the relation between the geometrical NACA parameters and the resulting flow fields. The trained PINN was shown to be able to infer accurate solutions for all the analyzed airfoil geometries, which demonstrates the effectiveness of a trained PINN to solve a parametric problem when provided with some boundary samples. With conventional CFD such as OpenFOAM, every geometry change requires not only a new simulation, but also a new mesh needs to be created.

# Chapter 5

# PINNs on PARSEC airfoils in steady flow

After successfully training PINNs on parametric NACA airfoils, it is time to take the next step towards a more practical problem of airfoil design. Airfoil design requires analyzing numerous airfoils with different geometries in order to find one that has the desired characteristics to fulfill a specific purpose. It is convenient to use an airfoil parameterization scheme for this, which can a represent a large variety of airfoil geometries with a number of parameters. The PARSEC method was chosen for this problem for its intuitive parameters that are directly linked to geometrical features of the airfoil.

This chapter will describe two PARSEC problems, where the first has its independent geometrical parameters to only five to limit the design space to only symmetrical airfoils. The Reynolds number of this flow is raised to $\mathcal{R}e = 1000$ to match [45], whose drag optimization results will be used as a reference. The final problem will expand the design space to 10 of the 11 PARSEC parameters and adds the airfoil angle of attack to a total of 11 independent parameters with the Reynolds number further increased to $\mathcal{R}e = 5000$, pushing the limits of the PINN and finally reaching the Reynolds number range of MAVs. OpenFOAM will be used in the same way as for the previous problems to generate training and validation data for the PINN. After training the two PINNs they will be used in a numerical optimization routine in an attempt to find the optimal airfoil geometries, according to the objective function.

## 5.1 OpenFOAM simulations

The aforementioned PARSEC problems have more geometrical parameters compared to the previous problems resulting in higher dimensional design spaces. While they contain more diverse airfoil shapes, they are more difficult to learn for a PINN, requiring a larger number of solution samples. A similar design space sampling strategy is used as in the cambered NACA case, where all the combinations of the parameter bounds are taken. For the three parameter design space it was analogous to sampling all the corners of the cube shaped design

space resulting in $2^3 = 8$ samples. The five and 11 parameter design spaces a more difficult to visualize, but the required number of sampling points for those can be computed in the same way with $2^5 = 32$ and $2^{11} = 2048$. The c-meshes are constructed in a similar fashion as for the cambered NACA case, with the semicircle section has a radius of 10 measured from its center at $(1, 0)$. The rectangular extends to a distance of 20 behind the semicircle and the airfoil lies in the semicircle with the leading edge coinciding with $(0, 0)$. Boundary conditions are also the same for the five parameter PARSEC case with the arc on the left side as the velocity inlet, slip condition on the top and and bottom side, pressure outlet on the right and no-slip condition on the airfoil perimeter. However, it is different for the 11 parameter PARSEC case as it includes the angle of attack as an independent parameter. Changes in angle of attack are implemented by rotating the inlet velocity vector. The slip boundary conditions on the top and bottom boundaries need to be adapted as a result, because the slip boundary condition would force a horizontal flow like how wind tunnel walls would in contrast to the free flow that is modeled. So, to correctly model a free flow with a positive angle of attack, the top boundary will be changed to a pressure outlet matching the right boundary and the bottom boundary will match the velocity inlet of the left boundary. In the case of a negative angle of attack the top and bottom boundaries are reversed. The full OpenFOAM simulations for the five parameter PARSEC case completed without issue resulting in 32 data sets for the boundary samples as expected. Unfortunately, of the 2048 boundary cases for the 11 parameter PARSEC case 1892 simulations converged, while 156 did not. The higher Reynolds number makes the flow less stable and flow separation may occur for some geometries even at low angles of attack. This results in unsteady flow behavior near the training edge where the flow separates and causes the steady solver to fail. To compensate for these gaps in the boundary of the design space, an additional 64 samples have been sampled from the interior using Latin Hypercube Sampling (LHS), which is a near-random sampling method that tries to distribute the samples evenly over the sample space. In addition, some extra simulations will be performed for the optimized airfoils to create additional validation data sets. Using these, the PINN performance will be evaluated in terms of accuracy. to determine the suitability of PINNs as a surrogate flow solver for airfoil design.

## 5.2 Symmetrical five parameter PARSEC airfoil case

The PARSEC method defines airfoil shapes with 11 geometrical parameters in total. This section will consider a simplified symmetrical airfoil case that reduces the number of independent parameters. Due to the symmetry, only one side of the airfoil needs to be defined as the other will mirror it. The three lower side parameters $p_5$, $p_6$ and $p_7$ will become dependent on the upper side parameters $p_2$, $p_3$ and $p_4$. Also, the trailing edge vertical position and direction need to be zero to keep the symmetry intact, so $p_8$ and $p_{10}$ will have a constant value of zero. Finally, the trailing edge is kept sharp as a blunt trailing edge possibly complicates the flow with unsteady phenomena due to flow separation. So, the trailing edge thickness $p_9$ is fixed at zero. Table 5.1 lists the relations of the 11 PARSEC parameters to the remaining five independent parameters $p_1$, $p_2$, $p_3$, $p_4$ and $p_{11}$.

The PINN set-up keeps the same size as for the cambered NACA airfoils and spatial domain,

**Table 5.1:** PARSEC parameters for the symmetrical airfoil case.

| PARSEC parameter | Geometry parameter | Description |
|---|---|---|
| $p_1$ | $r_{le}$ | leading edge radius |
| $p_2$ | $x_{up}$ | upper crest horizontal position |
| $p_3$ | $z_{up}$ | upper crest vertical position |
| $p_4$ | $z_{xx,up}$ | upper curvature at crest position |
| $p_5 = p_2$ | $x_{lo}$ | lower crest horizontal position |
| $p_6 = -p_3$ | $z_{lo}$ | lower crest vertical position |
| $p_7 = -p_4$ | $z_{xx,lo}$ | lower curvature at crest position |
| $p_8 = 0$ | $z_{te}$ | trailing edge vertical position |
| $p_9 = 0$ | $\Delta z_{te}$ | trailing edge thickness |
| $p_{10} = 0$ | $\alpha_{te}$ | trailing edge direction |
| $p_{11}$ | $\beta_{te}$ | trailing edge wedge angle |

but the three NACA parameters are replaced by the five independent PARSEC parameters. The Reynolds number has been raised to $\mathcal{R}e = 1000$ to compare the drag optimization results with [45] later. Boundary conditions remain the same as well as the sampling density and the sampling density of the interior collocation points, but with modified weighting scheme. These PINN sample points were previously only weighted to assign a uniform importance across the spatial domain. This PINN will weight them using the Signed Distance Function (SDF), which is the distance of a point to the nearest boundary. The idea behind this weighting method is to make the PINN training focus on areas that are the furthest away from boundaries first, where the flow is less complex. Thus, the easier areas are prioritized before attempting to learn the more complex areas, which may avoid local minima and result in faster and smoother convergence as is claimed in [32]. Figure 5.1 shows the SDF values plotted on the spatial domain.

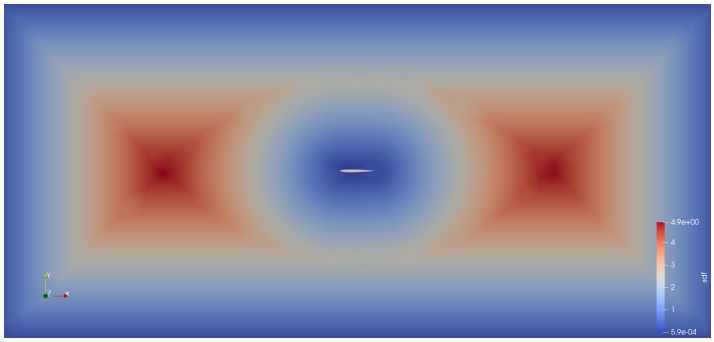


**Figure 5.1:** SDF values plotted on the spatial domain.

Unlike the NACA design space that has only a discrete number of airfoil designs, the PARSEC design space is a continuous one, requiring a decision on a sampling strategy. Taking a conservative approach, the design space is sampled in a discrete way, similar to how the boundary is sampled for the OpenFOAM simulations, except the mid point is added to the parameter ranges resulting in a total number of combinations of $3^5 = 243$. This results in

PINN only data for 243 airfoils, consisting of boundary samples and collocation points for the evaluation of continuity and momentum residuals. Of these, 32 airfoil geometries coincide with the OpenFOAM training data, resulting in 32 airfoil with complete solution data and 211 airfoils with only physical information. Also, training process is extended to 200k iterations. Table 5.2 lists the PINN settings.

**Table 5.2:** Summary of the parameters of the PINN on the five parameter PARSEC case.

| **Parameters** | **Setting** |
|---|---|
| PINN size | 14 layers of 96 neurons |
| Computational domain | $x = [-10, 11]$, $y = [-5, 5]$ |
| Geometrical parameters | $p_1 = [0.0025, 0.0075]$, $p_2 = [0.2500, 0.3500]$, $p_3 = [0.035, 0.045]$, $p_4 = [-0.1500, -0.3500]$, $p_{11} = [0.0873, 0.2618]$ |
| Design space sampling | 243 sample points, including 32 coinciding with OpenFOAM data |
| Boundary conditions | inlet, top, bottom: $u = 1$, $v = 0$<br>outlet: $p = 0$<br>no-slip airfoil: $u = v = 0$ |
| PINN sampling density boundaries | inlet, top, bottom, outlet: BPA=64, BPE=1000<br>airfoil: BPA=128, BPE=1000 |
| PINN sampling density interior | lr: BPA=16, BPE=100<br>hr ($x = [-1, 2]$, $y = [-1, 1]$): BPA=256, BPE=100<br>hr2 ($x = [-0.2, 1.2]$, $y = [-0.2, 0.2]$): BPA=1024, BPE=100 |
| Training iterations | 200,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 1000$ (by setting $\nu = 0.001$) |
| OpenFOAM training data | $u$, $v$, $p$ for the 32 PARSEC parameter boundary combinations |
| OpenFOAM data number of batches | BPE=1000 |

### 5.2.1 PINN training

The total loss progression in Figure 5.2 shows a fairly smooth curve, converging to a loss of 0.020 after 200k iterations. The PINN training was completed successfully in approximately six hours.

Loss components are shown in Figure 5.3 and Figure 5.4. In previous cases, the loss curves for the different components were very similar in shape, while here a clear differences can be observed. The loss curves for the prediction appear to be smoother than before, but the residual loss curves are rougher, especially the curves for the momentum in Figure 5.4b and Figure 5.4c. This can be explained by how the SDF weighting has increased the weights further away from the boundaries while reducing it closer to the boundaries. The easier parts of the spatial domain are prioritized first resulting in a rapid decline of the loss during the

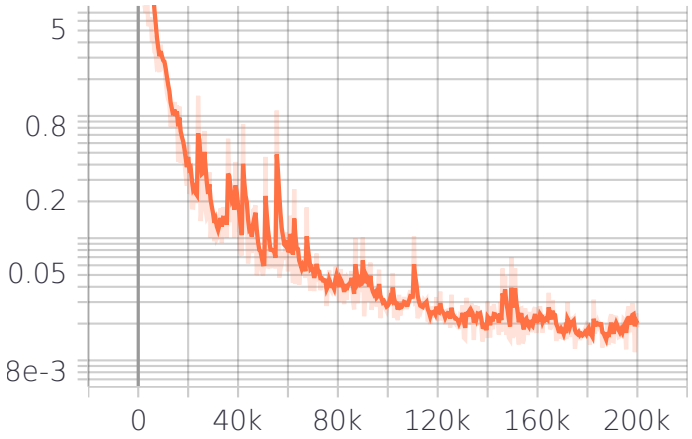
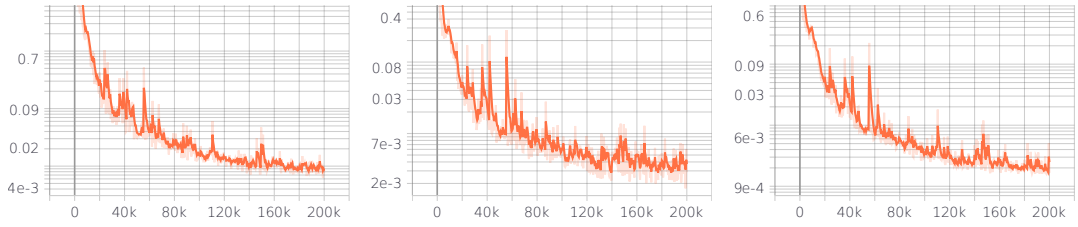


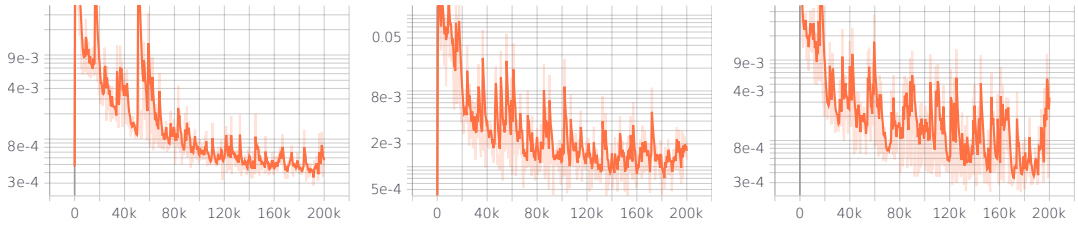

**Figure 5.2:** Total L2 loss of the PINN on the symmetrical PARSEC airfoil case.

earlier iterations, while the more difficult part near the airfoil in the boundary layer is left for the later iterations, making the oscillations in the loss curve more apparent. In this case, the use of the SDF weights seems to have a positive effect in the end, which can be seen in the next section, when the PINN accuracy will be investigated, however the SDF weights also assign lower weights to the areas where important flow features can be observed, which could lead to a lower accuracy in these areas at the end of the training process, despite the initial speed-up.




**(a)** Loss velocity $u$ **(b)** Loss velocity $v$ **(c)** Loss pressure $p$

**Figure 5.3:** L2 loss of prediction for the symmetrical PARSEC airfoil case.

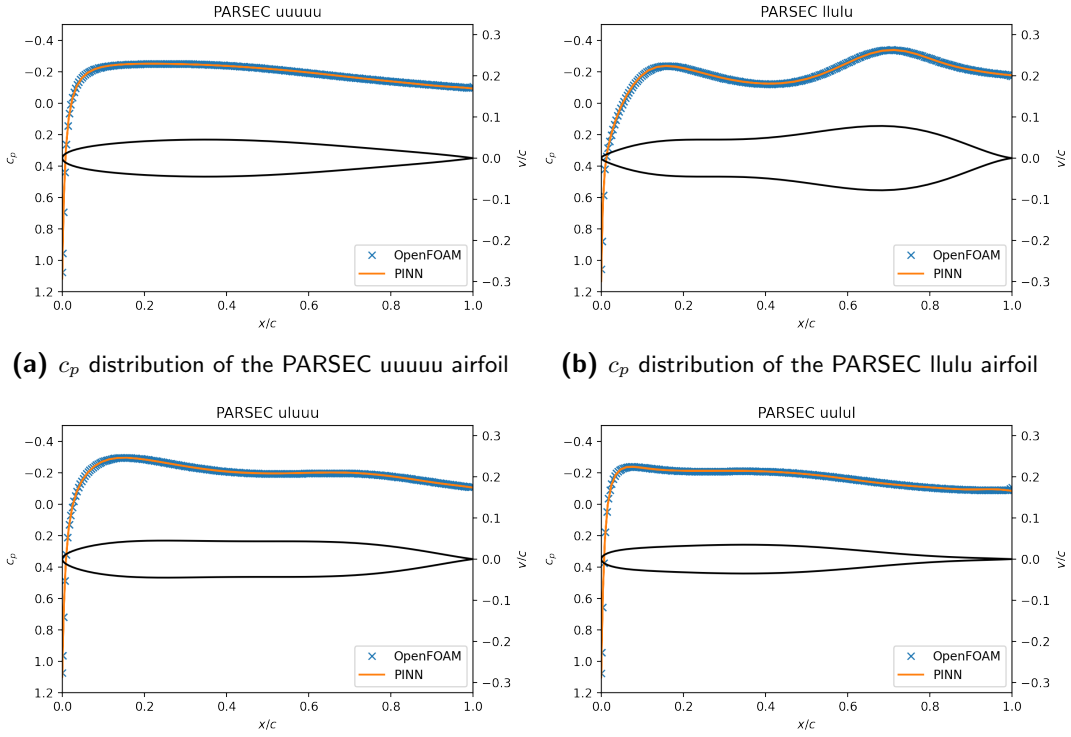



**(a)** Loss continuity **(b)** Loss momentum $x$ **(c)** Loss momentum $y$

**Figure 5.4:** L2 loss of Navier-Stokes residuals for the symmetrical PARSEC airfoil case.

### 5.2.2 PINN results

After completing the training of the PINN, it should be able to predict the flow for any airfoil described by the 5 parameters within their boundaries. The pressure plots for the 32 boundary samples and their airfoil shape are shown in Figure C.1 of section C.1. All of these PINN predictions match the OpenFOAM data closely. Figure 5.5 shows a selection of four of those airfoils as examples. The 5 letters in the airfoil name refer to the upper (u) and lower (l) bound of the independent parsec parameters in the ascending order ($p_1$, $p_2$, $p_3$, $p_4$, $p_{11}$). For example, the PARSEC uuuuu airfoil of Figure 5.5a has the PARSEC parameters (0.0075, 0.3500, 0.045, -0.3500, 0.2618).




**(a)** $c_p$ distribution of the PARSEC uuuuu airfoil

**(b)** $c_p$ distribution of the PARSEC llulu airfoil

**(c)** $c_p$ distribution of the PARSEC uluuu airfoil

**(d)** $c_p$ distribution of the PARSEC uulul airfoil

**Figure 5.5:** Pressure distribution of the 5 parameter PARSEC airfoils on the boundary.

Next, is the NACA 0008 airfoil expressed in the PARSEC parameters, which are (0.0065, 0.3006, 0.04, -0.2949, 0.1968) for the independent parameters. This airfoil coincides with neither the boundary points where OpenFOAM data is present, nor the PINN sampled points where only the physical data is available, thus making it a interesting validation point. Figure 5.6 does not show a near perfect fit as for the boundary points, but still a good fit with minor deviations.

