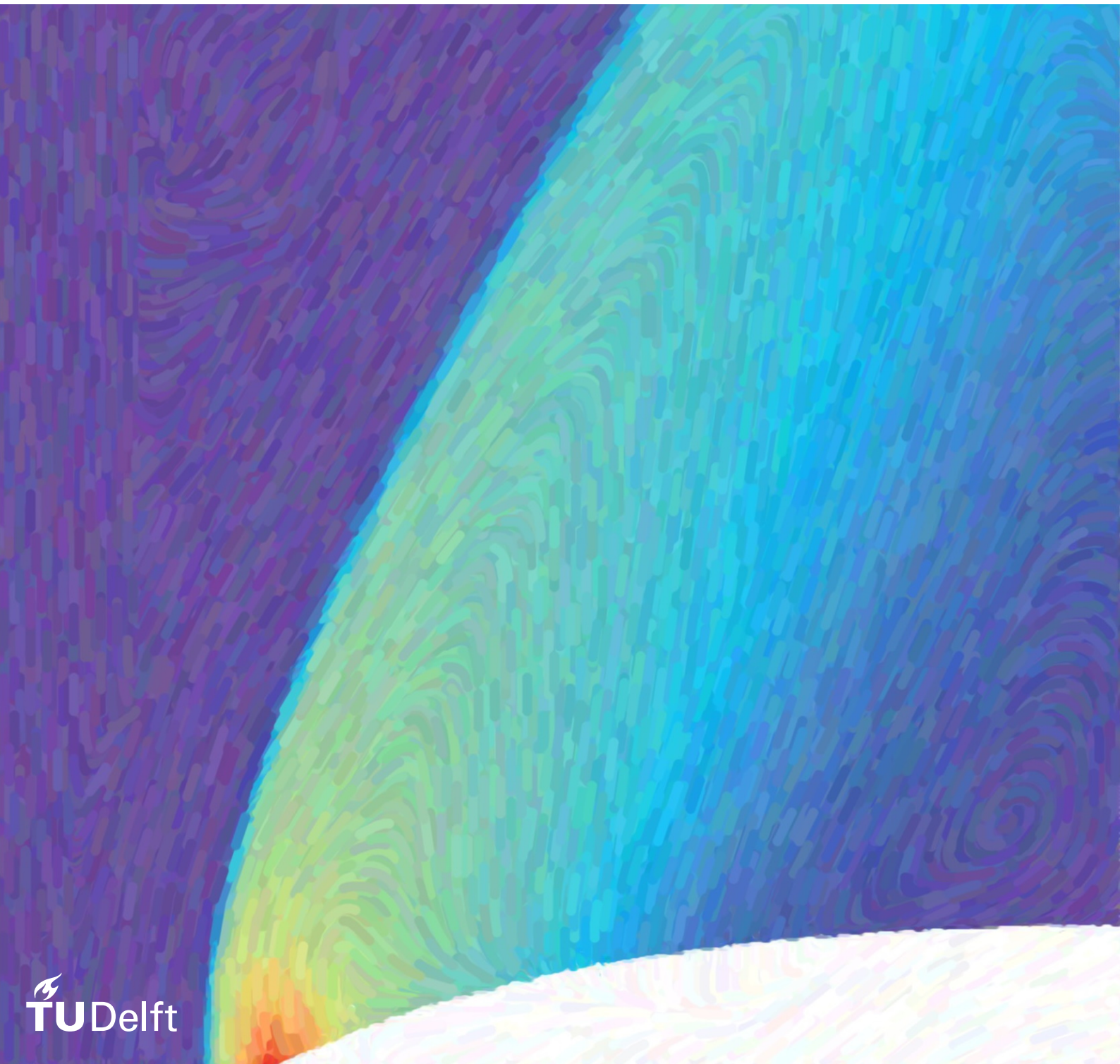
Physics-informed neural networks for highly compressible flows

Assessing and enhancing shock-capturing capabilities

by Thomas Wagenaar



TU Delft

Physics-informed neural networks for highly compressible flows

Assessing and enhancing shock-capturing capabilities

Thesis report

by

Thomas Wagenaar

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on September 15, 2023 at 14:00

Thesis committee:

Chair:	Dr.ir. M.I. Gerritsma
Supervisor:	Dr. N.A.K. Doan
External examiner:	Dr. X. Wang
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	November, 2022 - September, 2023
Student number:	5417414

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

My fascination for machine learning started long ago, even before I started studying. At the time, I developed an open-source machine-learning library which I initially created to train target-tracking aircraft. Despite my interest in machine learning, I decided to pursue a bachelor's in Applied Physics at Eindhoven University of Technology, reasoning that it is more beneficial to study a field in which machine learning can be used instead of machine learning itself. After I obtained my bachelor's degree in 2021, my growing interest in aerospace engineering persuaded me to pursue a master's in Aerospace Engineering at Delft University of Technology.

During both degrees, I tried to take as many courses on machine learning and artificial intelligence as possible, while at the same time working on various personal projects in the same fields. Towards the end of my master's degree, it was time to select a thesis topic. Although machine learning was still one of my biggest interests, I was not sure I wanted to pursue a thesis involving it. My bachelor's thesis on convolutional neural networks for turbulence measurements taught me that big machine-learning projects involve lots of hyperparameter tuning and consequently lots of waiting while models are training.

Nevertheless, other potential topics did not interest me enough. When I saw the thesis on competitive physics-informed neural networks for fluid dynamics proposed by Dr. Doan, I was immediately persuaded. While the initial thesis aimed to compare the recently introduced and promising competitive physics-informed neural networks to conventional physics-informed neural networks, the first results quickly tempered our expectations. However, one of the comparisons on highly compressible problems showed interesting and most importantly counterintuitive results, which sparked a change in topic.

Before you lies the resulting master thesis, *"Physics-informed neural networks for highly compressible flows"*. It is not only targeted at those who involve themselves with physics-informed neural networks but also at those who have an interest in highly compressible flows. The results provide a first step in the direction of fast and errorless shock-capturing methods. Based on my experience, I believe that physics-informed neural networks or a future variant thereof will become the future standard of such simulations.

I would like to thank my supervisor, Dr. Doan, for the independence he has given me while at the same time providing guidance whenever I needed it. While this thesis did not turn out to be what we intended, I undoubtedly think its findings are an interesting experience for both of us. I would also like to thank my family and the friends I have made along the way for their support.

Enjoy the read!

Thomas Wagenaar
Eindhoven, August 14th, 2023

Abstract

While physics-informed neural networks have been shown to accurately solve a wide range of fluid dynamics problems, their effectivity on highly compressible flows is so far limited. In particular, they struggle with transonic and supersonic problems that involve discontinuities such as shocks. While there have been multiple efforts to alleviate the nonphysical phenomena that arise on such problems, current work does not identify and address the underlying failure modes sufficiently. This thesis shows that physics-informed neural networks struggle with highly compressible problems for two independent reasons. Firstly, the differential Euler equations conserve entropy along streamlines, so that physics-informed neural networks try to find an isentropic solution to a non-isentropic problem. Secondly, conventional slip boundary conditions form strong local minima that result in fictive objects that simplify the flow. In response to these failure modes, two new adaptations are introduced, namely a local viscosity method and a streamline output representation. The local viscosity method includes viscosity as an additional network output and adds a viscous loss term to the loss function, resulting in localized viscosity that facilitates the entropy change at shocks. Furthermore, the streamline output representation provides a more natural formulation of the slip boundary conditions, which prevents zero-velocity regions while promoting shock attachment. To the author's best knowledge, this thesis provides the first inviscid steady solutions of curved and detached shocks by physics-informed neural networks.

Table of contents

Nomenclature

List of abbreviations

ACGD	Adaptive Competitive Gradient Descent
Adam	Adaptive moment estimation
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CFD	Computational Fluid Dynamics
CGD	Competitive Gradient Descent
CPINN	Competitive Physics-Informed Neural Network
GAN	Generative Adversarial Network
LAAF	Locally Adaptive Activation Function
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
RANS	Reynolds-Averaged Navier-Stokes

List of symbols

$\boldsymbol{\lambda}$	PDE parameters
$\boldsymbol{\theta}$	Model parameters
η	Learning rate
a	Activation values
b	Biases
f	Frequencies
u	Dependent variables
v	Velocity
x	Spatial variables
I	Identity matrix
W	Weights
α	Hyperparameter
β	Wave angle, momentum decay
δ	Shock detachment distance
ϵ	Small constant
γ	Heat capacity ratio
λ	Residual loss multiplier
B	Bézier curve
d	Discriminator parameters
g	Generator parameters
m	Moment estimates
ν	Viscosity
Ω	Spatial domain
Φ	Potential
ϕ	Flow direction
ρ	Density
θ	Surface inclination
a	Slope recovery term
c	Speed of sound
c_p, c_v	Specific heat capacity
D	Discriminator, Hessian
E	Total energy
e	Specific internal energy
f	Activation function
G	Generator
L	Loss
L_2	Error
M	Mach number
n	Slope recovery multiplier
P	Partial differential equation
p	Pressure
R	Ideal gas constant
S	Surface
s	Entropy
T	Temperature
t	Time
u	Horizontal velocity
V	Volume
v	Vertical velocity
w	Loss term weight
x	Horizontal coordinate
y	Vertical coordinate
z	Bézier parameter

”Much of computational fluid dynamics today is still more of an art than a science; each different problem usually requires special thought and originality in its solution.”

- **John D. Anderson**

1 Introduction

In recent years, the development of machine-learning methods has accelerated and the state-of-the-art has now reached a point where it is becoming the cornerstone of many disciplines. However, the field of Computational Fluid Dynamics (CFD) has remained relatively unaffected by these developments and machine-learning methods have been at most instrumental but not necessarily a replacement of traditional methods (Brunton et al., 2020). Although attempts have been made to use neural networks for simulations directly (Lagaris et al., 1998), they have not been able to compete with traditional solvers. It is only recently that this paradigm has started to shift, after the introduction of Physics-Informed Neural Networks (PINNs) by Raissi et al. (2018).

These PINNs work fundamentally differently than traditional solvers, as they do not rely on a discrete mesh and can produce a continuous solution. Furthermore, they are able to include measurement data or handle missing boundary conditions without significant effort, which is currently not possible with traditional solvers. While neural networks are complex mathematical structures, high-level machine-learning libraries allow them to be used by a wide audience. This contrasts with traditional solvers, which require significant background knowledge by the user to set up correctly. Despite these advantages, PINNs are relatively slow and cannot provide the same accuracy as traditional methods due to numerous failure modes (Krishnapriyan et al., 2021). Nevertheless, their simplicity compared to traditional solvers is appealing, and as a result, research is focused on fixing the failure modes.

Consequently, significant advancements have been made in identifying and fixing these failure modes, allowing a wide range of problems to be tackled by PINNs. For example, they have successfully been used for everything from electromagnetic analysis (Khan et al., 2022) to modeling of gravitational fields (Martin et al., 2022). Interestingly, it seems like fluid dynamics is a natural candidate as most papers that discuss features and properties of PINNs do so by applying them to fluid dynamics problems. In fact, the foundational paper by Raissi et al. (2018) that preceded the original PINN paper by Raissi et al. (2019) only considered fluid dynamics problems such as vortex shedding around a cylinder and an internal flow around an object. Nevertheless, both of these papers involved data assimilation and inverse problems, which rely on partial knowledge of the solution, unlike forward problems.

The first laminar forward problems were treated by Rao et al. (2020), who considered the Navier-Stokes equations at low Reynolds numbers to simulate the flow around a cylinder. While both steady and transient behavior is considered, it did not involve unsteady vortex shedding phenomena that occur at certain Reynolds numbers (Kármán, 1911). However, this was successfully done by Jin et al. (2021), who also showed that the results improved when larger weights are used for the initial and boundary conditions. Although laminar cylinder flows are relatively simple, their setup can easily be extended to more complex objects. For example, both Ang et al. (2022) and Sun et al. (2023) have shown that PINNs can successfully model the laminar flow around airfoils of various shapes and sizes at different angles of attack.

Naturally, the flow becomes more complex at larger Reynolds numbers as the onset of turbulence begins. Even for traditional CFD solvers these flows are troublesome and require either a very fine discretization or the usage of the Reynolds-Averaged Navier-Stokes (RANS) equations with empirical models of turbulence. These methods can certainly be combined with PINNs too, for example Eivazi et al. (2022) has shown that PINNs with RANS can successfully simulate boundary layers, even without the specification of a turbulence model. Instead, the fluctuation terms are specified at the boundaries and the PINN is free to devise its own turbulence model.

When simulating the Navier-Stokes equations directly, PINNs struggle due to the causality issues that arise from the strong transient nature of turbulence. Turbulence is a chaotic process, implying that a small difference in the past flow state can have large implications on the future flow state. Since PINNs minimize the residual uniformly over the temporal domain and do not take into account causality, they converge slowly. Nevertheless, Jin et al. (2021) have shown that a turbulent channel flow can be successfully simulated over short time intervals. Furthermore, the causality alleviations by for example Wang et al. (2022a) are promising and provide indications that PINNs can sustain accurate turbulence predictions over longer time intervals.

In terms of highly compressible flows, Mao et al. (2020) have shown that PINNs can capture both unsteady and steady oblique shocks, although they only treated those that arise directly from boundary conditions and not around objects. A slightly more applied shock is considered by Laubscher et al. (2022), namely an oblique shock over a wedge. They show that PINNs fail on such a problem, although some adjustments can lead to more accurate shocks. Furthermore, while simulations of more complicated detached shocks exist, they rely on partial knowledge of the solution (Cai et al., 2021; Wassing et al., 2023) or on modifications of the Partial Differential Equations (PDEs) (Liu et al., 2022). In short, the current state-of-the-art in terms of highly compressible flows is limited, despite their relatively low complexity. While multiple authors have proposed adaptations to achieve the capture of more complex shocks, their underlying theories are very diverse. From a global point of view, they form pieces of a puzzle that is yet to be solved.

Research objectives

In short, the research objectives of this thesis are:

> To provide a unifying theory on why PINNs fail on highly compressible problems and to design new adaptations that can alleviate these failure modes to allow for the simulation of more complex shocks such as curved and detached shocks.

These objectives can be divided into three fundamental research questions, namely:

1. **Why do PINNs fail on highly compressible problems?** Although PINNs suffer from a large range of failure modes, unique failure modes seem to arise on highly compressible problems. To answer this question, it is important to understand the general PINN failure modes so that they can be isolated from highly compressible failure modes.

2. **How do existing adaptations relate to these failure modes?** Multiple authors have proposed PINN adaptations that promise to improve their performance on highly compressible problems, although each of them is based on a different explanation. It is therefore important to relate these adaptations to the failure mode theory to correctly understand why they work but also to formulate their advantages and disadvantages.

3. **What adaptations can alleviate these failure modes?** While existing adaptations enhance the performance of PINN on highly compressible problems, they have still not been shown to simulate complex shocks. Therefore new adaptations must be designed that not only anchor themselves in the failure mode theory but also allow for the simulation of curved and detached shocks.

Thesis outline

The fundamental principles of PINNs are explained in Chapter 2, along with an overview of their known general failure modes. In addition, a new variant of PINNs is discussed as it displays interesting behavior on highly compressible problems. Afterward, Chapter 3 provides an overview of compressible flow theory, together with a summary of phenomena that can arise in highly compressible flows. It also shows how PINNs can be used to simulate highly compressible flows. Then, Chapter 4 applies PINNs to simple compressible problems to form a theory on their failure modes, which is related to existing adaptations. This theory is used in Chapter 5 to design novel adaptations that can alleviate the failure modes, which are then applied to oblique, curved and detached shocks in Chapter 6. Lastly, the findings are concluded in Chapter 7 and recommendations for further research are made in Chapter 8.

Part I

Background

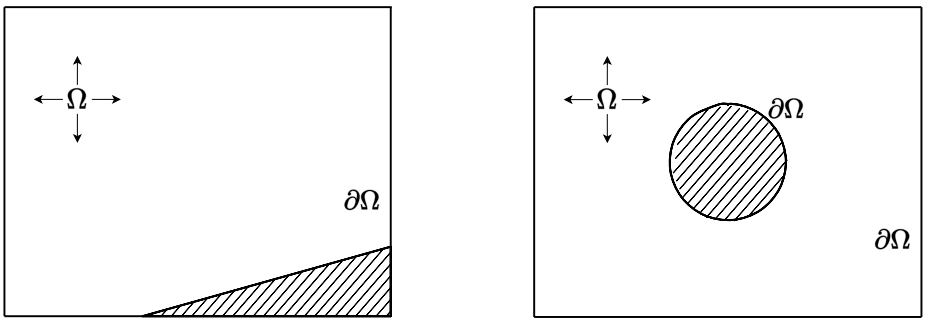
2

Physics-informed neural networks

While PINNs can successfully solve a wide range of fluid dynamics problems, they are certainly not perfect and suffer from a large catalog of failure modes. Therefore, it is important to understand their fundamental principles and general failure modes before trying to explain their behavior on highly compressible problems. This also allows to separate the general failure modes from the highly compressible failure modes, paving the way for tailored adaptations. The purpose of this chapter is to provide such an overview, as well as discuss the various general adaptations that have already been proposed. The fundamental principles of PINNs are introduced in Section 2.1, followed by an overview of failure modes and corresponding adaptations in Section 2.2. In addition, one recent adaptation is highlighted in Section 2.3 as it turns out to be beneficial for highly compressible problems.

2.1 Fundamental principles

The first step in understanding PINNs is to define a generic problem described by PDEs. First of all, each problem involves a domain Ω that is enclosed by a surface $\partial\Omega$, which can consist of both exterior and interior surfaces as shown in Fig. 2.1. Inside the domain, there is a set of variables $\mathbf{u}$ that is governed by a set of PDEs $\{P_i\}$, typically the Euler or Navier-Stokes equations in case of fluid dynamics problems. As a result, the variables may vary over both space, $\mathbf{x} \in \Omega$, and time, $t \in \mathbb{R}^+$. These variations should satisfy the PDEs, which essentially specify that some combination of the variables and their derivatives should be zero, giving rise to the first equation in Eq. (2.1).



(a) Irregular domain **(b)** Hollow domain

Figure 2.1: Examples of two-dimensional domains.

Generally, the set of PDEs alone does not uniquely prescribe the solution for $\mathbf{u}(\mathbf{x}, t)$ (Haberman, 2014). Instead, they must be accompanied by a set of boundary conditions and a set of initial conditions. The boundary conditions $\mathbf{u}_b(\mathbf{x}, t)$ prescribe the value of the variables or their derivatives at the surface $\partial\Omega$ and may vary as a function of time, while the initial conditions $\mathbf{u}_0(\mathbf{x})$ prescribe the value of the variables on the entire domain Ω at $t = 0$. When the initial and boundary conditions are formulated in such a way that they confine the problem to having a single unique solution, the problem defined in Eq. (2.1) is *well-posed*.

Note that such problems are generally not solved for all times t, but rather until some fixed end time.

$$\begin{aligned} \forall_i P_i(\mathbf{u}(\mathbf{x},t)) &= 0, & \mathbf{x} &\in \Omega, & t &\in \mathbb{R}^+, \\ \mathbf{u}(\mathbf{x},t) &= \mathbf{u}_b(\mathbf{x},t), & \mathbf{x} &\in \partial\Omega, & t &\in \mathbb{R}^+, \\ \mathbf{u}(\mathbf{x},0) &= \mathbf{u}_0(\mathbf{x}), & \mathbf{x} &\in \Omega. \end{aligned} \tag{2.1}$$

Traditionally, these problems are solved by either finite element methods (Reddy, 2019), finite difference methods (Versteeg et al., 2007) or spectral methods (Canuto et al., 2012). Each of these methods subdivides the domain Ω into many smaller subdomains, after which the discretized system of equations is iteratively solved. While this approach is reliable, it involves complex numerical schemes and requires considerable manual work to set up. Furthermore, the accuracy of the resulting solution is inherently limited by the resolution of the mesh. For turbulent flows, this can be detrimental because the smallest scales are of vital importance for the overall solution (Nieuwstadt et al., 2018). Nevertheless, advances in turbulence modeling allow for fairly accurate results even for turbulent flows.

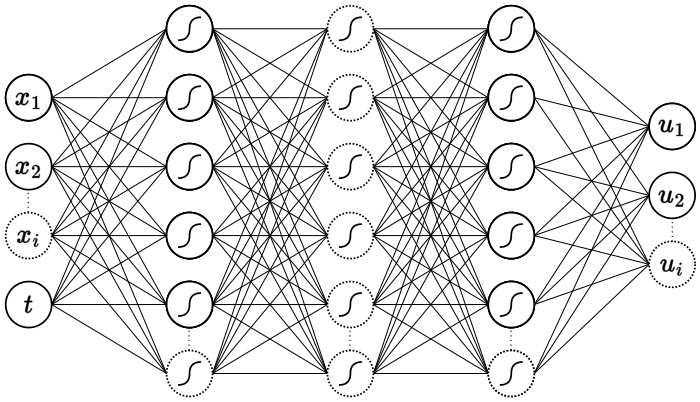


Figure 2.2: A layered neural network with inputs $(\mathbf{x}, t)$ and outputs $\mathbf{u} = [u_1, u_2, ...]$.

Recently, Raissi et al. (2019) proposed a new method based on the original idea of Lagaris et al. (1998). Instead of discretizing the domain and the governing PDEs, they proposed to represent the solution $\mathbf{u}(\mathbf{x}, t)$ with a neural network as shown in Fig. 2.2. Neural networks consist of neurons that are connected to each other through weights. Each neuron calculates its state by summing the product of activation values of incoming connections with these weights, after which a typically nonlinear activation function is applied to attain its activation value. The neurons are generally grouped in layers, where all neurons from one layer are connected to all the neurons of the succeeding layer as shown in Fig. 2.2. At the first layer, the activation functions are set to the input variables. In other words $\mathbf{a}_0 = [x_1, x_2, ..., t]$, where $\mathbf{a}_0$ are the activation values of the first layer. For each layer i after the input layer, Eq. (2.2) is used where f is the activation function, W is a weight matrix and $\mathbf{b}$ is a bias vector. If the activation functions are continuous, the outputs of the neural network will also be continuous. This is a significant advantage compared to traditional solvers, which only provide a discretized solution at specific locations on a mesh.

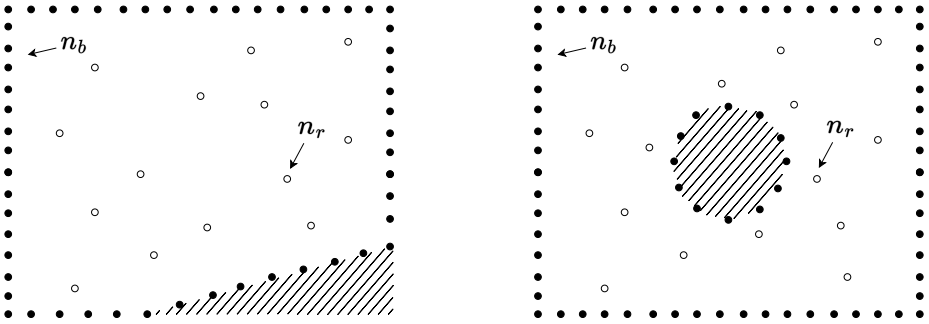
$$\mathbf{a}_i = f(W_{i-1,i}\mathbf{a}_{i-1} + \mathbf{b}_i) \tag{2.2}$$

The nonlinearity of the activation functions allows neural networks to approximate complex functions and they have even been proven to be universal function approximators (Cybenko, 1989; Hornik et al., 1989). In other words, they can in theory approximate any function to any desired accuracy given the neural network consists of enough neurons. The process of approximating a given function is done through a *loss function*, which measures how well the neural network approximates the function. Given certain inputs, the loss of the neural network can be calculated based on its corresponding outputs. This loss can then be backpropagated so that the gradient of each of the weights and biases with respect to the loss is known. Through gradient descent algorithms, this gradient can be used to incrementally change these parameters $\boldsymbol{\theta}$ to minimize the loss. For a neural network to learn the solution to a PDE problem, the loss must thus express how wrong its current solution is. Since the target solution is not known, Raissi et al.

(2019) introduced the loss function

$$\begin{gathered} L = L_0 + L_b + L_r, \\ L_0 = \|\mathbf{u}(\mathbf{x}, 0) - \mathbf{u}_0(\mathbf{x})\|^2, \quad L_b = \|\mathbf{u}(\mathbf{x}, t) - \mathbf{u}_b(\mathbf{x}, t)\|^2, \quad L_r = \|P(\mathbf{u}(\mathbf{x}, t))\|^2. \end{gathered} \tag{2.3}$$

Essentially, it consists of three different terms which respectively represent the loss over the initial conditions, the boundary conditions and the PDEs. The initial and boundary losses are calculated by taking the squared difference of the current solution with the prescribed conditions, while the PDE loss is calculated by taking the squared *residuals*. The residuals are the values that remain on the right-hand side when applying the PDEs to a trial solution; lower residuals imply that the trial solution satisfies the PDEs better. Note that it is typical for losses to be expressed in squared instead of absolute differences because the loss remains differentiable when the difference is zero. In addition, it also penalizes outliers. Neural networks that are trained using the above loss function are referred to as PINNs because the loss contains a term that is based on physical PDEs.



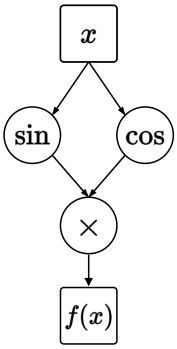
(a) Irregular domain **(b)** Hollow domain

Figure 2.3: Examples of collocation points for two-dimensional domains, excluding the initial condition points.

In practice, it is unfeasible to calculate the exact values of the loss terms in Eq. (2.3) through integration because neural networks are highly nonlinear, making it hard to obtain a primitive of the loss function in terms of their parameters. Therefore, each loss term has an associated set of collocation points that are spread over space and time, as shown in Fig. 2.3. At these points, the local loss is calculated to estimate the overall loss as is done in Eq. (2.4). The different sampling methods that can be used to generate the collocation points are discussed in Section 2.2.1, along with their advantages and disadvantages.

$$\begin{gathered} L = L_0 + L_b + L_r, \\ L_0 = \frac{1}{n_0}\sum_{i=1}^{n_0} \|\mathbf{u}(\mathbf{x}_i, 0) - \mathbf{u}_0(\mathbf{x}_i)\|^2, \; L_b = \frac{1}{n_b}\sum_{i=1}^{n_b} \|\mathbf{u}(\mathbf{x}_i, t_i) - \mathbf{u}_b(\mathbf{x}_i, t_i)\|^2, \; L_r = \frac{1}{n_r}\sum_{i=1}^{n_r} \|P(\mathbf{u}(\mathbf{x}_i, t_i))\|^2. \end{gathered} \tag{2.4}$$

One might argue that the usage of collocation points bears similarities with the discrete meshes used in traditional solvers, but there is an important difference. While the usage of meshes requires the discretization of derivatives in the PDEs, introducing discretization errors into the solution, the derivatives of the solution provided by the neural network can be calculated analytically. Although the activation functions of neurons are nonlinear, they are generally differentiable to allow for backpropagation of the loss. As a result, the derivatives of the solution with respect to the spatial and temporal variables can also be calculated through repeated application of the product rule. If all the activation functions in the neural network are differentiable up to the same order as the PDEs, the derivatives are also continuously defined. This process is called *automatic differentiation* (Baydin et al., 2017) and is a key advantage of PINNs. Note that automatic differentiation is different from symbolic differentiation, as the full neural network is never differentiated symbolically but rather the derivatives are calculated at each computational node during forward passes as shown in Fig. 2.4.



Symbolic	$f'(2) = \frac{d}{dx}(\sin(x)\cos(x))\Big\|_{x=2} = \|\cos(x)^2 - \sin(x)^2\Big\|_{x=2}$
Automatic	$f'(2) = \sin(2)\frac{d}{dx}(\cos(x))\Big\|_{x=2} + \frac{d}{dx}(\sin(x))\Big\|_{x=2}\cos(2)$
Numerical (e.g.)	$f'(2) = \frac{\sin(2+h)\cos(x+h) - \sin(2-h)\cos(x-h)}{2h}$

Figure 2.4: Comparison of symbolic, automatic and numerical differentiation.

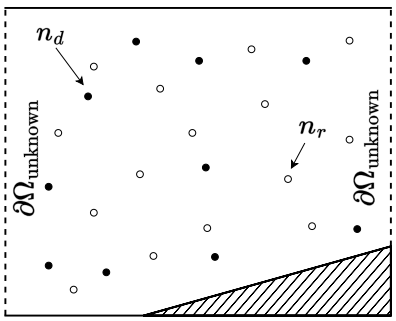
However, similar to meshes, the number and placement of collocation points influence the accuracy of the obtained solution. This is because the neural network only minimizes the loss at the specified points, and not necessarily over the whole domain. When the loss over the set of collocation points used for training is significantly lower than over a different set of collocation points, the neural network is said to be *overfitting*. This can be monitored by keeping track of the loss on a separate set of collocation points that is not used for gradient descent. Alternatively, if the exact solution is available, the L_2 error can be calculated via Eq. (2.5) to monitor the deviation with respect to the true solution.

$$L_2 = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{u}(\mathbf{x}_i, t_i) - \mathbf{u}_{\text{true}}(\mathbf{x}_i, t_i)\|^2} \tag{2.5}$$

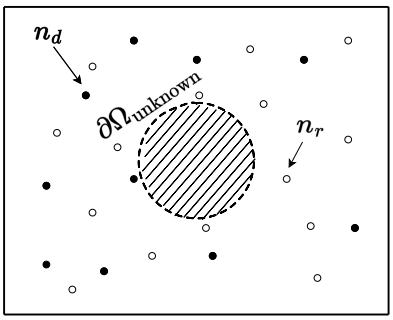
The error defined in Eq. (2.5) and the residual used in the loss function in Eq. (2.3) are generally not proportional to each other, although they are related. When the residuals are low, it only implies that the obtained solution satisfies the PDEs and is therefore physical. Nevertheless, if the initial and boundary conditions losses are large, the solution will be incorrect because it is a physical solution to a different set of initial and boundary conditions. But even when all losses are relatively low, locally high residuals can lead to incorrect propagation of the boundary conditions to the rest of the domain. Therefore, only *uniformly* low residuals lead to a low error with respect to the true solution. However, keep in mind that the relationship between the loss and error is nonlinear due to the nonlinear nature of PDEs, although some authors are investigating alternative loss definitions to linearize this relationship (Taylor et al., 2023).

2.1.1 Data assimilation and inverse problems

Problems that have a loss function of the form given in Eq. (2.3) are called forward problems and are most similar to traditional CFD methods. Essentially, they involve finding a physical solution to a set of initial and boundary conditions. However, one of the main advantages of PINNs is that their loss function is flexible, implying it can also incorporate additional loss terms. This allows PINNs to be applied to data assimilation problems for example, which are problems where additional knowledge on the solution or its derivatives is available at some points in the domain.



(a) Unknown inlet and outlet conditions



(b) Unknown surface conditions

Figure 2.5: Sketch of two-dimensional data assimilation problems with unknown boundaries.