The pressure distributions suggest a great prediction accuracy for this PINN also on the more complex five parameter PARSEC design space, but it still needs to be confirmed by inspecting the full solutions. First is the PARSEC llulu airfoil of Figure 5.5b whose prediction may be

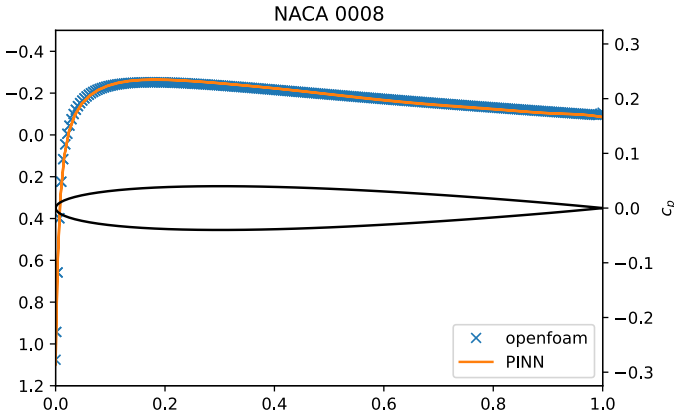
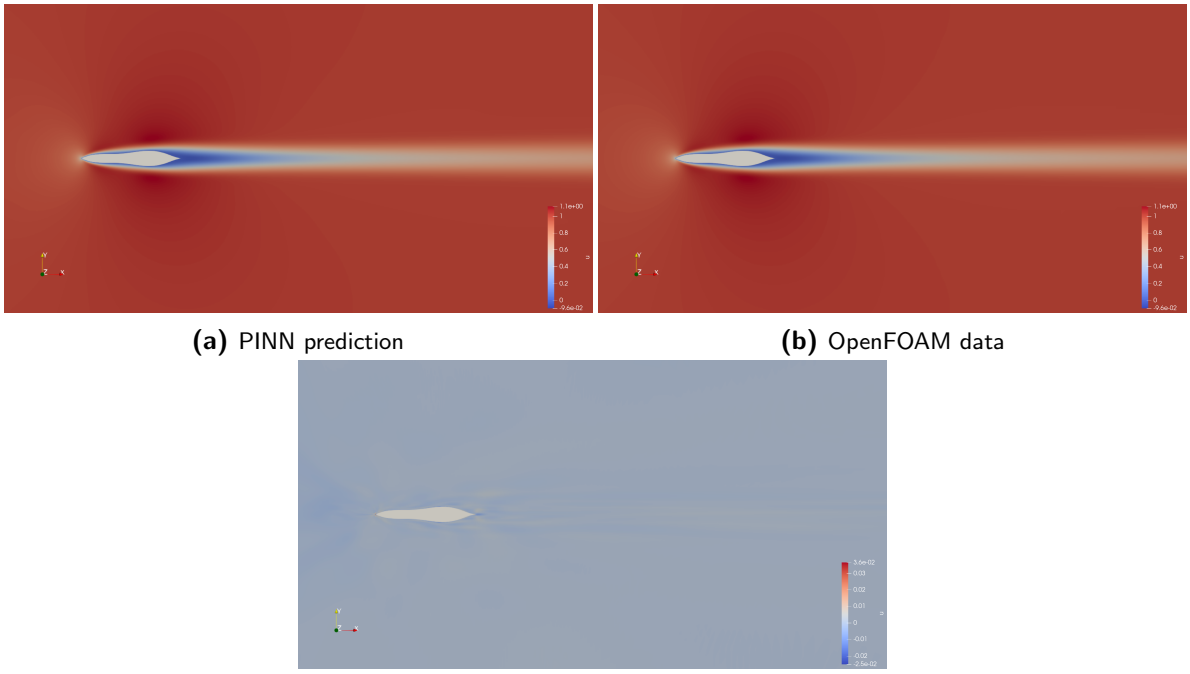



**Figure 5.6:** Pressure distribution of a NACA 0008 airfoil equivalent, expressed in PARSEC parameters.

more likely to turn out to be inaccurate due to its unusual airfoil geometry. Figure 5.7 confirms this with the $u$-velocity fields indistinguishable in a side-by-side comparison. The difference plot in Figure 5.7c appears nearly uniform, showing minimal differences. Pressure and $v$-velocity are found in Appendix C



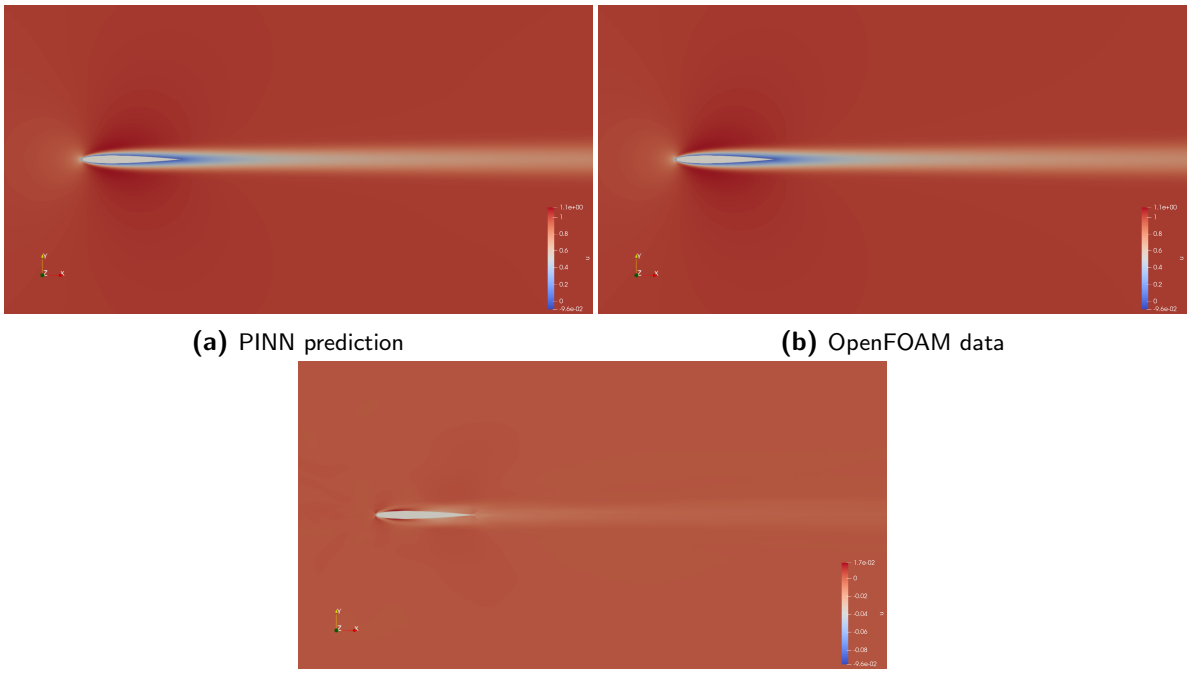
**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure 5.7:** $u$-velocity field of the PARSEC Ilulu airfoil.

Similarly, the NACA 0008 airfoil in PARSEC parameters show a close match between the PINN prediction and OpenFOAM data in Figure 5.8. The difference plot shows small differ-

ences concentrated in the area close to the airfoil where the boundary layer is and downstream of the airfoil in the wake in Figure 5.8c.



**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure 5.8:** $u$-velocity field of the NACA 0008 equivalent PARSEC airfoil.

Compared to velocity fields of last chapter such as Figure 4.21, thinner boundary layers are observed in Figure 5.8 as a consequence of the higher Reynolds number. This results in larger velocity gradients at higher flow Reynolds numbers, which makes PINN training more difficult. Despite this, the trained PINN has been shown able to predict accurate solutions within the geometrical parameter ranges and should be sufficiently accurate to serve as a surrogate model for an airfoil design optimization.

### 5.2.3 Drag optimization

The computational TensorFlow graph of the PINN in SimNet is large and bloated with numerous nodes and structures required for training or SimNet features simplifying producing, recording and analyzing results. In this form, it is difficult to implement and use the PINN as a surrogate model in an optimization loop, thus the graph needs to be exported in a more convenient form by freezing the model. This trims away all unnecessary parts of the graph till only the core remains that takes the tensor with all the spatial and PARSEC parameters and computes the output tensor with the velocity and pressure fields. The trainable parameters are converted to constants and the whole PINN model will be exported in TensorFlow's protobuff format as a singular file that can be loaded and used for inference with TensorFlow outside the SimNet framework.

The airfoil design optimization here will start with the NACA 0008 airfoil geometry and will try to improve the geometry to reduce the drag. For this case, the objective function is simply the drag coefficient as given by Equation 5.1, so the minimization of the objective functions leads to a drag minimization.

$$I(p_1, p_2, p_3, p_4, p_{11}) = c_d(p_1, p_2, p_3, p_4, p_{11}) \tag{5.1}$$

The optimization loop is implemented in Python using the L-BFGS-B optimizer of the SciPy library. A flow chart of the optimization loop is shown in Figure 5.9.

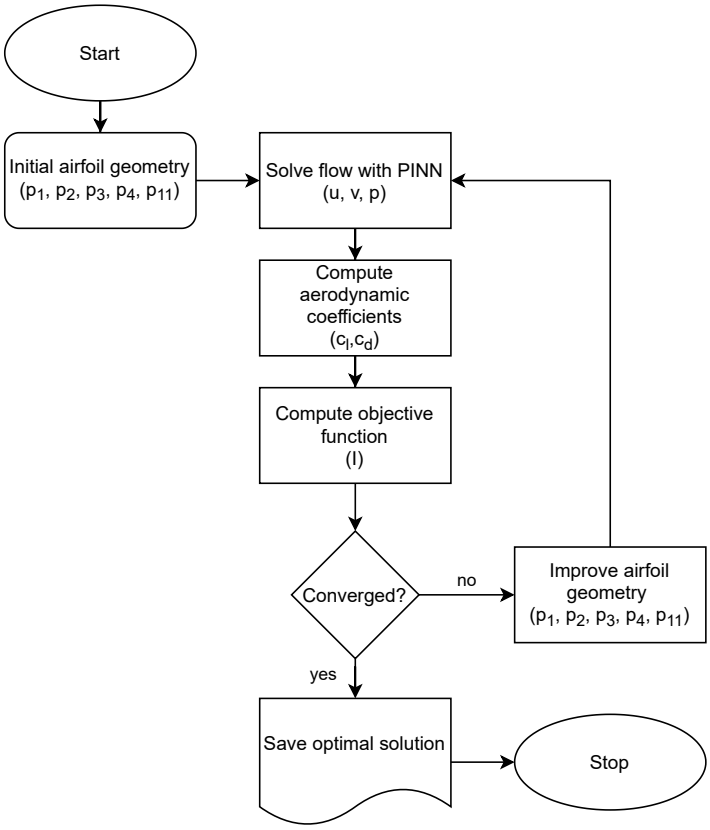



**Figure 5.9:** Flow chart of the optimization loop for the five parameter PARSEC airfoil.

Since the PINN showed accurate predictions for its entire design space, the optimization was performed under the same parameter constraints. The optimization completed with 54 function evaluations in a negligible computation time of a few seconds, resulting in the geometry shown in Figure 5.10.

Table 5.3 shows the PARSEC parameters of the optimized airfoil compared to the starting NACA 0008 airfoil. The resulting drag of 0.0982 is a decrease of 5.5% compared to the original value of 0.1039, confirming a successful optimization. From Figure 5.10 and Table 5.3 a number of geometrical changes can be observed that are directly related to the PARSEC parameters. First, the leading edge of the optimized airfoil became sharper by reducing the leading edge radius parameter to $p_1 = 0.0025$, which is the lower bound of this parameter.

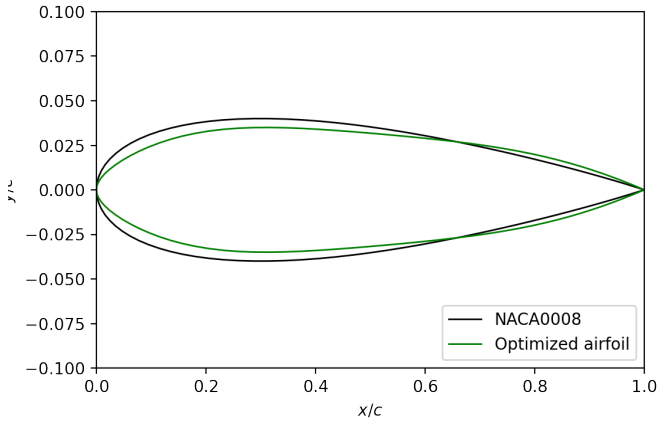



**Figure 5.10:** Airfoil geometry of the optimized airfoil.

**Table 5.3:** PARSEC parameters of the optimized airfoil.

| | $(p_1, p_2, p_3, p_4, p_{11})$ | $c_d$ |
|---|---|---|
| NACA 0008 | (0.0065, 0.3006, 0.04, -0.2949, 0.1968) | 0.1039 |
| Optimized airfoil | (0.0025 , 0.3091, 0.0350 , -0.2893, 0.2618) | 0.0982 |

Similarly, the maximum thickness has been reduced with the vertical upper crest position parameter reduced to its lower bound of $p_3 = 0.035$, which seems reasonable as it makes the airfoil more slender thus reducing its form drag. The trailing edge wedge angle has been increased to its upper bound of $p_{11} = 0.2618$ [rad]. These geometrical changes is what the numerical optimization process has found to be effective in reducing the drag.

Similar results were found in [45], where their drag optimization of the NACA 0008 airfoil had similar results and concluded that a sharp leading edge is a characteristic of optimal airfoil shapes at very low Reynolds numbers. This finding is further supported by [18], [12] and [24].

The final step is to compare the pressure distribution and flow fields predicted by the PINN to OpenFOAM simulation results of the optimized airfoil. This is to confirm that the optimization results are based on accurate predictions of the PINN and is able to function as a surrogate model with similar accuracy to the full CFD simulations with OpenFOAM at a lower computational cost. Figure 5.11 presents the pressure curve of the optimized airfoil, which matches the OpenFOAM results closely to a similar degree as for the NACA 0008 validation point in Figure 5.6.

Figure 5.12 shows the same for the $u$-velocity field with the PINN prediction and OpenFOAM results indistinguishable side-by-side. The plot of the difference between them in Figure 5.12c shows some small differences in the proximity of the airfoil where the boundary layer is present and in the wake.

Overall, the trained PINN is able to predict accurate solutions for the flow fields for all the symmetrical PARSEC airfoils within the parameter ranges. This PINN has been implemented

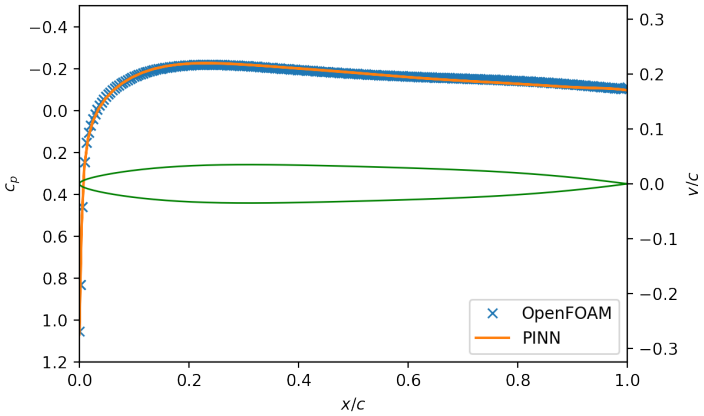
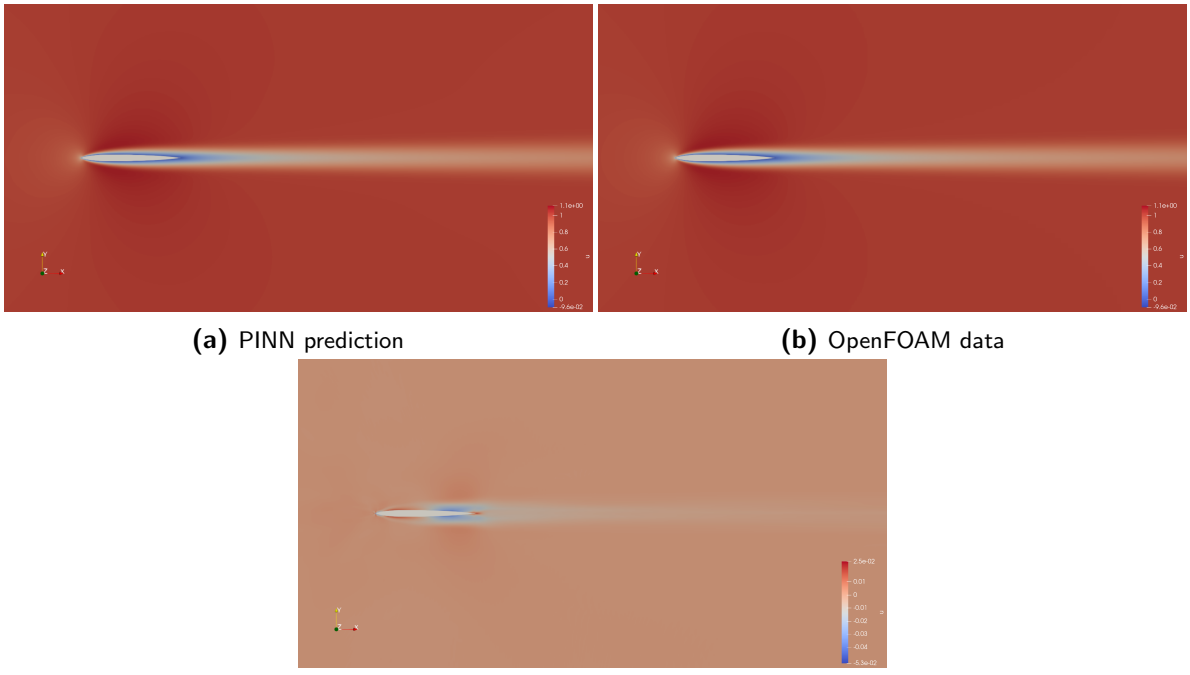



**Figure 5.11:** $c_p$ distribution of the drag optimized PARSEC airfoil



**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure 5.12:** $u$-velocity field of the drag optimized PARSEC airfoil.

in a numerical optimization loop to find an airfoil profile that has a lower drag compared to the starting geometry of the NACA 0008 airfoil. The trained PINN functions as a surrogate model that solves the flow at a lower computational cost compared to conventional CFD solvers as OpenFOAM. Inference with the trained PINN is near instantaneous and offers a tremendous speed boost over OpenFOAM, which takes up to 20 minutes to solve the flow problem per airfoil geometry. Even when accounting for the PINN training time of around six hours, a significant reduction in computational time has been achieved, considering the 54 function evaluations that were used. These would take around 18 hours with OpenFOAM,

so using the PINN shortens the computation time with a factor of three, while still retaining a high accuracy, comparable to a CFD solver. The PINN surrogate model is able to infer the complete velocity and pressure fields, whereas most conventional surrogate models are based on finding a statistical relation between the inputs and outputs. This obscures the physics of the flow and information about the flow is lost, while the PINN is trained to satisfy the Navier-Stokes equations in their full form by minimizing the Navier-Stokes residuals across the computational domain during training.

## 5.3 Full 11 parameter PARSEC airfoil case

The final case of this thesis will try to push the limits of the current PINN set-up by expanding the design space to 10 of the 11 PARSEC parameters and adds the angle of attack as a free parameter. Only $p_9$ of the PARSEC parameters remain zero to maintain a sharp trailing edge. Also the Reynolds number is increased to $\mathcal{R}e = 5000$. The OpenFOAM training data reaches a total of 1892 airfoils on the boundary and 64 airfoils that are sampled from the interior of the design space. As the problem complexity and the amount of training data increases, some changes in the PINN set-up are required. The PINN size is increased to 16 layers with 196 neurons each to raise the capacity to deal with the increased complexity. Boundary conditions of the inlet, top and bottom are dependent on the angle of attack with the velocity components prescribed as $u = \cos(\alpha)$ and $v = \sin(\alpha)$. Design space sampling also needs to be preformed in a different way, as the previous strategy does not scale well with the number of dimensions. This method would require generating $3^{11} = 177,147$ airfoils, which is a time consuming task taking a few hours and results in an excessive number of total boundary and interior sampling points exceeding memory limits. Reduction of the sampling density to a practical level results in very sparsely sampled domains of every airfoil, so in this case it is better to treat the geometrical parameters in a continuous way where the sampling points are directly randomly sampled from the full 13 dimensional space, consisting of the 2D spatial coordinates ($x$, $y$), 10 PARSEC parameters ($p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, $p_7$, $p_8$, $p_{10}$, $p_{11}$) and angle of attack $\alpha$. Near the airfoil however, sampling will still be done in a discrete manner by first randomly sampling 4096 airfoil geometries from the design space, followed by sampling physical information from the boundaries and interior for those airfoil geometries in the spatial domain. The discrete approach is taken near the airfoil to avoid having to generate the airfoil geometry for every generated sample, because they need to lie outside of the airfoil section and removed if not. Since the airfoil geometry has been parameterized, it is not constant, but depends on the geometrical parameters of the sample point. Thus the corresponding airfoil geometry needs to be generated to determine whether a sample point lies outside the airfoil or not, which is very time consuming to do for every randomly generated sampling point. For this reason, the sampling points close to the airfoil in the hr2 area is generated in batches for 4096 airfoil geometries, randomly taken from the design space. Table 5.4 summarizes the PINN settings for this case.

Loss weighting has been reverted to the original uniform weighting scheme that keeps the importance of different areas constant regardless of sampling density. The SDF weighting method resulted in worse prediction in this case, which may be caused by the higher Reynolds

**Table 5.4:** Summary of the parameters of the PINN on the 11 parameter PARSEC case.

| Parameters | Setting |
|---|---|
| PINN size | 16 layers of 196 neurons |
| Computational domain | $x = [-10, 11]$, $y = [-5, 5]$ |
| Geometrical parameters | $p_1 = [0.0020, 0.0040]$, $p_2 = [0.3000, 0.6000]$, $p_3 = [0.0200, 0.04000]$, $p_4 = [-0.2200, -0.1200]$, $p_5 = [0.3000, 0.6000]$, $p_6 = [-0.0400, -0.0200]$, $p_7 = [0.1200, 0.2200]$, $p_8 = [-0.0100, 0.0100]$, $p_{10} = [-0.0873, 0.0873]$, $p_{11} = [0.1397, 0.2618]$, $\alpha = [-3, 3]$ |
| Design space sampling | lr, hr: direct random sampling, hr2:4096 airfoils |
| Boundary conditions | inlet, top, bottom: $u = \cos(\alpha)$, $v = \sin(\alpha)$<br>outlet: $p = 0$<br>no-slip airfoil: $u = v = 0$ |
| PINN sampling density boundaries | inlet, top, bottom, outlet: BPA=64, BPE=1000<br>airfoil: BPA=128, BPE=10 |
| PINN sampling density interior | lr: BPA=16, BPE=1000<br>hr ($x = [-1, 2]$, $y = [-1, 1]$): BPA=256, BPE=1000<br>hr2 ($x = [-0.2, 1.2]$, $y = [-0.2, 0.2]$): BPA=1024, BPE=10 |
| Training iterations | 200,000 |
| Learning rate decay steps | 2000 |
| Reynolds number | $\mathcal{R}e = 5000$ (by setting $\nu = 0.0002$) |
| OpenFOAM training data | $u$, $v$, $p$ for the 1892 PARSEC parameter boundary samples and 64 interior samples |
| OpenFOAM data number of batches | BPE=1000 |

number. Compared to the previous case, the area where the boundary layer is present has grown smaller resulting in a smaller contribution to the total loss. Also, the SDF weights of the sample points in the boundary layer, which were already small due their proximity to the airfoil, will be lowered even more when the boundary layer becomes thinner. This results in a very low relative importance of the boundary layer area compared to the rest of the spatial domain, where even large errors in this area will barely affect the total loss.

### 5.3.1 PINN training

The complexity of this problem has increased significantly from the larger number of geometrical parameters, increasing the dimensionality of the problem. Also, the higher Reynolds number makes the flow problem for the individual airfoils more difficult caused by the thinner boundary layers, resulting in larger gradients. Figure 5.13 shows the overall loss decreasing at a stable rate on average till 140k iterations are reached, but training slows down after. The loss is reduced to $9.20 \times 10^{-3}$ after 200k iterations, with a training time of 19 hours.

Figure 5.14 and Figure 5.15 shows the makeup of the overall loss. The loss components show a

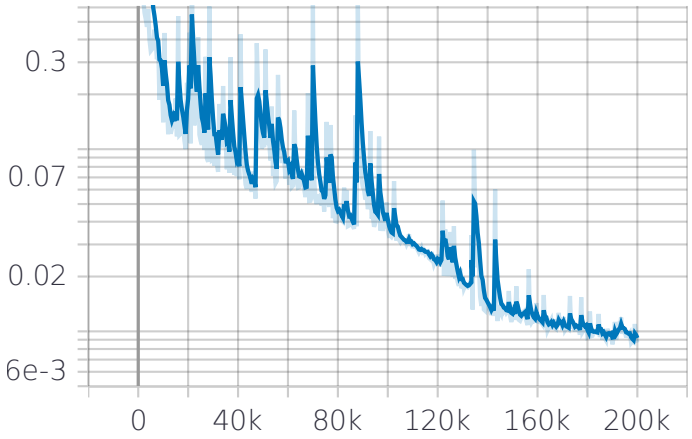
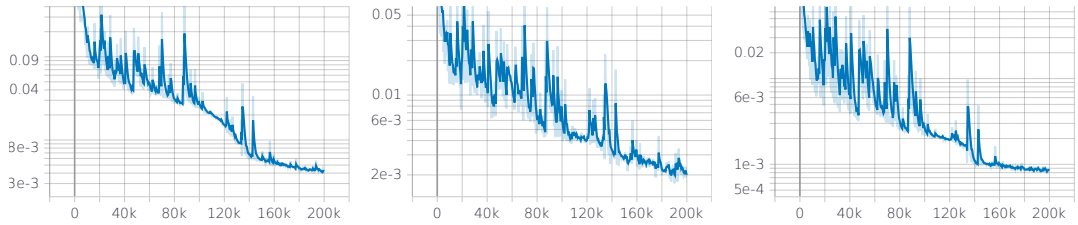


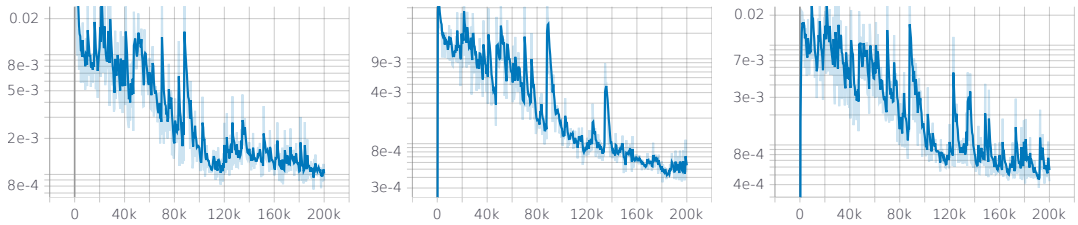

**Figure 5.13:** Total L2 loss of the PINN on the 11 parameter PARSEC airfoil case.

similar, steady loss reduction up to 140k iterations and improvements made after small. The training appears to have completed without problems and 200k iterations seem sufficient.




**(a)** Loss velocity $u$ **(b)** Loss velocity $v$ **(c)** Loss pressure $p$

**Figure 5.14:** L2 loss of prediction for the 11 parameter PARSEC airfoil case.




**(a)** Loss continuity **(b)** Loss momentum $x$ **(c)** Loss momentum $y$

**Figure 5.15:** L2 loss of Navier-Stokes residuals for the 11 parameter PARSEC airfoil case.

### 5.3.2 PINN results

As the number of training airfoil samples have increased to 1956, validating the PINN predictions on those by hand would be a tedious process. For this reason, the differences between the PINN prediction and OpenFOAM results for the pressure distribution are calculated as in Equation 5.2 and plotted in a comparison in Figure 5.16.

$$\varepsilon_i = \frac{||c_{p,OF}(x) - c_{p,NN}(x)||^2}{(\int c_{p,OF}(x)dx)^2} \tag{5.2}$$

When comparing the errors of the boundary airfoils in Figure 5.16a with the interior airfoils in Figure 5.16b, the accuracy of the PINN is noticeably better for the interior points. The prediction accuracy for the airfoils on the boundary of the design space varies, with some similar to the interior airfoils, while others have a far larger error. The three most accurate predictions are shown in Figure 5.17. Notably, all three of them lie in the interior of the design space and show an good accuracy comparable to PINNs from earlier cases.

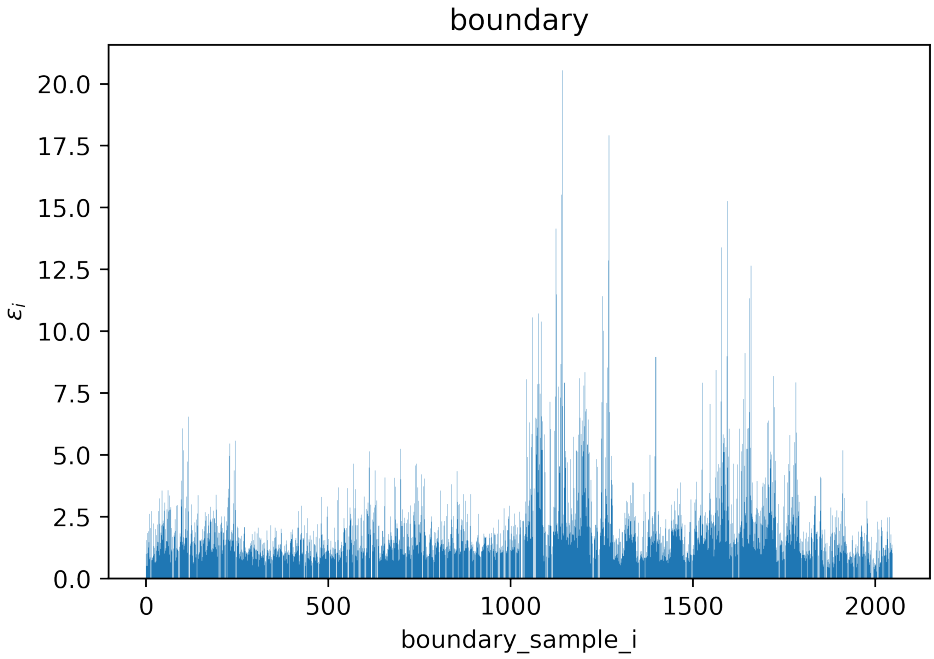
The three worst predictions in Figure 5.18, where significant differences between the PINN prediction and OpenFOAM data can be observed that are large enough to have an effect on the optimization result. When comparing the geometry of these airfoils, some common features can be found. They are thin airfoils with a concave part on the lower surface near $x/c = 0.2$ and are all located on the boundary of the design space. This hollow section may lead to flow separation at some angles of attack, which result in a unsteady flow. The PINN in this current set-up is not equipped to deal with unsteady behavior and will not likely be able to infer accurate solutions in such cases. Despite lower accuracy of some prediction compared to previous PINNs that were accurate across their design space, the PINN seems to perform fairly well in the interior of the design space and the inaccurate predictions are clustered near some parts of the boundary. As the PINN appears to be decently accurate for the most part of the design space, it should still be able to function as the flow solver for the optimization.

### 5.3.3 Lift optimization

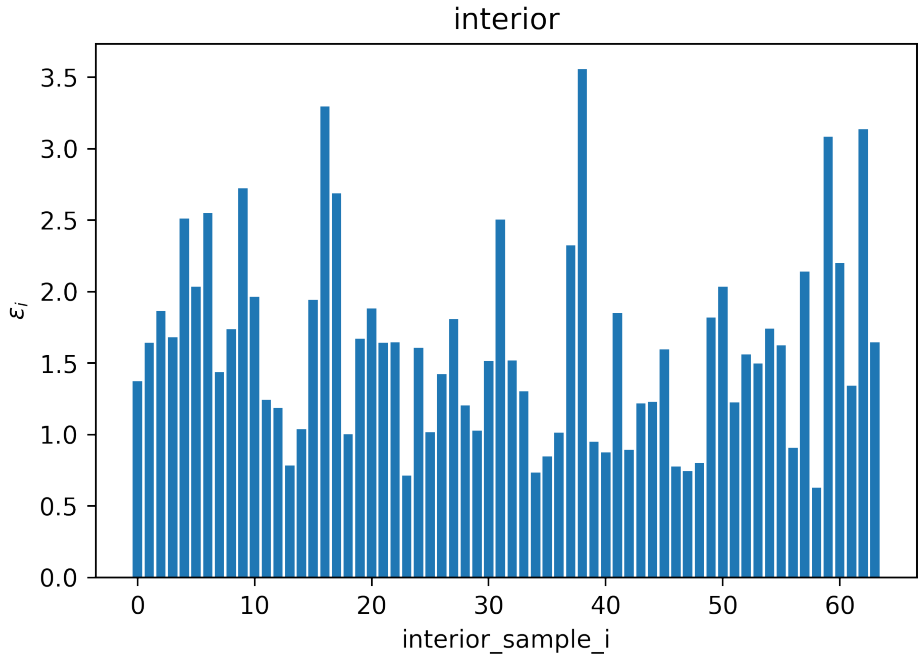
The airfoil geometry will this time be optimized for lift, while minimizing the drag. Also, a minimum lift coefficient of $c_{l,ref} = 0.0705$ will be added as an constraint, which is the lift coefficient of the initial airfoil geometry. This section will first perform a design optimization with one starting geometry, like for the symmetrical PARSEC case. The initial airfoil geometry, where the optimization starts, is the airfoil geometry at the center of the design space. For this optimization, the angle of attack will be set to a constant $\alpha = 1$ [deg], resulting in a design space with 10 free PARSEC parameters. The objective function of this optimization, adapted from [41], is given in Equation 5.3.

$$I = -0.5c_l^2 + 0.5c_d^2 + 40 \cdot \max(0, c_{l,ref} - c_l)^2 \tag{5.3}$$

Here, the first and second term will maximize lift and minimize drag and the last term is a exterior penalty function that increases the loss if the lift coefficient drops below the minimum reference lift coefficient, but is zero when the lift coefficient is higher than the minimum required value. The coefficient of 40 sets the steepness of the penalty function when the $c_l$ drops below $c_{l,ref}$. The final optimization will generate 20 random airfoil geometries to start the optimization with. This will take advantage of the fast PINN prediction speed to look into the effects of different starting points on the resulting optimized airfoil shapes.

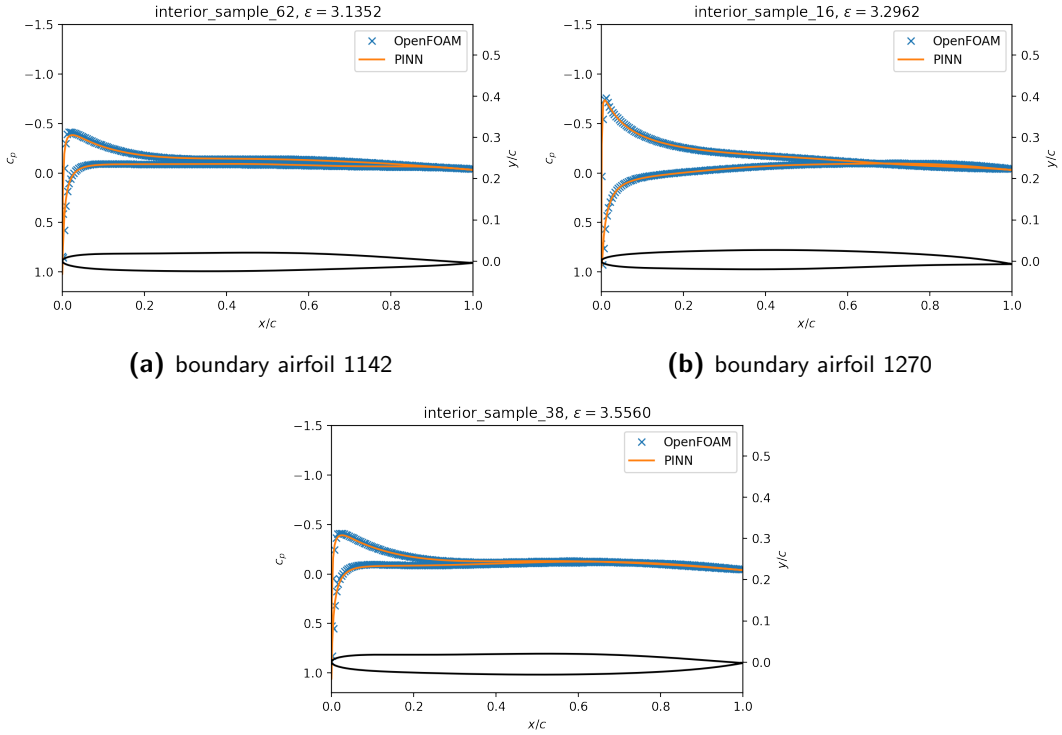



**(a)** boundary airfoils




**(b)** interior airfoils

**Figure 5.16:** Normalized error of the airfoil pressure distribution for the 11 parameter PARSEC airfoil case.




**(a)** boundary airfoil 1142

**(b)** boundary airfoil 1270

**(c)** boundary airfoil 1140

**Figure 5.17:** Pressure distribution of airfoils with the smallest deviation from OpenFOAM data.

**Table 5.5:** The 10 PARSEC parameters of initial center airfoil and resulting optimum with their lift and drag coefficient.