The mentioned knowledge could be provided by other numerical methods, but more interestingly it could come from experimental measurements. While this knowledge can help the convergence of PINNs on well-posed problems, it also allows solutions to be found to underdetermined problems where the initial and boundary conditions are partly unknown or simply uncertain. Furthermore, data assimilation can also be applied as a physics-informed interpolation scheme to provide super-resolution on existing provided numerical or experimental data, which has been performed by both Jiang et al. (2020) and Wang et al. (2020). A more practical example is visualized in Fig. 2.5a, where the inlet and outlet conditions might be uncertain but measurements are conducted inside the domain at various locations. To include measurements or any additional knowledge, an extra term can be added to Eq. (2.4) of the form

$$L_d = \frac{1}{n_d} \sum_{i=1}^{n_d} \|\mathbf{u}(\mathbf{x}_i, t_i) - \mathbf{u}_d(\mathbf{x}_i, t_i)\|^2. \tag{2.6}$$

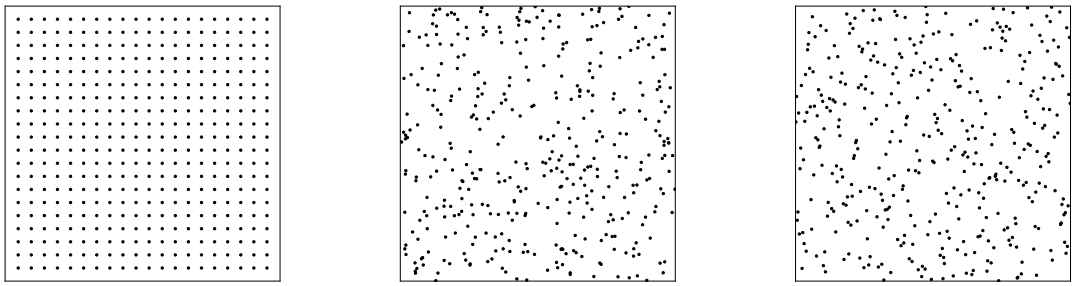
where $\mathbf{u}_d(\mathbf{x}_i, t_i)$ is the known value of the solution or its derivatives provided at some position $\mathbf{x}_i$ at some time t_i. Of course, it does not make much sense to apply data assimilation on problems with unknown inlet conditions since the PDEs often also contain constants based on these conditions. This could be for example the density or viscosity of the fluid or the dimensionless Reynolds number. Since the PDEs are usually nonlinear, a small change in such a constant could have a large consequence on the flow phenomena. As a result, it might not even be possible to perform data assimilation if the incorrect values for these constants are assumed. Therefore, an approach is used based on the field of model identification (Wang et al., 2018), where the model constants are learned during training as well. They are simply considered additional parameters to be learned during backpropagation and gradient descent, similar to the parameters of the neural network. This class of problems is called *inverse problems* since the goal is to infer the PDEs from flow phenomena instead of the other way around. Note that this introduces no additional loss term in Eq. (2.3), as the model constants are already part of the PDEs required to calculate the residual loss term.

2.2 Limitations and adaptations

Although PINNs have been shown to effectively solve a wide range of fluid dynamics problems, they require various tweaks and tricks to do so. This is because PINNs suffer from two issues: first of all, Markidis (2021) has shown that PINNs are generally slower than traditional solvers based on iterative methods. Second of all, they are generally unable to easily reach the same residuals and therefore errors as traditional solvers. In the case of highly compressible fluid dynamics, the lack of applied cases is likely related to the latter issue. Therefore, this section provides an overview of the current knowledge of the failure modes that underlay this issue, as well as the methods that have been proposed to mitigate their effects. For more information about the failure modes that affect the computational cost of PINNs, the reader is referred to domain decomposition (Karniadakis, 2020; Han et al., 2023) and meta-learning (Markidis, 2021; Liu et al., 2022) methods.

2.2.1 Sampling methods

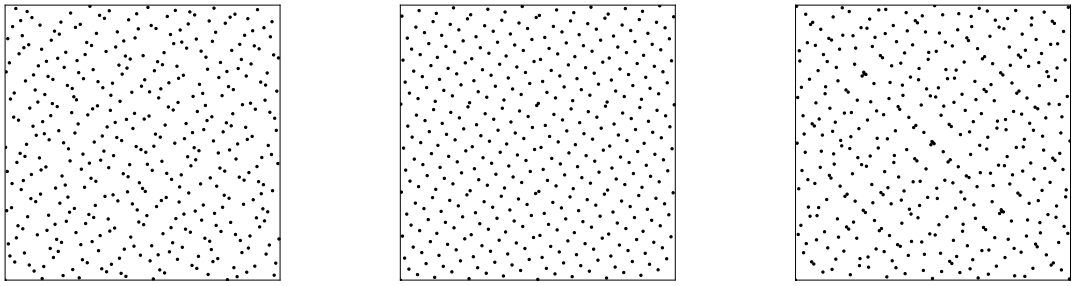
Although PINNs provide continuous solutions, a finite number of collocation points is used to assess the loss function as shown in Eq. (2.4). Similar to the quality of meshes in traditional solvers, the placement of these collocation points has a large influence on the error of the obtained solution. As mentioned, this is partly a consequence of PINNs minimizing the residuals rather than the errors, since low residuals only imply that the solution at the collocation points satisfies the PDEs. When the residual points fail to sufficiently map regions with high gradients, Daw et al. (2022) argue that the boundary corrections incorrectly propagate to the rest of the domain, leading to low residuals yet high errors. In other words, the solution is physical but represents different boundary conditions. This can not only occur in the spatial domain but also in the temporal domain: when initial conditions incorrectly propagate because of bad sampling, (Wang et al., 2022a) argue that the principle of causality is violated and the errors at future times can be high despite low residuals. To counteract this, several sampling methods have been proposed.



(a) Grid sampling **(b)** Uniform random sampling **(c)** Latin hypercube sampling

Figure 2.6: Selection of uniform non-adaptive sampling methods.

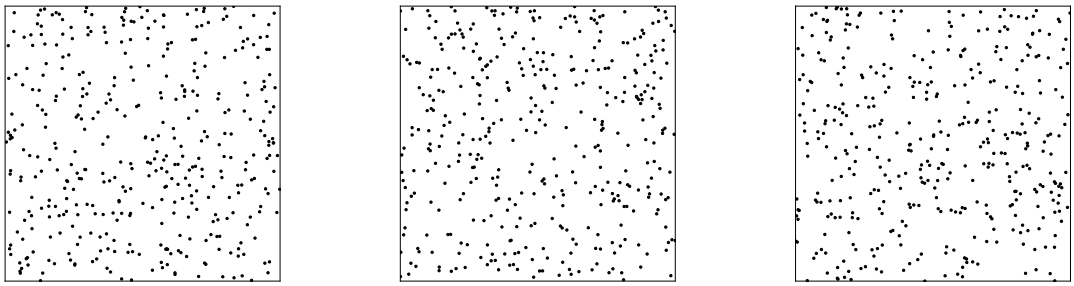
The first class of sampling methods involves static or non-adaptive sampling methods, which provide collocation points independent of the problem. An extensive overview of these methods is provided by Wu et al. (2023), who identify six main methods that are visualized in Fig. 2.6 and Fig. 2.7 and apply these to a set of problems to compare their performance. The first method is grid sampling, where points are placed at the nodes of a regular lattice. This method has the worst performance, which also explains why there are not many papers using it. The second method is uniform random sampling, where points are sampled independently from a uniform distribution. It performs slightly better and is more consistent, perhaps because it is invariant to rotations. Nevertheless, uniform random sampling can lead to large regions without any collocation points because they are independently sampled, which can worsen propagation failures. It is therefore no surprise that this method only appears in literature to highlight its problems (e.g. Mao et al. (2020)). The third method is Latin hypercube sampling, which is similar to uniform random sampling but makes sure that the samples uniformly cover the ranges of both variables. Interestingly, Wu et al. (2023) do not find it to improve over uniform random sampling, but this sampling method is particularly prevalent and is used by for example Raissi et al. (2019) and Rao et al. (2020).



(a) Halton sampling **(b)** Hammersley sampling **(c)** Sobol sampling

Figure 2.7: Selection of low-discrepancy non-adaptive sampling methods.

While the above uniform sampling methods are simple to use, they do not take into account that the sampled points are used to approximate an integral of the loss function in Eq. (2.3). There are in fact several sampling methods that optimize the placement of points for faster convergence to the true value of integrals. An example is the Sobol sampling method, which is a quasi-random low-discrepancy sequence. Wu et al. (2023) show that this method and the other low-discrepancy methods in Fig. 2.7 improve the accuracy greatly over a large class of problems. Markidis (2021) confirms this finding, although the improvement is less pronounced which is argued to be the result of a larger number of collocation points. This is understandable because, in the limit of infinite collocation points, all the mentioned static methods essentially uniformly cover the domain. Therefore, there will be little difference between the sampling methods if the collocation point density is far higher than the feature density in the solution.

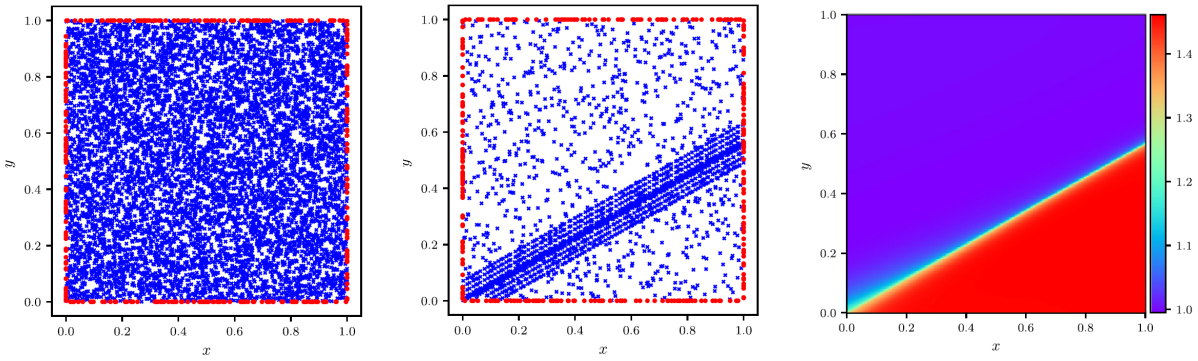


(a) Iteration i **(b)** Iteration $i+1$ **(c)** Iteration $i+2$

Figure 2.8: Resampling collocation points at each iteration using a uniform random sampling method.

The comparison of sampling methods so far assumes that the set of collocation points is fixed. However, it is also possible to resample the collocation points at fixed intervals during the iterative training procedure, as shown in Fig. 2.8. As a result, the mean value of the loss over different iterations converges to the true value in the limit of many iterations because the domain is more uniformly sampled. In addition, the stochasticity of the resulting gradient descent may drive the neural network out of local minima, which are further elaborated upon in Section 2.2.2. Wu et al. (2023) have shown that combining uniform random sampling with resampling leads to a considerable improvement over a large variety of problems.

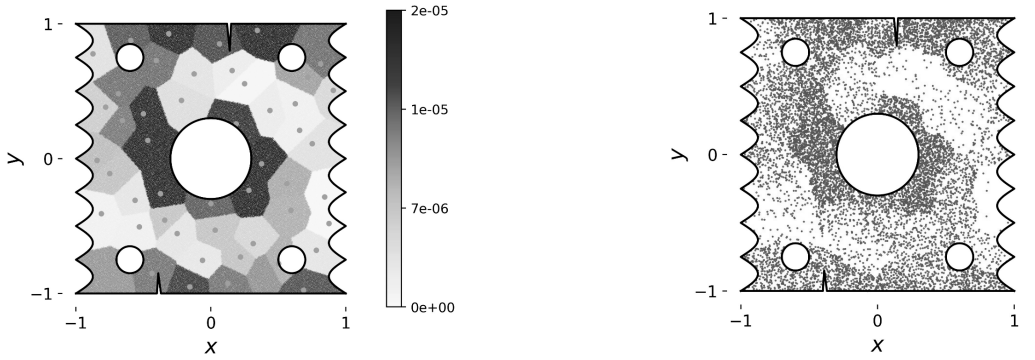
While resampling leads to a significant accuracy increase, it is not clear why an approximately uniform set of collocation points would lead to the lowest error. In fact, Mao et al. (2020) show that they can achieve a considerable performance increase by clustering more collocation points near large gradients in the solution, as shown in Fig. 2.9 for a shock in a highly compressible flow. Intuitively this makes sense because fewer collocation points are necessary to accurately approximate the loss function in areas where there are low gradients compared to areas where there are high gradients. Based on this finding, multiple adaptive sampling methods have been developed that continuously add or relocate collocation points to achieve better accuracy.



(a) Uniformly sampled points **(b)** Clustered points **(c)** Obtained solution

Figure 2.9: Uniform sampled points versus clustered points (Mao et al., 2020).

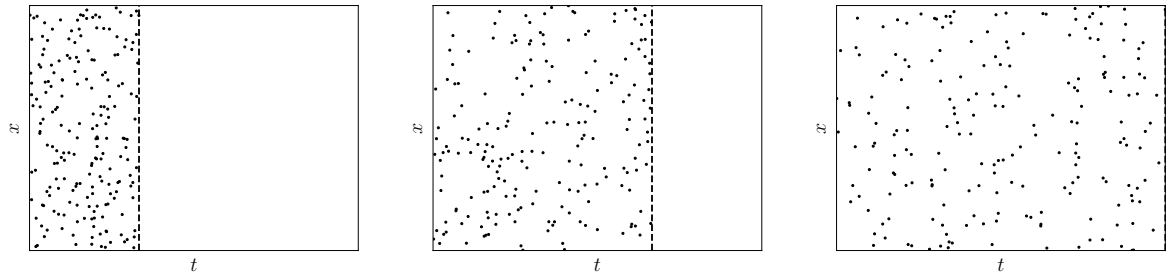
The first class of adaptive methods are based on the residuals at the collocation points and are hence named residual-based adaptive methods. Lu et al. (2021) proposed a method that regularly samples new points uniformly and then selects those with the highest residual to be added to the current set of collocation points. A more sophisticated approach is proposed by Daw et al. (2022), who use an evolutionary algorithm to select and remove collocation points based on a fitness proportional to their residual value. Although both these methods adaptively place points in areas with high residuals, Wu et al. (2023) argue that this approach is too greedy and can cause low residual areas to become underrepresented. A less greedy approach is proposed by Nabian et al. (2021), who sample new collocation points from a probability density function that is proportional to the residuals, as shown in Fig. 2.10. Since sampling from a probability density function can be expensive, Tang et al. (2023) have proposed to use a deep generative model to generate samples instead.



(a) Piecewise probabilities **(b)** Sampled collocation points

Figure 2.10: Sampling from a residual-based piecewise probability density function (Nabian et al., 2021).

While the residual-based adaptive methods provide better results than their non-adaptive counterparts, Wu et al. (2023) show that the accuracy increase is not significant when compared to resampling. In addition, these methods fail to account for the principle of causality: solutions at future times depend on earlier solutions, therefore it is not useful to place many collocation points in the future even though the residuals there are large since lowering the residuals will not necessarily lead to a low error. This has led to the time marching approach by Wight et al. (2021), where collocation points are spread progressively as shown in Fig. 2.11. Intuitively, this method seems most relevant for highly unsteady fluid dynamics problems such as those involving turbulence. However, keep in mind that for steady highly compressible problems in the hypersonic regime, spatial variables can turn time-like as is demonstrated by the hypersonic equivalence principle (Anderson, 2006).



(a) Start of training **(b)** Halfway through training **(c)** End of training

Figure 2.11: Progression of residual points when using the time marching technique by Wight et al. (2021).

2.2.2 Loss simplification

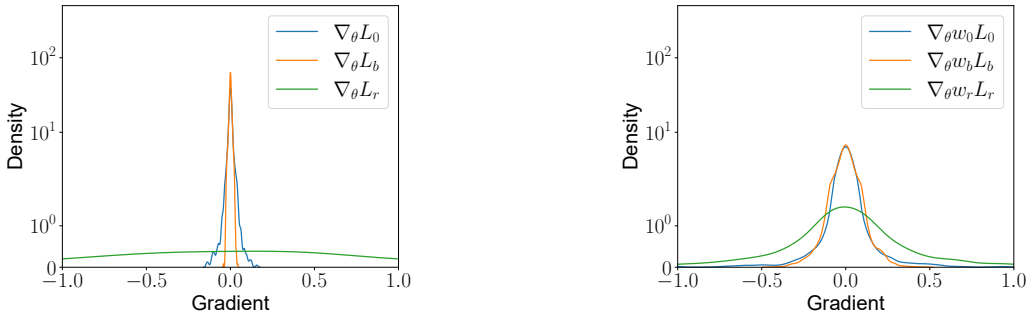
Although advanced sampling methods increase the accuracy of PINNs, Zeng et al. (2022) note that the errors obtained rarely reach machine-precision levels, even on simple problems. This is interesting because, in theory, there exists a solution that exactly satisfies the constraints in Eq. (2.1) given that the problem is well-posed. Intuitively, one might argue that the universal approximation property of neural networks only holds in the limit of infinite neurons (Hornik et al., 1989). Nevertheless, the typical sizes of the networks in PINN papers should give enough expressive power for far better accuracies, giving reason to believe that there is a different issue. A more compelling argument is given by Krishnapriyan et al. (2021), who argue that the residual loss term in Eq. (2.3) complexifies the loss landscape. Since gradient descent algorithms are prone to local minima, they might converge at relatively high loss values. Although Choromanska et al. (2015) argue that convergence to local minima is not inherently disadvantageous because the global minimum might lead to overfitting and thus bad generalization, a balance must be sought between a low loss and proper generalization.

Since the residual loss term complexifies the loss landscape, Wang et al. (2022b) considered a weighted variant of the loss function as shown in Eq. (2.7) and studied the effect of varying just the residual loss

weight w_r. In theory, this should not affect the optimal solution although it can reshape the loss landscape. Interestingly, they found that lower weights lead to a lower final L_2 error. Intuitively this can be understood with the propagation failure theory discussed in Section 2.2.1: if the boundary conditions are hardly satisfied, it makes little sense to start minimizing the residual because it will lead to a solution that satisfies other boundary conditions. This is especially problematic because gradient descent algorithms can converge to a local minimum before the boundary conditions are reasonably satisfied. By increasing the relative importance of the boundary conditions, the correct boundary conditions can propagate before the residual is minimized. As a result, a lower final error is generally obtained.

$$L = w_0 L_0 + w_b L_b + w_r L_r \tag{2.7}$$

While lowering the weight of the residual loss term generally improves the error, the optimal value is highly problem dependent and can depend on the magnitude of the solution if it is not made dimensionless. Therefore, multiple adaptive weighting methods have been developed that automatically adjust the weights of the different loss terms. Wang et al. (2021a) proposed to balance the loss terms in such a way that their contribution to the backpropagated parameter changes is approximately equal, as shown in Fig. 2.12. They showed that this indeed leads to a larger weighting of the initial and boundary conditions, resulting in better overall losses on selected problems. Nevertheless, it is not clear why balancing the contributions to weight changes is the optimal strategy.



(a) No loss balancing

(b) Loss balancing

Figure 2.12: Distribution of backpropagated gradients in a single layer, with and without the loss balancing technique proposed by Wang et al. (2021a).

Another weighting approach is introduced by Liu et al. (2021), who instead propose to adapt the weights using a competitive minimax game. In this game, the relative weights of the loss terms are varied in such a way that the total loss is maximized, as shown in Eq. (2.8). The weights are essentially considered to be parameters that are also updated but using gradient *ascent* instead of gradient descent. Unfortunately, this approach does not improve the L_2 error significantly, although it converges considerably faster.

$$\begin{gathered} \min_{\boldsymbol{\theta}} \max_{w_0, w_b, w_r} L = w_0 L_0 + w_b L_b + w_r L_r, \\ w_0 + w_b + w_r = 1. \end{gathered} \tag{2.8}$$

A more refined approach that uses a similar strategy is introduced by McClenny et al. (2022), who instead weigh the contribution of individual collocation points to the loss function instead of the contribution of entire loss terms. The weights can also be provided by spatiotemporal weighting functions $w(\mathbf{x}, t)$ as shown in Eq. (2.9), which are represented with Gaussian processes. The parameters of these Gaussian processes are then optimized using a minimax game, resulting in an L_2 error that can be orders of magnitude lower compared to no weighting. This considerable increase is likely not only the result of a better loss landscape but also of the fact that the neural network can be punished for having locally high losses. As a result, the network equalizes its loss over space and time which reduces propagation failures.

$$\begin{gathered} \min_{\boldsymbol{\theta}} \max_{w_0, w_b, w_r} L = L_0 + L_b + L_r, \\ L_0 = w_0(\mathbf{x}) \|\mathbf{u}(\mathbf{x}, 0) - \mathbf{u}_0(\mathbf{x})\|^2, \quad L_b = w_b(\mathbf{x}, t) \|\mathbf{u}(\mathbf{x}, t) - \mathbf{u}_b(\mathbf{x}, t)\|^2, \quad L_r = w_r(\mathbf{x}, t) \|P(\mathbf{u}(\mathbf{x}, t))\|^2. \end{gathered} \tag{2.9}$$

Note that the approach of McClenny et al. (2022) bears similarities with self-paced learning methods for neural networks (Kumar et al., 2010), which start by using easy samples for the loss and progressively include harder samples. A self-paced adaptive method has been applied to PINNs by Gu et al. (2021), resulting in the error improving by an order of magnitude. Another example of a self-paced weighting method for PINNs is the casual approach by Wang et al. (2022a), although it was developed to alleviate temporal propagation failures. This method applies a weighting function to each collocation point that depends on the loss at earlier collocation points, as shown in Eq. (2.10) with ϵ a small constant and $t_{i+1} > t_i$. It is designed to respect causality, as small weights are given to collocation points further in the future, but their weights progressively increase between training iterations. Again, this temporal weighting method can not only be relevant for fluid dynamics problems involving turbulence but also for steady highly compressible flows in the hypersonic regime.

$$L_r = \frac{1}{n_r}\sum_{i=1}^{n_r} w_i L_{r,i}, \quad L_{r,i} = \|P(\mathbf{u}(\mathbf{x}_i, t_i))\|^2, \quad w_i = \exp\left(-\epsilon \sum_{k=1}^{i-1} L_{r,k}\right). \tag{2.10}$$

Instead of using weighting methods to alleviate the loss landscape issues, it is also possible to consider more advanced gradient algorithms. In particular, to mitigate convergence to local minima, it is possible to select a different algorithm or even switch between different algorithms during training. The most basic gradient descent algorithm is simple gradient descent as given in Eq. (2.11) (Ruder, 2016), where the update of the parameters $\boldsymbol{\theta}$ only depends on the current gradient of the loss with respect to the parameters. The learning rate η dictates how large the parameter updates should be. Generally, lower learning rates lead to earlier convergence to local minima. On the other hand, a lower learning rate is also necessary to make fine adjustments towards the center of minima once the loss is sufficiently low. If the learning rate is too large, the parameter updates will ”jump” over these minima causing the loss to stagnate.

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_i) \tag{2.11}$$

To avoid premature convergence to local minima, many adaptive gradient descent algorithms have been proposed. One example is Adaptive Moment Estimation (Adam), which has been introduced by Kingma et al. (2014) and has been used by most authors when training PINNs. Similar to other adaptive gradient descent algorithms (Ruder, 2016), it keeps track of the *momentum* of previous gradients. Its workings are given by Algorithm 1, where β_1 and β_2 are hyperparameters that dictate how fast the contribution of past gradients and squared gradients should decay respectively. Furthermore, ϵ is a small non-zero value to avoid division by zero. By using momentum, Adam is more likely to pass over premature local minima.

Algorithm 1: Adaptive Moment Estimation (Kingma et al., 2014)

require base learning rate η
require exponential decay rates β_1, β_2
require loss function $L(\boldsymbol{\theta})$
require initial parameters $\boldsymbol{\theta}_0$

initialize moment estimates $\mathbf{m}_1 = 0$, $\mathbf{m}_2 = 0$

for i *in* $\{0, 1, ...\}$ **do**
  $\mathbf{m}_1 = \beta_1 \mathbf{m}_1 + (1 - \beta_1)(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_i))$
  $\mathbf{m}_2 = \beta_2 \mathbf{m}_2 + (1 - \beta_2)(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_i))^2$
  $\mathbf{m}_1 = \mathbf{m}_1 / (1 - \beta_1^i)$
  $\mathbf{m}_2 = \mathbf{m}_2 / (1 - \beta_2^i)$
  $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \mathbf{m}_1 / (\sqrt{\mathbf{m}_2} + \epsilon)$

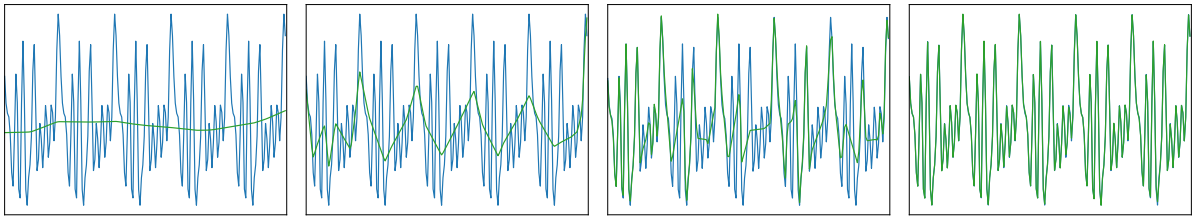
Despite the popularity of the Adam optimizer in literature, the resulting accuracy when using it for PINNs is limited. Therefore, multiple authors such as Markidis (2021) and Jin et al. (2021) have opted to combine Adam with the second-order L-BFGS optimizer, introduced by Liu et al. (1989). In particular, Adam is used as the gradient descent algorithm in the first stage, after which L-BFGS is used to further refine the solution when the loss stagnates. As a result, the loss might decrease by a further two orders of magnitude. In short, the L-BFGS optimizer is a limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which is essentially a *second-order* gradient descent algorithm. Instead of just

using the gradient to determine the descent direction, it also computes the Hessian matrix which includes all second-order derivatives and can be used to estimate the curvature of the loss function. In other words, it also takes into account how much changing a parameter affects the gradients of all parameters. The update rule is given in Eq. (2.12), where $D^2_{\boldsymbol{\theta}\boldsymbol{\theta}} L(\boldsymbol{\theta}_i)$ is the Hessian. In practice, the limited-memory approximation of BFGS is often used because the Hessian matrix grows squared with the number of neural network parameters.

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta D^2_{\boldsymbol{\theta}\boldsymbol{\theta}} L(\boldsymbol{\theta}_i) \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_i) \tag{2.12}$$

2.2.3 Network architecture

Given that an appropriate sampling method is used, smoothening the loss landscape can certainly benefit the accuracy of PINNs. However, it is important to remember that although the PDEs complexify the loss landscape, this landscape itself is expressed in terms of the model parameters. Instead of smoothening the landscape by adapting the loss function, one can therefore also use a different network architecture to change how the loss function is parameterized. In fact, many breakthroughs in machine learning are the result of discipline-specific architectures, such as convolutional neural networks for image processing (O'Shea et al., 2015) and transformers for natural language processing (Vaswani et al., 2017). Therefore, this section gives an overview of the various architectural developments for PINNs.



(a) Iteration 100 **(b)** Iteration 1000 **(c)** Iteration 10000 **(d)** Iteration 80000

Figure 2.13: An illustration of spectral bias adapted from Rahaman et al. (2019), where the blue line is the target function consisting of a superposition of sine waves with different frequencies and the green line is the neural network approximation. It shows how low frequencies are learned first and high frequencies only afterward.

Neural networks inherently suffer from *spectral bias*, which can greatly decrease their convergence rate. In short, it is a bias towards learning lower frequency components in the solution over higher frequency components (Rahaman et al., 2019), as illustrated in Fig. 2.13. This can be especially problematic for flow problems involving turbulence, where solutions consist of a wide range of frequencies that interact with each other. Because the different frequencies are not learned at the same rate, the solution converges slowly or does not converge at all. Interestingly, McClenny et al. (2022) have shown that spectral bias is partly addressed by adaptive loss balancing methods. However, a more direct solution is Fourier feature networks introduced by Tancik et al. (2020), which transform the network inputs to a higher-dimensional feature space by applying a Fourier transformation to them as shown in Eq. (2.13), where $\mathbf{f}$ is a set of frequencies that is predefined or trainable. By applying this transformation to the spatial and temporal inputs of PINNs, both Wang et al. (2021b) and Hennigh et al. (2021) have shown to obtain errors that are orders of magnitude smaller with a faster convergence.

$$\tilde{\mathbf{x}} = [\sin(2\pi \mathbf{f} \times \mathbf{x}); \cos(2\pi \mathbf{f} \times \mathbf{x})]^\mathsf{T} \tag{2.13}$$

Note that spectral bias is primarily an issue when the target solution consists of a wide range of frequencies, which can be determined by taking its Fourier transform. Again, it is natural to take turbulence as a prime example of a flow consisting of a wide range of frequencies, as it involves an energy cascade consisting of eddies of many scales (Nieuwstadt et al., 2018). Nevertheless, highly compressible flows often involve discontinuities that are essentially step functions, which in the Fourier domain are also represented by a wide range of frequencies.

While Fourier feature networks transform the inputs of a neural network, they do not modify the rest of their architecture. A different approach is taken by Wang et al. (2021a), who use the transformer architecture of Vaswani et al. (2017) from the natural language processing field. Essentially, this architecture converts the spatial and temporal inputs to highly-dimensional feature spaces that are connected to future hidden

layers through residual connections, as shown in Fig. 2.14. Similar to Fourier feature networks, they have been shown to obtain an improvement of two orders of magnitude on the L_2 error compared to conventional layered networks. Furthermore, Hennigh et al. (2021) combine Fourier feature networks with the transformer architecture, using the Fourier transformed inputs as the transformer feature space. Compared to plain Fourier feature networks, the performance of these modified Fourier feature networks is only marginally better.

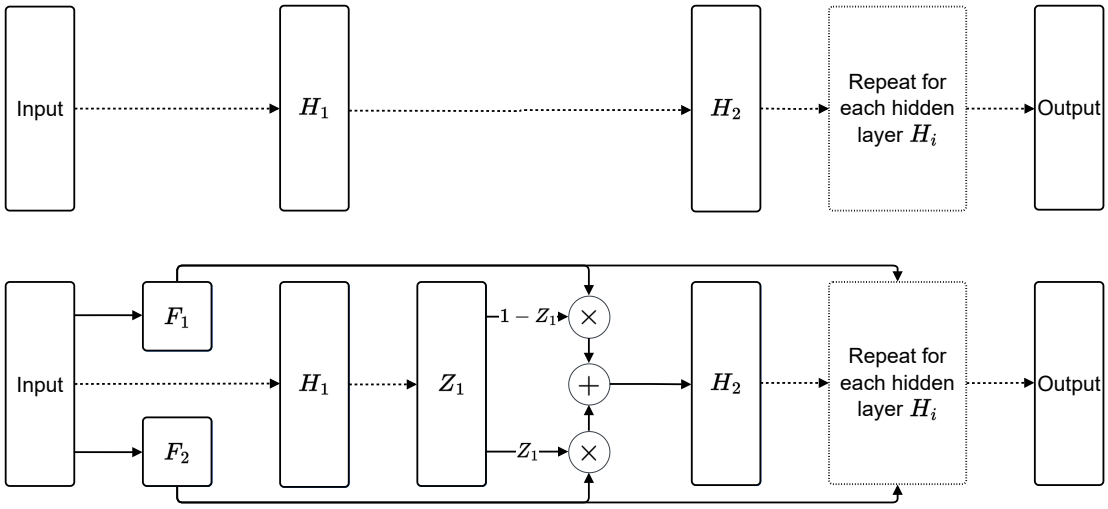


Figure 2.14: Comparison of a standard neural network architecture (top) to a transformer architecture (bottom), dashed arrows indicate matrix multiplications while solid arrows indicate value transfer. Here F_1 and F_2 are layers representing feature spaces and Z_1 is an intermediate layer.

Apart from input transformations and alternative architectures, one can also consider different activation functions for the neurons. Although there is a wide range of activation functions that can be chosen (Lederer, 2021), a class of activation functions has been designed by Jagtap et al. (2020b) for PINNs specifically. These Locally Adaptive Activation Functions (LAAFs) take any activation function $f(x)$ and transform it as shown Eq. (2.14), where a is a continuous parameter that is learned through gradient descent and $n \geq 1$ is an integer hyperparameter. These two parameters can either be defined for the whole network (Jagtap et al., 2020a), per layer or for each neuron. The main motivation of the extra parameters is that they can increase the gradient of the activation functions, potentially mitigating the vanishing gradient problem (Pascanu et al., 2013) leading to faster convergence. Generally, the larger n is chosen, the faster the convergence, up and until a critical value. Furthermore, a slope recovery term can be added to the loss function that rewards the network for larger values of a.

$$f(x) \Rightarrow f(nax) \tag{2.14}$$

Although the additional parameters increase the computational cost slightly, Jagtap et al. (2020b) and Markidis (2021) have shown that all variants of LAAFs lead to drastically faster convergence compared to using non-adaptive activation functions. In general, using LAAFs with parameters for each neuron leads to faster convergence than using LAAFs with layer-shared parameters, which leads again to faster convergence than using LAAFs with network-shared parameters. However, it is important to reiterate that using LAAFs can also lead to a performance decrease if the value for n is chosen too large.

2.3 Adversarial training

Section 2.2.1 shortly discussed some weighting methods that are based on maximizing and minimizing the loss at the same time, which are also known as adversarial training methods. Recently, a more compelling adversarial training method was introduced by Zeng et al. (2022), which redefines the foundation of PINNs while maintaining their fundamental principles and desirable properties. These so-called Competitive Physics-Informed Neural Networks (CPINNs) were actually the starting point of this thesis as they show interesting behavior on highly compressible problems, which will be further analyzed and discussed in

Chapter 4. In short, CPINNs are similar to the self-adaptive PINNs of McClenny et al. (2022), but they use a second neural network instead of a Gaussian process to weigh the collocation points. Furthermore, the loss function no longer involves squared but rather pure differences and residuals, leading to Eq. (2.15).

$$\begin{gathered}\max_{\mathbf{d}} \min_{\mathbf{g}} L = L_0 + L_b + L_r, \\ L_0 = D(\mathbf{x}, 0)(\mathbf{u}(\mathbf{x}, 0) - \mathbf{u}_0(\mathbf{x})), \quad L_b = D(\mathbf{x}, t)(\mathbf{u}(\mathbf{x}, t) - \mathbf{u}_b(\mathbf{x}, t)), \quad L_r = D(\mathbf{x}, t)P(\mathbf{u}(\mathbf{x}, t)).\end{gathered} \tag{2.15}$$

Thus, instead of only involving a single neural network, this loss function involves a *discriminator* network $D(\mathbf{x}, t)$ as shown in Fig. 2.15 and a *generator* network $G(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t)$ which is architecturally equivalent to the PINN in Fig. 2.2. These two neural networks play a competitive game, where the discriminator is tasked with predicting the mistakes of the generator. The loss is now defined as a minimax problem, where the discriminator optimizes its parameters **d** to maximize the loss and the generator optimizes its parameters **g** to minimize the loss. Essentially, the discriminator is placing bets on whether the generator will over- or undershoot the initial conditions, boundary conditions and PDEs. A correct bet results in a reward for the discriminator and a penalty for the generator, whereas a wrong bet has the reverse impact. Most importantly, the game has a Nash equilibrium at $G(\mathbf{x}, t) = \mathbf{u}_{\text{true}}(\mathbf{x}, t)$ and $D(\mathbf{x}, t) = 0$, where the generator produces the exact solution to the problem given enough collocation points are used.

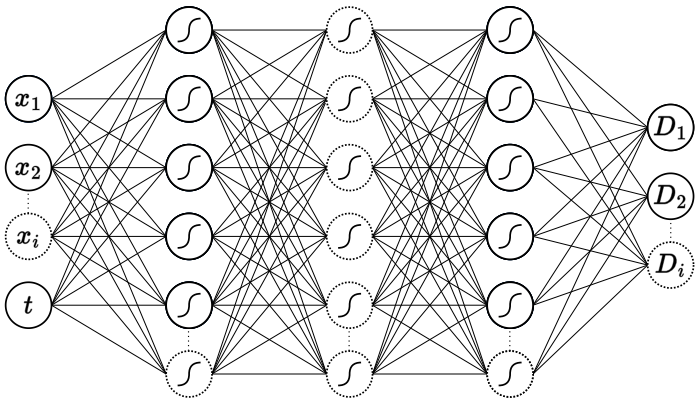
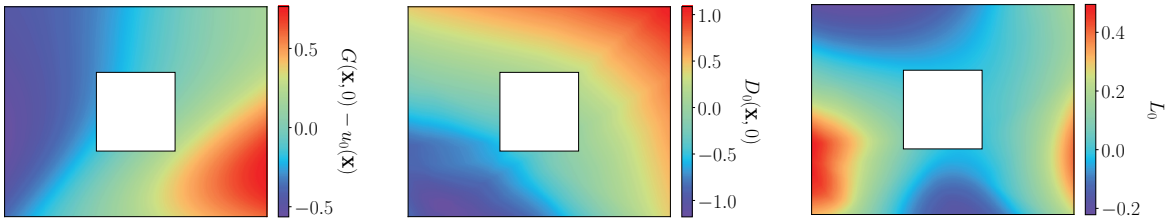


Figure 2.15: Architecture of a discriminator network with inputs $(\mathbf{x}, t)$. It has an output for every initial condition, boundary condition and PDE in order to evaluate the loss term in Eq. (2.15).

At the time of writing, the only examples involving CPINNs are provided by Zeng et al. (2022) who show that CPINNs can reach an error that is several orders of magnitude lower compared to PINNs. Nevertheless, the results are somewhat limited because they are achieved on univariate problems that only involve a single PDE. Furthermore, the problems are either unsteady and one-dimensional or steady and two-dimensional. In other words, they involve only two independent variables. Therefore, one must be careful to extrapolate these results to typical fluid dynamics problems, which involve multiple PDEs and often many more variables. In addition, the authors do not sufficiently elaborate on how the output of the discriminator should be structured when more complex boundary conditions are prescribed. For example, it is straightforward to use a single output for the initial condition of a continuous domain as shown in Fig. 2.16. However, it remains unclear if the discriminator should have a separate output for each boundary segment in case directional boundary conditions such as Neumann conditions are prescribed on a non-differentiable boundary, for example on the surface of the square. A single output introduces discontinuities in the loss function that could worsen the loss landscape, while separate outputs increase the network complexity.



(a) Initial condition difference **(b)** Discriminator prediction **(c)** Initial condition loss field

Figure 2.16: Example of the calculation of the adversarial loss for some initial condition $u_0(\mathbf{x})$.

While CPINNs seem to have the ability to drastically improve over the performance of conventional PINNs, it is important to note that it is not clear yet why exactly they do so. At the start of this section, it was mentioned that CPINNs are similar to the self-adaptive PINNs introduced by McClenny et al. (2022), except for two differences. Since these differences are relatively subtle, it could be argued that CPINNs are simply a weighting method. However, Zeng et al. (2022) attribute the success of CPINNs to the replacement of the least-squares problem by a competitive game without any squared error terms. They argue that when the squared loss function in Eq. (2.3) is used for a PDE of order n, minimizing it involves solving a system of order $2n$. As a result, the condition number of the problem is squared, which is indicative of the sensitivity of the loss with respect to the parameters. In essence, the authors claim that the competitive loss leads to a significant simplification of the complex loss landscape as discussed in Section 2.2.2, effectively reducing the probability that gradient descent algorithms prematurely converge to local minima.

2.3.1 Failure modes

Although CPINNs seem to address one or more failure modes of PINNs, one might argue that CPINNs themselves might suffer from other undiscovered failure modes. There is certainly an element of truth in this rhetoric, but it is good to realize that the competitive loss function is based on the loss function of Generative Adversarial Networks (GANs), which were introduced by Goodfellow et al. (2014) almost a decade ago and have since matured. This class of neural networks is used to generate samples from a probability distribution that must be inferred from a dataset, for example to generate new faces based on a dataset of real faces. Similarly, training them involves a generator network and discriminator network, where the generator is responsible for generating new samples, as shown in Fig. 2.17. The discriminator is then randomly given a real sample from the dataset or a fake sample from the generator, and it must give the probability that the sample is real or fake.

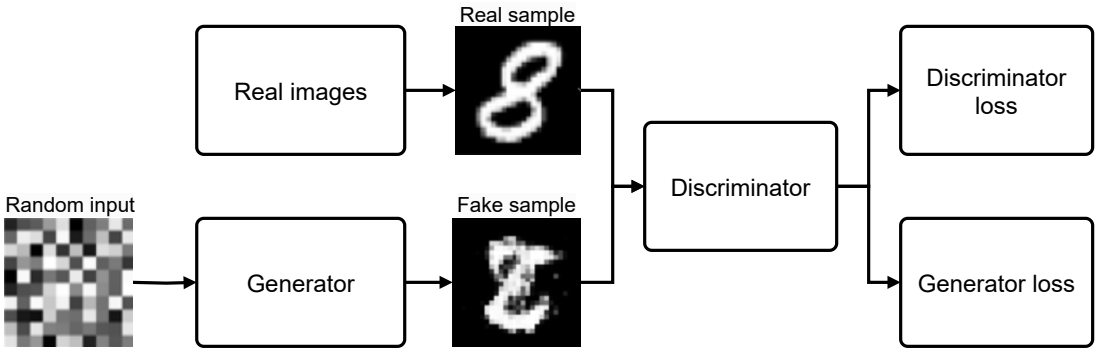
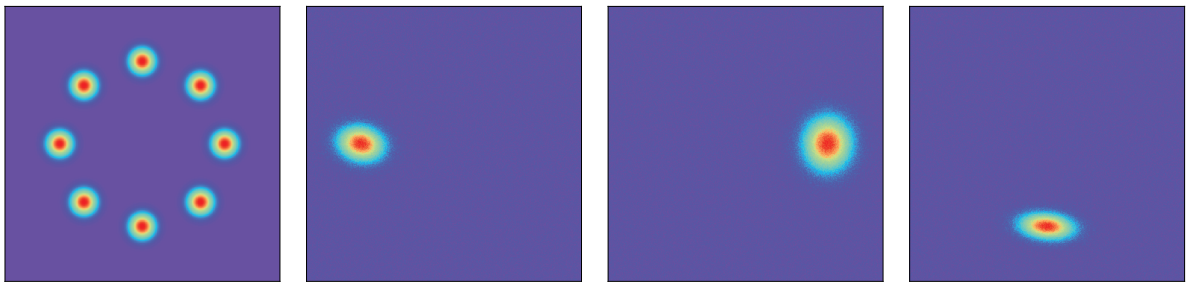


Figure 2.17: Architecture of a GAN setup involving a generator and discriminator, example data by Deng (2012).

Theoretically, this game also results in a Nash equilibrium where the generator perfectly mimics the probability distribution of the dataset and the discriminator outputs $\frac{1}{2}$ on both real and fake samples. This adversarial setup works remarkably well, and it resulted in the first photorealistic generated faces (Karras et al., 2017). One theory that attempts to explain this success is called implicit regularization (Schäfer et al., 2020b), which postulates that the generator is forced to generalize better because overfitting on simple patterns in the dataset also makes it easy for the discriminator to understand how the generator

operates. Nevertheless, GANs also suffer from their own failure modes (Wiatrak et al., 2019). Since they might also manifest in CPINNs, it is instrumental to understand them.



(a) Target distribution **(b)** Iteration i **(c)** Iteration $i + 5000$ **(d)** Iteration $i + 10000$

Figure 2.18: Example of mode collapse on a 2D target probability distribution, adapted from Metz et al. (2016). During training, the generator keeps moving to a new subspace and fails to sample the full probability distribution.

In particular, GANs suffer from three main failure modes, namely mode collapse, instability and vanishing gradient (Saxena et al., 2020). Mode collapse occurs when the generator only generates samples with limited diversity, which might be beneficial because the generator has control over which fake images the discriminator sees. In other words, the generator can focus its limited approximation capacity on a sample subspace, without having to suffer the consequences of not sampling the full probability distribution. As a result, the generator and discriminator play a cat-and-mouse game where the generator keeps changing to a different subspace and the discriminator trails behind, as illustrated in Fig. 2.18. Fortunately, CPINNs do not suffer from this since the generator does not decide on which regions of the domain the loss is evaluated, as this is determined by the collocation points that should uniformly cover the domain.

Unlike mode collapse, instability and vanishing gradient can occur within CPINNs. The former is related to the fact that the Nash equilibrium is essentially a saddle point on the loss landscape, so applying simultaneous gradient descent as given in Eq. (2.16) might not pave the way to this optimum. Instead, oscillations can occur and the loss might even diverge from the Nash equilibrium completely, as shown in Fig. 2.19. In addition, when the discriminator overpowers the generator with excellent predictions, it results in a vanishing gradient for the generator causing the learning process to stagnate. To avoid instability and vanishing gradient, one must carefully select the hyperparameters of the networks. In particular, the learning rates of the networks have the most effect on the learning dynamics. In practice, this is a tedious procedure, which is one of the reasons why GANs have been superseded by a new class of generative models, namely diffusion models introduced by Sohl-Dickstein et al. (2015).

$$\begin{aligned} \mathbf{g}_{i+1} &= \mathbf{g}_i - \eta_g \nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i) \\ \mathbf{d}_{i+1} &= \mathbf{d}_i + \eta_d \nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i) \end{aligned} \tag{2.16}$$

2.3.2 Adversarial gradient descent

Unfortunately, the subtle differences between CPINNs and GANs do not allow diffusion-like training to be applied to CPINNs. Therefore, Zeng et al. (2022) have had to look elsewhere to mitigate the instability issues that occur as a result of adversarial loss functions. One prevalent method is to use gradient descent algorithms that model the interaction between the two networks (Liang et al., 2018), unlike the individualistic approach described in Eq. (2.16). One algorithm is proposed by Yadav et al. (2017), where the generator updates its parameters normally, but the discriminator uses a prediction of the generator parameters to update its parameters based on the resulting predicted loss as given in Eq. (2.17). This approach bears similarities with implicit derivative schemes (Butcher, 2003), which are often used in traditional solvers because they are more stable than their explicit counterparts.

$$\begin{aligned} \mathbf{g}_{i+1} &= \mathbf{g}_i - \eta_g \nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i) \\ \mathbf{g}^* &= \mathbf{g}_{i+1} + (\mathbf{g}_{i+1} - \mathbf{g}_i) \\ \mathbf{d}_{i+1} &= \mathbf{d}_i + \eta_d \nabla_{\mathbf{d}} L(\mathbf{g}^*, \mathbf{d}_i) \end{aligned} \tag{2.17}$$

An extension of this algorithm is given in Eq. (2.18) and was proposed by Metz et al. (2016), who let the discriminator make multiple projected parameter updates after which the generator makes a parameter update based on the resulting projected loss. As a result, the discriminator is not likely to overpower the generator which avoids the vanishing gradient failure mode. They also postulate that this method could be applied to both networks, resulting in a recursive gradient descent algorithm where the generator and discriminator make moves that are optimal over a finite horizon. While there are other effective gradient descent algorithms, such as consensus optimization by Mescheder et al. (2017), these implicit algorithms are most prevalent and are essentially variations of the classic extragradient method by Korpelevich (1977).

$$\begin{aligned} \mathbf{d}_{i+1} &= \mathbf{d}_i + \eta_d \nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i) \\ \mathbf{d}^* &= \mathbf{d}_i \\ \mathbf{d}^* &= \mathbf{d}^* + \eta_d \nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}^*), \text{ repeat } k \text{ times} \\ \mathbf{g}_{i+1} &= \mathbf{g}_i - \eta_g \nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}^*) \end{aligned} \tag{2.18}$$

While extragradient-based methods are successful, they are essentially using implicit schemes to turn first-order gradient descent schemes into higher-order schemes. Furthermore, it is tedious to figure out how many steps to project gradients and the optimal settings depend on the problem at hand. As a result, Schäfer et al. (2020a) have introduced Competitive Gradient Descent (CGD) which inherently incorporates second-order effects by using the Hessian that contains the interaction effect of the two networks on the loss. The performance of different gradient descent algorithms on CPINNs was investigated by Zeng et al. (2022), showing that CGD performs orders of magnitude better compared to the other methods. It appears that CGD is a vital ingredient to the effectivity of CPINNs, hence it is instrumental to understand its workings and limitations.

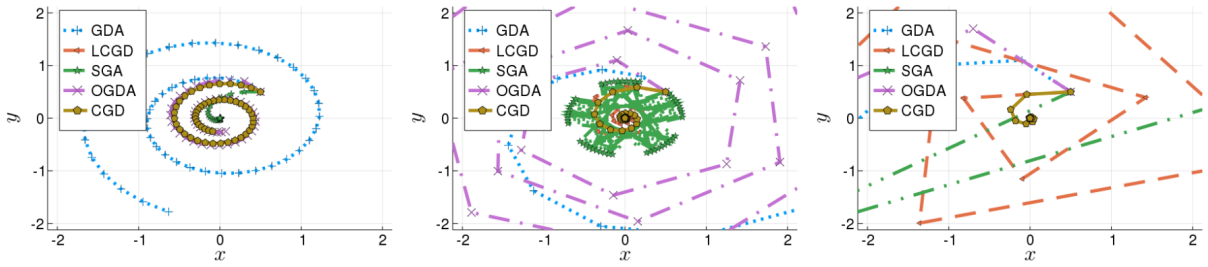


Figure 2.19: Comparison of CGD to other gradient descent algorithms on a simple problem with a Nash equilibrium at $(0, 0)$ (Schäfer et al., 2020a). The learning rate increases from left to right. Notice that all gradient descent algorithms diverge for a high learning rate, while CGD converges.

As explained earlier, in simultaneous gradient descent as given by Eq. (2.16) the parameter updates do not take any interaction into account. This is quite naive because both networks in fact "know" what parameter update the other network is going to take. Without using this knowledge, the networks fail to take into account each other's future parameter updates, which can result in oscillations and divergence as shown in Fig. 2.19. The emergence of this behavior can be better explained by recasting the parameter updates into the form given below.

$$\begin{aligned} \mathbf{g}_{i+1} &= \arg\min_{\mathbf{g}} \; +\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{g} - \mathbf{g}_i) + \frac{1}{2\eta} ||\mathbf{g} - \mathbf{g}_i||^2 \\ \mathbf{d}_{i+1} &= \arg\min_{\mathbf{d}} \; -\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{d} - \mathbf{d}_i) + \frac{1}{2\eta} ||\mathbf{d} - \mathbf{d}_i||^2 \end{aligned} \tag{2.19}$$

This form shows how the learning rate is essentially a parameter that indicates the confidence in the local gradient. Since the local gradient is only a first-order approximation of the loss function, a learning rate that is too large results in overconfidence and can cause stability issues. While decreasing the learning rate might avoid divergence, it does not necessarily guarantee convergence to the Nash equilibrium because the parameter updates might circle around it since no interaction is taken into account. To take into account

interaction, Schäfer et al. (2020a) included mixed second derivatives as shown in Eq. (2.20).

$$\begin{aligned}\mathbf{g}_{i+1} &= \arg\min_{\mathbf{g}} \; +\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{g} - \mathbf{g}_i) + (\mathbf{g} - \mathbf{g}_i)^\intercal D^2_{\mathbf{gd}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{d} - \mathbf{d}_i) + \frac{1}{2\eta}||\mathbf{g} - \mathbf{g}_i||^2 \\ \mathbf{d}_{i+1} &= \arg\min_{\mathbf{d}} \; -\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{d} - \mathbf{d}_i) - (\mathbf{d} - \mathbf{d}_i)^\intercal D^2_{\mathbf{dg}} L(\mathbf{g}_i, \mathbf{d}_i)(\mathbf{g} - \mathbf{g}_i) + \frac{1}{2\eta}||\mathbf{d} - \mathbf{d}_i||^2\end{aligned} \tag{2.20}$$

Although the updates in Eq. (2.20) contain the unknown future parameters of the other network, it is possible to solve the system for the local Nash equilibrium which results in CGD as shown in Eq. (2.21). Notice how it contains the matrix inverse of a matrix with dimensions equal to the number of parameters of the model, which together with the calculations of the Hessians greatly increases the computational cost compared to simultaneous gradient descent. Furthermore, the matrix inverse is also the reason that no pure second derivatives such as $D^2_{\mathbf{gg}}$ and $D^2_{\mathbf{dd}}$ are used in Eq. (2.20), as Schäfer et al. (2020a) argue they can ill-condition the matrix or even make it singular.

$$\begin{aligned}\Delta\mathbf{g}_{i+1} &= -\eta(I + \eta^2 D^2_{\mathbf{gd}} L(\mathbf{g}_i, \mathbf{d}_i) D^2_{\mathbf{dg}} L(\mathbf{g}_i, \mathbf{d}_i))^{-1}(\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i) + \eta D^2_{\mathbf{gd}} L(\mathbf{g}_i, \mathbf{d}_i)\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i)) \\ \Delta\mathbf{d}_{i+1} &= +\eta(I + \eta^2 D^2_{\mathbf{dg}} L(\mathbf{g}_i, \mathbf{d}_i) D^2_{\mathbf{gd}} L(\mathbf{g}_i, \mathbf{d}_i))^{-1}(\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i) - \eta D^2_{\mathbf{dg}} L(\mathbf{g}_i, \mathbf{d}_i)\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i))\end{aligned} \tag{2.21}$$

Although CGD has proven to be effective on problems where simultaneous gradient descent results in divergence, it requires some adaptations when applied to neural networks in an adversarial training setting for better performance. Recall that momentum is combined with simultaneous gradient descent to improve its convergence properties, as is done with the Adam optimizer introduced in Algorithm 1. Similarly, Schäfer et al. (2020b) combined momentum with CGD to form Adaptive Competitive Gradient Descent (ACGD) which is shown in Algorithm 2. Unlike Adam, it only uses the momentum of the squared gradient, similar to the non-adversarial gradient descent algorithm RMSprop, introduced by Tieleman et al. (2012).

Algorithm 2: Adaptive Competitive Gradient Descent (Schäfer et al., 2020b)

require base learning rate η
require exponential decay rate β
require loss function $L(\mathbf{g}, \mathbf{d})$
require initial parameters $\mathbf{g}_0, \mathbf{d}_0$

initialize second moment estimates $\mathbf{m}_g = 0$, $\mathbf{m}_d = 0$

for i *in* $\{1, 2, ...\}$ **do**
  $\mathbf{m}_g = \beta\mathbf{m}_g + (1-\beta)(\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i))^2$
  $\mathbf{m}_d = \beta\mathbf{m}_d + (1-\beta)(\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i))^2$
  $\mathbf{m}_g = \mathbf{m}_g/(1-\beta^i)$
  $\mathbf{m}_d = \mathbf{m}_d/(1-\beta^i)$
  $\boldsymbol{\eta}_g = \eta/(\sqrt{\mathbf{m}_g} + \epsilon)$
  $\boldsymbol{\eta}_d = \eta/(\sqrt{\mathbf{m}_d} + \epsilon)$

  $\Delta\mathbf{g}_i = -\boldsymbol{\eta}_{g,i}^{1/2}(I + \boldsymbol{\eta}_{g,i}^{1/2} D^2_{\mathbf{gd}} \boldsymbol{\eta}_{d,i} D^2_{\mathbf{dg}} \boldsymbol{\eta}_{g,i}^{1/2})^{-1} \boldsymbol{\eta}_{g,i}^{1/2}(\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i) + D^2_{\mathbf{gd}} L(\mathbf{g}_i, \mathbf{d}_i)\boldsymbol{\eta}_{d,i}\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i))$
  $\Delta\mathbf{d}_i = +\boldsymbol{\eta}_{d,i}^{1/2}(I + \boldsymbol{\eta}_{d,i}^{1/2} D^2_{\mathbf{dg}} \boldsymbol{\eta}_{g,i} D^2_{\mathbf{gd}} \boldsymbol{\eta}_{d,i}^{1/2})^{-1} \boldsymbol{\eta}_{d,i}^{1/2}(\nabla_{\mathbf{d}} L(\mathbf{g}_i, \mathbf{d}_i) - D^2_{\mathbf{dg}} L(\mathbf{g}_i, \mathbf{d}_i)\boldsymbol{\eta}_{g,i}\nabla_{\mathbf{g}} L(\mathbf{g}_i, \mathbf{d}_i))$
  $\mathbf{g}_i = \mathbf{g}_{i-1} + \Delta\mathbf{g}_i$
  $\mathbf{d}_i = \mathbf{d}_{i-1} + \Delta\mathbf{d}_i$

Essentially, computing the parameter updates boils down to solving the two linear systems of the form $A\mathbf{x} = \mathbf{b}$ as highlighted in Eq. (2.22). Fortunately, it is sufficient to solve only one of the systems and use the resulting updated parameters for one network to calculate the parameters for the other network using simple gradient descent. Nevertheless, solving such a system is computationally expensive as direct algorithms typically scale as $O(n^3)$, where n is the dimension of the system (Ferziger et al., 2012). However, this computational cost can be reduced by using approximate iterative solvers that solve the systems up to a specified tolerance, similar to what is done in traditional CFD solvers. Nevertheless, the derivation of the full Hessian required to calculate the matrix to be inverted in the first place is a

computationally intensive but also memory-intensive procedure.

$$\begin{aligned}(I+\eta^2 D^2_{\mathbf{gd}} L(\mathbf{g}_i,\mathbf{d}_i) D^2_{\mathbf{dg}} L(\mathbf{g}_i,\mathbf{d}_i))\Delta\mathbf{g}_{i+1} &= -\eta(\nabla_{\mathbf{g}} L(\mathbf{g}_i,\mathbf{d}_i) + \eta D^2_{\mathbf{gd}} L(\mathbf{g}_i,\mathbf{d}_i)\nabla_{\mathbf{d}} L(\mathbf{g}_i,\mathbf{d}_i)) \\ \underbrace{(I+\eta^2 D^2_{\mathbf{dg}} L(\mathbf{g}_i,\mathbf{d}_i) D^2_{\mathbf{gd}} L(\mathbf{g}_i,\mathbf{d}_i))}_{A}\underbrace{\Delta\mathbf{d}_{i+1}}_{\mathbf{x}} &= \underbrace{+\eta(\nabla_{\mathbf{d}} L(\mathbf{g}_i,\mathbf{d}_i) - \eta D^2_{\mathbf{dg}} L(\mathbf{g}_i,\mathbf{d}_i)\nabla_{\mathbf{g}} L(\mathbf{g}_i,\mathbf{d}_i))}_{\mathbf{b}}\end{aligned} \tag{2.22}$$

Fortunately, most iterative matrix inverse methods such as the conjugate gradient method (Shewchuk, 1994) and the generalized minimal residual method (Saad et al., 1986) only involve vector-matrix products with the Hessian matrix. This allows for a computational trick in most neural network libraries, where the Hessian vector product can be calculated incrementally without having to store the entire matrix in memory (Martens, 2010). Nevertheless, CPINNs remain considerably more computationally expensive than PINNs since they involve two neural networks. In addition, the computational cost increases as the loss decreases because the linear systems in Eq. (2.22) tend to require more iterations to be solved up to the same threshold.

3

Compressible fluid dynamics

Now that an overview has been provided on the fundamental principles, limitations and possible adaptations to PINNs, the next step is to understand what highly compressible flows are and how PINNs can be used to simulate them. In short, the field of compressible fluid dynamics concerns flows that involve large changes in density, which most commonly occur around objects that travel near or beyond the speed of sound. The theoretical foundations of such flows are explained in Section 3.1, followed by a brief explanation of how PINNs can be used to simulate them in Section 3.2. Furthermore, the various phenomena that occur in these flows are explained in Section 3.3, which is relevant to understand the behavior of PINNs on highly compressible problems in the upcoming chapter. Note that in essence, a selection of relevant theories from Anderson (2006) and Anderson (2021) are considered in this chapter. For a more elaborate explanation, the reader is referred to these books.

3.1 Euler equations

In practice, it is possible to simulate highly compressible flows using either the Navier-Stokes equations or the Euler equations. The difference between the two is that the Euler equations assume that the fluid is not viscous, and as a result complex phenomena such as boundary layers and turbulence are not modeled. Nevertheless, the Euler equations are useful because they are easier to simulate and therefore allow for faster iterations in the design of objects subjected to highly compressible flow. To the authors' best knowledge, there are currently no papers that apply PINNs to the compressible Navier-Stokes equations to solve supersonic flows. This is no surprise given the lack of complex supersonic flows simulated with the Euler equations in literature. Therefore, this thesis will focus on the Euler equations, although most of the findings are also relevant to simulations with the Navier-Stokes equations.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho \,\mathrm{d}V + \int_{\partial V} \rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S &= 0,\\ \frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho\mathbf{v}\,\mathrm{d}V + \int_{\partial V} (p+\rho|\mathbf{v}|^2)\cdot\mathbf{n}\,\mathrm{d}S &= 0,\\ \frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho E\,\mathrm{d}V + \int_{\partial V} (p+\rho E)\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S &= 0. \end{aligned} \tag{3.1}$$

The integral form of the Euler equations is given in Eq. (3.1), which can be derived by taking some volume element V and applying conservation of mass, momentum and energy over it respectively. Note that it is assumed that there are no viscous and body forces, as well as that no heat is added to the fluid. By assuming that the density ρ, pressure p, velocity $\mathbf{v}$ and total energy E vary smoothly (LeVeque, 1992), the divergence theorem and gradient theorem can be applied to derive the differential form of the Euler equations, namely

$$\begin{aligned} \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) &= 0,\\ \frac{\partial(\rho\mathbf{v})}{\partial t} + \nabla\cdot(p+\rho\mathbf{v}\mathbf{v}^{\mathsf{T}}) &= 0,\\ \frac{\partial(\rho E)}{\partial t} + \nabla\cdot((p+\rho E)\mathbf{v}) &= 0. \end{aligned} \tag{3.2}$$

Notice that the above PDEs have more variables ($\rho, p, E, \mathbf{v}$) than equations, making the system underdetermined. To admit a unique solution, an equation of state is required. It is common to use the ideal gas law for this, of which one form is $p = \rho RT$ with R the ideal gas constant and T the temperature. Using this ideal gas law together with the other relations in Eq. (3.3), Eq. (3.4) can be derived. Note that e is the specific internal energy, E is the total energy, c_p and c_v are the specific heat capacities at constant pressure and volume respectively, and γ is the heat capacity ratio.

$$p = \rho RT, \quad e = c_v T, \quad E = e + 1/2||\mathbf{v}||^2, \quad \gamma = c_p/c_v, \quad R = c_p - c_v \tag{3.3}$$

$$\rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho||\mathbf{v}||^2 \tag{3.4}$$

3.2 Physics-informed solutions

By using Eq. (3.4) and defining the set of variables as $\mathbf{u} = [\rho, p, \mathbf{v}]$, Eq. (3.2) can be used to provide the residuals for the residual loss as given in Eq. (2.3), which is visualized in Fig. 3.1 for a two-dimensional domain. Typically, the activation functions of the neurons corresponding to the density and pressure output are chosen to be positive to avoid nonphysical negative values. Note that Eq. (3.2) is written in the conservative form, as the different terms are grouped based on their derivatives. The conservative form is often also used in traditional solvers because it improves the capability of numerical discretization schemes to capture shocks. However, this is not relevant for PINNs because they can be differentiated without discretization errors through automatic differentiation. Nevertheless, it is possible to use the conservative variables $\mathbf{u} = [\rho, \rho E, \rho\mathbf{v}]$ as the output of the PINN as is done by Jagtap et al. (2022), but no comparison is done to using the primitive variables. Note that using the conservative variables involves division by the density ρ to obtain the primitive variables, which could lead to numerical issues.

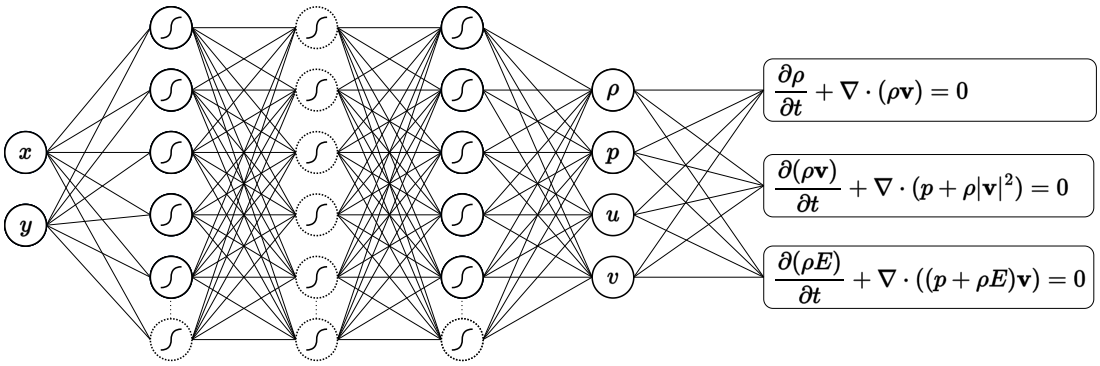


Figure 3.1: Example of a PINN applied to the two-dimensional Euler equations.