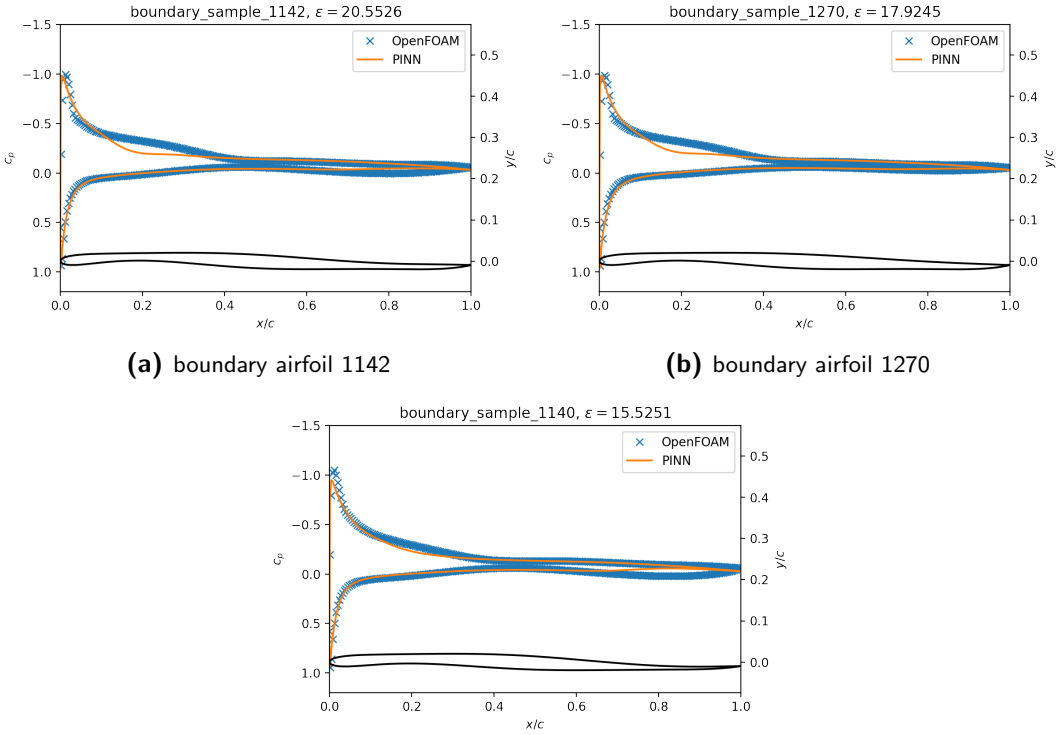
| | $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11})$ | $c_l$ | $c_d$ |
|---|---|---|---|
| Initial center airfoil | (0.0030, 0.4500, 0.0300, -0.1700, 0.4500, -0.0300, 0.1700, 0, 0, 0.2007) | 0.0705 | 0.0433 |
| Optimized airfoil | (0.0023, 0.4644, 0.0400, -0.1646, 0.4810, -0.0245, 0.1735, 0.0006, 0.0025, 0.2067) | 0.0860 | 0.0434 |

### Optimization with one starting geometry

The optimization took 583 function evaluations to complete and Table 5.5 presents the resulting parameters and aerodynamic coefficients.

Figure 5.19a shows the geometries of the starting and optimized geometry, corresponding to the parameter values. The net effect of the optimization seems to be an increased camber compared to the starting airfoil. This results in a higher lift coefficient, while the drag coefficient remains nearly the same as can be seen from the values in Table 5.5.

An additional OpenFOAM simulation has been performed on the optimized airfoil geometry to investigate the PINN prediction accuracy. Figure 5.19b shows a comparison between the pressure distribution of the airfoil inferred by the PINN and OpenFOAM. The results

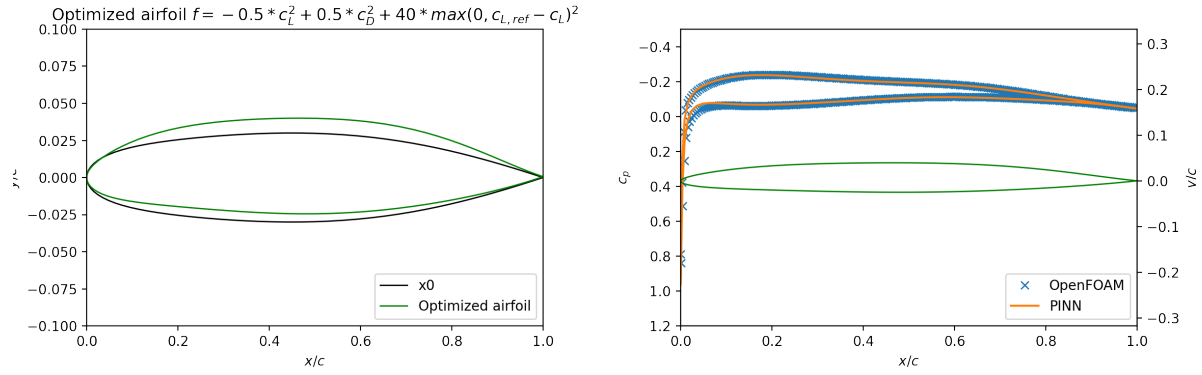



**(a)** boundary airfoil 1142

**(b)** boundary airfoil 1270

**(c)** boundary airfoil 1140

**Figure 5.18:** Pressure distribution of airfoils with the largest deviation from OpenFOAM data.

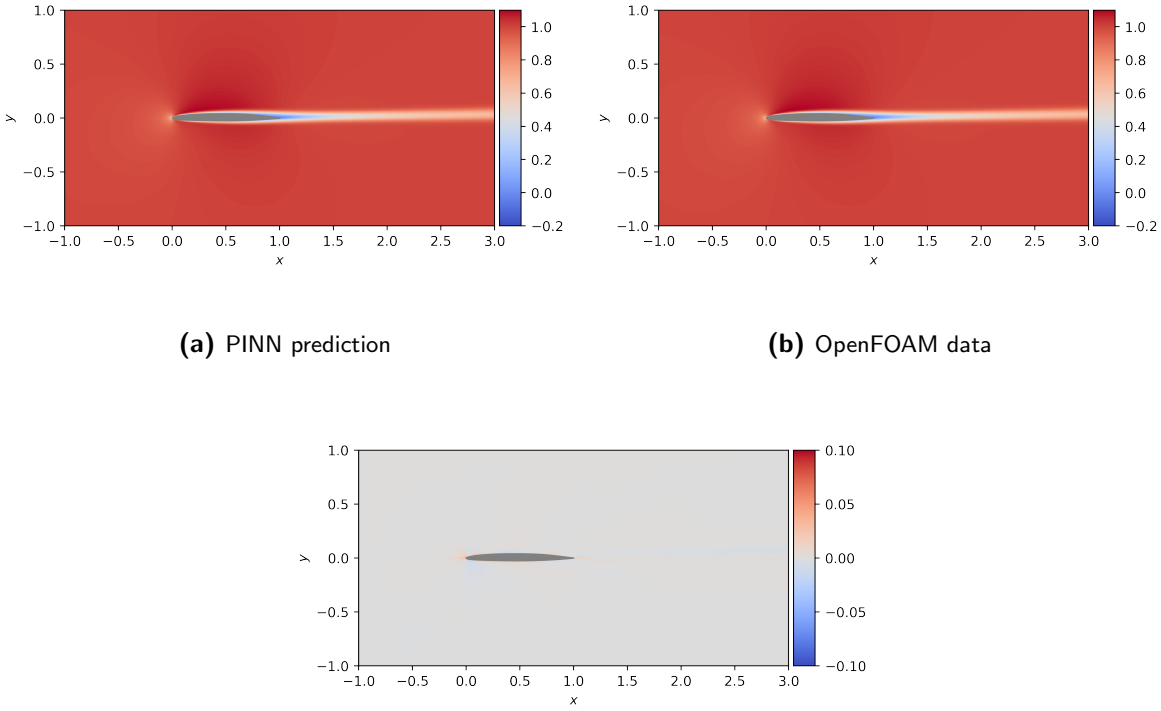



**(a)** optimized geometry

**(b)** $c_p$ distribution

**Figure 5.19:** Optimized PARSEC airfoil with the design space center as initial geometry.

are mostly the same, but an error can be observed on the curve for the lower side of the airfoil near the leading edge. Here, the PINN predicts a lower pressure compared to the OpenFOAM simulation. Like discussed in the previous section, the accuracy of this PINN has a comparatively lower accuracy than the previous PINNs on the less complex problems that have been able mirror the OpenFOAM simulations near perfectly. Overall, the discrepancy is still acceptable, as the PINN gets the shape of the curve largely correct. Next, the $u$-velocity field will be inspected in Figure 5.20. Figure 5.20a and Figure 5.20b actually look very similar




**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure 5.20:** $u$-velocity field of the lift optimized PARSEC airfoil with initial geometry in the center of the design space.

with no discernible difference. However, Figure 5.23c reveals the inconsistencies taking place in close proximity of the airfoil where the boundary layer is present are the highest. This was something something already observed earlier in the pressure plot of Figure 5.19b and highlights the issue with thinner boundary layers at higher Reynolds numbers. With uniform weighting by area, a thinner boundary layer results in a lower amount of samples inside the boundary layer area, thus a lower importance of the boundary layer area during training, relative to the rest of computational domain. Consequently, the errors in the boundary layer remain the highest as it is also the most difficult area for the PINN to predict accurately due to the large gradients. Unfortunately, this area is also the most important for computing the aerodynamic coefficients as they are computed from integrating quantities along the airfoil perimeter.

### Optimization with 20 random starting geometries

The final optimization will start the optimization not with only one airfoil geometry, but 20 randomly selected airfoil geometries to find the global optimum. Using the high inference speed of the PINN, the full optimization completed in 87 seconds with a total of 8107 function evaluations. Of the 20 starting geometries, five failed to converge, suggesting a complex

**Table 5.6:** The 10 PARSEC parameters of initial airfoil 11 and resulting optimum with their lift and drag coefficient.

| | $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11})$ | $c_l$ | $c_d$ |
|---|---|---|---|
| Initial center airfoil | (0.0027, 0.3581, 0.0292, -0.1280, 0.5637, -0.0349, 0.1548, -0.0063, 0.0701, 0.2259) | 0.1097 | 0.0428 |
| Optimized airfoil | (0.0020, 0.3000, 0.0400, -0.1272, 0.6000, -0.0200, 0.1719, -0.0100, 0.0861, 0.2298) | 0.1839 | 0.0378 |

optimization landscape. The successfully optimized geometries are plotted in Figure 5.21. Despite the 20 random starting points, the optimized airfoils have limited variation. They

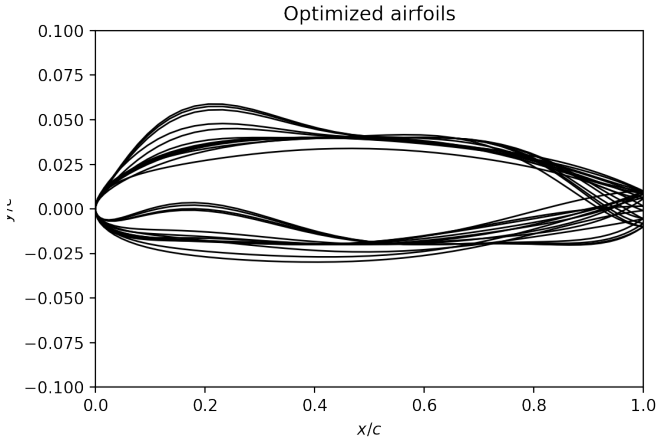


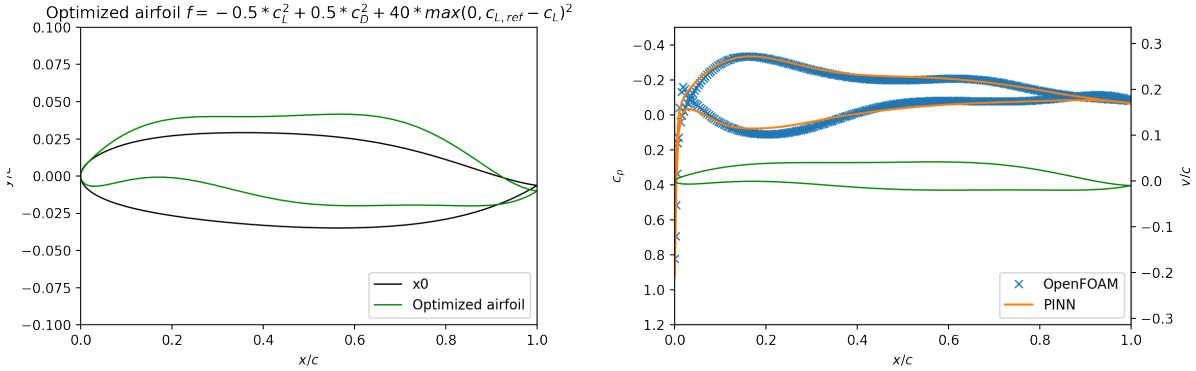

**Figure 5.21:** Converged airfoil geometries.

either fall in the category of airfoils with the more unusual shape with a concave part on the bottom surface near the leading edge or the more common cambered airfoils. Starting with the 11th geometry, belonging to the former category, has the lowest objective function value after optimization. Its parameters and those of the optimized are shown in Table 5.6 together with the lift and drag.

The corresponding profiles are shown in Figure 5.22a and the pressure distribution in Figure 5.22b. This airfoil geometry is close to the boundary as can seen from the parameter values, of which many are at or close to their limits leading to mediocre PINN accuracy. In particular, the accuracy of the prediction is the least accurate on the bottom surface where the PINN has difficulty with the flow in the concave area and near the trailing edge on the top surface.

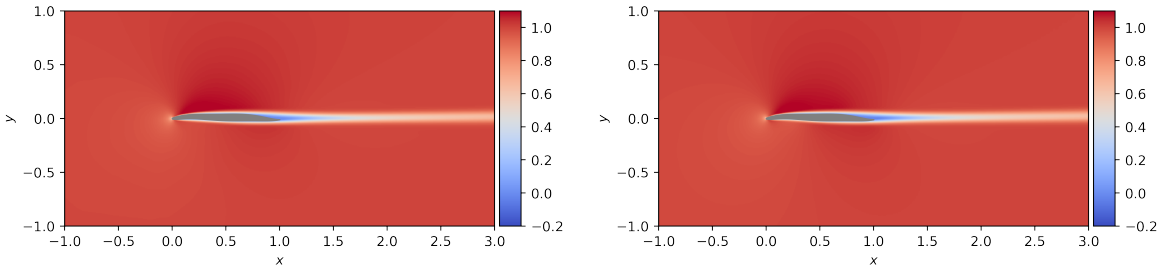
Figure 5.23c confirms this with clear errors visible in wake and close to the airfoil in the boundary layer area, while the predicted velocity field of Figure 5.23a appears fine compared tot the OpenFOAM result of Figure 5.23b.

The flow appears to separate near the trailing edge on the top surface, which may lead to vortex shedding. This is not within the steady flow assumption of this case with this current

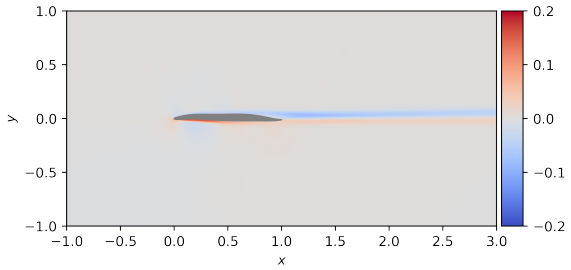



**(a)** optimized geometry **(b)** $c_p$ distribution

**Figure 5.22:** Optimized PARSEC airfoil with sample 11 as initial geometry.




**(a)** PINN prediction **(b)** OpenFOAM data




**(c)** difference

**Figure 5.23:** $u$-velocity field of the lift optimized PARSEC airfoil with initial geometry 11.

set-up resulting possible inaccurate results for the PINN. Flow separation and vortex shedding is likely to affect the airfoil performance negatively, so the lift and drag values from Table 5.6 may be too optimistic.

Thus this optimized design and the other similar shapes will be set aside to focus on airfoils

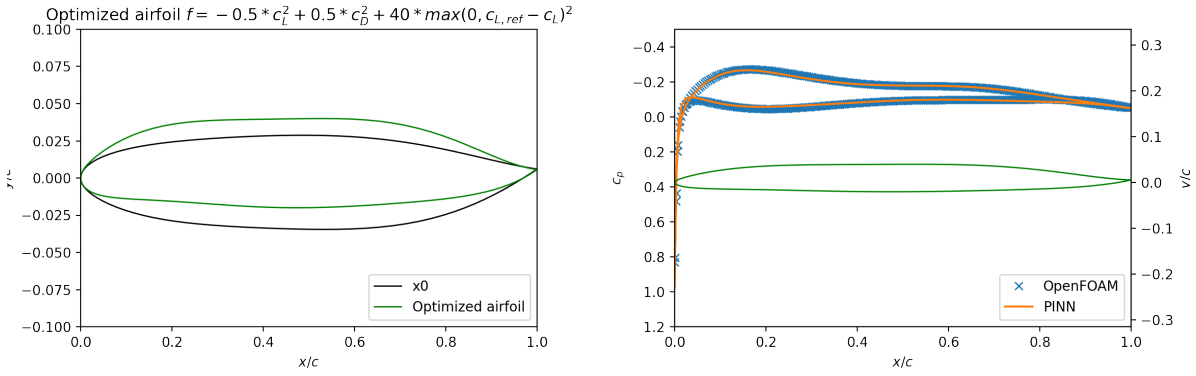
**Table 5.7:** The 10 PARSEC parameters of initial airfoil 8 and resulting optimum with their lift and drag coefficient.

| | $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11})$ | $c_l$ | $c_d$ |
|---|---|---|---|
| Initial airfoil | (0.0024, 0.4866, 0.0288, -0.1415, 0.5340, -0.0345, 0.1476, 0.0060, 0.0800, 0.2466) | 0.0442 | 0.0429 |
| Optimized airfoil | (0.0025, 0.5297, 0.0400, -0.1308, 0.4747, -0.0200, 0.1604, 0.0057, 0.0873, 0.2487) | 0.0872 | 0.0431 |

that do not cause problematic flow behavior.

The best performing airfoil among the remaining is the result from the eighth starting geometry, whose parameters are presented in Table 5.7.

As the parameter values show, it is further away from the design space boundary than the previously discussed airfoil, where the PINN should predict more accurate solutions. Figure 5.24 confirms this indeed the case with a good match between the prediction and the OpenFOAM results. The airfoil shape is similar to the initial optimization, starting from the design space center. This optimum has a comparable sharp leading edge and a positive camber, resulting in similar lift and drag coefficient values. The lift coefficient is slightly higher at a value of 0.0872 compared to the previous 0.0860 and the drag coefficient sees a minor improvement to 0.0431 from 0.0434.

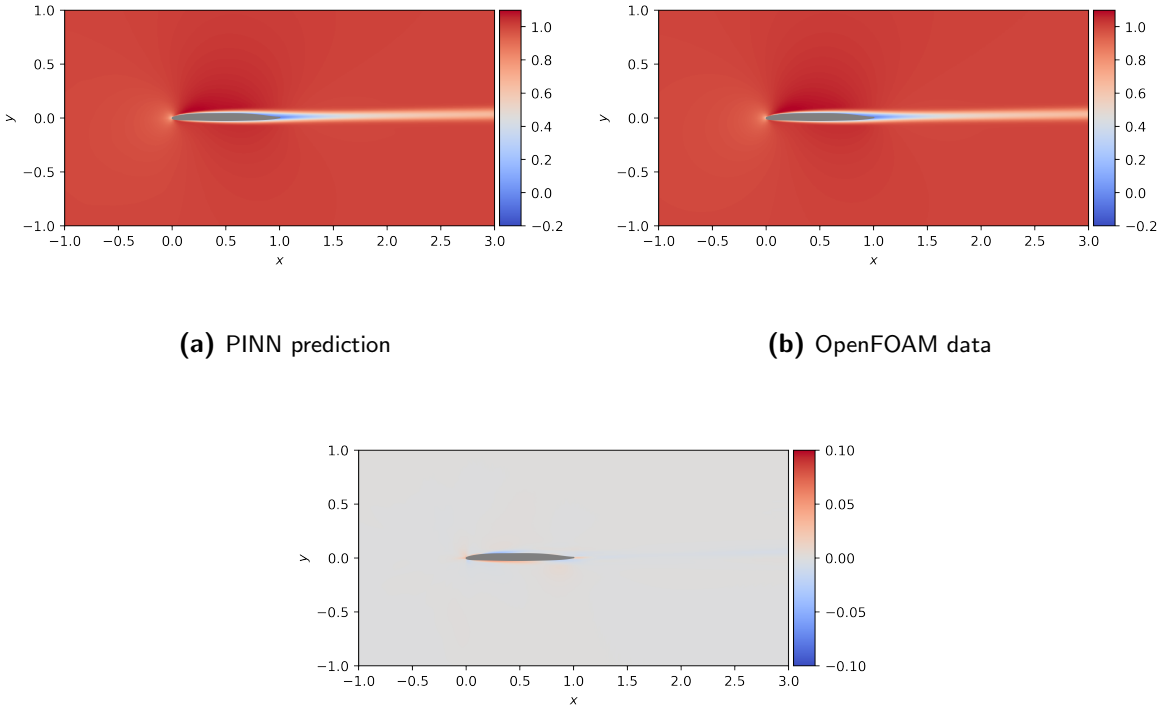



**(a)** optimized geometry **(b)** $c_p$ distribution

**Figure 5.24:** Optimized PARSEC airfoil with random sample 8 as initial geometry.

Figure 5.25 shows a good match for the $u$-velocity field, with the largest differences found close to the airfoil. The velocity is slightly underestimated on the lower side, but overestimated on the top side.