There are many more forms of the Euler equations and their associated variables $\mathbf{u}$, such as the characteristic form treated by Jagtap et al. (2022) or the vorticity form. Each of these forms can affect the loss landscape and consequently simplify or complexify the problem at hand. One interesting form is often used for *incompressible* flows, where it is possible to define the velocities as a derivative of some potential Φ to inherently satisfy and thus eliminate the continuity equation. For a two-dimensional flow, Eq. (3.5) can be used and the variables become $\mathbf{u} = [p, \Phi]$. The concept can also be extended to eliminate the continuity equation for three-dimensional flows, but it requires more derivatives. Raissi et al. (2019) has successfully combined this potential approach with the Navier-Stokes equations, but it can also be combined with the Euler equations.

$$u = \frac{\partial \Phi}{\partial y}, \quad v = \frac{\partial \Phi}{\partial x} \tag{3.5}$$

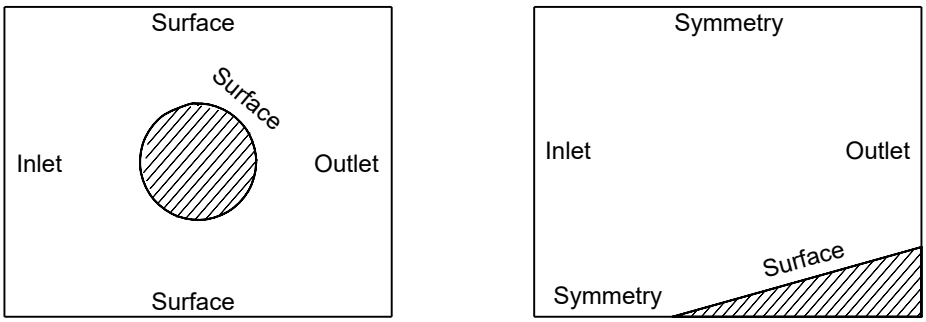
A similar approach can also be applied to compressible flows by simply modifying the equations to Eq. (3.6) and using the variables $\mathbf{u} = [\rho, p, \Phi]$. The primitive velocities can then be obtained by dividing the conservative velocities by the density, which could again cause problems if the density is near-zero. To the authors' best knowledge, there are no counts of using this compressible potential form in literature. While eliminating an equation may seem desirable, note that it requires extra derivatives, increasing the computational and memory demands. Furthermore, hard-constraining the continuity equation could potentially

complexify the loss landscape, preventing convergence. Since the effect of the different formulations of the Euler equations and corresponding variables is not well-researched, the remainder of the thesis will use the conservative form for the residual losses and the primitive variables for the network outputs.

$$\rho u = \frac{\partial \Phi}{\partial y}, \quad \rho v = \frac{\partial \Phi}{\partial x} \tag{3.6}$$

3.2.1 Boundary conditions

As mentioned in Chapter 2, the Euler equations must always be accompanied by a domain with appropriate boundary conditions to ensure that a unique solution exists. Typical domains for two-dimensional highly compressible problems are shown in Fig. 3.2, one without a symmetry plane and one with a symmetry plane. The different boundaries can impose different kinds of constraints on the variables **u**, depending on the physical behavior that the boundary should emulate. Typically, each boundary applies a constraint to all of the variables to make the problem well-posed.



(a) Without symmetry plane **(b)** With symmetry plane

Figure 3.2: Examples of boundary conditions on two-dimensional domains for highly compressible problems.

An overview of the constraints for different physical boundaries is given in Table 3.1. At the inlet, the values of the variables **u** are prescribed through Dirichlet boundary conditions, essentially specifying the free-stream flow. At physical surfaces, slip boundary conditions prescribe that the surface is impenetrable through both Dirichlet and Neumann boundary conditions. For the top and bottom boundaries, it is also possible to use a symmetry boundary condition, which acts like a ”mirror”. However, in the case of inviscid flow, it is identical to the slip boundary condition. When the flow at the outlet is subsonic, Neumann boundary conditions are applied to all variables except for the pressure. In case the flow is supersonic, extrapolative Neumann conditions are applied to all variables, since no information enters the domain from the outlet (Liepmann et al., 2001). While a more general form of these boundary conditions would be characteristic boundary conditions (Pulliam, 1981), they are beyond the scope of this thesis.

Table 3.1: Overview of relevant boundary conditions. Subscripts n and t specify the normal and tangential components respectively.

Variable	Inlet	Surface	Symmetry	Outlet (subsonic)	Outlet (supersonic)
ρ	$\rho = \rho_\infty$	$\partial_n \rho = 0$	$\partial_n \rho = 0$	$\partial_n \rho = 0$	$\partial_n \rho = 0$
p	$p = p_\infty$	$\partial_n p = 0$	$\partial_n p = 0$	$\partial_n p = 0$	$p = p_\infty$
E	$E = E_\infty$	$\partial_n E = 0$	$\partial_n E = 0$	$\partial_n E = 0$	$\partial_n E = 0$
$\mathbf{v}$	$\mathbf{v} = \mathbf{v}_\infty$	$v_n = 0, \partial_n v_t = 0$	$v_n = 0, \partial_n v_t = 0$	$\partial_n v_n = 0$	$\partial_n v_n = 0$

Note that the formulation of these boundary conditions is based on their implementation in traditional solvers, which require constraints to make sure the solved linear system is not underdetermined. Interestingly, while there is plenty of research on findings ways to analytically impose boundary conditions on PINNs (Sukumar et al., 2022; Berrone et al., 2022), there is little research on their importance or even necessity since PINNs do not solve a linear system. While the absence of certain boundary conditions allows PINNs to choose any boundary condition they desire, their strong tendency to generalization avoids

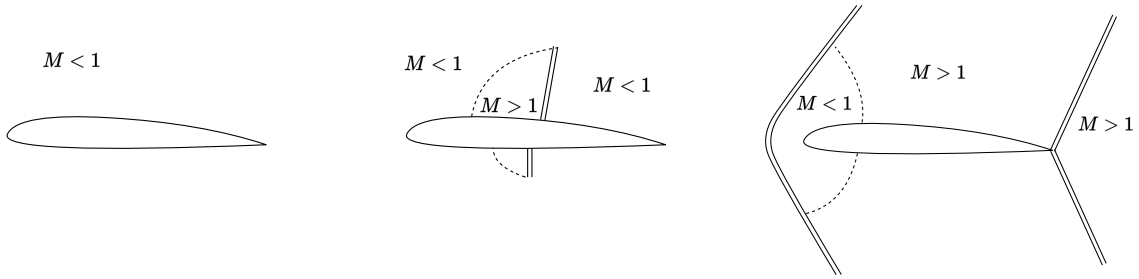
producing overly complicated solutions. For example, the outlet boundary conditions for supersonic flow merely serve as extrapolation and PINNs will likely converge to the correct solution without them. Indeed, Jin et al. (2021) show that the removal of certain boundary conditions hardly affects the errors obtained on a simple problem described by the Navier-Stokes equations.

3.3 Compressible phenomena

Before applying PINNs to problems described by the Euler equations, it is important to discuss the various phenomena that occur in highly compressible flows. The first step is to define when a flow is highly compressible, in other words, when variations in density start to play a significant role. Typically this threshold is established at the empirical value of $M \approx 0.3$ (Anderson, 2021), where M is the Mach number defined as $M = ||\mathbf{v}||/c$ with c the speed of sound. When an ideal gas is assumed, the speed of sound is given by Eq. (3.7). Under this threshold, the density is almost homogeneous and there is little difference between the solutions obtained from the incompressible and compressible Euler equations.

$$c = \sqrt{\gamma p/\rho} \tag{3.7}$$

Above the threshold, the flow will compress and subsequently expand around an object causing the density to vary more significantly. The variation in density is reversible, implying that the processes are isentropic so that the isentropic relations in Eq. (3.8) can be used to determine the pressure, temperature and density based on the local Mach number. Here p_t, ρ_t and T_t are the total pressure, density and temperature respectively, which are obtained when the flow is stagnated isentropically. Note that isentropic expansion and compression are smooth phenomena, so they do not involve discontinuities.



(a) Subsonic, $M_\infty < M_c$ **(b)** Transonic, $M_\infty > M_c$ **(c)** Supersonic, $M_\infty > 1$

Figure 3.3: Flow phenomena on a subsonic, transonic and supersonic airfoil.

When the free-stream Mach number is increased even more, compressibility effects around objects become more pronounced. Each object has a subsonic critical Mach number M_c, above which the flow expands so rapidly that it becomes locally supersonic. Since the flow properties must return to supersonic free-stream conditions far away from the object, the flow must recompress at some point. This transition of a supersonic flow to a subsonic flow can only occur through a shock wave, resulting in a normal shock on the surface of the object as shown in Fig. 3.3b. Such flows are generally called transonic flows because they involve both subsonic and supersonic regions.

$$\begin{aligned}
\frac{p}{p_t} &= \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{\gamma}{\gamma - 1}} \\
\frac{T}{T_t} &= \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-1} \\
\frac{\rho}{\rho_t} &= \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{1}{\gamma - 1}}
\end{aligned} \tag{3.8}$$

Essentially, a shock is a sharp compression wave where the fluid instantaneously compresses and decelerates. This process is no longer isentropic and the entropy increases, implying that the isentropic relations can no longer be used. Instead, the differential form of the Euler equations in Eq. (3.2) must be integrated over an infinitesimal shock to obtain the jump relations or Rankine-Hugoniot equations (Evans, 2022). By

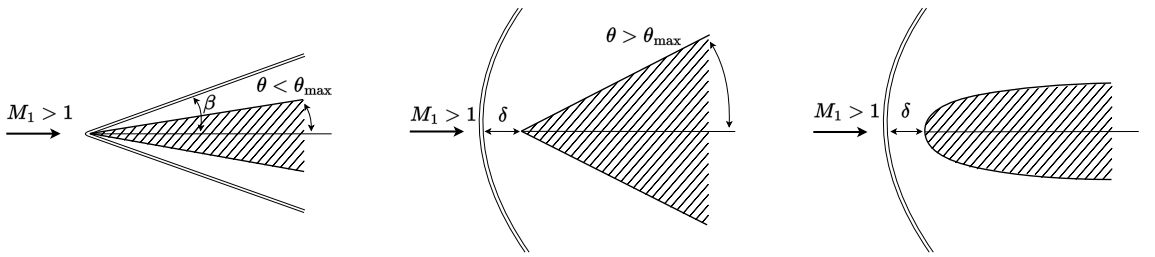
assuming an ideal gas, these can be converted to the normal shock relations as given in Eq. (3.9), with subscript 1 indicating the pre-shock values and subscript 2 the post-shock values. Note that velocities and Mach numbers in these equations are in the frame of reference of the shock itself, which may travel at some velocity with respect to an external observer. Interestingly, in the reference frame of the shock, the velocity component tangential to the shock does not change.

$$\begin{aligned}
\frac{p_2}{p_1} &= 1 + \frac{2\gamma}{\gamma+1}(M_1^2 - 1) \\
\frac{\rho_2}{\rho_1} &= \frac{(\gamma+1)M_1^2}{2+(\gamma-1)M_1^2} \\
\frac{T_2}{T_1} &= \left(1 + \frac{2\gamma}{\gamma+1}(M_1^2 - 1)\right)\frac{2+(\gamma-1)M_1^2}{(\gamma+1)M_1^2} \\
\frac{M_2}{M_1} &= \sqrt{\frac{M_1^2 + 2/(\gamma-1)}{2\gamma/(\gamma-1)M_1^4 - M_1^2}}
\end{aligned} \tag{3.9}$$

When the free-stream Mach number is increased even further so that $M_\infty > 1$, the flow is said to be supersonic. Instead of a shock occurring on the bottom or top surface of the object, it occurs at the tip of the object due to the sudden deceleration required to reach the subsonic stagnation conditions, as shown in Fig. 3.3c. The resulting shape and strength of the shock are completely determined by the geometry of the object. For a sharp object with an infinitesimal tip, the shock can be attached or detached depending on the angle θ of the tip. The maximum angle θ_{max} for which a shock remains attached can be determined using the θ-β-M relation in Eq. (3.10). If the shock is detached, this equation has no solution for the wave angle β. If the angle is small enough so that $\theta < \theta_{\text{max}}$, an oblique shock wave will attach to the tip as shown in Fig. 3.4a and the equation generally has two solutions for β. The small value is referred to as the weak shock solution and the large value is referred to as the strong solution. In the absence of strong back pressure, the weak solution emerges and the post-shock flow generally remains supersonic.

$$\tan(\theta) = 2\cot(\beta)\frac{M_1^2\sin^2(\beta) - 1}{M_1^2(\gamma + \cos(2\beta)) + 2} \tag{3.10}$$

Since the oblique shock forms an angle β with the flow, the jump equations in Eq. (3.9) can be converted to the oblique shock relations simply by substituting $M_1 = M_1\sin(\beta)$ and $M_2 = M_1\sin(\beta - \theta)$. If $\theta > \theta_{\text{max}}$ or if the tip of the object is blunt, a detached shock will occur that is no longer oblique but rather curved as shown in Fig. 3.4b and Fig. 3.4c. The detachment distance δ and the curvature of the shock cannot be solved analytically and depend on the size and shape of the body. However, it is possible to use experimental or numerical methods to obtain empirical correlations for classes of objects, as is done by Billig (1967) and Anderson et al. (1968) respectively. A general rule of thumb is that blunter objects have a larger shock detachment distance.

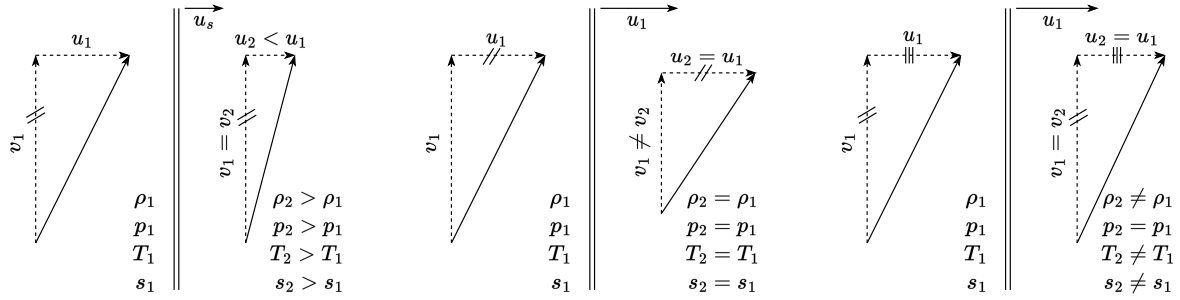


(a) Attached shock over sharp body **(b)** Detached shock over sharp body **(c)** Detached shock over blunt body

Figure 3.4: Attached and detached shocks over sharp and blunt bodies, based on Anderson (2021).

Apart from shocks, other discontinuities can also emerge in solutions of the Euler equations. Since there is no viscosity, there are no shear stresses and hence a discontinuity may exist where there is a jump in the tangential velocity. Such a discontinuity is called a shear wave, and it is shown in Fig. 3.5b. Furthermore, it is also possible that contact discontinuities arise in flows with constant velocity and pressure. These

contact discontinuities may show a jump in density, energy and even entropy. Nevertheless, they should not be confused with shocks as the normal velocity does not jump across a contact discontinuity. Note that a shear wave and contact discontinuity may also be combined.

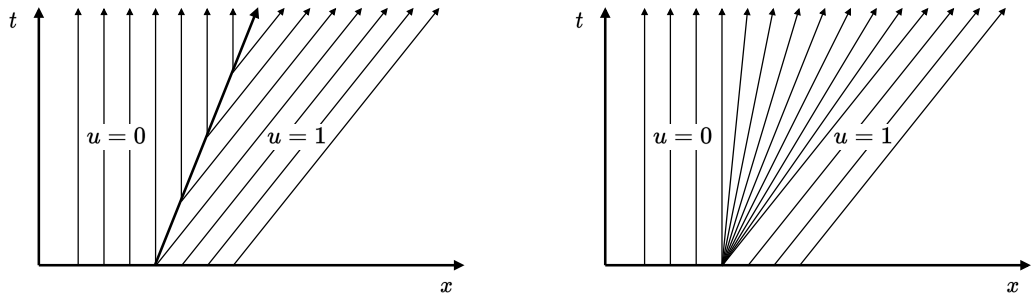


(a) Shock wave **(b)** Shear wave **(c)** Contact discontinuity

Figure 3.5: General types of discontinuities that can emerge in solutions of the two-dimensional Euler equations.

As a last remark, note that Eq. (3.9) is mathematically valid for $M_1 < 1$, which corresponds to an "expansion shock" as shown in Fig. 3.6a for a one-dimensional flow. However, such shocks are not seen in nature and instead, expansion always occurs through an isentropic rarefaction wave. To understand why expansion shocks are nonphysical, refer to the second law of thermodynamics, which essentially dictates that entropy must either remain constant or increase along a streamline. For a calorically perfect gas, the change in entropy across a shock is given by Eq. (3.11). When substituting the ratios from Eq. (3.9), it shows that the entropy change is negative when $M_1 < 1$, which violates the second law of thermodynamics. Therefore, the pre-shock flow velocity must be supersonic with respect to the shock, implying that $M_2 \leq 1$, $p_2/p_1 \geq 1$, $T_2/T_1 \geq 1$ and $\rho_2/\rho_1 \geq 1$, which are also referred to as entropy conditions.

$$s_2 - s_1 = c_p \ln \frac{T_2}{T_1} - R \ln \frac{p_2}{p_1} \tag{3.11}$$



(a) Nonphysical expansion **(b)** Physical expansion

Figure 3.6: Comparison of a nonphysical expansion shock to a physical rarefaction wave (Evans, 2022).

Part II

Results

4

Failure mode analysis

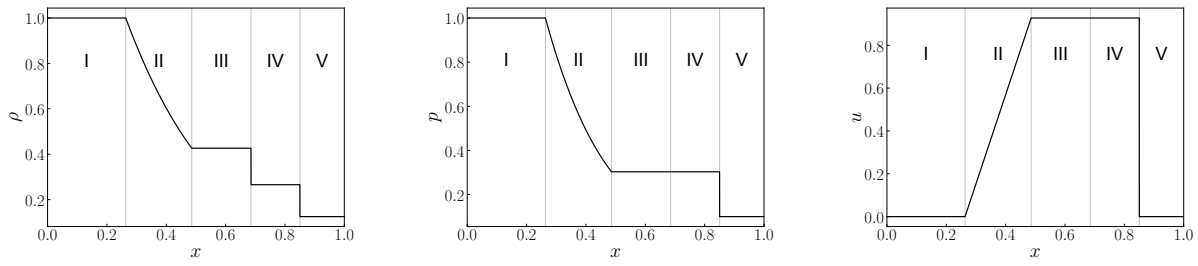
As mentioned in the introduction, PINNs appear to struggle with capturing shocks despite their advantages compared to traditional solvers. This chapter aims to answer the first two research questions, namely why PINNs fail on highly compressible problems and how existing adaptations relate to the failure modes. In Section 4.1, PINNs are applied to a simple problem, revealing some counterintuitive behavior. This behavior is related to the theory in the previous chapters in Section 4.2, leading to the identification of a first failure mode. Afterward, Section 4.3 introduces various existing adaptations and their underlying rationales to show that they unknowingly treat this failure mode. Lastly, a set of problems that isolate compressible phenomena is considered in Section 4.4 to show that a second failure mode may arise.

4.1 Riemann problem

Consider the one-dimensional Riemann problem given in Eq. (4.1) and its solution illustrated in Fig. 4.1. Essentially, this problem represents a tube in which a high-pressure fluid is separated from a lower-pressure fluid by a membrane. At $t = 0$ the membrane is removed, which creates an expansion wave traveling to the left and a compression or shock wave traveling to the right. In addition, a contact discontinuity travels to the right, following the shock wave. The solution of such problems can be calculated without error using exact Riemann solvers, interested readers are referred to Toro (2009).

Initial conditions	**Boundary conditions**
$\rho(x,0) = 1.0$ if $x < 0.5, 0.125$ if $x \geq 0.5$	$\partial_x \rho(x,t) = 0$ at $x = 0.0, 1.0$
$p(x,0) = 1.0$ if $x < 0.5, 0.1$ if $x \geq 0.5$	$\partial_x p(x,t) = 0$ at $x = 0.0, 1.0$
$u(x,0) = 0.0$	$\partial_x u(x,t) = 0$ at $x = 0.0, 1.0$

(4.1)

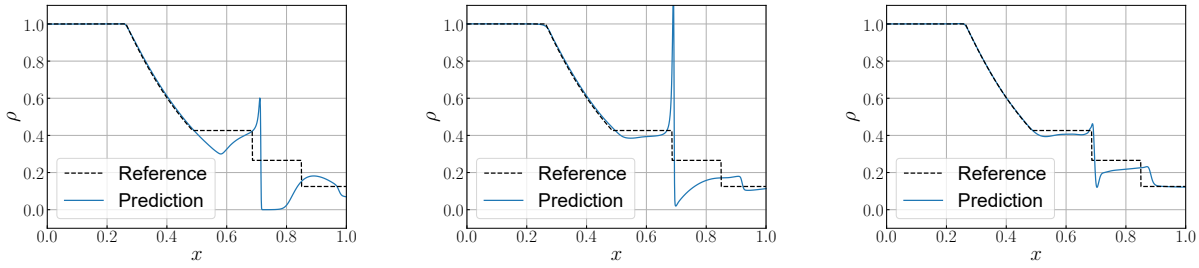


(a) Density **(b)** Pressure **(c)** Velocity

Figure 4.1: The exact solution of the Riemann problem in Eq. (4.1) at $t = 0.2$. A left-traveling expansion wave is located in region II, a contact discontinuity separates regions III and IV, and a compression wave (shock) separates regions IV and V. The regions I and V are undisturbed.

Riemann problems often consist of different wave types, making them excellent for assessing and comparing the capabilities of shock-capturing methods. They are extensively treated in literature on traditional methods, essentially serving as a standardized test. For the physics-informed approach, a PINN is set up

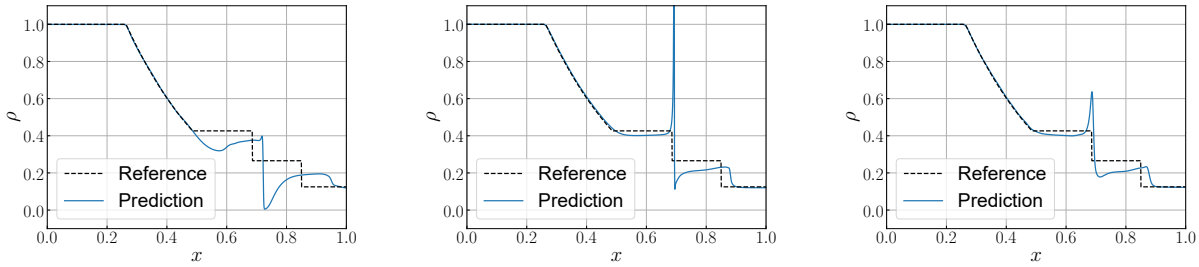
as given in Fig. 3.1, featuring 4 layers of 100 neurons. The `tanh` activation function is used for all neurons except the outputs, where `linear` is used for the velocities and `exp` is used for the density and pressure. A total of 10,000 residual points are sampled in the domain $(x, t) \in [0, 1] \times [0, 0.2]$ using the Hammersley sequence introduced in Section 2.2.1. Furthermore, 1,000 initial condition points are linearly sampled in $x \in [0, 1]$. Note that Neumann boundary conditions are not considered, technically making the problem underdetermined. However, they do not affect the solution significantly and can actually complexify the loss landscape as mentioned in Section 2.2.2. Furthermore, a weighted loss function is used with a variable weight w_0 for the initial condition loss term. Finally, the PINN is trained using the L-BFGS optimizer designed by Liu et al. (1989) with a learning rate of 0.1 until the loss no longer improves. The simulations are performed using the PyTorch machine learning library (Paszke et al., 2019) for Python.



(a) $w_0 = 1$ **(b)** $w_0 = 10$ **(c)** $w_0 = 100$

Figure 4.2: Density solutions at $t = 0.2$ of PINNs applied to the Riemann problem with different loss weights w_0.

The results are shown in Fig. 4.2, revealing that the PINN does not produce a correct solution to the problem. Increasing the weight of the initial condition loss improves the accuracy of the solution slightly, but it is still far from perfect. The PINN struggles mainly with the contact discontinuity and shock wave on the right side of the domain, while it has no issues with the expansion region on the left side of the domain. In particular, a distinct peak and dip form around the contact discontinuity, which has also been observed by Papados (2021). At first, it seems as if the PINN has converged to a local minimum as the residuals are fairly high: on the order of 10^{-3}, 10^{-2} and 10^{-1} for the different weights respectively. Based on this intuition, numerous adaptations have been proposed that aim to improve the convergence pathology.

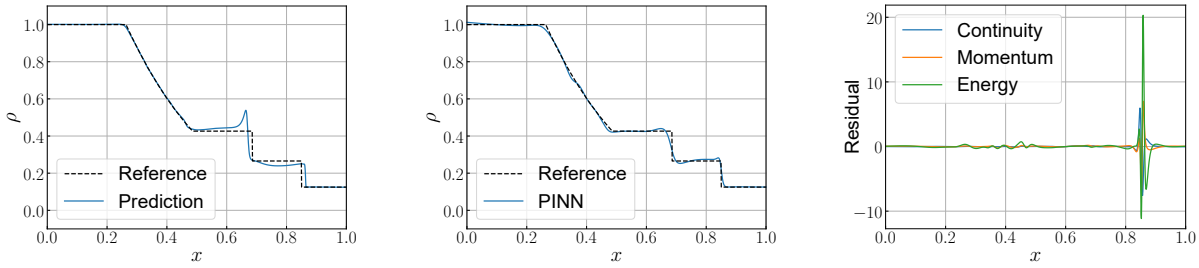


(a) Clustered points **(b)** More points **(c)** Domain extension

Figure 4.3: Density solutions at $t = 0.2$ of PINNs with adaptations applied to the Riemann problem with $w_0 = 100$.

One method was proposed by Mao et al. (2020), who observed that clustering more collocation points around shocks can improve their simulated sharpness. Intuitively this makes sense because shocks are very sharp features in the domain and therefore they require finer placement of collocation points, similar to refining meshes in traditional solvers. Of course, this requires foreknowledge of the shock location. Nevertheless, Fig. 4.3a shows that uniformly clustering half of the 10,000 collocation points in a band of $\Delta x = 0.1$ around the shock does not improve the solution. In fact, compared to the solution in Fig. 4.2c produced by a non-clustered but otherwise identical setup, the accuracy has deteriorated. A possible explanation is that refining the collocation points around the shock is not sufficient and instead the entire domain should be refined. However Fig. 4.3b shows that even with 100,000 collocation points and 10,000 initial condition points, a ten-fold increase, an accurate solution is still not obtained.

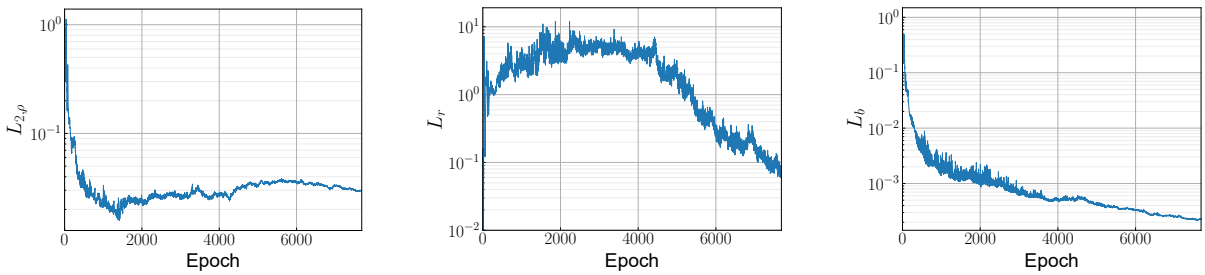
Instead of clustering, Papados (2021) proposes to extend the domain in which the collocation points are sampled. The rationale of this approach is that domain extension can suppress spurious oscillations, similar to domain extension methods in traditional solvers. It must be noted however that spurious oscillations are the result of numerical discretization schemes (Shu, 1998), which are not present in PINNs. Nevertheless, the authors report significantly better results, specifically on the same Riemann problem as Eq. (4.1). Fig. 4.3c shows the solution for a domain with twice the size, in other words, $x \in [-1, 2]$. To keep the density of the collocation points identical, the number of collocation points is increased to 20,000 and the number of initial condition points is increased to 2,000. Evidently, the solution is still far from accurate.



(a) Latest solution **(b)** Best solution **(c)** Residuals of best solution

Figure 4.4: Density solutions and residuals at $t = 0.2$ of a CPINN applied to the Riemann problem. The best solution is based on the lowest $L_{2,\rho}$ error.

The discrepancy between the obtained results and promised results with the adaptations requires some explanation. Before diving deeper into this, it is instrumental to evaluate the performance of CPINNs on the same problem. Essentially, the PINN setup is used together with a discriminator with 5 layers of 100 neurons with the `relu` activation function to replace the mean-squared error loss with an adversarial loss as given in Eq. (2.15). It has 6 output neurons with the `linear` activation function, one for each initial condition and PDE. The CPINN is trained using the ACGD optimizer introduced in Section 2.3.2 and training is terminated after 24 hours, which is significantly longer than the approximate 30 minutes required to train the PINNs. Interestingly, a more accurate result is obtained as shown in Fig. 4.4a, although the spurious peak at the contact discontinuity remains. Even more intriguing, the history of the error in Fig. 4.5a shows that initially, the CPINN produces an even more accurate solution shown in Fig. 4.4b, even though this solution has huge associated residuals near the shock wave as shown in Fig. 4.4c.



(a) Density error **(b)** Residual loss **(c)** Initial condition loss

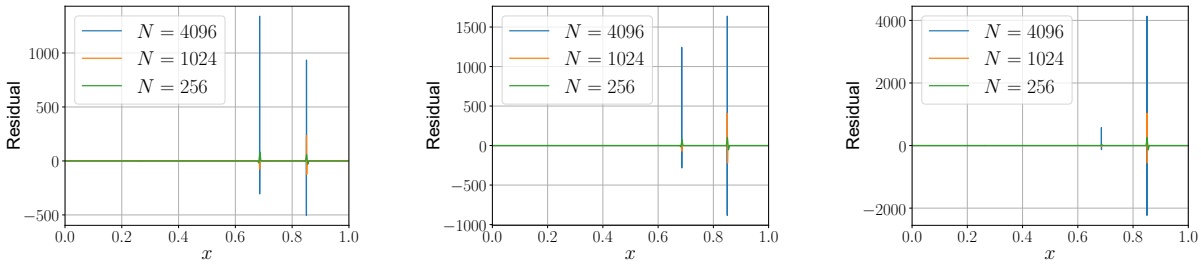
Figure 4.5: Pathology of the error and losses for the CPINN solution.

Upon closer inspection of the different loss terms in Fig. 4.5, an interesting loss pathology can be observed: instead of minimizing the residuals, the CPINN initially *increases* the residuals. In contrast, the initial condition loss in Fig. 4.5c decreases as expected. To some extent, the loss pathology can be explained by the fact that an adversarial loss does not necessarily minimize the loss directly. However, it does not explain why the best solution in Fig. 4.4 has a substantially larger residual loss than the latest and worse solution. This is counterintuitive because better solutions generally have lower residuals, although this is a highly nonlinear relationship. At first glance, it is possible to argue that the CPINN is simply ”lucky” in the initial phase. However, this pathology is consistently observed, independent of the parameter initialization

and also the size of the networks. Based on this behavior, but also the behavior of the PINN, it seems as if lowering the residual does not necessarily improve the solution.

4.2 Discontinuities and entropy

The latter statement is not far off from the truth. In fact, the behavior of PINN and its various adaptations on the Riemann problem is related to a crucial fact that was purposely not mentioned in Section 3.1: the differential form of the Euler equations is not valid at discontinuities. This is because the derivation of the differential form from the integral form assumes smoothness of the solution, but Fig. 4.1 shows that the solution is not smooth at the contact discontinuity and the shock. As a result, the first derivatives in Eq. (3.2) are not defined because the jump in variables is instantaneous. If the derivatives are approximated by a finite discretization scheme for instance, the residuals might be nonzero even for the exact solution.



(a) Conservation of mass **(b)** Conservation of momentum **(c)** Conservation of energy

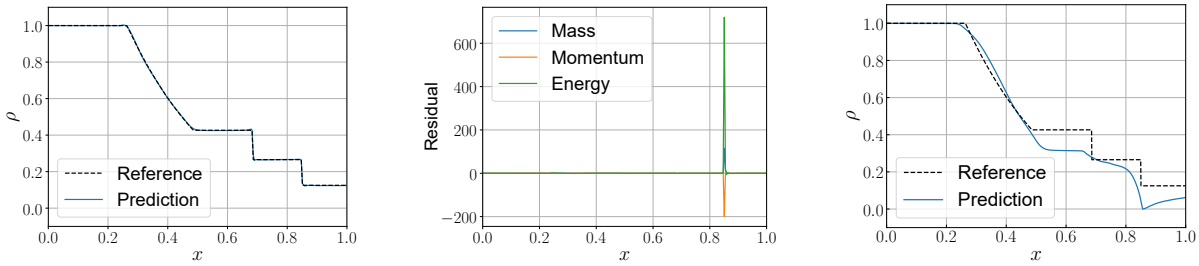
Figure 4.6: Residuals after discretizing the solution of the Riemann problem into N chunks in space and time.

This is visualized in Fig. 4.6, which shows high residuals at the contact discontinuity and the shock, but also small nonzero residuals at the start of the expansion wave. As the discretization is refined, the residuals at the expansion wave decrease, while the residuals at the contact discontinuity and the shock wave *increase*. One might argue that this phenomenon does not occur in PINNs because they provide a continuous solution that can be differentiated exactly using automatic differentiation. However, numerical discretization is only part of the explanation of why large residuals occur at discontinuities. Even for a continuous approximation of the discontinuities, the target solution simply does not satisfy the Euler equations in Eq. (3.2). This fundamental issue becomes clear once applying the first law of thermodynamics to the equations, as done by Liepmann et al. (2001).

$$
\begin{aligned}
T\mathrm{d}s &= \mathrm{d}e + p\mathrm{d}\left(\frac{1}{\rho}\right) \\
\frac{\mathrm{D}s}{\mathrm{D}t} &= \frac{1}{T}\left(\frac{\mathrm{D}e}{\mathrm{D}t} - \frac{p}{\rho^2}\frac{\mathrm{D}\rho}{\mathrm{D}t}\right) \\
\frac{\mathrm{D}s}{\mathrm{D}t} &= \frac{1}{T}\left(-\frac{1}{\rho}\nabla\cdot p\mathbf{v} - \mathbf{v}\cdot\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} + \frac{p}{\rho^2}\rho\nabla\cdot\mathbf{v}\right) \\
\frac{\mathrm{D}s}{\mathrm{D}t} &= \frac{1}{T}\left(-\frac{p}{\rho}\nabla\cdot\mathbf{v} - \frac{\mathbf{v}}{\rho}\cdot\nabla p + \mathbf{v}\cdot\frac{\nabla p}{\rho} + \frac{p}{\rho}\nabla\cdot\mathbf{v}\right) \\
\boxed{\frac{\mathrm{D}s}{\mathrm{D}t} = 0}
\end{aligned}
\tag{4.2}
$$

Note that Eq. (4.2) uses some of the relations in Eq. (3.3) and Eq. (3.4) for the derivation. As the equation shows, the differential form of the Euler equations prescribes that the entropy s remains constant along a streamline, which is not prescribed by the integral form. However, as discussed in Section 3.3, an entropy jump occurs across shocks. The fundamental issue is therefore that PINNs are trying to find an isentropic solution to a non-isentropic problem. Expansion waves are isentropic, hence this does not pose an issue. Although a jump in entropy can occur across a contact discontinuity, it does not happen along the streamline so it does not violate the PDEs. Therefore, the *entropy failure mode* will only occur when shock waves are present.

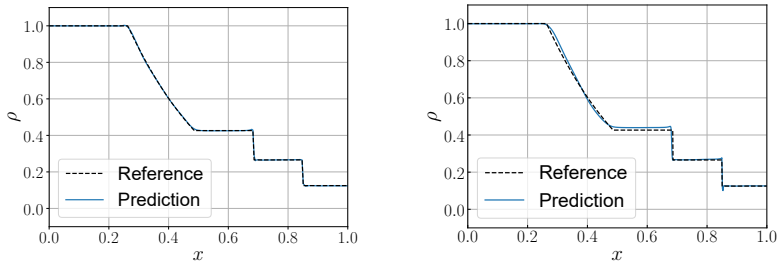
To show the severity of the issue, consider a neural network that is trained directly on the known solution. In other words, no residual loss and boundary loss are imposed, but rather a conventional target loss is imposed of the form $L = ||\mathbf{u} - \mathbf{u}_{\text{true}}||^2$. The solution of the trained network is shown in Fig. 4.7a, showing that it closely resembles the target solution in Fig. 4.1. Notice also that the peak in the residuals in Fig. 4.7b only occurs at the shock, confirming that the peak at the contact discontinuity in Fig. 4.6 is merely a consequence of discretization. When this pre-trained network is used as the initial network for a PINN, training it with a residual loss and boundary loss actually deteriorates the solution, as shown in Fig. 4.7b. Again, this is because the PINN is attempting to find an isentropic solution while the target solution is non-isentropic. This also means that any of the adaptive sampling techniques in Section 2.2.1 will only worsen the failure mode as they will place more points near the shock to enforce isentropy.



(a) Pre-trained solution **(b)** Residuals of pre-trained solution **(c)** PINN solution

Figure 4.7: Deterioration of the solution when using a network that is pre-trained on the true solution as an initial network for a PINN with an initial condition loss weight of $w_0 = 100$.

Just because PINNs try to find an isentropic solution does not necessarily mean the obtained solution will be isentropic. For example, the network may be overfitting, implying that the solution is isentropic at collocation points but not in regions in between the collocation points. However, the network may also be underfitting, implying that there are finite and possibly significant residuals at the collocation points. Nevertheless, it is not likely that a PINN will purposely underfit the residual points at the shock wave because the squared nature of the loss function tends to spread out the residuals. In addition, PINNs will spread the residuals over the different PDEs instead of containing them in the energy equation, which is the equation that indirectly prescribes the conservation of entropy along streamlines in the absence of dissipation terms. Evidently, Fig. 4.4c and Fig. 4.5b show that CPINNs are initially able to underfit the residuals to produce accurate shocks, although this is not a result of their non-squared loss but rather of the adversarial training method which does not explicitly prescribe that the residual must be minimized.



(a) Pre-trained solution **(b)** PINN solution

Figure 4.8: Deterioration of the solution when using a network that is pre-trained on the true solution as an initial network for a PINN with an initial condition loss weight of $w_0 = 100$. Only 1,000 collocation points and 100 initial condition points have been used.

Although it is not likely that a PINN underfits, it is considerably more likely that it overfits. To show this, again consider the PINN that is initialized with the pre-trained solution. However this time, only a fraction of the original collocation points and initial conditions points are used. As a result, it is considerably less likely that collocation points are located at the shock. Therefore, the PINN is not aware of the large residual

peaks and therefore it does not deteriorate the solution by minimizing them, as is shown in Fig. 4.8. This is somewhat remarkable, as the jump conditions remain to be satisfied correctly.

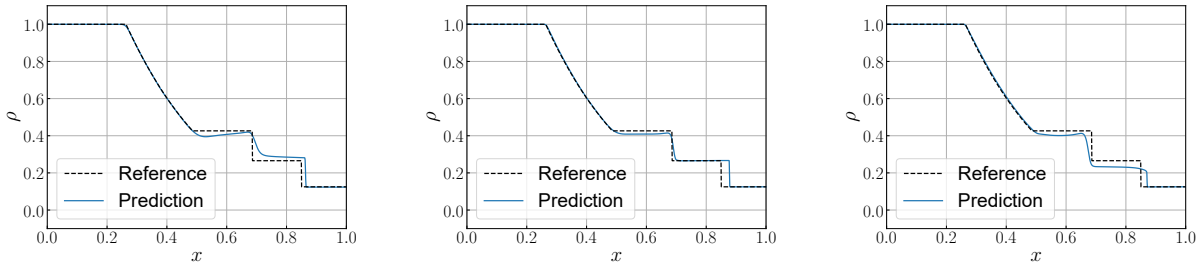
In real scenarios the PINN is not initialized with the correct solution, making it harder for the PINN to produce a shock in between collocation points by overfitting. In practice, the placement of the shock is sensitive to the density but also the placement of the collocation points. This also explains why Fig. 4.3 demonstrates that the adaptations from Mao et al. (2020) and Papados (2021) are not effective; different sampling procedures are used. Upon closer inspection, Mao et al. (2020) clustered points around the shock not through random sampling, but by placing them along lines parallel to the shock in space and time as shown in Fig. 2.9b. Therefore, it is more likely that the shock locates itself between one of the parallel lines. In addition, they considered a simple oblique shock and were able to reasonably capture the shock already without any adaptations. Furthermore, Papados (2021) applied domain extension, but did not change the number of residual points, effectively decreasing the density of the collocation points almost fivefold. In addition, the author used a variant of grid sampling discussed in Section 2.2.1 which increases the likelihood of the formation of a shock wave that does not intersect with collocation points.

4.3 Failure mode alleviations

While the adaptations proposed by Mao et al. (2020) and Papados (2021) lack a theoretical foundation, more substantiated adaptations do exist. However, to the author's best knowledge, none of them explicitly mention that the fundamental issue is that the PDEs are simply not valid at the approximated shock wave. Nevertheless, it is important to review the different methods and their rationales, placing them in the context of the theory introduced in the previous section. Instead of changing the distribution of the collocation points, Liu (2022) propose to alter the PDEs to suppress the residual peaks that occur at shocks. They correctly realize that points that are placed near shocks, which they term transition points, have a high residual. While this can be avoided by the PINN overfitting and placing the shock in between the collocation points, they argue that this is not likely to happen due to the paradoxical status of transition points: to make the shock thinner and fall in between collocation points, the gradients at the shock must be increased temporarily, which increases the residuals. Since PINNs directly minimize the residuals, this is generally not favorable. To alleviate this paradoxical status, they introduce a weight to the residual loss as shown in Eq. (4.3).

$$L_r = \|\lambda P(\mathbf{u}(\mathbf{x}, t))\|^2, \quad \lambda = \frac{1}{\epsilon\left(|\nabla \cdot \mathbf{v}| - \nabla \cdot \mathbf{v}\right) + 1} \tag{4.3}$$

Here ϵ is a small hyperparameter that influences the magnitude of the weight. Effectively, the denominator resembles the rate of compressibility, since $\nabla \cdot \mathbf{v} = 0$ for incompressible flows. By taking the absolute and subtracting the plain value, only compression ($\nabla \cdot \mathbf{v} < 0$) is considered but not expansion ($\nabla \cdot \mathbf{v} > 0$). However, note that this assumes that the velocity components are positive. Introducing such a weight to the Riemann problem gives considerably better results, as is shown in Fig. 4.9a. However, as the authors mention themselves in their source code, while the method is highly effective at producing discontinuities, the location of the discontinuity might not necessarily converge to the correct location. In addition, the location seems to be highly dependent on the setup of the PINN, as is shown in Fig. 4.9 for different collocation point sampling methods. Furthermore, it is observed that once a shock is produced, it will only intensify and its location does not change much since the PINN is essentially overfitting.



(a) Hammersley initialization **(b)** Uniform initialization **(c)** Grid initialization

Figure 4.9: PINN solution obtained when the residuals are weighted by the compressibility term in Eq. (4.3) with $\epsilon = 0.1$, for different collocation point initializations. The initial condition loss weight is $w_0 = 10$.

Nevertheless, this method performs better than the other adaptations so far and it has even been used to simulate the first unsteady bow shock around a transonic cylinder without any data (Liu, 2022). At the time of writing, a similar weighting as Eq. (4.3) was proposed by Ferrer-Sánchez et al. (2023), although the authors do not elaborate much on the underlying failure mode. This weighting is given in Eq. (4.4), with α_i hyperparameters. Note that there are essentially three main differences compared to Eq. (4.3): (1) it only applies to one-dimensional flows, (2) it considers the gradients of all primitive variables instead of only the velocity and (3) it allows for higher-order exponents of the gradients to suppress residuals even more. The presented results do not indicate that this weighting definition is more favorable than Eq. (4.3). Note that both weighting definitions do not take into account the direction of the flow, implying that they will also unnecessarily affect contact discontinuities in multi-dimensional flows. Furthermore, it remains unclear how these methods affect regions of isentropic expansion or compression.

$$L_r = \|\lambda P(\mathbf{u}(\mathbf{x}, t))\|^2, \quad \lambda = \frac{1}{1 + \alpha_1 |\partial_x \rho|^{\alpha_2} + \alpha_3 |\partial_x u|^{\alpha_4} + \alpha_5 |\partial_x p|^{\alpha_6}} \tag{4.4}$$

While weighing the residuals solves the failure mode up to some extent, it is more of a remedy than a proper solution to the underlying physical issue of entropy. A better approach can be found by realizing that traditional solvers require seemingly no adaptations of the Euler equations to capture shocks correctly. While some finite volume solvers essentially solve the integral form and bypass the entropy issue, even finite difference solvers do not struggle with capturing shocks. In essence, while the lack of numerical discretization is presented as an advantage of PINNs, it is a disadvantage in terms of their capability to capture shocks. To illustrate this, consider a general scalar conservation law of the form

$$u_t + F(u)_x = 0 \tag{4.5}$$

When such a conservation law is solved by traditional methods, discretization schemes introduce discretization errors into the PDE, which are proportional to derivatives of the conserved variable. Errors proportional to odd derivatives introduce a dispersive effect into the equation, while those proportional to even derivatives introduce a dissipative effect into the equation. For example, a dissipative scheme might effectively modify the above conservation law to

$$u_t + F(u)_x = \epsilon u_{xx}, \tag{4.6}$$

where $\epsilon > 0$ is a small coefficient that scales with the mesh discretizations Δx and Δt. This additional term effectively represents the addition of viscosity to the solution, even though the original equations are inviscid. Although this numerical viscosity term implies that the obtained solution is not inviscid, especially for coarse mesh refinements, it is essential to produce discontinuities. While the addition of viscosity smears out discontinuities over a larger area, it also provides a form of dissipation. In the case of the Euler equations, this dissipation provides a means of entropy change which can make shocks an admissible flow feature. Since ϵ scales with the mesh size, finer meshes will result in finer shocks and more localized entropy changes. The vanishing viscosity theorem prescribes that solutions obtained from conservation laws, such as the one in Eq. (4.5), are only admissible or physical if they resemble the same solution that is obtained in the limit of vanishing viscosity (LeVeque, 1992). In other words,

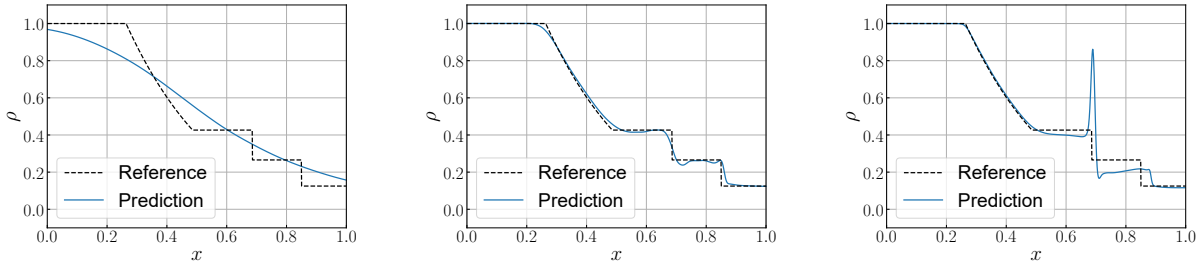
$$u_{\text{physical}} = \lim_{\epsilon \to 0} u_\epsilon \tag{4.7}$$

This limit is also important for satisfying the entropy conditions that were introduced in Section 3.3, as numerical viscosity will never decrease the entropy and therefore no nonphysical expansion shocks can occur as detailed by Evans (2022). Therefore, the physical solution given by Eq. (4.7) is often called the entropy solution. In short, the addition of numerical viscosity has substantial benefits. While PINNs do not involve discretization and hence do not experience numerical viscosity, a viscosity term can simply be added to the PDEs. In fact, PINNs are still advantageous compared to traditional solvers in this sense because the viscosity term can be set explicitly and it can even be made a function of space and time so that its effects are only local and do not negatively affect isentropic regions of the flow. In other words, the differential equations in Eq. (3.2) can be augmented to Eq. (4.8), where Δ is the Laplace operator and $\nu(\mathbf{x}, t) > 0$. Note that the evaluation of the right-hand sides involves an additional two derivatives per

spatial variable, which considerably increases the computational and memory requirements.

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= \nu(\mathbf{x}, t)\Delta\rho, \\ \frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (p + \rho \mathbf{v}\mathbf{v}^{\mathsf{T}}) &= \nu(\mathbf{x}, t)\Delta(\rho\mathbf{v}), \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((p + \rho E)\mathbf{v}) &= \nu(\mathbf{x}, t)\Delta(\rho E). \end{aligned} \tag{4.8}$$

Before considering various viscosity forms proposed in literature, first consider simply that the viscosity is constant, in other words $\nu(\mathbf{x}, t) = \nu$. Fig. 4.10a shows that for large amounts of constant viscosity, the dissipative effect is so strong that the solution of the viscous PDEs does not match the entropy solution. When the viscosity is decreased, as in Fig. 4.10b, the solution closely resembles the entropy solution. However, when the viscosity is decreased even more, the entropy failure mode resurfaces as the effect of viscosity on the PDEs becomes negligible. A very low viscosity likely makes the target shock so thin that it is no longer captured by the collocation points, implying that a higher density of collocation points would possibly allow lower viscosity values. Note that this observation also suggests that the entropy failure mode can also manifest in the Navier-Stokes equations when the viscosity is too small.



(a) $\nu = 1 \times 10^{-1}$ **(b)** $\nu = 1 \times 10^{-3}$ **(c)** $\nu = 1 \times 10^{-4}$

Figure 4.10: Density solutions at $t = 0.2$ of a PINN with viscosity applied to the Riemann problem for different levels of constant artificial viscosity ν.

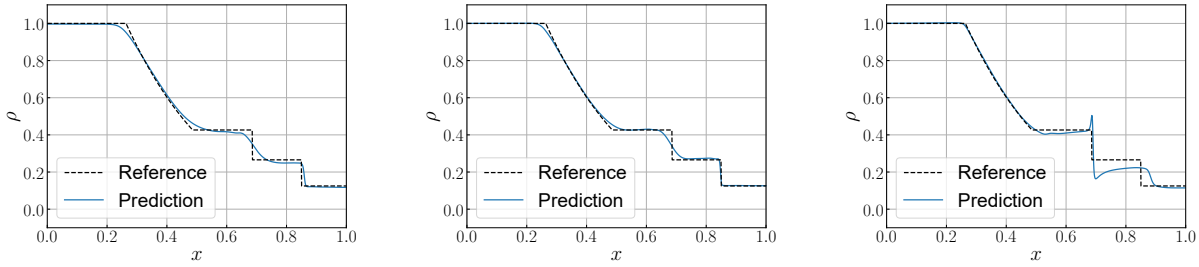
Since trying out different viscosity values in a trial-and-error process is computationally expensive, Wassing et al. (2023) propose a different approach. Instead, the artificial viscosity ν is made variable and it is initialized with a relatively large value. The PINN is then trained for a large number of epochs until reasonable convergence has been reached as in Fig. 4.10a. From that point on, the viscosity is linearly decreased to zero. However, the authors note that in practice zero viscosity is never reached, which is expected as Fig. 4.10 shows that the entropy failure mode reappears when the viscosity becomes too low. Furthermore, the authors have only applied this method to subsonic flows, implying that it is not proven to be effective at capturing shocks.

Although the approach appears more refined, it remains unclear when, how fast and to what level the viscosity much be reduced. This introduces a large number of additional hyperparameters, which again require a process of trial and error to optimize. Therefore, Coutinho et al. (2023) instead propose to include the viscosity as a parameter that is optimized alongside the neural network parameters. Since a large viscosity simplifies the target solution by diffusing it, a viscosity loss term is added to the loss function to reward lower viscosity values. The new weighted loss function is shown in Eq. (4.9), with w_ν a weight that prescribes the importance of minimizing the viscosity. Note that there are many possible choices for L_ν, but a typical choices are $L_\nu = \nu$ and $L_\nu = \nu^2$.

$$L = w_0 L_0 + w_b L_b + w_b L_r + w_\nu L_\nu \tag{4.9}$$

While the addition of a variable viscosity term allows the PINN to automatically reduce the viscosity as much as possible, it still includes the hyperparameter w_ν. As a result, behavior that is similar to using constant artificial viscosity arises again. Fig. 4.11 shows that when the weight is too large, the viscosity is minimized too quickly and the entropy failure mode is not avoided. When the weight is too small, the viscosity remains large and the solution is diffused too much. Nevertheless, the method seems to be far

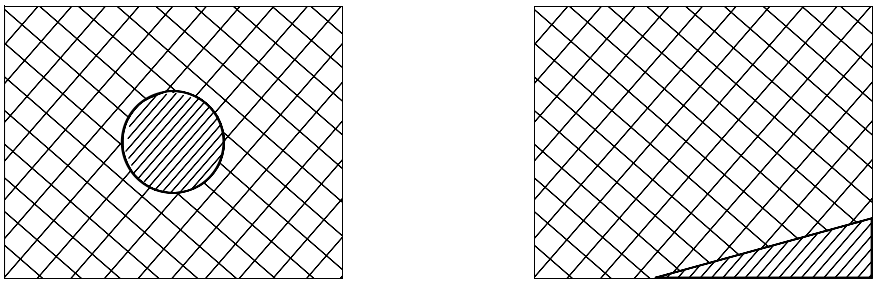
less sensitive to changes in w_ν compared to the sensitivity of the constant viscosity method to changes in ν. This is likely because a low but nonzero viscosity turns the problem well-posed, implying that both the residual and initial condition losses can be minimized far more effectively. Another advantage is that a suitable weight will generally yield physical solutions with a lower viscosity than can be achieved with a constant viscosity. In other words, they more closely approximate the entropy solution.



(a) $w_\nu = 1 \times 10^{-6}$, $\nu = 1.1 \times 10^{-3}$ **(b)** $w_\nu = 1 \times 10^{0}$, $\nu = 7.0 \times 10^{-4}$ **(c)** $w_\nu = 1 \times 10^{6}$, $\nu = 2.2 \times 10^{-5}$

Figure 4.11: Density solutions at $t = 0.2$ of a PINN with global viscosity applied to the Riemann problem for different viscosity loss weights w_ν with a viscosity loss of $L_\nu = \nu^2$. Note that ν indicates the final viscosity value.

While global artificial viscosity alleviates the entropy failure mode to some extent, it is an indiscriminate method in the sense that viscosity is added everywhere in the domain instead of only at the shock. As a result, regions that are normally isentropic become non-isentropic, such as the expansion wave to the left of the domain. Therefore, Coutinho et al. (2023) also propose to use a parametric viscosity map of the form $\nu(\mathbf{x}, t, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are parameters that define the shape and intensity of the map. For example, to capture shocks one could define a linear map that has its angle and width as parameters. While this method is effective, it requires foreknowledge of the solution which is not readily available when dealing with more complex shocks. Last but not least, the authors also propose a residual-based approach where the current residuals are used to assign adaptive viscosity values to the collocation points.



(a) Without symmetry plane **(b)** With symmetry plane

Figure 4.12: Examples of control volumes on two-dimensional domains for highly compressible problems.

Although the essence of the method is good, it involves a calculation based on neighboring collocation points, which requires the definition of a mesh. The usage of meshes or control volumes to deal with the discontinuities that arise in hyperbolic PDEs has earlier already been proposed by Patel et al. (2022). Essentially, the integral form in Eq. (3.1) is applied to control volumes similar to those in Fig. 4.12 instead of the application of the differential form to collocation points. While mesh-based methods certainly have their advantages, they always involve some form of numerical discretization which clashes with the fundamental principles of PINNs.

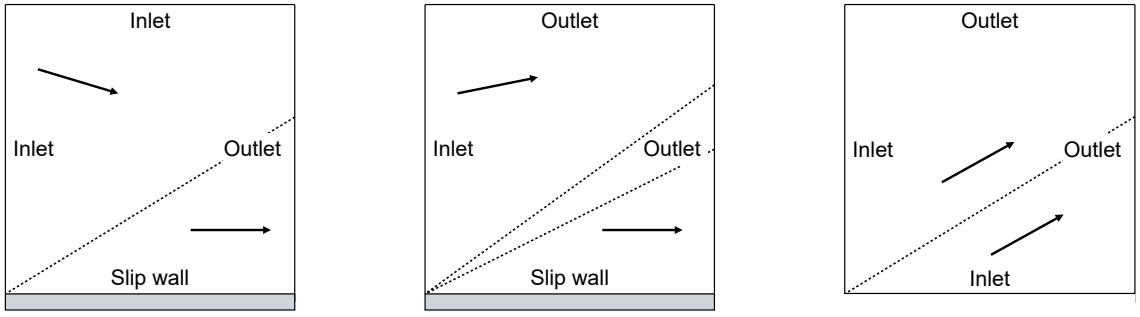
4.4 Steady waves

The analysis of the Riemann problem has given valuable insights into the entropy failure mode that is prohibiting PINNs from simulating highly compressible flows. However, its solution includes three different

wave types, which does not allow to distinguish the capabilities of PINNs capturing the waves individually. Therefore, the purpose of this section is to judge how well PINNs can capture different individual wave types for different numbers of collocation points, boundary weightings and viscosity levels. Interestingly, this will reveal another failure mode. From this point on, only steady problems will be considered as it can be expected that the Euler equations will converge to a steady solution when well-posed and time-independent boundary conditions are considered.

Shock	**Expansion**	**Contact discontinuity**
Left: $\mathbf{u} = (1.0, 1.0, 2.29, -0.40)$	Left: $\mathbf{u} = (1.0, 1.0, 2.54, 0.45)$	Left: $\mathbf{u} = (1.0, 1.0, 2.05, 1.18)$
Top: $\mathbf{u} = (1.0, 1.0, 2.29, -0.40)$	Top: $\partial_y \mathbf{u} = 0$	Top: $\partial_y \mathbf{u} = 0$
Bottom: $\partial_y p = \partial_y \rho = \partial_y u = v = 0$	Bottom: $\partial_y p = \partial_y \rho = \partial_y u = v = 0$	Bottom: $\mathbf{u} = (2.0, 1.0, 2.05, 1.18)$
Right: $\partial_x \mathbf{u} = 0$	Right: $\partial_x \mathbf{u} = 0$	Right: $\partial_x \mathbf{u} = 0$

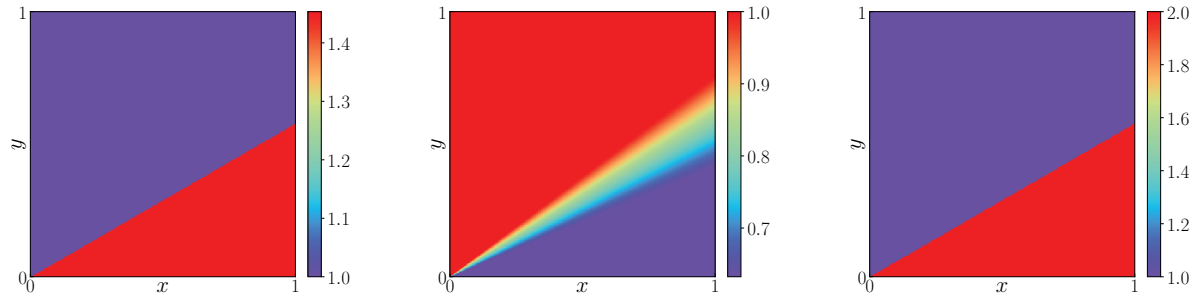
(4.10)



(a) Shock **(b)** Expansion **(c)** Contact discontinuity

Figure 4.13: Sketch of the domains for the steady waves problems.

The considered problems are all fully supersonic and defined over the domain $(x, y) \in [0, 1] \times [0, 1]$ as shown in Fig. 4.13, their boundary conditions are given in Eq. (4.10) with $\mathbf{u} = (\rho, p, u, v)$. The shock wave problem is equivalent to the flow over a wedge with an angle of $\theta = 10°$, except rotated by $-10°$. An oblique shock forms over the flat plate, allowing the solution to be analytically determined using the oblique shock relations in Eq. (3.9). The expansion wave problem is similar, except the inlet flow is angled *upwards* at an angle of $10°$, resulting in a rarefaction wave over the flat plate. This problem is particularly interesting because a nonphysical expansion "shock" also satisfies the boundary conditions, as discussed in Section 3.3. Last but not least, the contact discontinuity problem has two inlets with different flow densities, resulting in a contact discontinuity at an angle of $30°$. The respective density solutions are given in Fig. 4.14, showing that the solutions consist of identically angled waves, namely $30°$. For the expansion fan, this is the central angle.

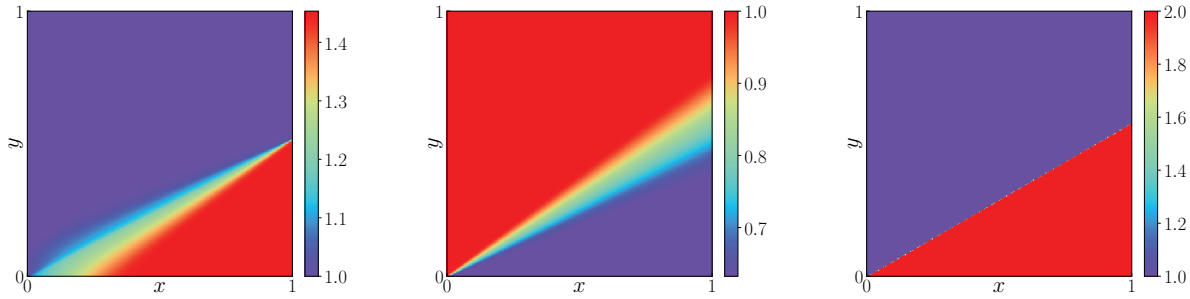


(a) Shock **(b)** Expansion **(c)** Contact discontinuity

Figure 4.14: The analytic solutions of the densities for the steady wave problems.

For each problem, a PINN is set up with 4 layers of 50 neurons with the `tanh` activation function. Except for the output neurons, where `linear` is used for the velocity components while `exp` is used for the

density and pressure to enforce positive values. Furthermore, 10,000 collocation points are sampled from the Hammersley sequence and 1,000 boundary points are linearly sampled per boundary. Again, the L-BFGS optimizer is used with a learning rate of 0.1. Note that Neumann conditions are not imposed, technically making the problems underdetermined. However, it is observed that including these conditions can worsen the solutions by complexifying the loss landscape mentioned in Section 2.2.2.