In general, the flow is inferred with high accuracy by the PINN as has been shown in a comparison with OpenFOAM validation data. This optimum, found by starting the optimization from the eighth random starting geometry, is a good candidate for the global optimum as the better scoring optimums have similar shapes to the 11th optimum, which all lie close to the




**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure 5.25:** $u$-velocity field of the lift optimized PARSEC airfoil with initial geometry 8.

boundary of the design space where prediction accuracy have shown to be lower. Here, the predicted aerodynamic performance is questionable due to potential unsteady flow behavior from flow separation, while the set-up for this case assumes steady flow. In regard to the computational work, the PINN was able to complete the optimization in 87 seconds, requiring 8107 function evaluations. These evaluations would take 2702 hours of simulation time with OpenFOAM at 20 minutes each or 113 days. Compared to this, PINN training itself only took 19 hours, but the 2112 OpenFOAM simulations to produce training data required 704 hours. Even after taking the cost of producing the training data, solving this design problem with a PINN reduces the computational cost by a factor of 3.7. If the problem would already have a sufficient amount of training data available, the cost savings would increase to a factor of 142.

## 5.4 Conclusion

Two airfoil optimization cases were studied in this chapter using PINNs as surrogate models to solve the parametric flow problems. A drag optimization at a zero angle of attack and $\mathcal{R}e = 1000$ is treated first, which should have a symmetrical airfoil as the optimal shape. This

simplifies the 11 parameter PARSEC airfoil geometry to a problem with only 5 independent PARSEC parameters. The same approach has been taken for training this PINN as for the previous cambered NACA case and completed without issues. The PINN, trained for this case, was able solve the flow problems with a comparable accuracy to the OpenFOAM solver, but is able to predict the full velocity and pressure fields instantaneously. Even when accounting for the time it takes to produce the training data and completing the training, it still features a net speedup of a factor three. With this, the PINN combines high accuracy and faithfulness to the flow physics of CFD simulations with cheap computational cost of conventional surrogate models, making it a powerful tool for solving parametric flow problems and is able to compete with the current solution methods. The resulting optimal geometry showed an expected increase in leading edge sharpness and reduction of airfoil thickness.

The final case raises the number of free parameters to a total of 11 with 10 PARSEC parameters and the angle of attack. Furthermore, the Reynolds number has been increased to $\mathcal{R}e = 5000$, bringing it in range of MAV flow conditions. This parametric flow problem has proven to be more difficult to train a PINN on, because some parameter combinations exist near the design space boundary that result in unsteady flows due to flow separation. The current set-up for the PINN assumes steady flow, resulting in varying PINN accuracy near the boundaries. Fortunately, most of the design space does not have this problem, so the PINN is still an usable flow solver for the optimization. The numerical optimization was performed for lift with a minimum reference lift coefficient as a constraint at a fixed angle of attack of one degree. In the first optimization attempt, the center airfoil geometry was taken as the starting point. The optimized geometry features a sharper leading edge and a positive camber resulting in a higher lift and lower drag. Since function evaluations are cheap to compute with the PINN, the optimization was repeated for 20 randomly generated starting points to increase the chance of finding the global optimum. About half of the found optimal shapes ended up close to the design space boundary, where the PINN accuracy has been uncertain. After discarding these, the best of the remaining showed similar results to the initial optimization attempt.

Overall, both trained PINNs have shown fair inference accuracy qualitatively and have shown their value as a surrogate flow solver for parametric flow problems that is able to infer the full velocity and pressure fields, similar to a CFD simulation, but reduces the computation time up to a factor of 3.7 when accounting for training and training data generation. During the numerical optimizations using the trained PINNs as flow solvers, plausible optimal shapes were found confirming their applicability to the practical flow problem of numerical design optimization.

# Chapter 6

# Conclusion

Starting with the objective of investigating the prospects of PINNs in fluid mechanics problems, this project has started an exploratory research in order to answer this question. Specifically, the focus has been on flow problems with an airfoil in freestream flow as the basis, governed by the incompressible Navier-Stokes equations. Attempts have been made to train PINNs on several cases to investigate the practical aspects of training and using PINNs to solve flow problems.

The first problem considered a fixed airfoil geometry at an angle of attack of 25 degrees at a low Reynolds number of 200 to induce vortex shedding. This case assumes an incompressible, transient and laminar flow, where both the data on boundary conditions and the flow field are incomplete: besides the initial condition, only the solution data on the airfoil perimeter and in a square area downstream of the airfoil were used as training data. Using these two incomplete data sets to train the PINN, the goal was to infer the complete solution with the trained PINN to reconstruct the flow field in effect. The accuracy of the PINN inferred solution varied depending on whether only the full solution was available in the square or only the pressure. Both cases saw a steady decreasing accuracy with the increase of time and convergence during training was not very smooth, as the time parameter has been treated as an continuous parameter by the PINN. As information should only flow along the positive time axis, some mechanism is needed to prevent the opposite from happening. Possible solutions were the implementation of temporal weighting or temporal discretization of the problem. Temporal weighting increases the weights on the losses at a lower time and reduces the losses at higher times. This encourages the PINN to prioritize reducing the loss at lower time values first before attempting higher time values, thus mimicking a time marching approach. While this may make the PINN training more smooth and avoid local minima at the start, the accuracy problem may not be solved at the end of the training. Due to the lower weights at higher time values, larger error will be tolerated there compared to the solution at lower time values, which still results in a lower accuracy at higher time values than lower time values. Discretizing the computational domain along the time axis is an alternative, more forceful approach to introduce time marching. With this method, the computational domain is divided

into multiple small time windows on which separate smaller PINNs are trained sequentially with limited exchange of information between the adjacent PINNs. Each resulting PINN will be able to predict a time-accurate flow field on a small part of the time domain, but will cover the entire time range together. With the aim of expanding the flow problem to a parametric airfoil problem, both aforementioned solutions still have their issues: temporal weighting may still result in varying accuracy depending on time, while temporal discretization is difficult to implement. Therefore, steady flow was assumed for the parametric flow problems.

The investigation of parametric problems starts with the NACA 4-series airfoils and was performed in two parts. A symmetrical NACA airfoil was the first part of the problem with a Reynolds number of 600. This time, the training data set was composed of the full solution for two airfoil geometries, obtained by OpenFOAM simulations. The boundary conditions were also complete with the velocity components known on the inlet, top and bottom boundary, pressure at the outlet and the no-slip condition on the airfoil perimeter. The PINN was trained on this 1D parametric problem that has the airfoil thickness as a free parameter. After training this PINN, it revealed accurate predictions for geometries that had a thickness between the thickness of the two airfoils from the training data. However, inferences of airfoils with thickness values outside this range were considerably less accurate. This highlights the importance of choosing the sampling points of the design space of the parametric problem as PINNs seem to be good at interpolating between the external training data sets, but less effective when extrapolating.

Using this knowledge, the following case of the cambered NACA airfoils sampled the eight corners of the 3D design space for to generate the training data with OpenFOAM. With this, every geometry can be learned in the design space from interpolating between those training data sets, which should result in a trained PINN with the ability to predict accurate solutions for the entire design space. This was indeed confirmed after inspecting the predicted solutions on the corners and validated for a few geometries where training data was not available. The success of this trained PINN demonstrates the technical feasibility of training a PINN to solve a parametric flow problem, where the trained PINN is able to infer the flow not only for one airfoil geometry, but all the geometries inside the design space.

Next is the demonstration of the practical use of a PINN in two numerical airfoil design optimization cases. For this the PARSEC method is used to describe a larger variety of airfoils for airfoil design with 11 parameters. The first case was a drag optimization of a symmetric airfoil at a Reynolds number of 1000 and a zero degree angle of attack. Assuming symmetry, the number of free parameters of the PARSEC method drops to five. Following the previous method of sampling geometries for the generating training data, the design space can be seen as a 5-dimensional hypercube. There are eight corners or vertices in the 3D case of a cambered NACA airfoil. A 5-dimensional hypercube has 32 vertices, for which training data was created with OpenFOAM. The PINN completed training with success and the validation showed that the trained PINN was able to infer accurate solutions for the entire design space. Following this, the trained PINN was integrated in a optimization loop to demonstrate an application of the PINN. The optimization successfully reduced the drag of the starting geometry by increasing the leading edge sharpness and lowering the airfoil thickness, similar to optimization results in literature for very low Reynolds number flows. Also, the PINN showed the possibility of a significant amount of time saved, depending on the number of

function evaluations are needed. PINNs have a large upfront cost from producing training data and PINN training, but computing a solution with the PINN is nearly instantaneous and has a negligible cost. With the numerous function evaluations needed for the numerical optimization, a net gain has been observed of a computational work reduction of a factor three compared to using OpenFOAM as solver. This demonstrates the practical value of the PINN as a surrogate model that can infer a full flow field with a comparable accuracy to conventional CFD at lower cost, even when taking the upfront costs into account.

The final case raises the complexity by increasing the number of free parameters to 11, consisting of 10 PARSEC parameters and the angle of attack. Also, the Reynolds number was further increased to 5000 to match the flow conditions of a practical problem of airfoil design of a small fixed-wing MAV. Airfoil design optimization was performed to maximize lift and minimize drag at the same time, with a minimum reference lift as a constraint. The design space was sampled on the boundary in a similar way as before for generating the training data, but due to some of the OpenFOAM simulations failing to converge an addition number of samples of the interior were included. Using this data, the PINN was trained on the 11 parameter design space, but the accuracy in some areas of the design space were noticeably lower, in particular near the boundary. Some combinations of the parameters appeared to create conditions that resulted in unsteady flow, which is more difficult for the PINN to learn. Also, a slightly lower accuracy was observed in the interior, likely caused by the thinner boundary layer due to the Reynolds number increase. This reduced the total area the boundary layer occupies in the spatial domain, thus reducing the relative importance of the boundary layer for the PINN, which is reflected in a smaller contribution to the total loss. However, the main issue is likely to be the higher complexity of the problem with more geometrical parameters, resulting in a much larger variety of airfoil shapes captured within the design space boundaries. The more diverse flow cases resulting from this have a more complicated relation with each other, which is more difficult for the PINN to learn. Despite the lower accuracy of this trained PINN compared to those of previous cases, its accuracy was sufficient for a preliminary design optimization, focusing on the qualitative aspects of the resulting optimal shapes, rather than the quantitative aspects. The optimization was started from 20 randomly generated starting geometries resulting in 15 successfully converged optimal shapes. After removing the optimal shapes close to the boundary for their uncertain accuracy, the best of the remaining showed a sharper leading edge and a positive camber. A sharper leading edge has been shown to reduce drag in the drag optimization case and the addition of camber increases the pressure difference between the upper and lower side of the airfoil, thus increases lift. As the optimization produced a plausible optimal shape, the PINN fulfilled its role as a flow solver successfully in the optimization. Performance-wise, the PINN showed a computational work reduction of a factor 3.7, after taking training time and training data production into account.

With all these cases together, this project takes a step beyond the use of PINNs on academic benchmark problems in literature, by demonstrating its capabilities on a series of airfoil flow problems. The research objective is reached by presenting the technical feasibility of training a PINN on parametric flow problems and its potential as a flow solver for practical problems, by demonstrating its implementation in a numerical airfoil design optimization loop.

## 6.1 Recommendations

PINNs is still an immature technology with a low technological readiness level, meaning it needs much research and investment to reach widespread adoption. Subjected to the resource and time constraints, this project has explored some topics in order to advance our knowledge on PINNs for fluid mechanics, but there are many more that deserve attention or further investigation.

One of these is the trade-off between early convergence and the final accuracy if the area of interest is also most difficult to learn for the PINN. In this case, it is beneficial to have the difficult area have a smaller contribution to the total error, so the easier and larger part of the computational domain is prioritized. This accelerates the convergence and prevents the training from getting stuck at a local minima. However, the lower contribution of these losses to the total loss also results in higher tolerance for errors in these areas, possibly resulting in a lower final accuracy here compared to the rest of the computational domain. This mismatch of high importance and its under-representation in the total loss can simply be the result of how the computational domain looks like as was the case for the flow problems where the boundary layer became thinner with increasing Reynolds number or a consequence of a loss weighting scheme. A possible solution that may warrant investigation is the use of spatial discretization to divide the computational domain into smaller parts and split them between multiple PINNs that are individually easier to train on. Such a concept for domain decomposition was presented in [20] with a number of benchmark problems including the Burger's equation and lid-driven cavity problem governed by the incompressible Navier-Stokes equations, but its application to larger problems needs research. Alternatively, a transfer learning approach can be taken to use the current results to partially retrain the PINN with different loss weighting. This bears some resemblance to the method of [8], but uses a combination of a PINN and a regular deep neural network.

All the PINNs in this project have used the fully connected network architecture. While this architecture has wide applicability, it is not optimized for use in PINNs nor for flow problems. Alternative network architectures are mentioned in SimNet's manual [32]. Most of them are a variation of the Fourier network, which aims to mitigate the spectral bias of the PINN towards low frequencies [35], by using a Fourier feature mapping on the inputs to learn high frequencies [44].

The currently used velocity-pressure formulation of the incompressible Navier-Stokes equations is widely used, but does have some issues, which may affect convergence and accuracy. The first problem is that pressure boundary conditions are not given naturally in this formulation [13]. The second problem is the absence op pressure in the continuity equation. The vector potential formulation of [50] offers an alternative, where the velocity field automatically satisfies the continuity equation and is divergence free. There are some comparisons between the alternative architectures and the Navier-Stokes formulations in [32], but no published works that have studied those in more detail.

Another area where improvements can be made is the optimizer. The Adam optimizer has been the default choice for awhile in deep learning for their ability to deal with large amounts

of data. This optimizer choice is inherited from the regular deep neural networks, but may not be the optimal choice for PINNs. In many ways PINNs are similar to regular deep neural networks, but there are still some clear differences. For this reason, the empirical evidence supporting the optimizer choice in [23] may not apply to PINNs. One of the differences is the quality and quantity of data. While external data from either measurements or simulations contain noise like training data for data-driven deep learning, data sampled from the boundary conditions or the collocation points are perfect. Also, in data-driven machine learning the accuracy requirements are less strict, where a perfect prediction of the trained network is not expected. In physical problems this is usually not the case as a decent local minimum is not enough but the global optimum needs to be found so the trained PINN will predict a solution that is as close to reality as possible.

These recommendations are not an exhaustive list of all areas in which physics-informed machine learning can improve, but just a number of topics encountered during this project that could not investigated in more detail considering the practical constraints of time and resources. Fortunately, this is not the only research being performed on PINNs as the interest in physics-informed machine learning has been growing since the introduction of PINNs in [36] with the amount of papers referencing this or including PINN in their keywords or abstract title is growing exponentially [9], hinting at a bright future for PINNs in the scientific community. Out the many potential applications of PINNs, this work focused on solving several incompressible airfoil flow problems. Hopefully the knowledge gained in this work will contribute to the foundation of PINNs in the field of fluid mechanics.

# Bibliography


[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. May 2016.

[2] John D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill Education, 6 edition, 2016. ISBN 9781259129919.

[3] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, February 2015.

[4] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, sep 1995. doi: 10.1137/0916069.

[5] Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, jun 2012. doi: 10.1007/s10107-012-0572-5.

[6] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *arXiv:2105.09506 [physics]*, May 2021. URL http://arxiv.org/abs/2105.09506. arXiv: 2105.09506.

[7] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young-Jin Jeon, Callum Gray, and George Em Karniadakis. Flow over an espresso cup: Inferring 3d velocity and pressure fields from tomographic background oriented schlieren videos via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, March 2021. doi: 10.1017/jfm.2021.135.

[8] Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426:109942, May 2020. doi: 10.1016/j.jcp.2020.109942.

[9] Salvatore Cuomo, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maizar Raissi, and Francesco Piccialli. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next. January 2022.

[10] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[11] Ronen Eldan and Ohad Shamir. The Power of Depth for Feedforward Neural Networks. *arXiv:1512.03965 [cs, stat]*, May 2016. URL http://arxiv.org/abs/1512.03965. arXiv: 1512.03965 version: 4.

[12] R. Glowinski and O. Pironneau. On the numerical computation of the minimum-drag profile in laminar flow. *Journal of Fluid Mechanics*, 72(02):385, nov 1975. doi: 10.1017/s0022112075003436.

[13] Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1111–1145, oct 1987. doi: 10.1002/fld.1650071008.

[14] Ehsan Haghighat and Ruben Juanes. SciANN: A Keras/Tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 373:113552, January 2021. ISSN 00457825. doi: 10.1016/j.cma.2020.113552. URL http://arxiv.org/abs/2005.08803. arXiv: 2005.08803.

[15] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2015. doi: 10.1109/ijcnn.2015.7280578.

[16] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. Nvidia simnet$^{TM}$: an ai-accelerated multi-physics simulation framework. December 2020.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[18] Jianchun Huan and Vijay Modi. Design of minimum drag bodies in incompressible laminar flow. *Inverse Problems in Engineering*, 3(4):233–260, sep 1996. doi: 10.1080/174159796088027626.

[19] Eastman N. Jacobs, Kenneth E. Ward, and Robert M. Pinkerton. The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel. Technical Report PB-177874, NACA, January 1933. URL https://ntrs.nasa.gov/citations/19930091108. NTRS Author Affiliations: NTRS Document ID: 19930091108 NTRS Research Center: Legacy CDMS (CDMS).

[20] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, jun 2020. doi: 10.1016/j.cma.2020.113028.

[21] Stephen Wright Jorge Nocedal. *Numerical Optimization*. Springer-Verlag GmbH, December 2006. ISBN 9780387400655.

[22] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. URL https://www.nature.com/articles/s42254-021-00314-5.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

[24] Tsuguo Kondoh, Tadayoshi Matsumori, and Atsushi Kawamoto. Drag minimization and lift maximization in laminar flows via topology optimization employing simple objective function expressions based on body force integration. *Structural and Multidisciplinary Optimization*, 45(5):693–701, May 2012. ISSN 1615-1488. doi: 10.1007/s00158-011-0730-z. URL https://doi.org/10.1007/s00158-011-0730-z.

[25] Dongzhuo Li, Kailai Xu, Jerry M. Harris, and Eric Darve. Coupled Time-lapse Full Waveform Inversion for Subsurface Flow Problems using Intrusive Automatic Differentiation. *arXiv:1912.07552 [physics]*, May 2020. URL http://arxiv.org/abs/1912.07552. arXiv: 1912.07552.

[26] Jichao Li, Jinsheng Cai, and Kun Qu. Adjoint-based two-step optimization method using proper orthogonal decomposition and domain decomposition. *AIAA Journal*, 56 (3):1133–1145, mar 2018. doi: 10.2514/1.j055773.

[27] Lu Lu, Ming Dao, Punit Kumar, Upadrasta Ramamurty, George Em Karniadakis, and Subra Suresh. Extraction of mechanical properties of materials through deep learning from instrumented indentation. *Proceedings of the National Academy of Sciences*, 117 (13):7052–7062, March 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1922210117. URL https://www.pnas.org/content/117/13/7052.

[28] Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *arXiv:1907.04502 [physics, stat]*, February 2020. URL http://arxiv.org/abs/1907.04502. arXiv: 1907.04502.

[29] D. Mateescu and M. Abdo. Analysis of flows past airfoils at very low reynolds numbers. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 224(7):757–775, jul 2010. doi: 10.1243/09544100jaero715.

[30] Nirmal J. Nair and Andres Goza. Leveraging reduced-order models for state estimation using deep learning. *Journal of Fluid Mechanics*, 897:R1, jun 2020. ISSN 0022-1120, 1469-7645. doi: 10.1017/jfm.2020.409. URL http://arxiv.org/abs/1912.10553. arXiv: 1912.10553.

[31] Maryam M. Najafabadi, Flavio Villanustre, Taghi M. Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1), 2015. doi: 10.1186/s40537-014-0007-7.

[32] NVIDIA. *Simnet User Guide Release v21.06*, June 2021.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. December 2019.

[34] David Pfau, James S. Spencer, Alexander G. de G. Matthews, and W. M. C. Foulkes. Ab-Initio Solution of the Many-Electron Schrödinger Equation with Deep Neural Networks. *Physical Review Research*, 2(3):033429, September 2020. ISSN 2643-1564. doi: 10.1103/PhysRevResearch.2.033429. URL http://arxiv.org/abs/1909.02487. arXiv: 1909.02487.

[35] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *ICML 2019*, June 2018.

[36] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

[37] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020. ISSN 0036-8075. doi: 10.1126/science.aaw4741.

[38] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. October 2017.

[39] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking State-of-the-Art Deep Learning Software Tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104, November 2016. doi: 10.1109/CCBD.2016.029.

[40] Helmut Sobieczky. Parametric airfoils and wings. In *Notes on Numerical Fluid Mechanics (NNFM)*, pages 71–87. Vieweg+Teubner Verlag, 1999. doi: 10.1007/978-3-322-89952-1_4.

[41] D. N. Srinath and S. Mittal. Optimal airfoil shapes for low reynolds number flows. *International Journal for Numerical Methods in Fluids*, 61(4):355–381, oct 2009. doi: 10.1002/fld.1960.

[42] Vis Sripawadkul, Mattia Padulo, and Marin Guenov. A Comparison of Airfoil Shape Parameterization Techniques for Early Design Optimization. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Multidisciplinary Analysis Optimization Conferences. American Institute of Aeronautics and Astronautics, September 2010. doi: 10.2514/6.2010-9050. URL https://arc-aiaa-org.tudelft.idm.oclc.org/doi/10.2514/6.2010-9050.

[43] Lieven Vandenberghe Stephen Boyd. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

[44] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *arXiv:2006.10739 [cs]*, June 2020. URL http://arxiv.org/abs/2006.10739. arXiv: 2006.10739.

[45] Meedhu Geogy Ukken and M Sivapragasam. Aerodynamic shape optimization of airfoils at ultra-low reynolds numbers. *Sādhanā*, 44(6), may 2019. doi: 10.1007/s12046-019-1115-z.

[46] Pierluigi Della Vecchia, Elia Daniele, and Egidio D'Amato. An airfoil shape optimization technique coupling PARSEC parameterization and evolutionary algorithm. *Aerospace Science and Technology*, 32(1):103–110, 1 2014. doi: 10.1016/j.ast.2013.11.006.

[47] Sifan Wang and Paris Perdikaris. Deep learning of free boundary and stefan problems. June 2020. doi: 10.1016/j.jcp.2020.109914.

[48] Colby L. Wight and Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *arXiv:2007.04542 [cs, math]*, July 2020. URL http://arxiv.org/abs/2007.04542. arXiv: 2007.04542.

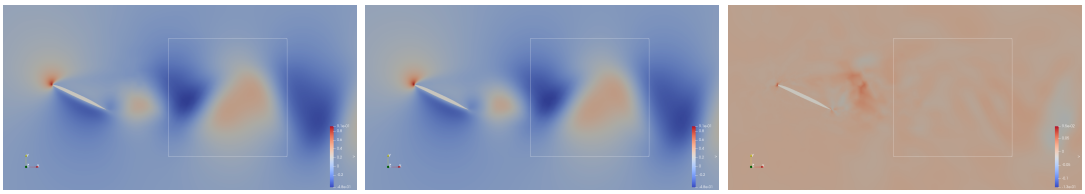
[49] Justin Winslow, Hikaru Otsuka, Bharath Govindarajan, and Inderjit Chopra. Basic understanding of airfoil characteristics at low reynolds numbers (104–105). *Journal of Aircraft*, 55(3):1050–1061, may 2018. doi: 10.2514/1.c034415.

[50] D. L. Young, C. H. Tsai, and C. S. Wu. A novel vector potential formulation of 3D Navier-Stokes equations with through-flow boundaries by a local meshless method. *Journal of Computational Physics*, 300:219–240, 2015. ISSN 0021-9991. doi: 10.1016/j.jcp.2015.07.040. URL https://www.sciencedirect.com/science/article/pii/S0021999115004854.

# Appendix A

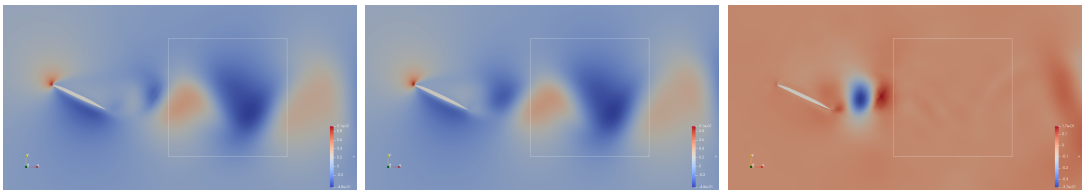
# Flow reconstruction additional results

## A.1 Velocity and pressure in measurement window



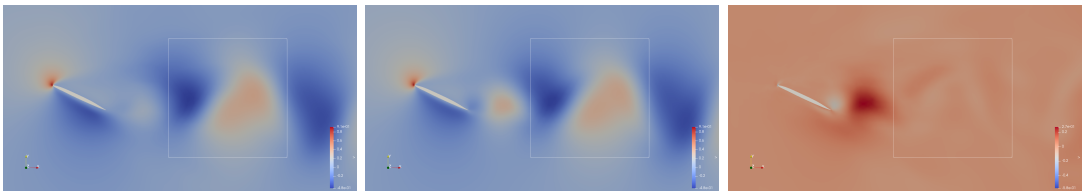
**(a)** velocity ($v$) PINN **(b)** velocity ($v$) OpenFOAM **(c)** velocity ($v$) difference

**Figure A.1:** Velocity ($v$) comparison at $t = 30.04$ for the PINN trained on the full solution in the measurement window.



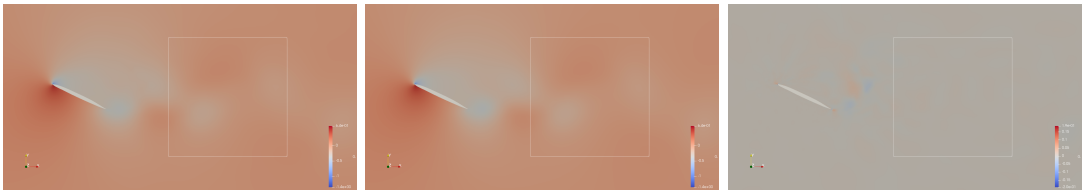
**(a)** velocity ($v$) PINN **(b)** velocity ($v$) OpenFOAM **(c)** velocity ($v$) difference

**Figure A.2:** Velocity ($v$) comparison at $t = 31.52$ for the PINN trained on the full solution in the measurement window.



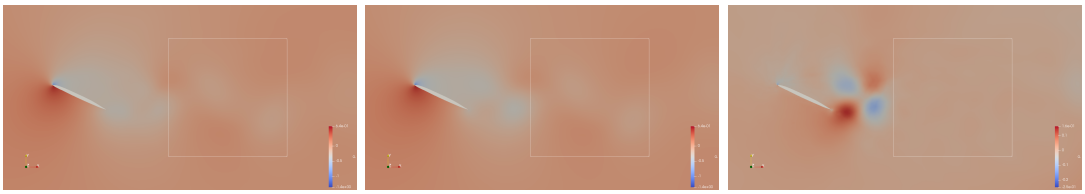
**(a)** velocity ($v$) PINN **(b)** velocity ($v$) OpenFOAM **(c)** velocity ($v$) difference

**Figure A.3:** Velocity ($v$) comparison at $t = 33.00$ for the PINN trained on the full solution in the measurement window.



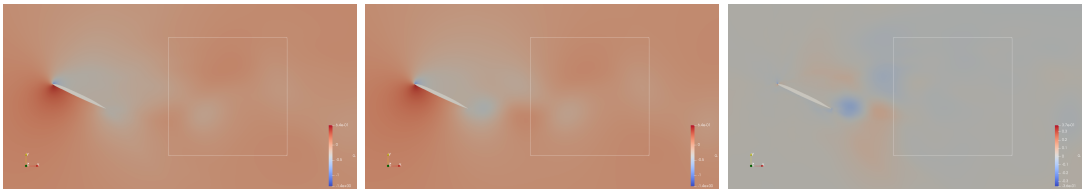
**(a)** pressure ($p$) PINN **(b)** pressure ($p$) OpenFOAM **(c)** pressure ($p$) difference

**Figure A.4:** Pressure ($p$) comparison at $t = 30.04$ for the PINN trained on the full solution in the measurement window.



**(a)** pressure ($p$) PINN **(b)** pressure ($p$) OpenFOAM **(c)** pressure ($p$) difference

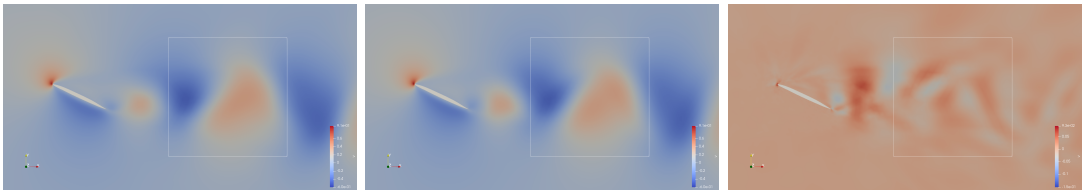
**Figure A.5:** Pressure ($p$) comparison at $t = 31.52$ for the PINN trained on the full solution in the measurement window.



**(a)** pressure ($p$) PINN **(b)** pressure ($p$) OpenFOAM **(c)** pressure ($p$) difference

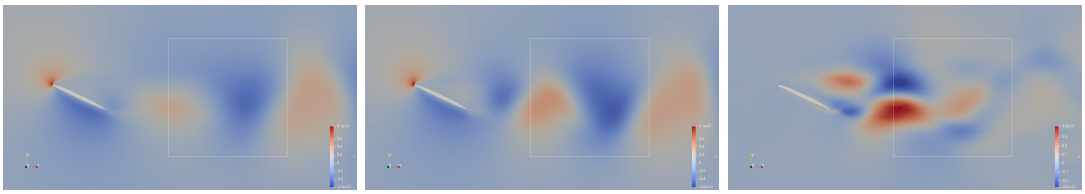
**Figure A.6:** Pressure ($p$) comparison at $t = 33.00$ for the PINN trained on the full solution in the measurement window.

## A.2 Pressure only in measurement window



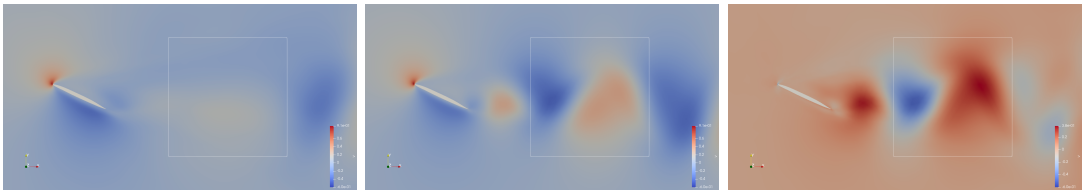
**(a)** velocity $(v)$ PINN **(b)** velocity $(v)$ OpenFOAM **(c)** velocity $(v)$ difference

**Figure A.7:** Velocity $(v)$ comparison at $t = 30.04$ for the PINN trained on pressure in the measurement window.



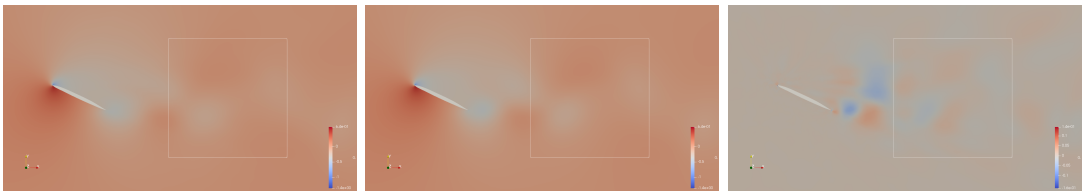
**(a)** velocity $(v)$ PINN **(b)** velocity $(v)$ OpenFOAM **(c)** velocity $(v)$ difference

**Figure A.8:** Velocity $(v)$ comparison at $t = 31.52$ for the PINN trained on pressure in the measurement window.



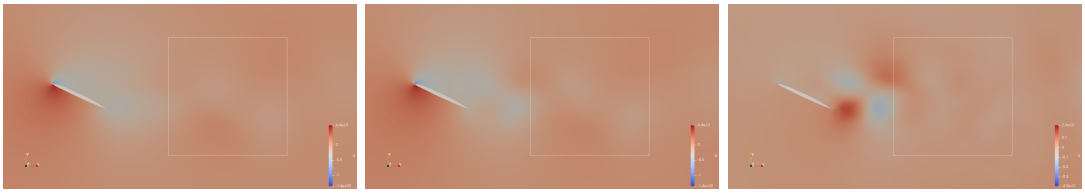
**(a)** velocity $(v)$ PINN **(b)** velocity $(v)$ OpenFOAM **(c)** velocity $(v)$ difference

**Figure A.9:** Velocity $(v)$ comparison at $t = 33.00$ for the PINN trained on pressure in the measurement window.



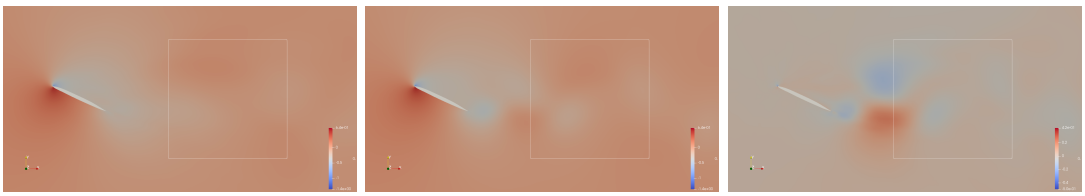
**(a)** pressure $(p)$ PINN **(b)** pressure $(p)$ OpenFOAM **(c)** pressure $(p)$ difference

**Figure A.10:** Pressure $(p)$ comparison at $t = 30.04$ for the PINN trained on pressure in the measurement window.



**(a)** pressure $(p)$ PINN **(b)** pressure $(p)$ OpenFOAM **(c)** pressure $(p)$ difference

**Figure A.11:** Pressure $(p)$ comparison at $t = 31.52$ for the PINN trained on pressure in the measurement window.



**(a)** pressure ($p$) PINN **(b)** pressure ($p$) OpenFOAM **(c)** pressure ($p$) difference

**Figure A.12:** Pressure ($p$) comparison at $t = 33.00$ for the PINN trained on pressure in the measurement window.

# Appendix B

# NACA cases additional results

## B.1 NACA 00XX

NACA 0010

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.1:** $v$-velocity field of the NACA 0010 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.2:** Pressure field $p$ of the NACA 0010 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.3:** $u$-velocity field of the NACA 0006 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

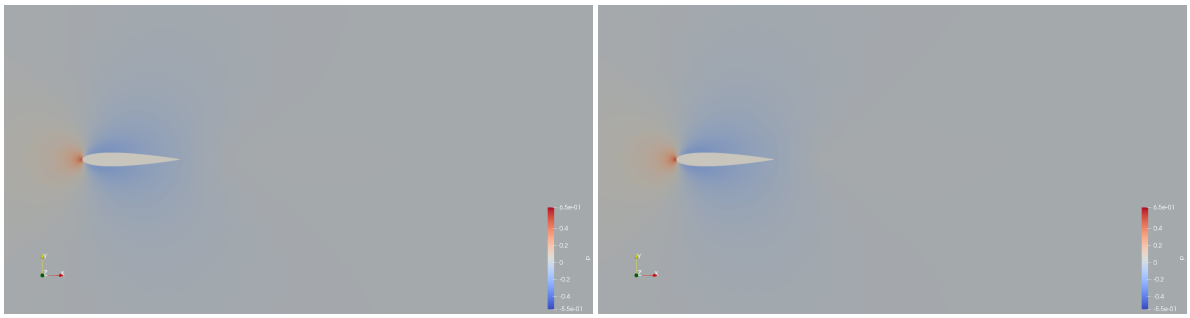
**Figure B.4:** Pressure field $p$ of the NACA 0006 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.5:** $v$-velocity field of the NACA 0014 airfoil.