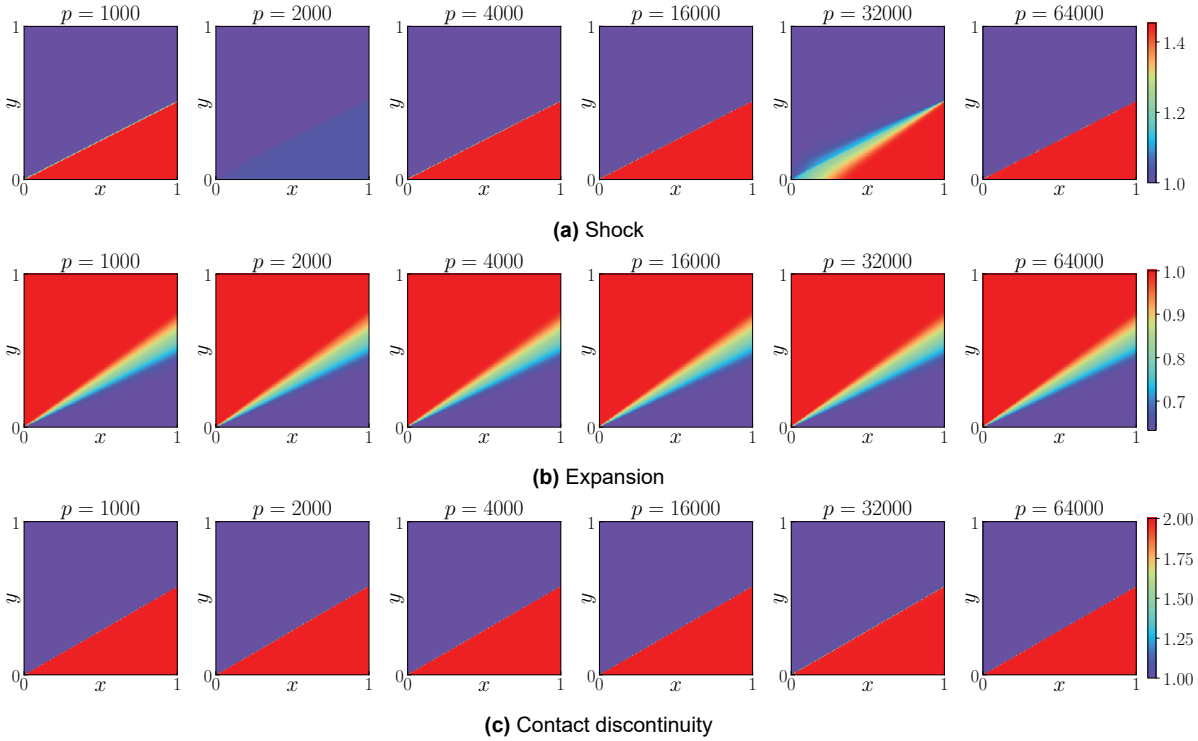
(a) Shock
$L_{2,\rho} = 7.2 \times 10^{-2}$

(b) Expansion
$L_{2,\rho} = 3.6 \times 10^{-3}$

(c) Contact discontinuity
$L_{2,\rho} = 1.2 \times 10^{-2}$

Figure 4.15: Density solutions of the steady wave problems without loss weighting or other adaptations.

Fortunately, Fig. 4.15b shows that PINNs correctly simulate the expansion wave even without a weighted loss. This makes sense because nonphysical ”expansion shocks” involve a negative entropy change which *also* conflicts with the conservation of entropy along streamlines, making them unfavorable compared to isentropic rarefaction waves. In other words, the entropy condition is satisfied naturally. Furthermore, Fig. 4.15a shows that the shock is partially modeled by the PINN, becoming more diffused towards the lower-left corner as is also observed by Mao et al. (2020). Compared to the Riemann problem, the shock follows more directly from the boundary conditions as there is no opportunity for the temporal failure mode discussed in Section 2.2.1 to manifest. Note that the $L_{2,\rho}$ error is fairly large because the shock angle is too small. Lastly, Fig. 4.15c confirms that contact discontinuities are not problematic for PINNs.



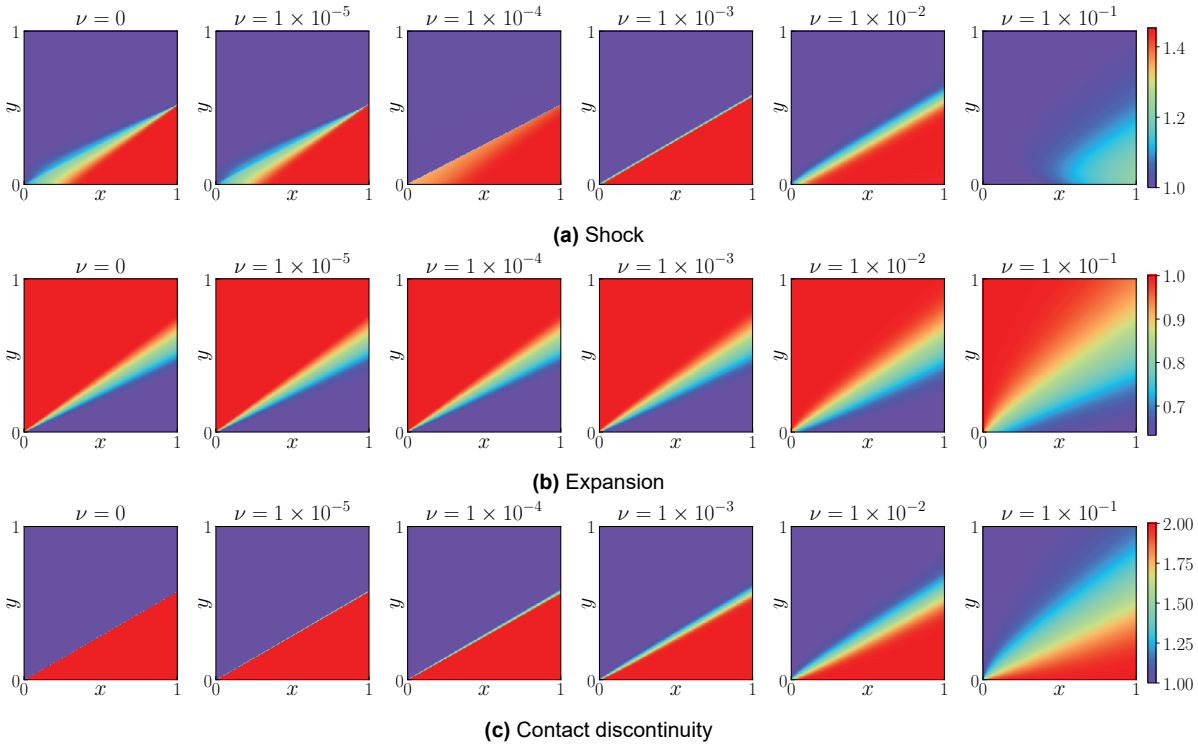
(a) Shock

(b) Expansion

(c) Contact discontinuity

Figure 4.16: Density solutions of the steady wave problems for different numbers of collocation points p.

Although the shock in Fig. 4.15a is diffused, Fig. 4.16a shows that the sharpness of the shock is sensitive to the number of collocation points. In reality, the PINN is likely sensitive to the location of the collocation points, as it is not always able to overfit and collapse to a sharp shock due to their paradoxical status discussed in Section 4.3. When it is not able to do so, it deals with the high residuals by underfitting the boundary conditions. Note that for $p = 2000$ a sharp shock is produced, but the jump relations are not satisfied correctly due to the PINN overfitting incorrectly. In addition, while the shocks are all reasonably sharp, keep in mind that the shock angles are all incorrect. While the PINNs are sensitive to the number of points on the shock problem, it has no substantial effect on the expansion and contact discontinuity problems, which further confirms the entropy failure mode.



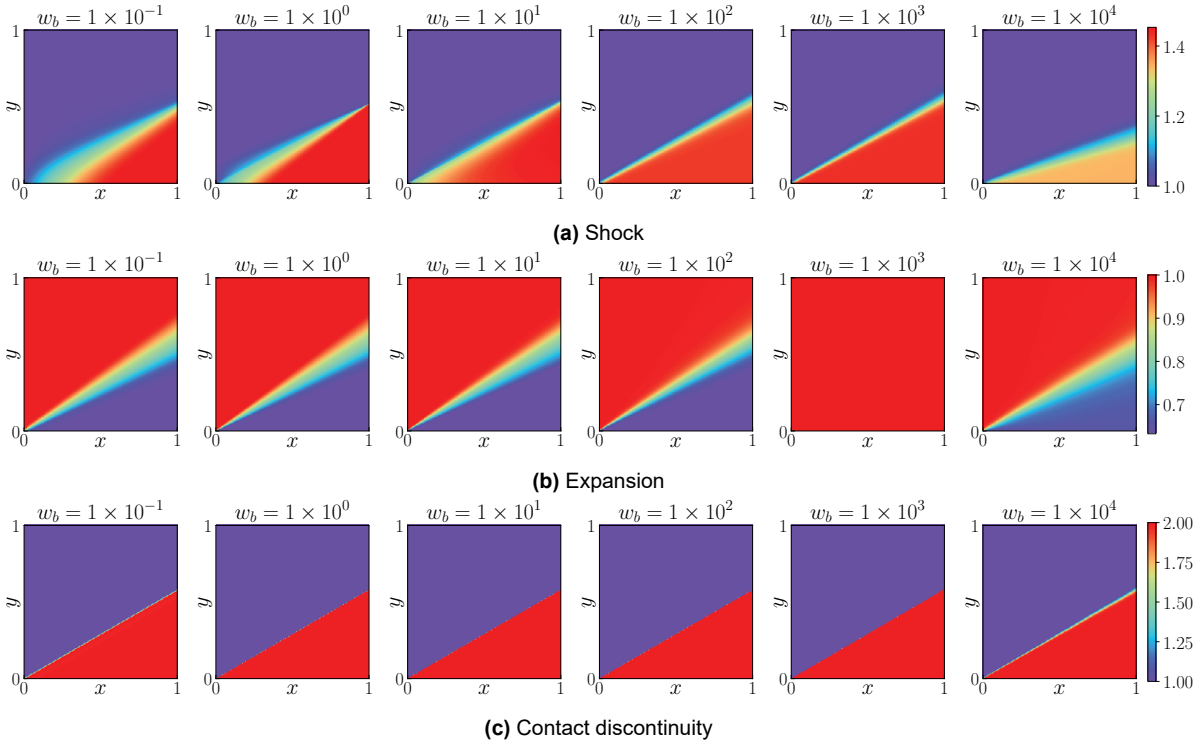
(a) Shock

(b) Expansion

(c) Contact discontinuity

Figure 4.17: Density solutions of the steady wave problems for different artificial viscosity levels ν.

To avoid diffused shock waves, it is possible to include viscosity as was done for the Riemann problem. Fig. 4.17 shows the effect of viscosity on each of the waves, confirming that isentropic regions of the flow such as expansion and contact discontinuities are negatively affected by viscosity. It also shows again how a large viscosity can negatively affect a shock, while a small viscosity has no effect. Interestingly, for $\nu = 1 \times 10^{-3}$ and $\nu = 1 \times 10^{-2}$ the shock is for the first time produced at the correct angle, yielding an accurate solution. Notice also that for $\nu = 1 \times 10^{-4}$ the shock is sharp, but the jump conditions are not satisfied correctly everywhere along the shock due to overfitting.



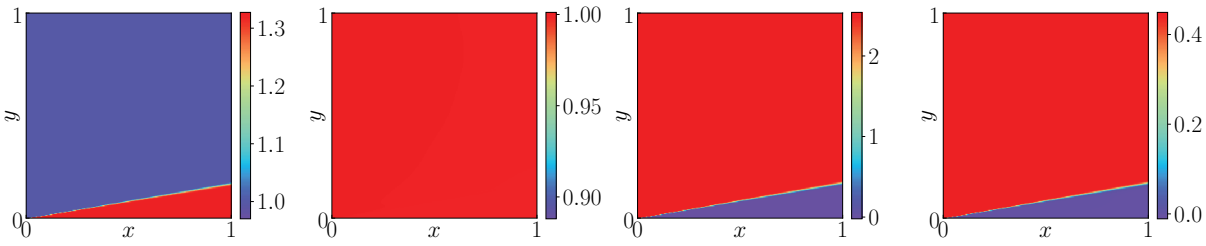
(a) Shock

(b) Expansion

(c) Contact discontinuity

Figure 4.18: Density solutions of the steady wave problems for different boundary condition loss weights w_b.

Since the diffused shock solution in Fig. 4.15a is underfitting the boundary conditions, it is also possible to sharpen it by increasing the boundary condition loss weight. Indeed, Fig. 4.18 confirms that the shock becomes sharper. Notice that the solution of each of the wave types starts to deteriorate for very large boundary condition loss weights, as satisfying the PDEs and therefore providing a physical solution is less important. Another phenomenon is seen in Fig. 4.18b for $w_b = 1 \times 10^3$, where the density seems to be uniform. Upon closer inspection of the solution in Fig. 4.19, something interesting can be noticed: a zero-velocity region is formed instead of an expansion fan.

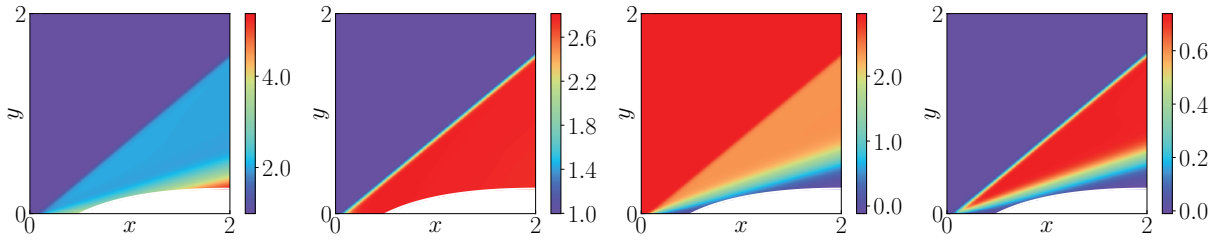


(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 4.19: Solution of the steady expansion problem for a boundary loss weight of $w_b = 1 \times 10^3$.

This phenomenon has also been observed by Laubscher et al. (2022) and Wassing et al. (2023) in highly compressible flows, particularly on skewed surfaces with a slip boundary condition. Laubscher et al. (2022) argue that this is the result of insufficient enforcement of the boundary conditions and show that it can be solved by not prescribing the ideal gas law explicitly but rather by including the equation as an additional term in the residual loss. They show that this approach successfully removed the zero-velocity region in front of wedges when simulating oblique shocks. On the other hand, Wassing et al. (2023) do not give much explanation and simply mention that it is a nonphysical solution. They show that by adding viscosity, the sharp interface between the zero-velocity and the rest of the flow is diffused and the nonphysical

solution is eliminated. However, they are only able to do so for subsonic flows. Upcoming chapters confirm that such phenomena can still occur despite the addition of viscosity, as shown in Fig. 6.11.



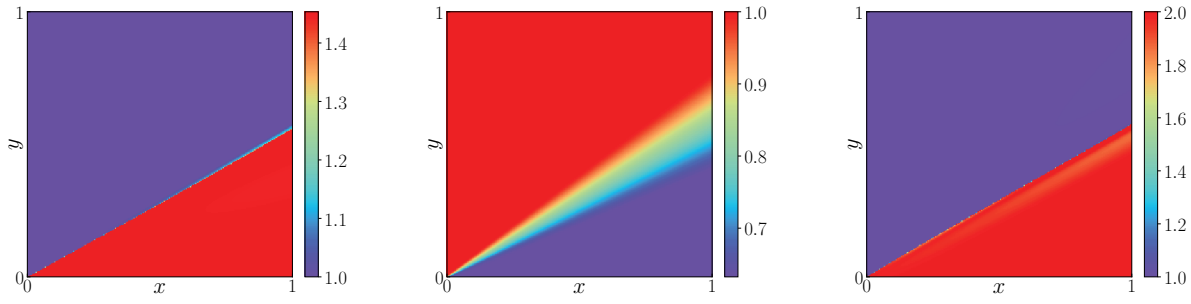
(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Preview of Figure 6.11: Solution of the PINN with global viscosity on the curved shock problem, with $w_\nu = 1 \times 10^1$.

While both methods work, they fail to address the underlying issue: the slip boundary condition is not properly designed for PINNs and can drive them into local minima. To understand this, consider the general form of the slip condition given in Eq. (4.11). While this condition can be satisfied by aligning the velocity with the surface angle θ, it is also satisfied by a trivial zero velocity, which is effectively the no-slip condition. While the latter solution to the boundary conditions is nonphysical, it is easier for the PINN to produce because it does not have to learn a dependency between the velocity components. As a result, PINNs have an initial tendency of producing a zero velocity region at surfaces. Note that in the case of the shock and expansion problems in Eq. (4.10) this phenomenon rarely happens because $\theta = 0°$, so that the slip condition simplifies to $v = 0$, which decouples the horizontal and vertical velocity components.

$$u \cos(\theta) + v \sin(\theta) = 0 \tag{4.11}$$

This initial tendency to produce a zero-velocity region drives PINNs into a local minimum, which can only be solved by forming a sharp interface as seen in Fig. 4.19. Although this solution is nonphysical and has high associated residuals, the PINN overfits and these high residuals become located in between collocation points. It is interesting to note that in Fig. 4.19, but also in the papers by Laubscher et al. (2022) and Wassing et al. (2023), the zero-velocity region effectively represents a ”fake object” surrounded by a combination of a shear wave and a contact discontinuity. The flow outside and inside this fake object is considerably simpler, allowing the PINN to reduce the residuals easier, which increases the attractiveness of the local minimum.

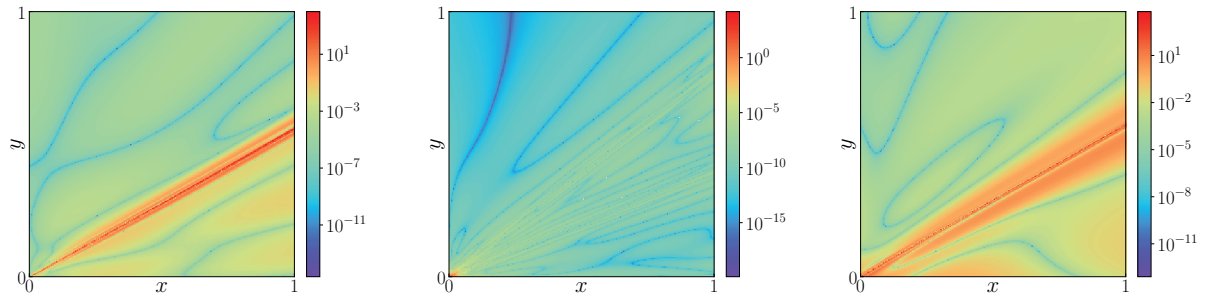


(a) Shock $L_{2,\rho} = 3.7 \times 10^{-2}$ **(b)** Expansion $L_{2,\rho} = 4.6 \times 10^{-3}$ **(c)** Contact discontinuity $L_{2,\rho} = 3.5 \times 10^{-2}$

Figure 4.20: Density solutions by a CPINN applied to the steady wave problems.

Given the interesting results of CPINNs on the Riemann problem, it is also worthwhile to apply them to the wave problems. Therefore, the previous PINN setup is used as the generator and combined with a discriminator consisting of 5 layers of 50 neurons with the `relu` activation function. Furthermore, it has a total of 8 or 9 output neurons with the `linear` activation function, one output for each boundary condition and PDE considered. The CPINNs are then trained using the ACGD optimizer with a learning rate of 0.001,

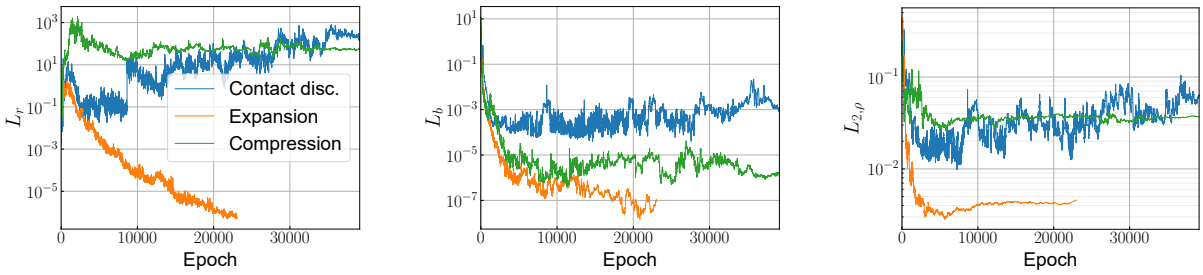
for a maximum time of 24 hours. Fig. 4.20 shows the results for each problem, revealing that CPINNs are able to produce a sharper shock than the PINN in Fig. 4.15. Nevertheless, keep in mind that PINNs can produce sharp shocks for different numbers of collocation points as shown in Fig. 4.16. Despite this, the shock angle is closer to the true wave angle compared to any of the weighted PINN solutions.



(a) Shock **(b)** Expansion **(c)** Contact discontinuity

Figure 4.21: Squared continuity residuals for a CPINN applied to the steady wave problems.

The CPINN solution to the expansion wave has a similar error as the PINN solution, but for the contact discontinuity, it is considerably worse. A possible explanation is the adversarial training style of CPINNs, which can cause the discriminator to exploit high gradients near the discontinuity to easily increase the residuals at collocation points, which creates a noisy convergence pathology. It is again interesting to note that for the solutions with discontinuities, CPINNs maintain high residuals instead of decreasing them as shown in Fig. 4.22a. Similar to the Riemann problem, the high residuals are located near the discontinuities as shown in Fig. 4.21. Furthermore, the error history in Fig. 4.22c shows that initially, the solution for the contact discontinuity is slightly better, but it deteriorates afterward. Based on the ability of CPINNs to underfit the residuals to circumvent the entropy failure mode, they remain interesting for shock capturing and will therefore also be assessed and compared on the applied cases in Chapter 6.



(a) Residual loss **(b)** Boundary condition loss **(c)** Density error

Figure 4.22: Loss and error pathology for the CPINN results in Fig. 4.20.

Closing remarks

Looking back at the research questions that this chapter intended to answer, it is now clear that PINNs fail on highly compressible problems due to two failure modes. The first, more fundamental entropy failure mode arises from the conservation of entropy along streamlines that is prescribed by the differential Euler equations. Unlike traditional solvers, PINNs do not use discretization and hence no numerical viscosity enters the PDEs. As a result, PINNs try to find an isentropic solution to a non-isentropic problem which leads to nonphysical phenomena. The second failure mode may arise on surfaces and is caused by the ambiguity of the slip boundary condition, which is also satisfied by a trivial zero velocity. The strong local minimum that emerges from this again leads to nonphysical phenomena, in particular fake objects that involve considerably simplified flows. While the symptoms of these failure modes have been observed in literature and adaptations have been proposed to alleviate them, the failure modes themselves are not explicitly recognized.

5

Shock-capturing methodology

Although the various adaptations in literature tackle the failure modes to a certain extent, they have not yet been successfully applied to more complex highly compressible problems. Essentially, there can be two reasons for this. Firstly, as mentioned, most adaptations are designed without correctly recognizing the failure modes. As a result, there might not be a proportional relationship between the parameters of the adaptation and the alleviation of the nonphysical phenomena, making it tedious for authors to obtain successful results. Secondly, the adaptations might simply not be sufficient to alleviate the failure modes. To holistically answer the last research question, namely what adaptations can alleviate the failure modes, both options must be considered. Therefore, new adaptations are proposed in Section 5.1 and Section 5.2 that may enhance the shock-capturing capabilities of PINNs by targeting the two failure modes. These are then assessed together with selected existing adaptations using a framework that is detailed in Section 5.3.

5.1 Local viscosity

As mentioned, the addition of artificial viscosity can alleviate the entropy failure mode and thereby improve the shock-capturing capabilities of PINNs. In particular, Section 4.3 has shown that a variable global viscosity together with a viscosity loss can achieve sharp shocks and relatively low artificial viscosity levels. However, global viscosity is indiscriminate in the sense that it affects the entire domain instead of only shocks. In particular, it can introduce errors in regions that should be isentropic, such as rarefaction waves or contact discontinuities. A natural solution is a local viscosity that only materializes near shocks, which has already been proposed by Coutinho et al. (2023). Their more general method involves some parametric map $\nu(\mathbf{x}, t, \boldsymbol{\theta})$, with $\boldsymbol{\theta}$ the parameters defining the map. However, as they mention themselves, an effective map requires foreknowledge of the solution. Therefore, they propose a residual-based method, however, it relies on the usage of a discretized mesh, which violates the fundamental principles of PINNs. Their first method is certainly a step in the right direction and serves as an inspiration for the proposed local viscosity method in Fig. 5.1, which harmonizes with the fundamental principles of PINNs.

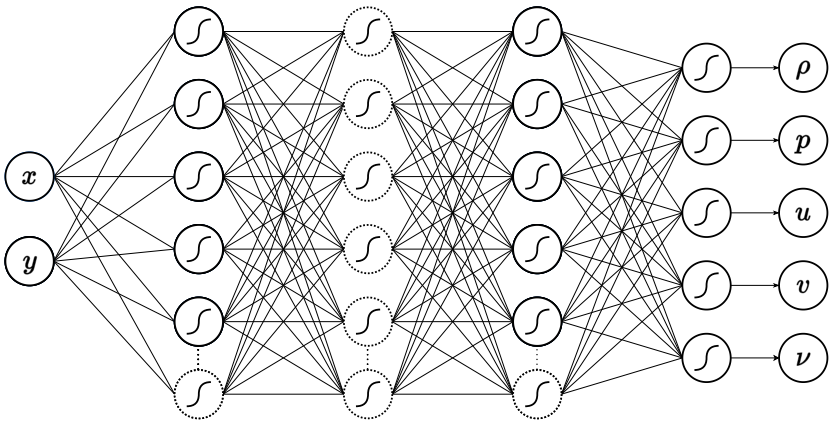
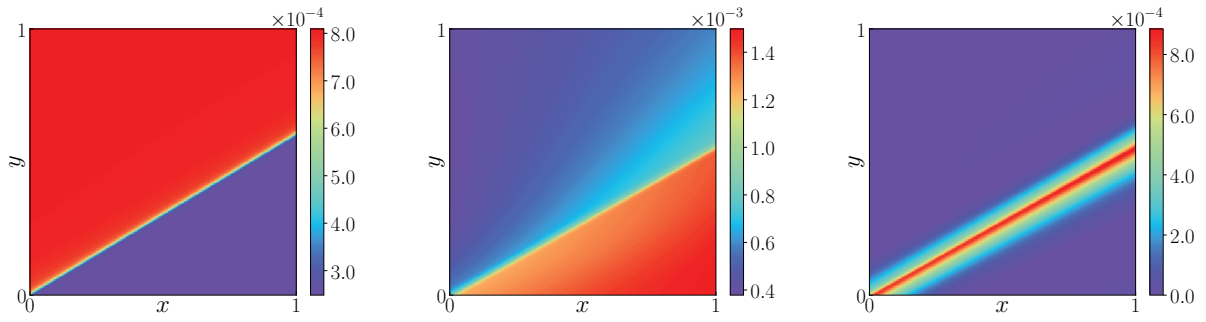


Figure 5.1: Architecture of a PINN with local viscosity for the two-dimensional Euler equations.

Essentially, the parameters of the network are used to define the parametric map. The local viscosity is

considered an additional output of the PINN, providing a continuous viscosity field. As a result, the same viscosity loss as in Eq. (4.9) can be used, but now averaged over the local values at the collocation points. Similar to the pressure and density, there are multiple activation functions that can enforce the positivity of the viscosity. For example, in the previous chapter, the exponential activation function is used for the pressure and density. Based on some empirical trials it is found that the pattern of local viscosity can vary drastically depending on the choice of the activation function and the viscous loss term in Eq. (4.9). For example, Fig. 5.2 shows that using an exponential activation function with $L_\nu = \nu^2$ produces viscosities that vary similarly to the other output variables. However, when a squared activation is used with $L_\nu = \nu$ the local viscosity becomes truly located at the shock. Fig. 5.2 also shows an interesting secondary benefit of the local viscosity method; it can be used as a shock detector as well, highlighting where shocks occur that might not be directly visible.



(a) `exp`, $L_\nu = \nu^2$, $w_\nu = 0.01$ **(b)** `square`, $L_\nu = \nu^2$, $w_\nu = 1$ **(c)** `square`, $L_\nu = \nu$, $w_\nu = 1$

Figure 5.2: Local viscosity solutions of the steady shock wave problem in Section 4.4, with an otherwise identical setup. The captions indicate the activation function, viscosity loss function and viscosity loss weight respectively.

Intuitively, the addition of an extra variable will increase the complexity of the output and therefore require a larger network. While this is true, realize that an optimal local viscosity that is only located at shocks highly correlates with the other output variables, implying that only limited additional approximation capacity may be necessary. This is a significant advantage of using the same neural network for the viscosity, instead of an additional smaller one.

$$L = L_0 + L_b + \lambda_r L_r + w_\nu L_\nu \tag{4.9 repeated}$$

5.2 Streamline representation

While Wassing et al. (2023) have shown that they can circumvent the slip failure mode by adding viscosity, this can again negatively affect isentropic regions of the flow. More importantly, the authors themselves note that the approach does not work on supersonic flows. To avoid the failure mode completely, the slip condition in Eq. (4.11) must be somehow reformulated so that reducing the velocity at the surface does not decrease the slip boundary loss. An intuitive approach is to normalize the slip condition by the velocity magnitude, as shown in Eq. (5.1). To avoid division by zero at stagnation points, a small constant ϵ must be added to the denominator. While this approach delays the initial formation of zero-velocity regions, they resurface after some time because decreasing the velocity magnitude without changing the flow angle still reduces the loss since $\epsilon > 0$.

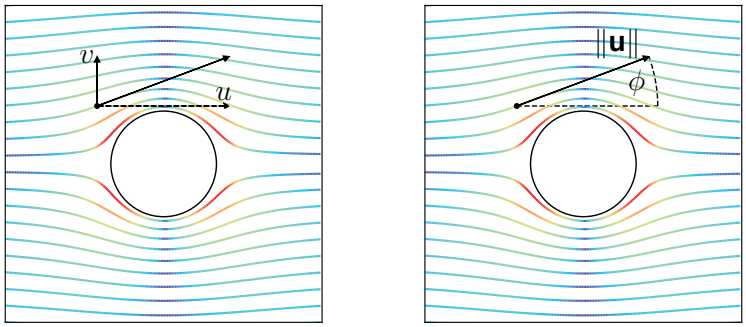
$$\frac{u\cos(\theta) + v\sin(\theta)}{||\mathbf{u}|| + \epsilon} = 0 \tag{5.1}$$

A better approach is to calculate the local flow angle $\phi = \operatorname{atan2}(v, u)$ at the slip surface and to constrain this angle instead, as done in Eq. (5.2). While division by zero is no longer an issue, atan is undefined when both $u = 0$ and $v = 0$. Since this can for example happen at the stagnation point on an object, this issue cannot be avoided. Even without stagnation regions in the target solution, this formulation can still cause gradient descent algorithms to fail due to large gradients, requiring them to be re-initialized.

$$\phi - \theta = 0 \tag{5.2}$$

While there are many other possible formulations of the slip boundary condition, it is more natural to consider a change from a component output representation to the streamline output representation visualized in Fig. 5.3. Instead of outputting the velocity components, the PINN can be designed to output the velocity magnitude $||\mathbf{u}||$ and direction ϕ instead, as shown in Fig. 5.4. As a result, Eq. (5.2) can be directly applied to the outputs of the PINN. Since the flow direction is now defined even when the velocity magnitude is zero, the previous numerical issues are avoided. Other boundary conditions can also be converted to a streamline form, although it is also possible to convert the streamline outputs back to the velocity components using Eq. (5.3). For the remainder of the thesis, the latter is done to isolate the effect of the reformulated slip boundary condition. To the author's best knowledge, there are currently no accounts of using PINNs with a streamline output representation, especially in the context of the Euler or Navier-Stokes equations.

$$\begin{aligned} u &= \|\mathbf{u}\| \cdot \cos(\phi) \\ v &= \|\mathbf{u}\| \cdot \sin(\phi) \end{aligned} \tag{5.3}$$



(a) Component representation **(b)** Streamline representation

Figure 5.3: Comparison of different output representations.