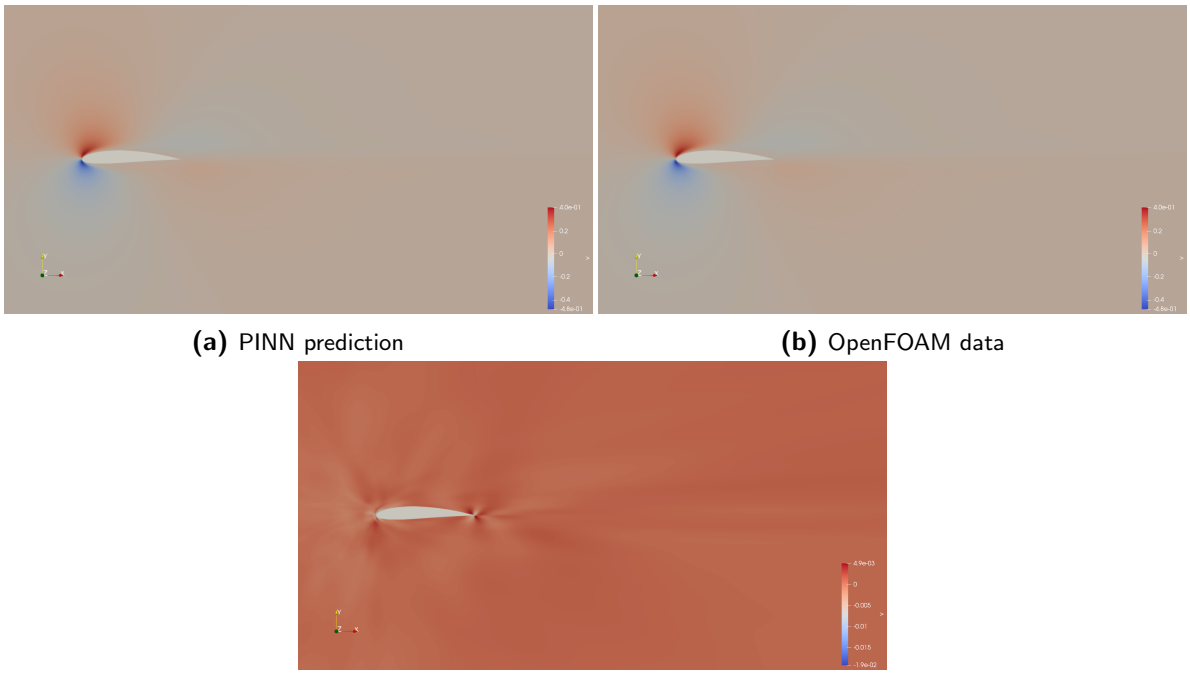


**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure B.6:** Pressure field $p$ of the NACA 0014 airfoil.

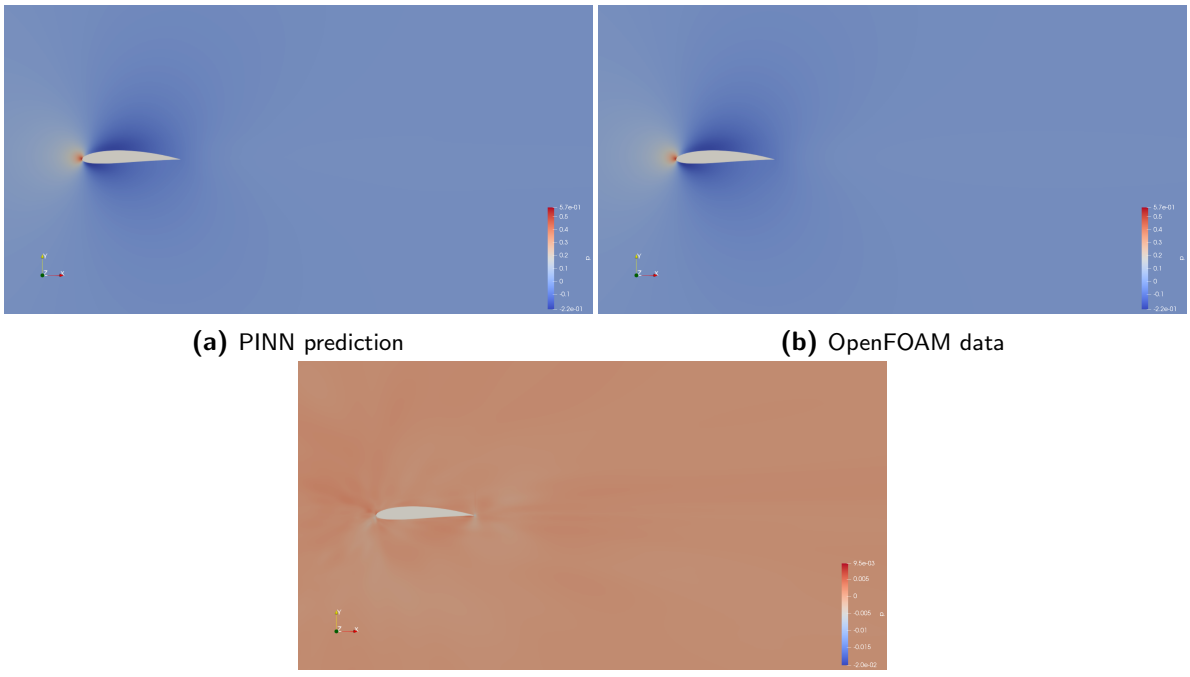
## B.2 NACA XXXX



**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure B.7:** $v$-velocity field of the NACA 3513 airfoil.



**(a)** PINN prediction **(b)** OpenFOAM data

**(c)** difference

**Figure B.8:** Pressure field $p$ of the NACA 3513 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.9:** $v$-velocity field of the NACA 2412 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.10:** Pressure field $p$ of the NACA 2412 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure B.11:** $u$-velocity field of the NACA 2413 airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

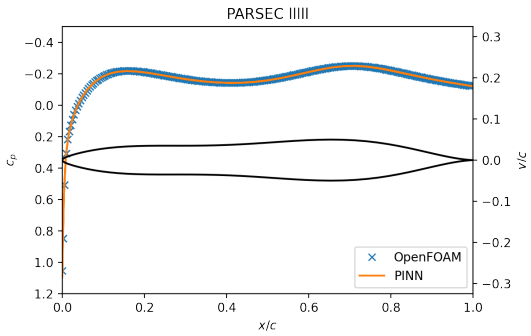
**(c)** difference

**Figure B.12:** Pressure field $p$ of the NACA 2413 airfoil.

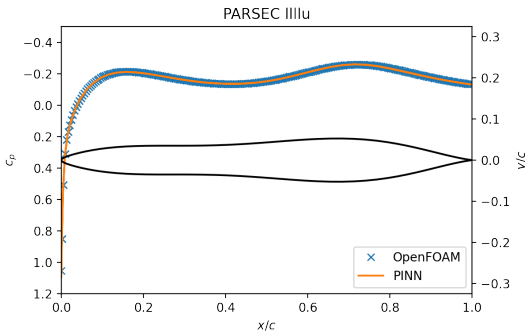
# Appendix C

# PARSEC cases additional results

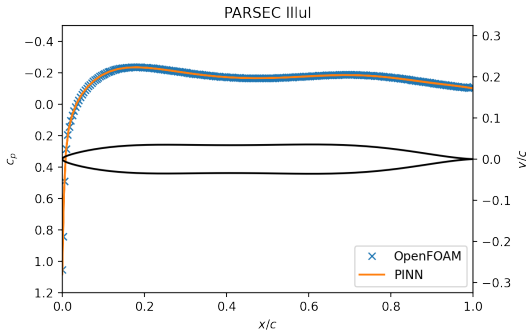
## C.1 PARSEC five parameter case



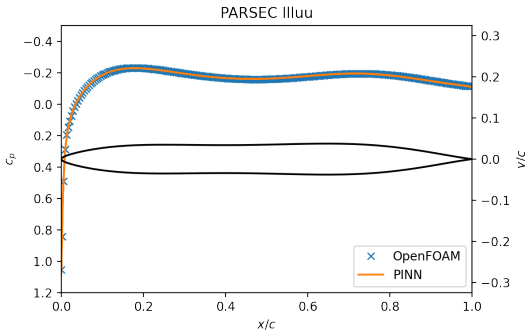

**(a)** $c_p$ distribution of the PARSEC lllll airfoil



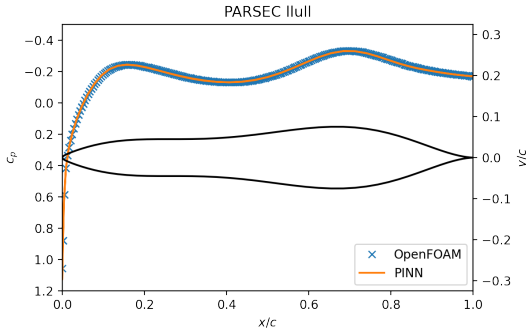

**(b)** $c_p$ distribution of the PARSEC llllu airfoil



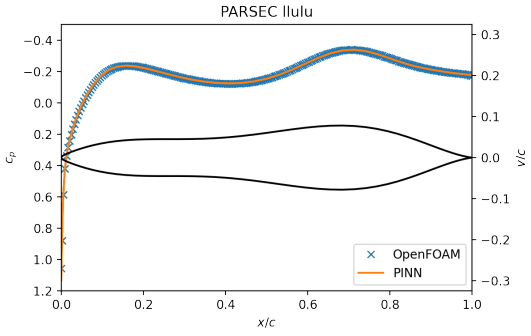

**(c)** $c_p$ distribution of the PARSEC lllul airfoil



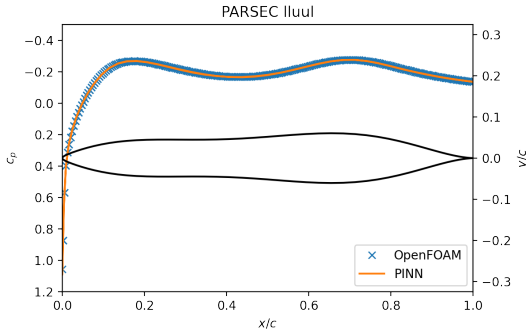

**(d)** $c_p$ distribution of the PARSEC llluu airfoil



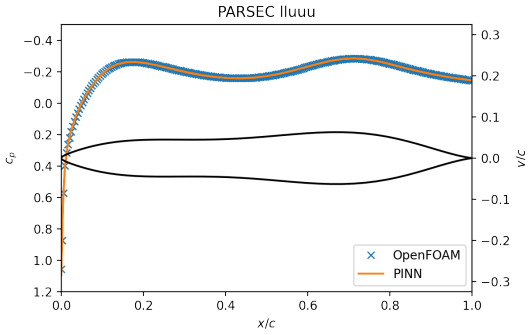

**(e)** $c_p$ distribution of the PARSEC llull airfoil




**(f)** $c_p$ distribution of the PARSEC llulu airfoil

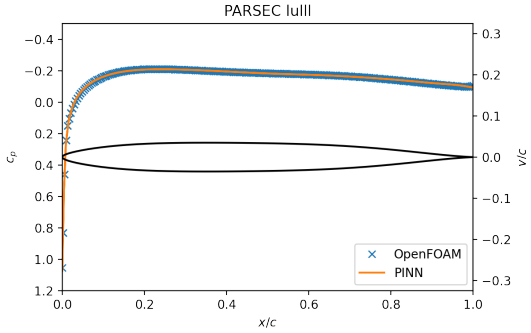



**(g)** $c_p$ distribution of the PARSEC lluul airfoil

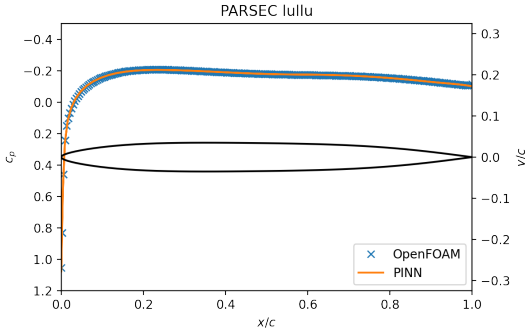



**(h)** $c_p$ distribution of the PARSEC lluuu airfoil

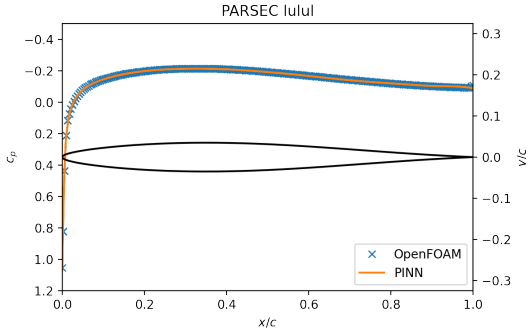
**Figure C.1:** Pressure distribution of the symmetric PARSEC airfoils on the boundary.



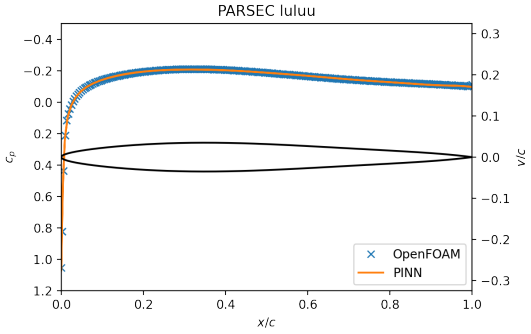

**(i)** $c_p$ distribution of the PARSEC lulll airfoil



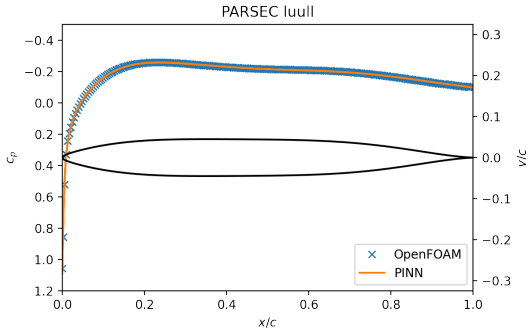

**(j)** $c_p$ distribution of the PARSEC lullu airfoil



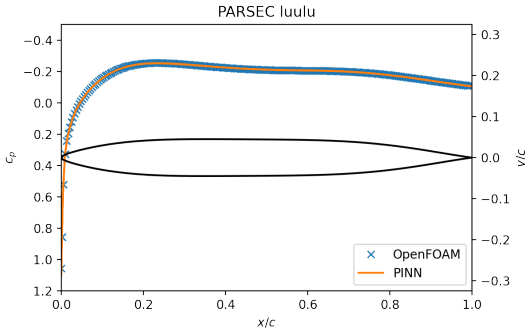

**(k)** $c_p$ distribution of the PARSEC lulul airfoil



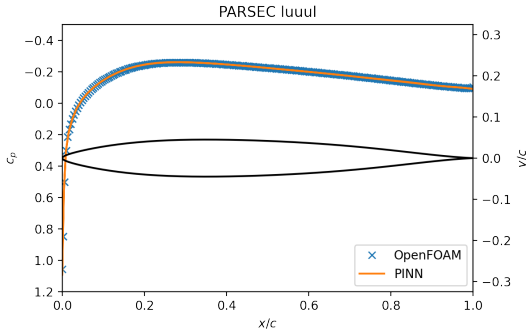

**(l)** $c_p$ distribution of the PARSEC luluu airfoil



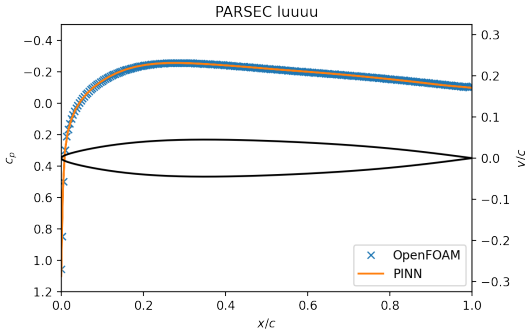

**(m)** $c_p$ distribution of the PARSEC luull airfoil




**(n)** $c_p$ distribution of the PARSEC luulu airfoil

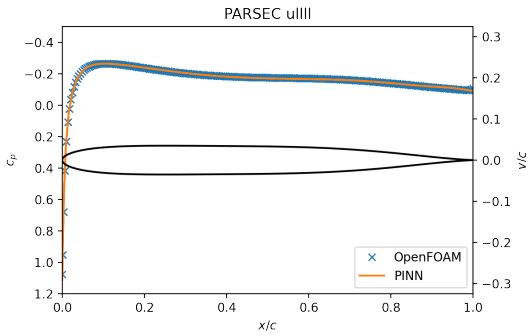



**(o)** $c_p$ distribution of the PARSEC luuul airfoil

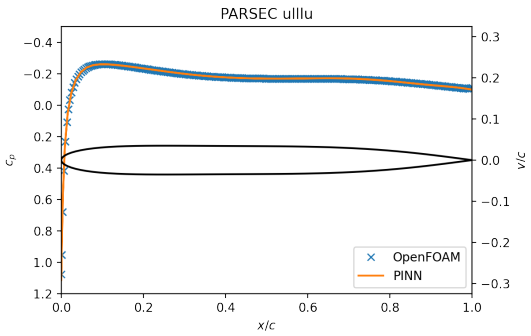



**(p)** $c_p$ distribution of the PARSEC luuuu airfoil

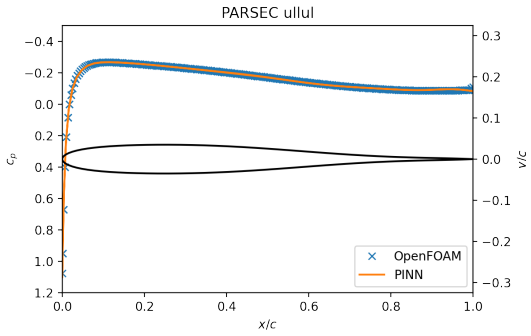
**Figure C.1:** Pressure distribution of the symmetric PARSEC airfoils on the boundary. cont.