Although the streamline output representation might avoid the zero-velocity phenomenon, note that it no longer penalizes the magnitude of the normal velocity on a slip surface. In other words, large normal velocities are penalized just as much as small normal velocities, which can possibly give rise to new failure modes. In addition, the magnitude of the streamline slip loss in Eq. (5.2) is lower than the component slip loss in Eq. (4.11) if the velocity magnitude is large. Therefore, it is likely that the streamline slip condition loss requires a larger weight to remain balanced with the other loss terms.

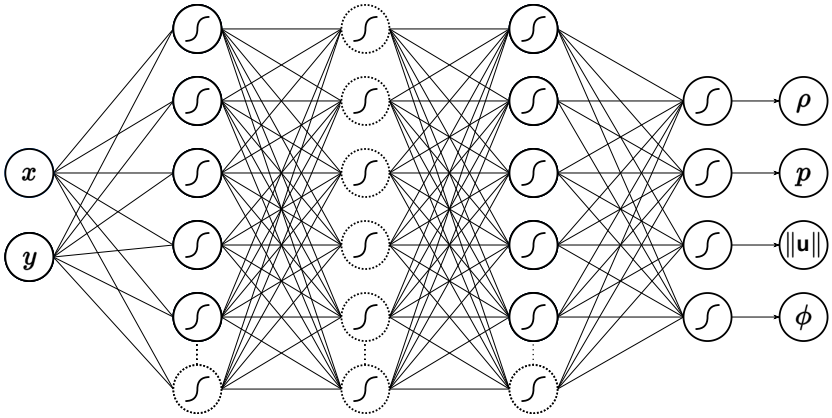


Figure 5.4: Architecture of a PINN with a streamline output representation.

Again, there are many possibilities in defining the activation functions of $\|\mathbf{u}\|$ and ϕ. While it may seem logical at first to apply the `exp` activation function to the magnitude and a multiple of `tanh` to the direction

as in Eq. (5.4), this assumes knowledge of the flow direction along the surface to evaluate Eq. (5.2). Even though the flow along an object is mostly in the same general direction as the free-stream flow, local backflow might occur especially for objects at high angles of attack.

$$\|\mathbf{u}\| : f(x) = \exp(x), \quad \phi : f(x) = \pi \tanh(x) \tag{5.4}$$

Therefore, it is better to apply the `linear` activation function the magnitude and to restrict the flow direction to only one side of the unit circle to avoid redundancy, as in Eq. (5.5).

$$\|\mathbf{u}\| : f(x) = x, \qquad \phi : f(x) = \frac{\pi}{2} \tanh(x) \tag{5.5}$$

5.3 Assessment framework

To assess and compare the shock-capturing capabilities of the above adaptations, PINNs and CPINNs will be applied to different steady shock problems. In other words, the steady Euler equations in Eq. (5.6) will be considered. Again, the rationale is that it can be expected that the flow around an object will converge to a steady solution when well-posed and time-independent boundary conditions are considered. While traditional solvers often make use of this assumption to obtain a steady solution through a time-marching technique, solving the unsteady equations to obtain a steady solution unnecessarily increases the computational and memory requirements for PINNs. In addition, this can unnecessarily give rise to the temporal failure mode which is mentioned in Section 2.2.1.

$$\begin{aligned} \nabla \cdot (\rho \mathbf{v}) &= 0, \\ \nabla \cdot (p + \rho \mathbf{v}\mathbf{v}^{\mathsf{T}}) &= 0, \\ \nabla \cdot ((p + \rho E)\mathbf{v}) &= 0. \end{aligned} \tag{5.6}$$

Just like all simulations so far, the ideal gas law will be used for the equation of state to relate the energy, pressure and density as in Eq. (3.4). The steady shock problems that are considered are, in increasing order of difficulty, an oblique shock, a curved shock and a detached shock. The setup of these problems will be discussed in the next chapter.

5.3.1 Cases

For each problem, a total of twelve (C)PINN setups will be considered as shown in Table 5.1. By considering these cases, it is possible to distinguish the effects of viscosity, local viscosity and the streamline output representation. In addition, it allows for the comparisons of PINNs and CPINNs. Note that CPINNs are not combined with local viscosity, as their adversarial loss requires the discriminator to predict the difference between the generated viscosity and a target value, which is zero. Since the viscosity is strictly positive, the generator can only gain an advantage by minimizing the viscosity and therefore it is quickly nullified. Several attempts have been made to overcome this issue, for example by using the raw viscosity before the positive activation function for the adversarial loss or simply allowing the viscosity to be negative, but these were unsuccessful.

Table 5.1: Considered cases of PINN adaptations for assessment and comparison.

No.	1	2	3	4	5	6	7	8	9	10	11	12
Type	PINN								CPINN			
Representation	Component				Streamline				Component		Streamline	
Viscosity	No	Fixed	Global	Local	No	Fixed	Global	Local	No	Fixed	No	Fixed

The above cases are initialized identically for the considered shock problems. In other words, the network architecture, initial network parameters and collocation points are identical. In the case of the CPINN, the generator is initialized identically to the other PINN networks. Each case has a single hyperparameter that will be varied to obtain the best results in terms of the $L_{2,\rho}$ error: the boundary condition loss weight w_{bc} for the inviscid cases, the viscosity ν for the fixed viscosity cases and the viscosity loss weight w_ν for the global and local viscosity cases. For the inviscid CPINN simulations, no hyperparameter is considered

because the adversarial training style provides a weighting mechanism of its own. While it is certainly possible to obtain better results by also varying the boundary condition loss weights for the PINNs with viscosity, the resulting optimization space is simply too large. Note that for the global viscosity cases $L_\nu = \nu^2$ is used, while for the local viscosity cases $L_\nu = \nu$ is used based on the findings in Section 5.1.

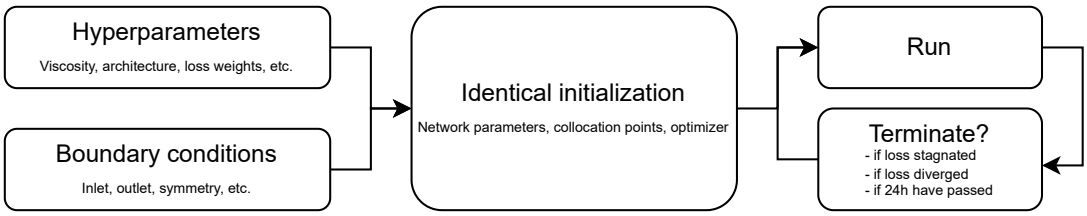


Figure 5.5: Overview of the assessment framework.

Furthermore, for all cases, the `exp` activation function is used for the pressure and density to enforce positive values. In addition, the `linear` activation function is used for the velocity components while the activation functions in Eq. (5.5) are used for the streamline components. Last but not least, the `square` activation function is used for the local viscosity. The internal activation functions and the network architecture are discussed in the next chapter as they differ for each problem considered. The PINNs are optimized with the second-order L-BFGS optimizer with a learning rate of 0.1, although Adam is sometimes used initially until the gradients are stable. The CPINNs are optimized with the second-order ACGD optimizer with a learning rate of 0.001. Simulations are terminated when a time limit of 24 hours has been exceeded, which only occurs when training CPINNs, or when the loss has stagnated. An overview of the assessment framework is shown in Fig. 5.5.

5.3.2 Software and hardware

All simulations are performed using PyTorch (Paszke et al., 2019) for Python, which has built-in automatic differentiation. Note that a custom implementation of ACGD has been developed for the optimization of CPINNs, which improves over existing implementations in terms of memory and speed. Simulations are performed on different devices, in particular an RTX A1000 on a laptop, a Quadro RTX 8000 on a workstation and a NVIDIA Tesla V100S 32GB on the DelftBlue cluster (DHPC, 2022). As a result, the training times cannot be directly compared, although this is not a significant issue since the focus is on assessing the qualitative shock-capturing capabilities in terms of accuracy. Furthermore, the results across the different devices might differ since PyTorch does not guarantee determinism and might even sample parameters and collocation points differently across devices. Therefore, groups of simulations are performed on the same device as much as possible to not affect the comparisons significantly.

Source code
The source code for a selection of the results will be made available at `https://github.com/wagenaartje/pinn4hcf`. The custom implementation of ACGD can be found at `https://github.com/wagenaartje/torch-cgd`. For specific results, feel free to contact the author via e-mail at `t.wagenaar-1@student.tudelft.nl`.

6 Results

In this chapter, the methodology in the previous chapter will be used to assess and compare existing and newly proposed adaptations to answer the last research question, namely what adaptations can alleviate the highly compressible failure modes. For this purpose, problems involving an oblique shock, a curved shock and a detached shock are considered, which increase in difficulty respectively. To the author's best knowledge, there are currently no examples in literature featuring steady curved or detached shocks. Based on the results, an answer to the last research question will be provided at the end of this chapter.

6.1 Oblique shock wave

Although an oblique shock has already been treated in Section 4.4, this is a rather simple case for two reasons. Firstly, the surface over which the shock is created is horizontal, which exempts PINNs from having to learn the relationship between the velocity components to satisfy the slip condition. Secondly, the shock emerges directly from the boundary conditions at the inlet, which avoids PINNs having to determine where a shock should be located. Therefore, the more complex case in Fig. 6.1 is considered where a wedge is placed at a nonzero distance from the inlet. A similar case has been considered by Laubscher et al. (2022), who observed the zero-velocity phenomenon and subsequently alleviated it by adding the ideal gas law as an additional residual equation. However, these results could not be reproduced.

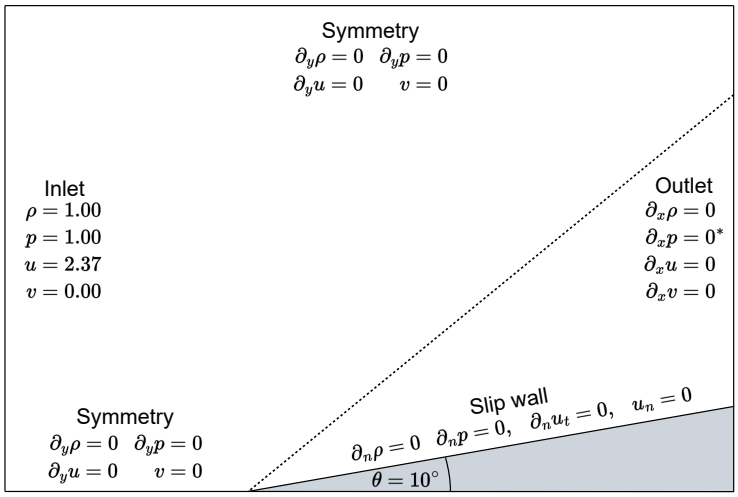
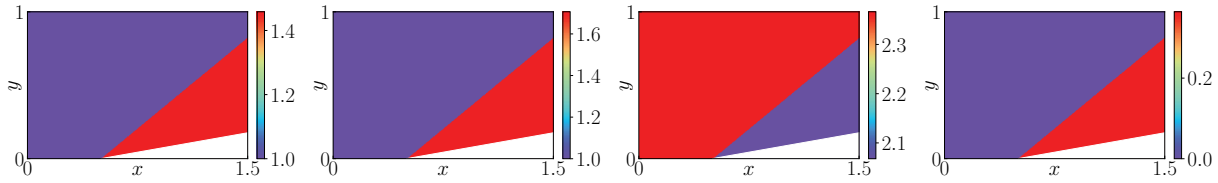


Figure 6.1: Domain and boundary conditions for the oblique shock wave problem. The Mach number is 2.0 and the domain size is $[0, 1.5] \times [0, 1]$ with the wedge starting at $x = 0.5$.

The solution to the problem in Fig. 6.1 can be determined analytically through the equations in Section 3.3. Since the wedge angle is below the maximum corner angle θ_{max}, the shock remains attached to the tip of the wedge. Using Eq. (3.10) with $\theta = 10°$, $M_1 = 2.0$ and $\gamma = 1.4$ gives that the shock angle is $\beta = 39.31°$. Furthermore, the jump relations in Eq. (3.9) give $\rho_2 = 1.46$, $p_2 = 1.71$, $u_2 = 2.07$ and $v_2 = 0.37$. The full solution is shown in Fig. 6.2. Note that Eq. (3.10) also gives a strong shock solution,

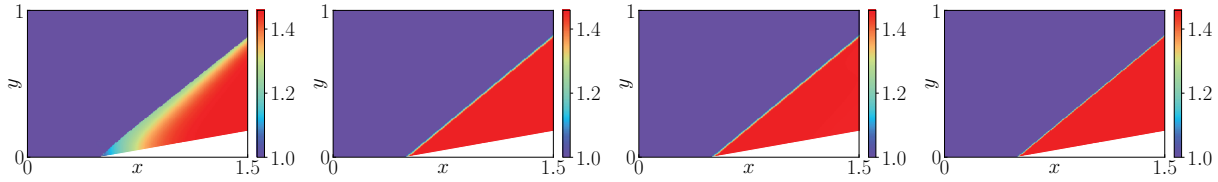
which as mentioned only appears in nature when a strong backpressure is imposed. It is likely that the tendency of generalization of PINNs will result in weak shock solutions as they involve smaller jumps in variables.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.2: Analytical solution of the oblique shock problem in Fig. 6.1.

For each PINN case in Table 5.1, 10,000 collocation points are sampled using the Hammersley sequence and 750 boundary conditions points are linearly sampled per boundary. Collocation points inside the wedge are removed, so the effective amount is slightly smaller. The network architecture consists of 4 layers of 50 neurons, each with the `tanh` activation function. Furthermore, the Neumann boundary conditions in Fig. 6.1 are ignored and so are the Dirichlet boundary conditions at the top symmetry plane. While excluding these boundary conditions certainly affects the results and technically makes the problem underdetermined, it does not considerably affect the comparison between the different adaptations. Last but not least, note that the Neumann outlet condition for the pressure is locally converted to a Dirichlet boundary condition prescribing $p = 1.00$ when the flow is locally subsonic. Such a dynamic approach to transonic outlet boundary conditions in PINNs has not yet been treated in literature and is therefore novel in its own right. However, it is also highly experimental because it can create a non-differentiable loss landscape that breaks memory-based optimizers such as L-BFGS, necessitating the usage of Adam until the solution at the outlet has stabilized.



(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

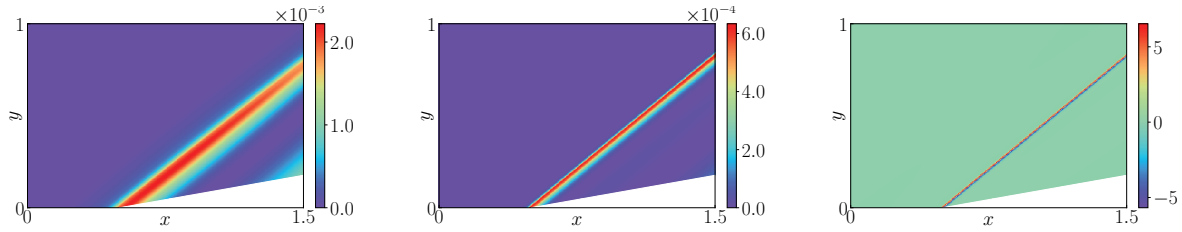
Figure 6.3: Density solutions for the oblique shock problem by PINNs with different viscosity methods.

Table 6.1: Errors and viscosity levels of the various PINN solutions for the oblique shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+1	4.55e-2	7.00e-2	1.51e-2	1.82e-1	-
PINN + viscosity	$\nu =$ 1e-3	1.68e-2	2.38e-2	5.68e-3	8.97e-2	1.0e-3
PINN + global viscosity	$w_\nu =$ 1e+2	1.66e-2	2.34e-2	5.64e-2	8.93e-2	9.1e-4
PINN + local viscosity	$w_\nu =$ 1e+1	**1.58e-2**	**2.22e-2**	**5.27e-3**	**8.47e-2**	**2.5e-5**

Fig. 6.3 shows that PINNs can create a shock without a zero-velocity region given a sufficiently large boundary condition weight. This is likely because the constant surface inclination makes it easier to produce the correct solution to the slip condition. In addition, previous and upcoming results show that zero-velocity regions are naturally wedge-like, making them unlikely to form over actual wedges. Nevertheless, the accuracy of the shock drastically improves when viscosity is added, as shown in Table 6.1. It is interesting to note that the optimal viscosity level is found to be $\nu = 1 \times 10^{-3}$, which is similar to the optimal values observed in Chapter 4. This is likely because approximately the same collocation point

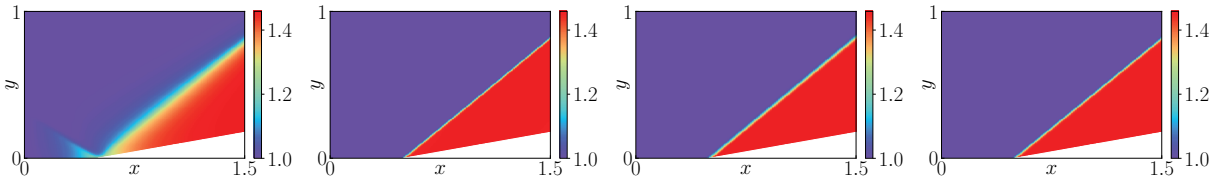
density has been used. Furthermore, the table shows that a global variable viscosity results in a lower viscosity level, although the errors hardly change. Using local viscosity results in an even better solution and the average viscosity decreases drastically. In general, it is interesting to note that the addition of viscosity allows an accurate solution to be formed without a large boundary condition weight, implying that it simplifies the loss landscape by making the problem more well-posed.



(a) Early local viscosity **(b)** Final local viscosity **(c)** Final inviscid continuity residual

Figure 6.4: Local viscosity evolution of a PINN with $w_\nu = 1 \times 10^1$ and the resulting inviscid continuity residual.

The local viscosity method is not considerably more accurate than the other viscosity methods despite its drastically lower average viscosity because the viscosity at the shock itself has a similar magnitude as the other viscosity methods, as shown in Fig. 6.4b. While the viscosity is zero outside the shock, this has virtually no effect on the error since the solution is constant in these regions implying that the second derivatives are zero. As a result, viscosity does not enter the PDEs outside of a correctly modeled oblique shock. This can be confirmed by calculating the inviscid residuals, which are the residuals of the PDEs without the viscosity term. Fig. 6.4c shows indeed that these are only significant around the shock, indicating that elsewhere viscosity plays a negligible role. Note that the residuals vary from positive to negative along the streamwise direction because the density increases across the shock, so that $\partial_{xx}\rho > 0$ before the shock and $\partial_{xx}\rho < 0$ after the shock.



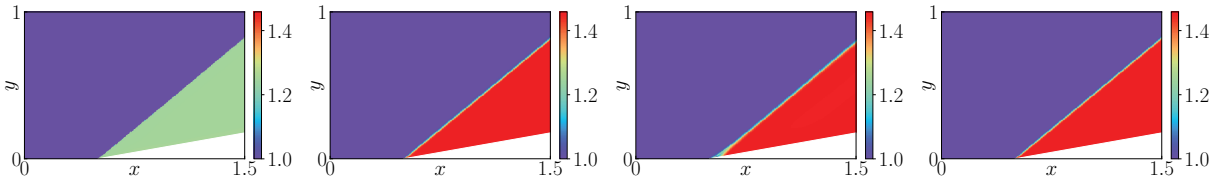
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.5: Density solutions for the oblique shock problem by streamline PINNs with different viscosity methods.

Table 6.2: Errors and viscosity levels of the various streamline PINN solutions for the oblique shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+3	4.68e-2	6.94e-2	1.69e-2	2.32e-1	-
PINN + viscosity	$\nu =$ 1e-3	**1.53e-2**	**2.24e-2**	**4.80e-3**	**7.62e-2**	1.0e-3
PINN + global viscosity	$w_\nu =$ 1e-1	2.05e-2	2.90e-2	6.76e-3	1.09e-1	1.8e-3
PINN + local viscosity	$w_\nu =$ 1e+0	1.87e-2	2.77e-2	5.69e-3	9.15e-2	**9.3e-5**

Fig. 6.5 and Table 6.2 show the results for the PINNs with a streamline output representation, which do not differ significantly from the results for the PINNs with a component output representation. This is expected to some extent because the streamline output representation is designed to tackle the zero-velocity phenomenon, which is not observed in the latter results. Furthermore, it is interesting to note that the viscosity levels for global and local viscosity in Table 6.2 are considerably higher than those in Table 6.1, even though the error of the solution is not considerably worse. It is therefore likely that the streamline PINNs satisfy the slip condition better leading to a slightly better positioning of the shock.



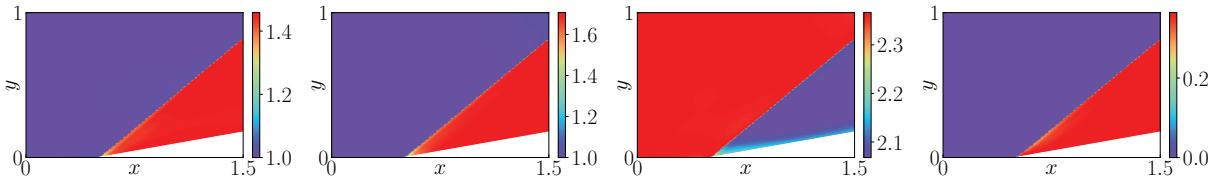
(a) Component & no viscosity **(b)** Component & viscosity **(c)** Streamline & no viscosity **(d)** Streamline & viscosity

Figure 6.6: Density solutions for the oblique shock problem by CPINNs with different adaptations.

Table 6.3: Errors and viscosity levels of the various CPINN solutions for the oblique shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$
CPINN	-	8.98e-2	2.06e-1	5.13e-2	1.35e-1
CPINN (lowest $L_{2,\rho}$)	-	1.57e-2	2.26e-2	8.10e-3	7.98e-2
CPINN + viscosity	$\nu =$ 1e-3	1.68e-2	2.45e-2	5.16e-3	8.27e-2
CPINN + streamline	-	5.19e-2	8.60e-2	1.67e-2	3.13e-1
CPINN + streamline + viscosity	$\nu =$ 1e-3	**1.51e-2**	**2.22e-2**	**4.62e-3**	**7.49e-2**

Last but not least, the previous PINN architecture is used as the generator network in a CPINN setup together with a discriminator architecture consisting of 5 layers of 50 neurons with the `relu` activation function. The discriminator has a total of 9 outputs with the `linear` activation function, consisting of one output for every boundary condition and PDE considered. Fig. 6.6 and Table 6.3 show an interesting difference compared to PINNs, namely that a sharp shock is produced even in the absence of viscosity, although the resulting jump in variables is not always correct as shown in Fig. 6.6a. The addition of viscosity again drastically improves the final results, but no improvement is obtained compared to PINNs with viscosity, although significantly more computational time is required. Note that without viscosity, CPINNs might produce better results earlier on in the training process as they underfit the residuals at this point. For example, Fig. 6.7 shows the checkpoint of the CPINN at which $L_{2,\rho}$ is smallest, which has similar errors as the final viscous solutions.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.7: Lowest $L_{2,\rho}$ solution obtained by the CPINN without viscosity on the oblique shock problem.

6.2 Curved shock wave

Although it is impressive that PINNs can simulate an oblique shock wave despite the entropy failure mode, analytical solutions already exist for such problems and the shocks around practical supersonic objects are usually not oblique. Therefore, it is more interesting to consider curved shocks, which can for example appear around curved supersonic objects. These are more complicated to model by PINNs as the variables after the shock are not homogeneous. In particular, the curved shock has a varying strength which results in a non-homentropic rotational flow behind the shock. To assess the shock-capturing capabilities of PINNs with and without the considered adaptations, the problem in Fig. 6.8 is considered.

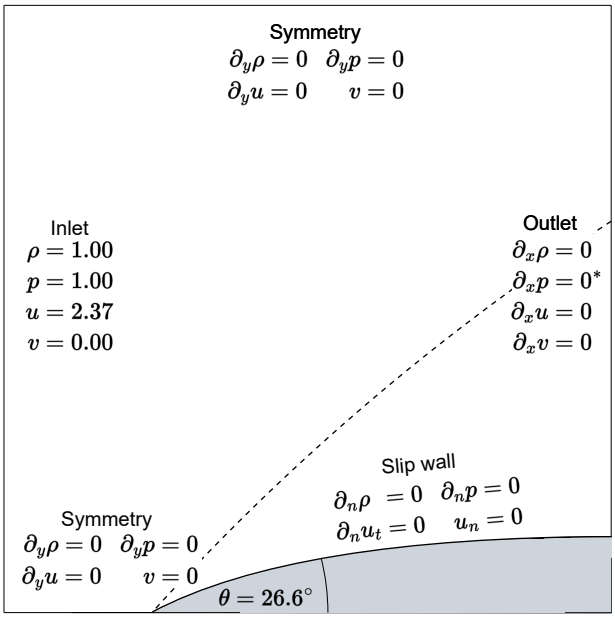
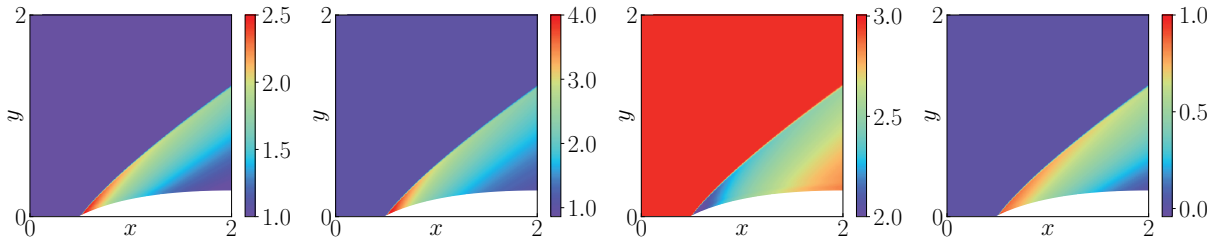


Figure 6.8: Domain and boundary conditions for the curved shock problem. The Mach number is 2.5 and the domain size is $[0, 2] \times [0, 2]$ with the object starting at $x = 0.5$.

The shape of the object is defined by a quadratic Bézier curve defined in Eq. (6.1) with points $\mathbf{P}_0 = (0.5, 0)$, $\mathbf{P}_1 = (1.0, 0.25)$ and $\mathbf{P}_2 = (2.0, 0.25)$. The advantage of a Bézier curve is that its position and slope are continuously defined, allowing it to be discretized to any desired degree. The surface inclination at the tip of the chosen Bézier curve is $\theta = 26.6°$, which is below θ_{max} so that the shock remains attached. Since the surface inclination decreases in the streamwise direction, so does the shock wave angle, resulting in a curved shock. While it is possible to solve these flows using fast methods such as the (rotational) method of characteristics (Shapiro, 1976), not many public solvers are available. Therefore, a reference solution is simulated via OpenFOAM (Weller et al., 1998), for which setup details are given in Appendix A.

$$\mathbf{B}(z) = (1-z)^2 \mathbf{P}_0 + 2(1-z)z\mathbf{P}_1 + z^2 \mathbf{P}_2, \quad 0 \leq z \leq 1 \tag{6.1}$$

The OpenFOAM solution is shown in Fig. 6.9, showing a sharp shock followed by a steady expansion due to the receding angle of the object. The solution is not perfect, partly because the solver has not converged to zero residuals but also because of the numerical viscosity resulting from the discretization schemes as discussed in Section 4.3. Nevertheless, it is important to reiterate that to the author's best knowledge, there are currently no documented PINN solutions for steady curved shocks. Therefore, the reference solution is sufficient to assess shock-capturing capabilities on a qualitative level. However, do keep in mind that the L_2 errors will be calculated with respect to these imperfect reference solutions and it is therefore not useful to compare errors below a certain threshold.

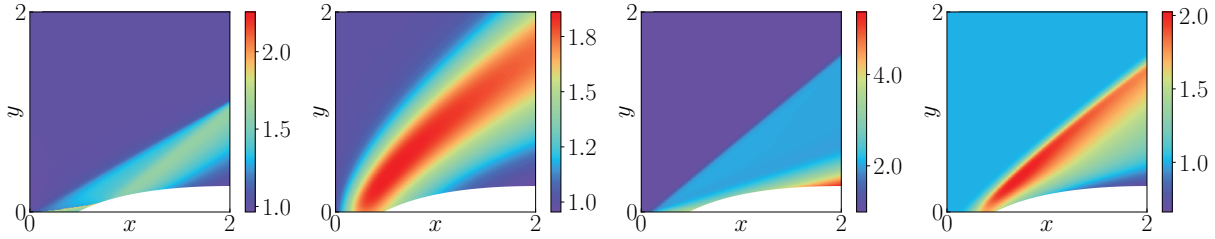


(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.9: Reference solution to the curved shock problem in Fig. 6.8.

The base PINN setup consists of 10,000 collocation points that are sampled using the Hammersley sequence, any points inside the object are removed. Furthermore, 500 boundary conditions points are

linearly sampled per boundary, except for the object. For the object, 1,000 points are sampled by linearly sampling the Bézier curve parameter z, which results in a slightly higher density of points near the tip of the object. Although the point nonuniformness is small, it can improve the obtained results because the tip angle completely determines the initial shock angle and thus if the shock remains attached or not. The network architecture again consists of 4 layers of 50 neurons, each with the `tanh` activation function. Furthermore, the Neumann boundary conditions as well as the Dirichlet boundary conditions of the top symmetry plane and possibly subsonic outlet are ignored, unless specified otherwise.



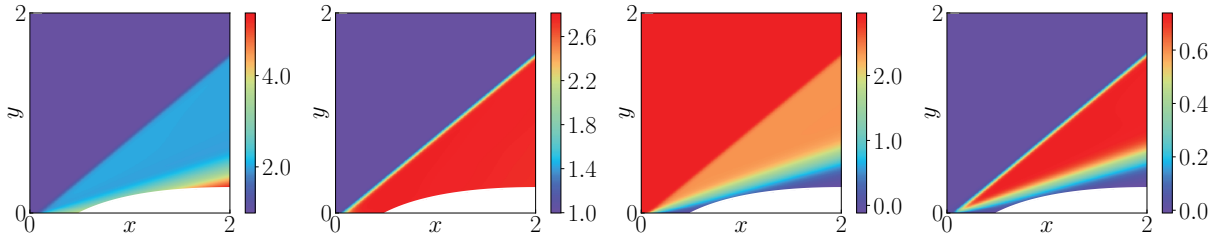
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.10: Density solutions for the curved shock problem by PINNs with different viscosity methods.

Table 6.4: Errors and viscosity levels of the various PINN solutions for the curved shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+0	**1.65e-1**	**2.69e-1**	**7.74e-2**	**5.72e-1**	-
PINN + viscosity	$\nu =$ 1e-1	3.80e-1	7.73e-1	2.39e-1	1.39e+0	1.0e-1
PINN + global viscosity	$w_\nu =$ 1e+1	5.45e-1	5.52e-1	2.24e-1	1.08e+0	6.0e-3
PINN + local viscosity	$w_\nu =$ 1e-1	2.29e-1	3.54e-1	9.01e-2	7.89e-1	**4.3e-3**

Fig. 6.10 shows the best results for PINNs without the streamline output representation, which reveals zero-velocity regions for the inviscid method and the global viscosity method. In Fig. 6.10a a small zero-velocity region occurs in front of the tip, which results in an oblique shock followed by a curved shock over the object. However, a much larger zero-velocity region forms in Fig. 6.10c. Upon closer inspection of this solution in Fig. 6.11, it even shows that a correctly angled oblique shock forms over the zero-velocity region. Note that this zero-velocity region is larger than the one observed in Fig. 6.10a because the pressure outlet condition is not satisfied, as the flow is subsonic and the Dirichlet boundary condition is not enforced. When enforcing the outlet condition, a zero-velocity region only forms at the tip. In general, it is observed that including the Dirichlet outlet condition in case the outlet flow is locally subsonic does not transform an incorrect solution into a correct solution.