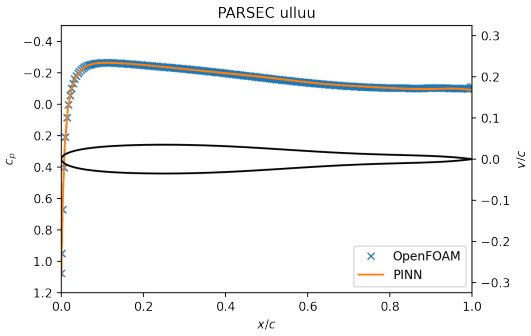

**(q)** $c_p$ distribution of the PARSEC ullll airfoil



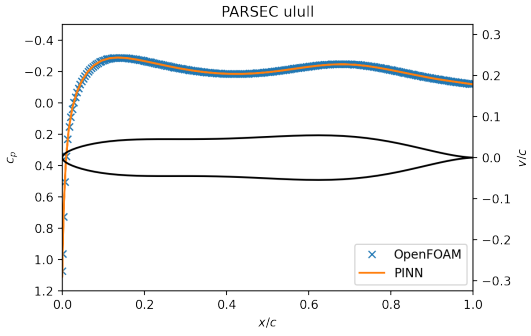

**(r)** $c_p$ distribution of the PARSEC ulllu airfoil



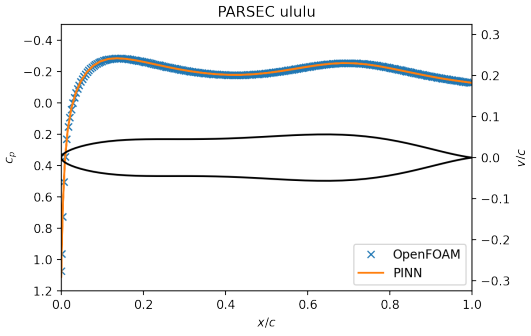

**(s)** $c_p$ distribution of the PARSEC ullul airfoil



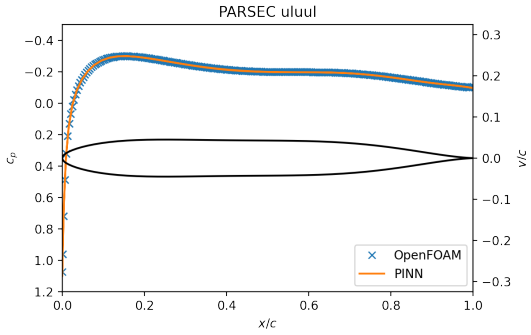

**(t)** $c_p$ distribution of the PARSEC ulluu airfoil



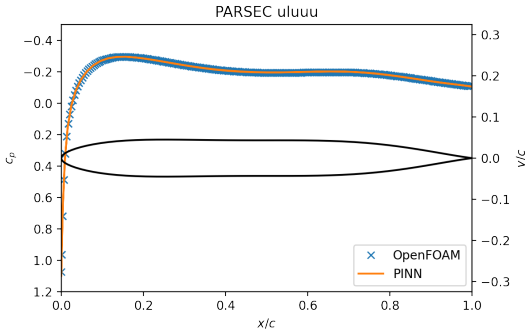

**(u)** $c_p$ distribution of the PARSEC ulull airfoil




**(v)** $c_p$ distribution of the PARSEC ululu airfoil

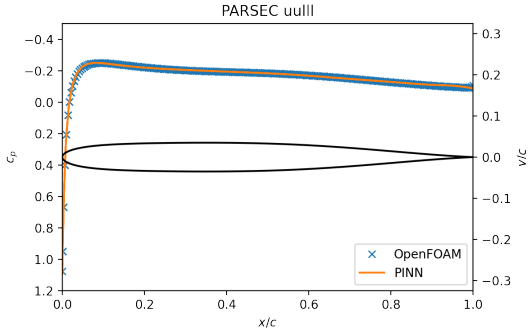



**(w)** $c_p$ distribution of the PARSEC uluul airfoil

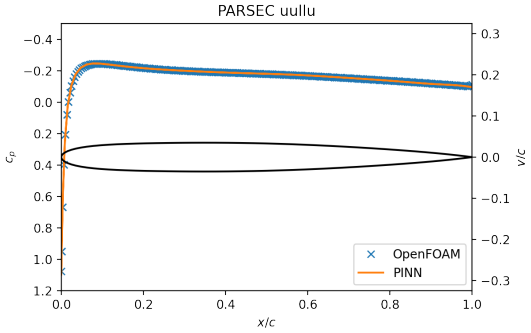



**(x)** $c_p$ distribution of the PARSEC uluuu airfoil

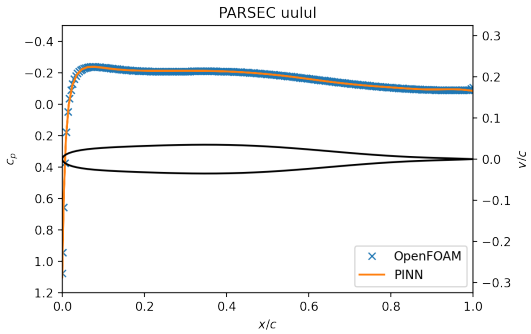
**Figure C.1:** Pressure distribution of the symmetric PARSEC airfoils on the boundary. cont.



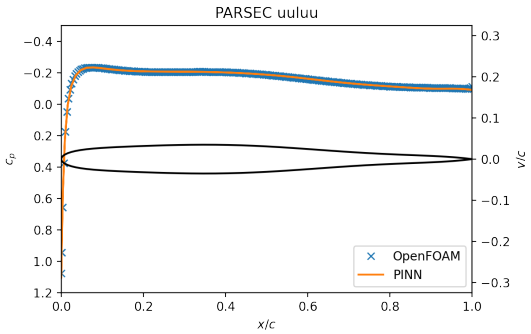

**(y)** $c_p$ distribution of the PARSEC uulll airfoil



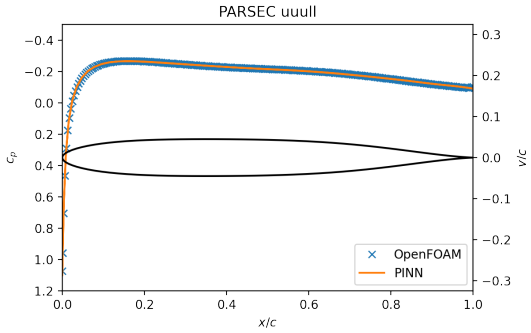

**(z)** $c_p$ distribution of the PARSEC uullu airfoil



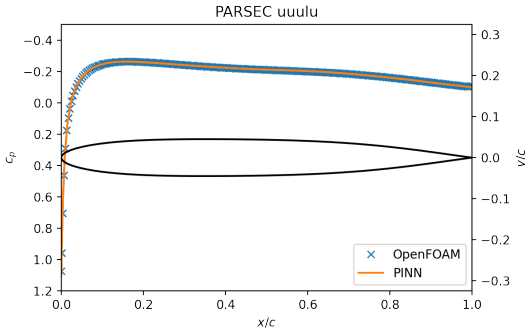

**(aa)** $c_p$ distribution of the PARSEC uulul airfoil



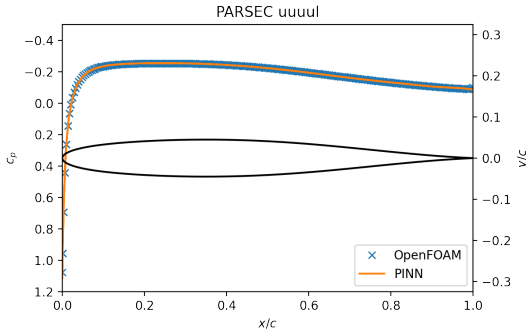

**(ab)** $c_p$ distribution of the PARSEC uuluu airfoil



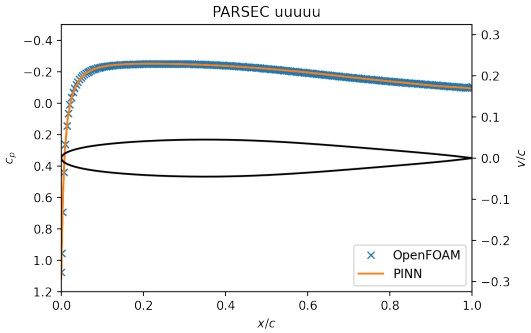

**(ac)** $c_p$ distribution of the PARSEC uuull airfoil




**(ad)** $c_p$ distribution of the PARSEC uuulu airfoil

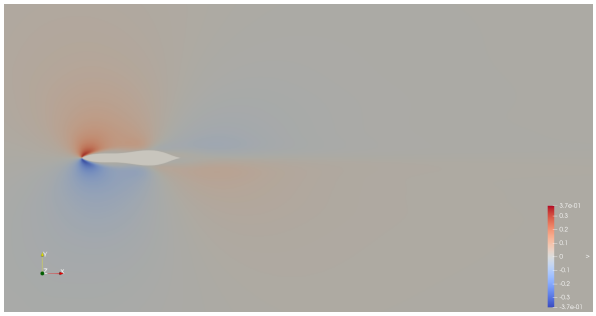



**(ae)** $c_p$ distribution of the PARSEC uuuul airfoil

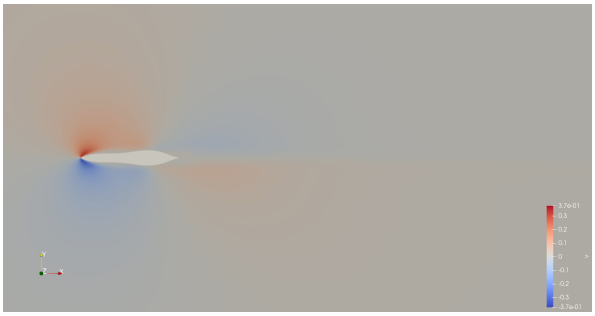



**(af)** $c_p$ distribution of the PARSEC uuuuu airfoil

**Figure C.1:** Pressure distribution of the symmetric PARSEC airfoils on the boundary. cont.



**(a)** PINN prediction



**(b)** OpenFOAM data

**(c)** difference

**Figure C.2:** Velocity field ($v$) of the PARSEC Ilulu airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure C.3:** Pressure field ($p$) of the PARSEC Ilulu airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure C.4:** Velocity field ($v$) of the NACA 0008 equivalent PARSEC airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure C.5:** Pressure field ($p$) of the NACA 0008 equivalent PARSEC airfoil.

**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

**Figure C.6:** Velocity field ($v$) of the drag optimized PARSEC airfoil.

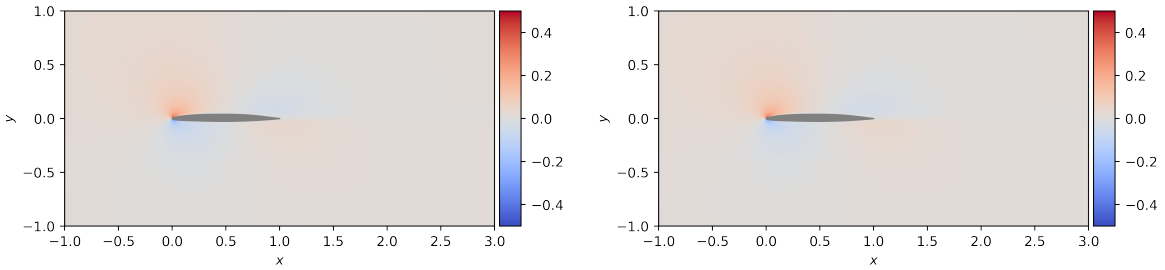
**(a)** PINN prediction

**(b)** OpenFOAM data

**(c)** difference

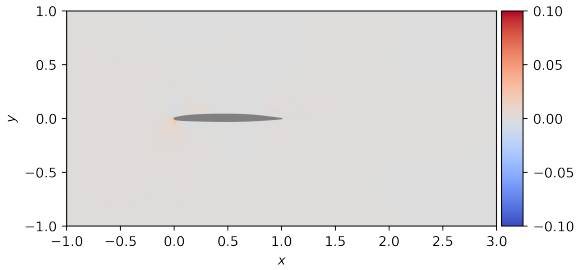
**Figure C.7:** Pressure field ($p$) of the drag optimized PARSEC airfoil.

## C.2 PARSEC 11 parameter case



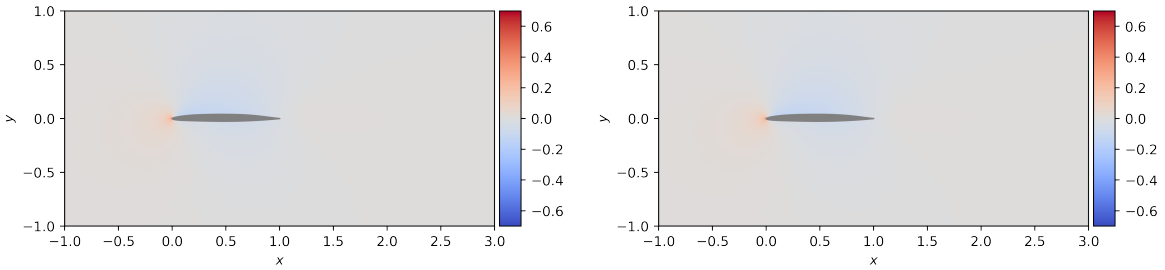

**(a)** PINN prediction

**(b)** OpenFOAM data



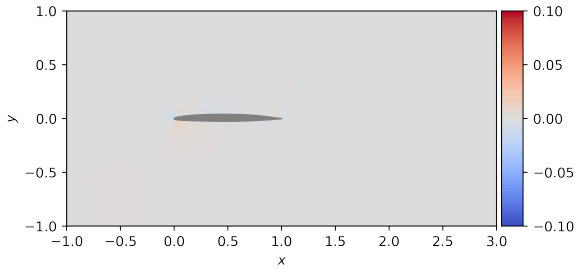

**(c)** difference

**Figure C.8:** Velocity field ($v$) of the lift optimized PARSEC airfoil with initial geometry in the center of the design space.



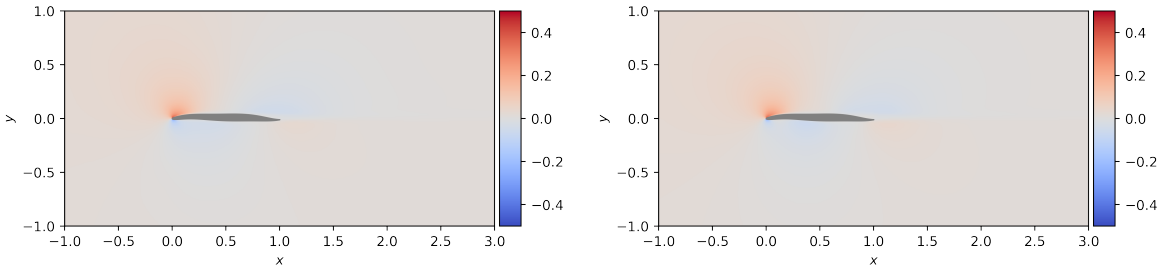

**(a)** PINN prediction

**(b)** OpenFOAM data



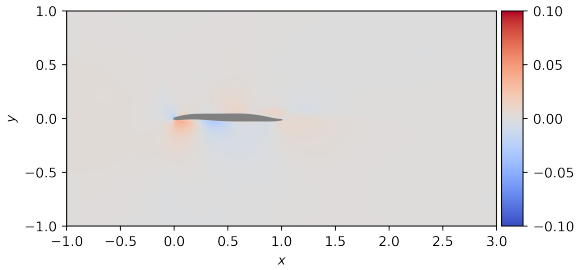

**(c)** difference

**Figure C.9:** Pressure field ($p$) of the lift optimized PARSEC airfoil with initial geometry in the center of the design space.



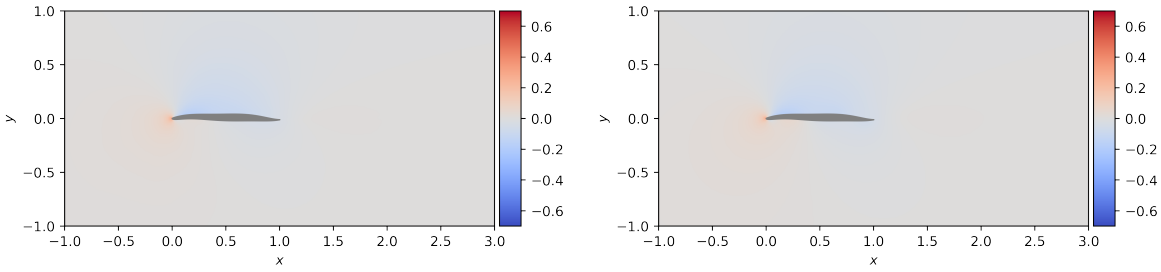

**(a)** PINN prediction

**(b)** OpenFOAM data



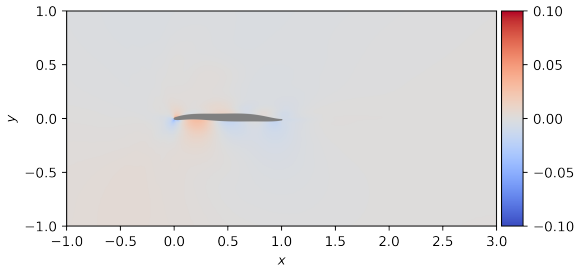

**(c)** difference

**Figure C.10:** Velocity field ($v$) of the lift optimized PARSEC airfoil with initial geometry 11.



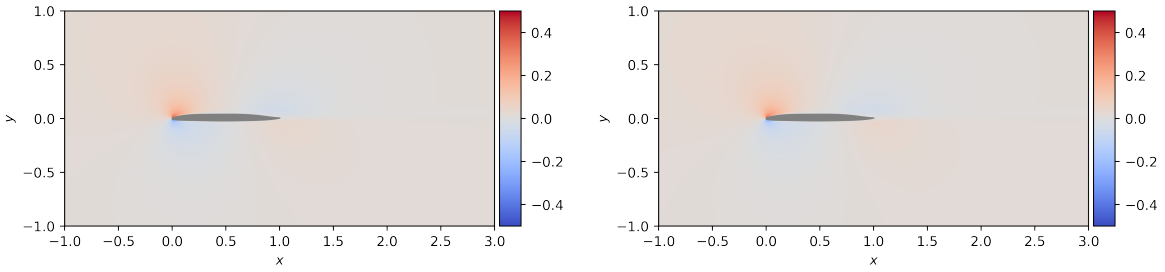

**(a)** PINN prediction

**(b)** OpenFOAM data



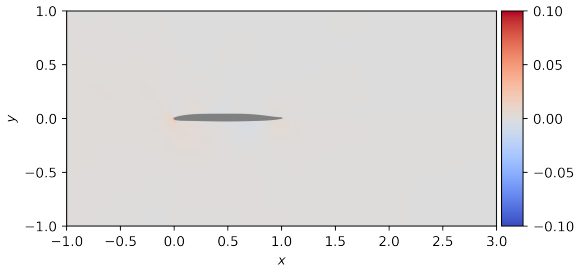

**(c)** difference

**Figure C.11:** Pressure field ($p$) of the lift optimized PARSEC airfoil with initial geometry 11.



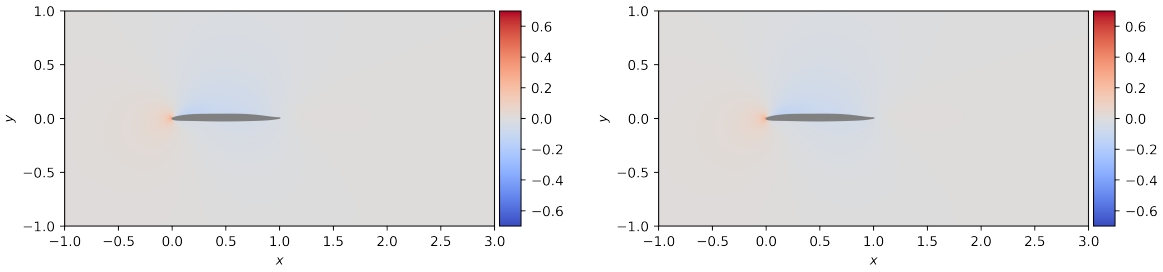

**(a)** PINN prediction

**(b)** OpenFOAM data



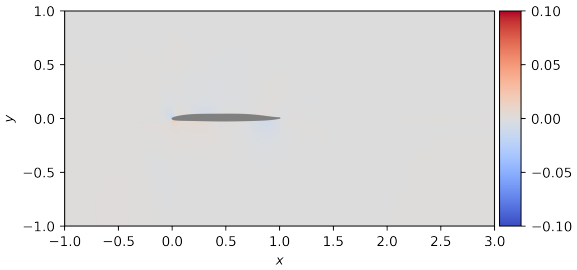

**(c)** difference

**Figure C.12:** Velocity field ($v$) of the lift optimized PARSEC airfoil with initial geometry 8.




**(a)** PINN prediction

**(b)** OpenFOAM data




**(c)** difference

**Figure C.13:** Pressure field ($p$) of the lift optimized PARSEC airfoil with initial geometry 8.