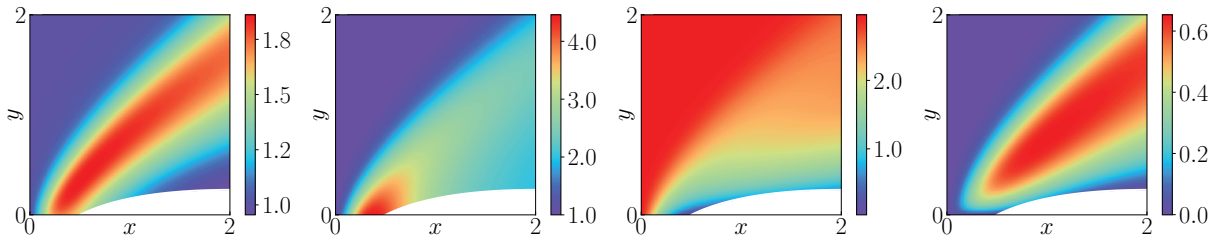


(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.11: Solution of a PINN with global viscosity on the curved shock problem, for $w_\nu = 1 \times 10^1$.

Zero-velocity regions do not form in Fig. 6.10b and Fig. 6.10d because the (local) viscosities are very large, on the order of 10^{-2} to 10^{-1}. This confirms the findings of Wassing et al. (2023), who show that

the zero-velocity region can be avoided by adding a sufficient amount of viscosity. However, in the case of the curved shock, the level of viscosity must be so large that the entropy solution is not approximated properly and the errors in Table 6.4 show only a slight improvement. Interestingly, Fig. 6.12 shows that the viscosity forms a boundary layer that creates a fictive, blunter object, resulting in a detached shock over the actual body. Therefore, it can be concluded that a low viscosity fictively sharpens the object, while a large viscosity fictively blunts the object.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.12: Solution of a PINN with fixed viscosity on the curved shock problem for $\nu = 1 \times 10^{-1}$.

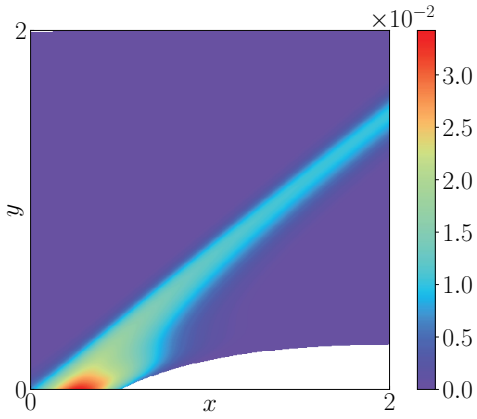
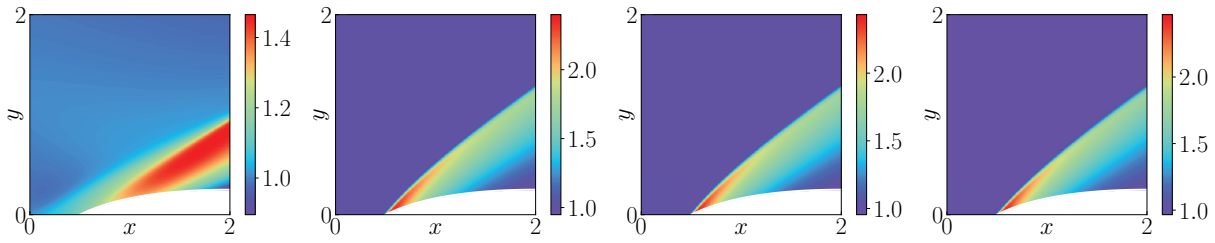


Figure 6.13: Local viscosity profile of the PINN with local viscosity for $w_\nu = 1 \times 10^{-1}$.

Interestingly, Fig. 6.13 reveals that the PINN with local viscosity does not only produce a large viscosity at the shock but also in the region in front of the tip. This is likely related to the symmetry conditions, which prescribe that $v = 0$ at the centerline even though $u = 0$ is prescribed at the stagnation point of the object on that same centerline. In other words, the flow must somehow decelerate from free-stream conditions to stagnation conditions without escaping upwards, which violates continuity. In reality, the centerline is infinitely thin and the stagnation point is infinitely small, but this might be hard for the PINN to model. Instead, the PINN opts to produce a large local viscosity so that stagnation is achieved through heat dissipation instead.



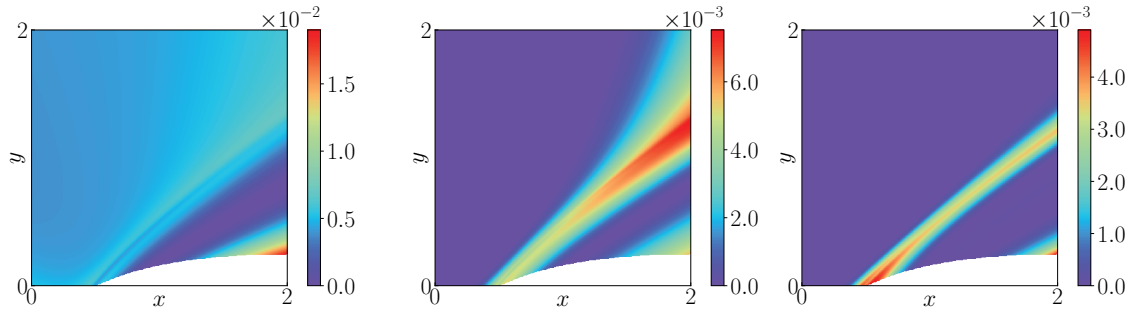
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.14: Density solutions for the curved shock problem by streamline PINNs with different viscosity methods.

Table 6.5: Errors and viscosity levels of the various streamline PINN solutions for the curved shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e-1	2.01e-1	3.26e-1	5.47e-2	7.01e-1	-
PINN + viscosity	$\nu =$ 5e-3	3.89e-2	6.41e-2	8.53e-3	1.21e-1	5.0e-3
PINN + global viscosity	$w_\nu =$ 1e-1	**3.45e-2**	**5.68e-2**	**7.64e-3**	**1.07e-1**	**5.2e-3**
PINN + local viscosity	$w_\nu =$ 1e-2	4.91e-2	8.01e-2	1.09-2	1.55e-1	**5.2e-3**

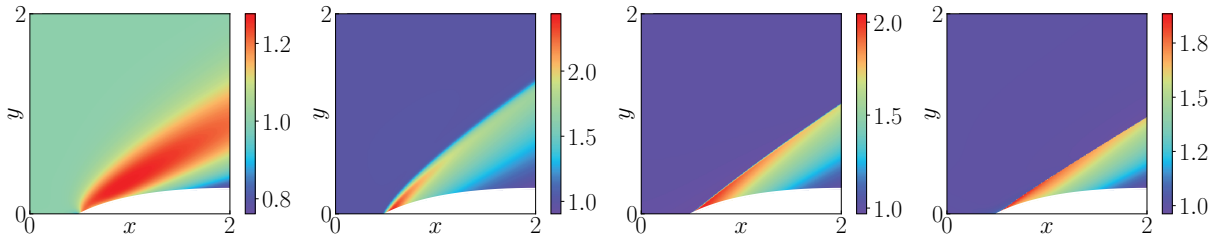
In contrast to plain PINNs, the results for the streamline PINNs in Fig. 6.14 are considerably better and show that the curved shock is accurately captured when viscosity is included. Even when viscosity is not included, the zero-velocity region is no longer present. Interestingly, the most accurate result is produced by the global viscosity method instead of the local viscosity method. Upon inspecting the profile of the local viscosity in Fig. 6.15a, it appears that viscosity is also located at the bottom right corner instead of only at the shock. However, this phenomenon lessens when the viscosity loss weight is increased, as shown in Fig. 6.15c. In other words, the viscosity at the corner is not necessary to capture the shock but it is likely a result of the correlation of the viscosity output to the other network outputs.



(a) $w_\nu = 1 \times 10^{-2}$ **(b)** $w_\nu = 1 \times 10^{-1}$ **(c)** $w_\nu = 1 \times 10^{0}$

Figure 6.15: Local viscosity profile for the streamline PINN with local viscosity for different viscosity loss weights.

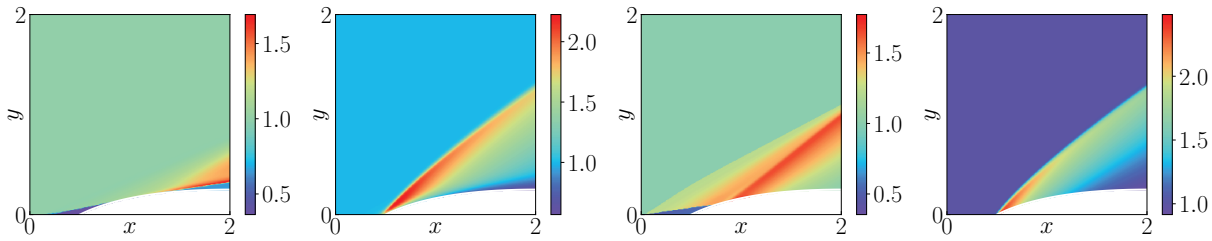
Although streamline PINNs produce a nicely attached shock, the curvature of this shock is highly dependent on the level of viscosity. Fig. 6.16a shows that for a large fixed viscosity, a thick and diffused shock is produced similar to the plain PINN. By decreasing the viscosity a more accurate shock is produced, but if it is decreased too much the shock loses its curvature and becomes oblique. The shock likely becomes so thin that it becomes located in between the collocation points, which can lead to overfitting so that the jump conditions are no longer satisfied. Behavior like this is problematic when the solution is not known because each of these solutions has low residuals and therefore it is hard to determine which one is correct. However, from these results and previous results it appears that the viscosity level that leads to the best solution is fairly constant. In other words, there is likely a relation between the optimal viscosity level and the density of the collocation points.



(a) $\nu = 1 \times 10^{-1}$ **(b)** $\nu = 1 \times 10^{-2}$ **(c)** $\nu = 1 \times 10^{-3}$ **(d)** $\nu = 1 \times 10^{-4}$

Figure 6.16: Density solutions for the curved shock problem by streamline PINNs with different fixed viscosity levels.

For the CPINN results, a discriminator with 5 layers of 50 neurons with the `relu` activation is again considered. Fig. 6.17 and Table 6.6 show that without viscosity, both the component CPINN and the streamline CPINN produce zero-velocity regions, although the latter produces a smaller one. These regions are produced early on in the training process so that, unlike the oblique shock case, the CPINNs do not initially produce an accurate shock. However, with the addition of viscosity, both output representations result in accurate shocks. Nevertheless, the component representation produces a low-density streak near the surface. In contrast, the streamline CPINN with viscosity produces the best results for all PINN adaptations considered.



(a) Component & no viscosity **(b)** Component & viscosity **(c)** Streamline & no viscosity **(d)** Streamline & viscosity

Figure 6.17: Density solutions for the curved shock problem by CPINNs with different adaptations.

Table 6.6: Errors and viscosity levels of the various CPINN solutions for the curved shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$
CPINN	-	2.66e-1	3.99e-1	1.24e-1	9.58e-1
CPINN + viscosity	$\nu = 5\text{e-}3$	7.68e-2	7.45e-2	1.72e-2	1.54e-1
CPINN + streamline	-	1.82e-1	2.82e-1	9.62e-2	5.88e-1
CPINN + streamline + viscosity	$\nu = 5\text{e-}3$	**3.03e-2**	**4.85e-2**	**7.61e-3**	**9.62e-2**

The reason that CPINNs can produce a better solution than PINNs is due to their inherent ability to balance the different loss terms. In particular, the discriminator assigns larger weights to the slip condition near the tip, which reduces the errors significantly compared to PINNs as shown in Fig. 6.18. Essentially, the generator is forced to model the tip region correctly despite its small size and negligible contribution to the loss function. This leads to a considerably better performance because the tip region is of vital importance to the overall curvature and placement of the shock. Note that the large deviation near $x = 0.5$ is a result of the ambiguous slip condition at the tip of the object, where the surface inclination is undefined due to the symmetry boundary conditions.

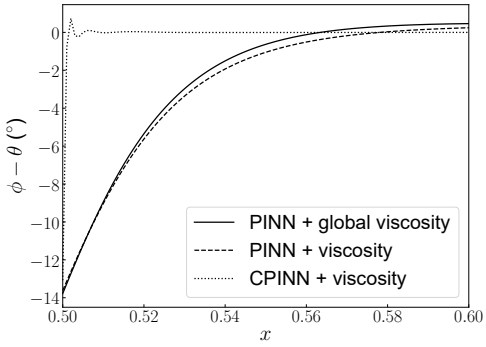


Figure 6.18: Difference between the flow angle and the surface inclination of the object for streamline (C)PINNs.

6.3 Detached shock wave

The streamline PINN and CPINN results for the curved shock wave problem are promising and give reason to attempt even more complex cases. To the author's best knowledge, no literature exists on the simulation of a steady bow shock with PINNs. In particular, current methods rely on known data (Cai et al., 2021; Jagtap et al., 2022) or can only simulate unsteady transonic shocks (Liu, 2022). To assess the performance of the considered adaptations, the problem in Fig. 6.19 is considered. It is essentially equivalent to the curved shock problem, except with a smaller Mach number such that $\theta_{\text{tip}} > \theta_{\text{max}}$, resulting in a detached shock. The reference solution is again calculated with OpenFOAM, details can be found in Appendix A.

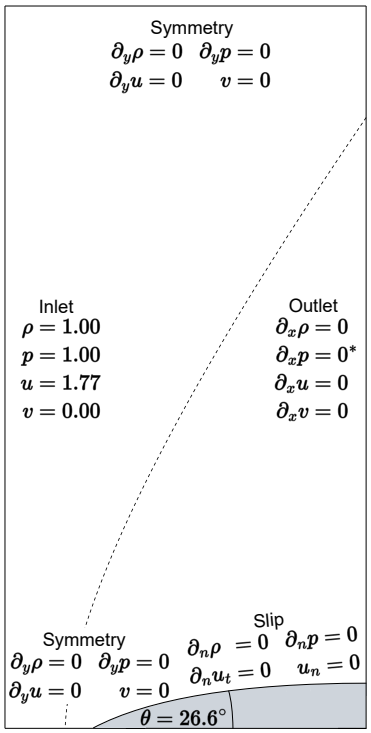
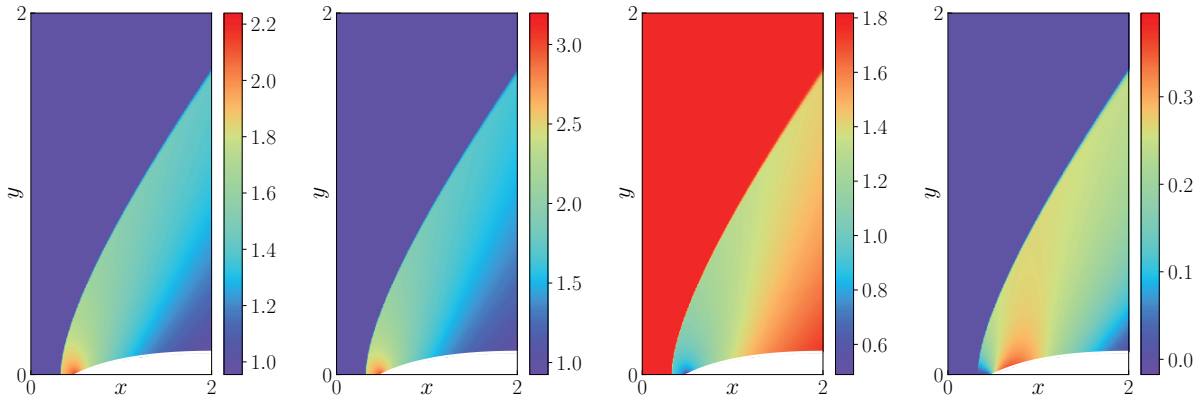


Figure 6.19: Domain and boundary conditions for the detached shock wave problem. The Mach number is 1.5 and the domain size is $[0, 2] \times [0, 4]$ with the object starting at $x = 0.5$.

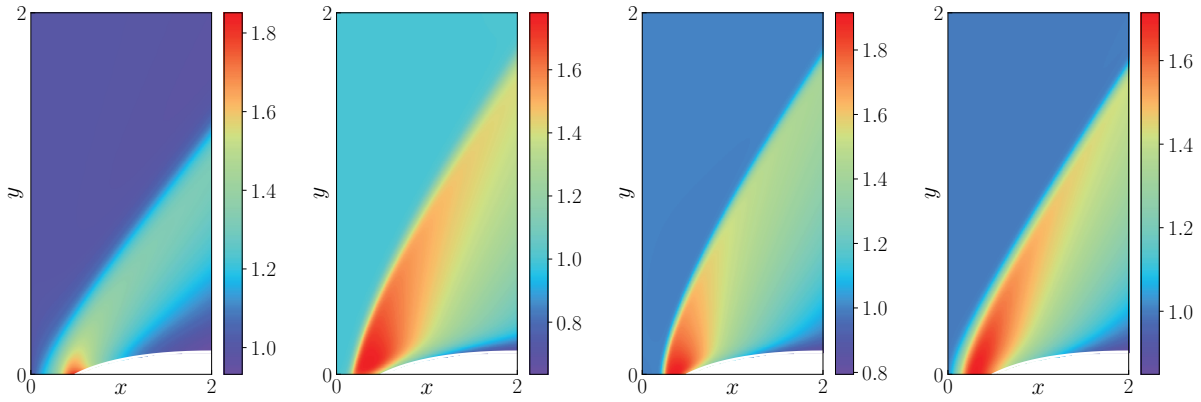
The reference solution is shown in Fig. 6.20, showing that the shock detaches at approximately $\delta = 0.16$ from the tip of the object. As a result, a small subsonic region is present near the tip. In particular, a smooth stagnation point now occurs at the tip which might provoke the creation of a zero-velocity region. After the subsonic tip region, the flow is expanded so that the flow at the outlet is completely supersonic again. The base PINN setup is identical to the curved shock problem, except for the domain being expanded to $(x, y) \in [0, 2] \times [0, 4]$. Furthermore, the number of collocation points is doubled to 20,000 to maintain the same point density. The number of boundary condition points is not increased, as the domain expansion only affects the boundary point density at the inlet, which has a negligible effect because the boundary conditions are uniform there.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.20: Reference solution of the detached shock problem in Fig. 6.19.

Interestingly, Fig. 6.21 shows that even without viscosity a detached shock wave can be produced by PINNs. However, this is only the case for very large boundary condition weights as evident from Table 6.7. The addition of viscosity greatly improves the shape of the produced shock wave, although all viscosity adaptations struggle with the stagnation zone in front of the tip. This is the result of the initial tendency to produce a zero-velocity region around the object, which also creates a low-density streak on the top surface. The local viscosity method produces the most accurate result overall, possibly owing due to its slightly better-modeled expansion region. Nevertheless, the detachment distance is far from correct and the shock is more diffused compared to the other viscosity methods.



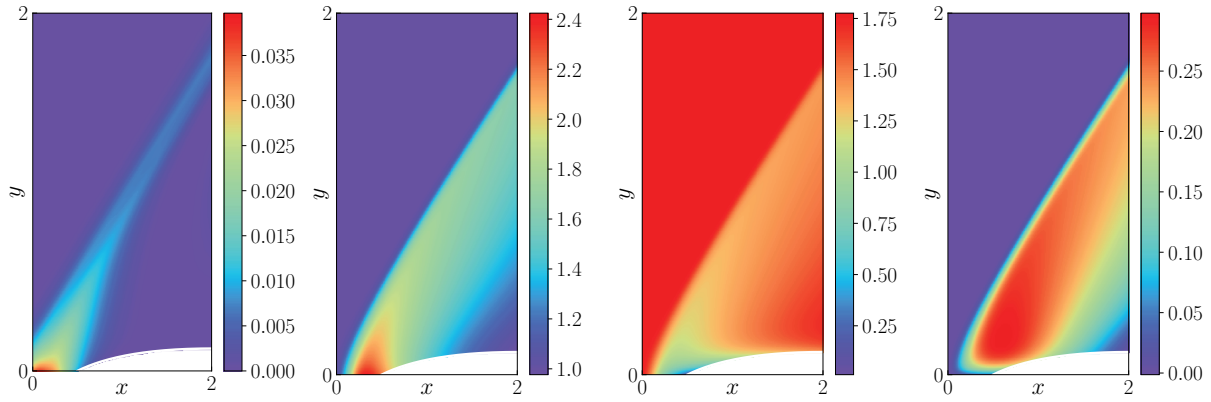
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.21: Density solutions for the detached shock problem by PINNs with different viscosity methods.

Table 6.7: Errors and viscosity levels of the various PINN solutions for the detached shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+4	1.33e-1	1.94e-1	9.17e-2	4.56e-1	-
PINN + viscosity	$\nu =$ 1e-2	9.64e-2	1.25e-1	6.23e-2	3.00e-1	1.0e-2
PINN + viscosity + Neumann	$\nu =$ 5e-3	1.18e-1	1.71e-1	8.73e-2	4.23e-1	5.0e-3
PINN + global viscosity	$w_\nu =$ 1e+0	8.37e-2	1.17e-1	**5.76e-2**	2.80e-1	6.4e-3
PINN + local viscosity	$w_\nu =$ 1e-2	**8.07e-2**	**1.13e-1**	5.99e-2	**2.55e-1**	**3.8e-3**

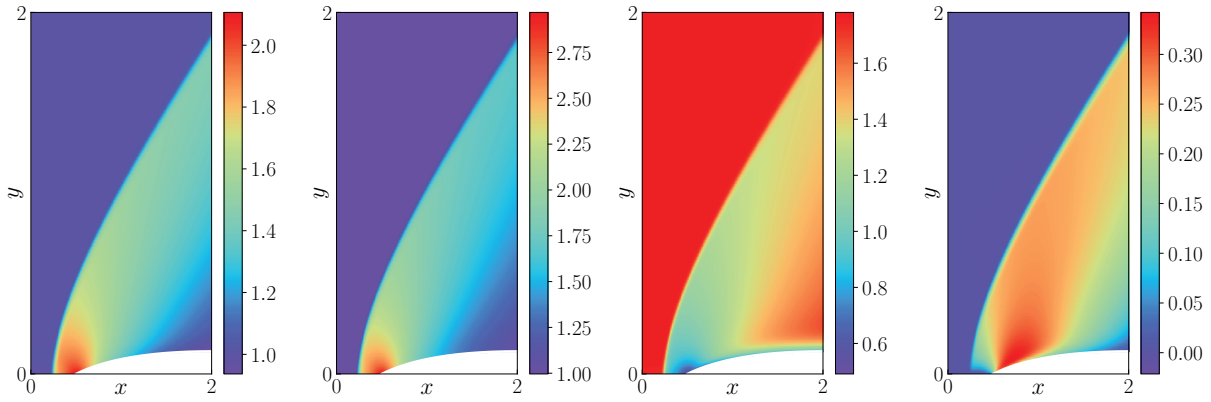
Upon inspection of the local viscosity profile of the PINN with the local viscosity method in Fig. 6.22a, a high viscosity region occurs in front of the tip similar to what is seen for the attached shock in Fig. 6.13. To reiterate, the bottom symmetry conditions cannot be properly satisfied due to the zero-velocity tip region in Fig. 6.22c. As a result, the PINN tries to solve this by adding viscosity in front of the tip so that the flow can stagnate through heat dissipation (friction).



(a) Local viscosity **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.22: Solution of a PINN with local viscosity on the detached shock problem for $w_\nu = 1 \times 10^{-2}$.

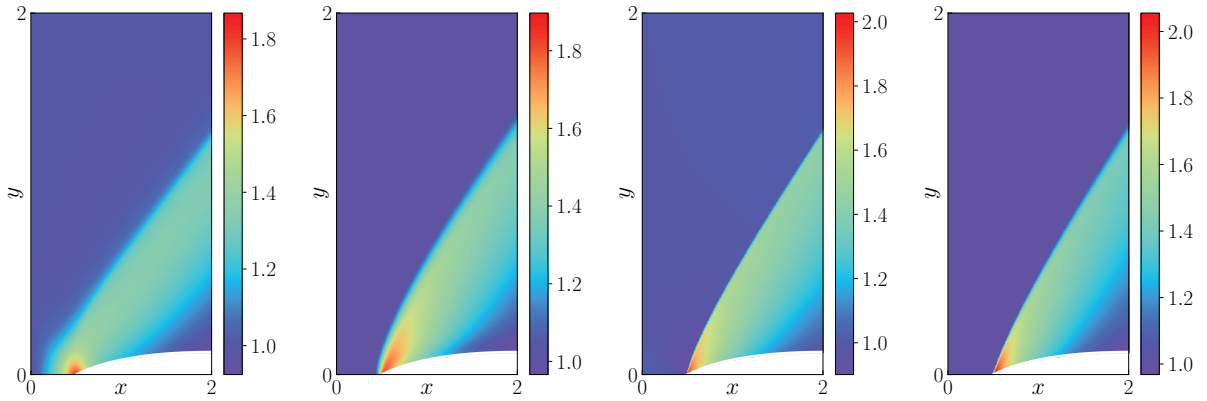
It is interesting to note that the local viscosity PINN solution is not fully symmetric at the bottom symmetry plane, similar to the fixed viscosity PINN solution in Fig. 6.21b. This is a direct consequence of not enforcing the Neumann boundary conditions. While excluding them makes the problem underdetermined, it does not mean that including them will necessarily improve the solution, as it can change the loss landscape. In fact, it is observed that the solutions hardly change when including these boundary conditions, although something interesting happens in the case of the fixed viscosity method. Fig. 6.23 shows that the stagnation region is modeled slightly better, although the solution overall is worse based on Table 6.7.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.23: Solution of a PINN with fixed viscosity and Neumann boundary conditions on the detached shock problem for $\nu = 5 \times 10^{-3}$.

Unlike on the curved shock problem, the solutions obtained with streamline PINNs deteriorate on the detached shock problem. Although Fig. 6.24 shows that sharp shocks are produced when including viscosity, they are attached and the errors in Table 6.8 are worse than before. This is because the PINNs are initially eager to fit the slip boundary conditions, but fail to detach the shock afterward as it requires a sharp transition of ϕ in front of the tip. In other words, streamline PINNs are prone to a new local minima.



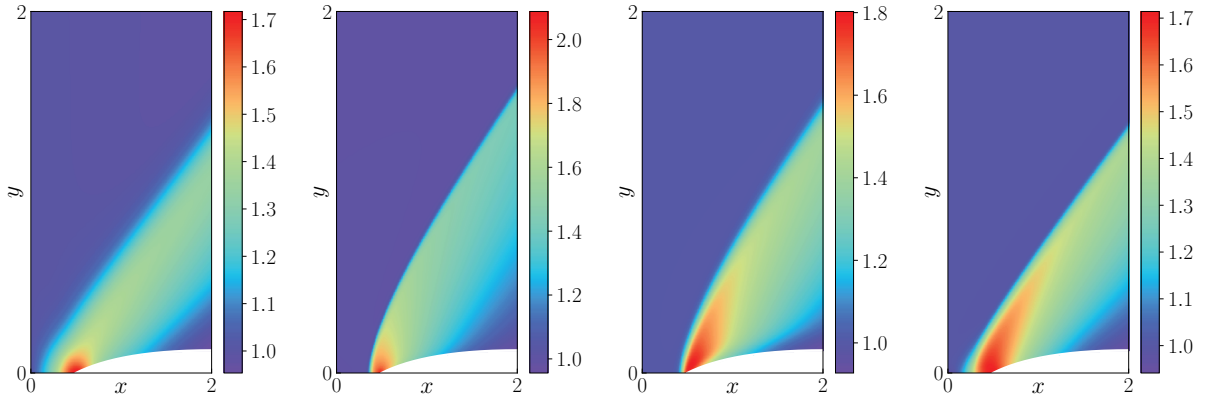
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.24: Density solutions for the detached shock problem by streamline PINNs with different viscosity methods.

Table 6.8: Errors and viscosity levels of the various streamline PINN solutions for the detached shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+1	**1.38e-1**	**2.02e-1**	**9.21e-2**	**4.83e-1**	-
PINN + viscosity	$\nu =$ 1e-2	1.49e-1	2.17e-1	1.13e-1	5.53e-1	1.0e-2
PINN + global viscosity	$w_\nu =$ 1e-1	1.75e-1	2.54e-1	1.19e-1	6.42e-1	**2.0e-3**
PINN + local viscosity	$w_\nu =$ 1e-4	1.70e-1	2.47e-1	1.12e-1	6.27e-1	2.4e-3

This behavior can be mitigated by including the Neumann boundary conditions at the bottom and object surface, as it prevents the initial attachment of the shock wave. This is clearly shown in Fig. 6.25, showing that all solutions with viscosity methods have now detached. As a result, the errors have significantly decreased, as shown in Table 6.9. It is particularly interesting to note that the fixed viscosity methods nicely model the stagnation region, although the errors are higher compared to some of the PINN results in Table 6.7. While the detachment that occurs by enforcing the Neumann boundary conditions is beneficial, the strong tendency of attachment displayed by the streamline representation is likely why it performed so well on the attached shock problem in the previous section.



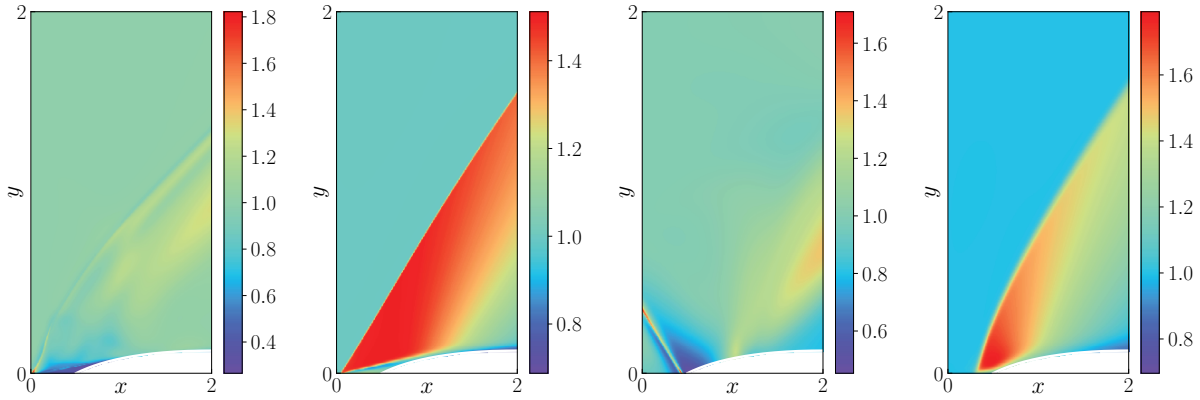
(a) No viscosity **(b)** Fixed viscosity **(c)** Global viscosity **(d)** Local viscosity

Figure 6.25: Density solutions for the detached shock problem by streamline PINNs with Neumann boundary conditions for different viscosity methods.

Table 6.9: Errors and viscosity levels of the various streamline PINN solutions with Neumann boundary conditions for the detached shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$	$\bar{\nu}$
PINN	$w_{bc} =$ 1e+2	1.35e-1	1.99e-1	9.08e-2	4.79e-2	-
PINN + viscosity	$\nu =$ 5e-3	**9.10e-2**	**1.32e-1**	**5.95e-2**	**3.33e-1**	5.0e-3
PINN + global viscosity	$w_\nu =$ 1e-1	1.30e-1	1.90e-1	9.07e-2	4.77e-1	1.1e-2
PINN + local viscosity	$w_\nu =$ 1e-1	1.29e-1	1.85e-1	8.70e-2	5.19e-1	**1.4e-3**

In contrast to streamline PINNs, streamline CPINNs do not produce an attached shock wave as shown in Fig. 6.26d, resulting in the best result so far as is evident from Table 6.10. It is therefore likely that the streamline PINNs are underfitting some boundary conditions to produce an attached solution, which can be successfully alleviated by employing a discriminator to weigh these points more heavily. The other CPINN solutions are not successful and the simulations without viscosity completely fail. Furthermore, Fig. 6.26b shows a zero-velocity region with an oblique wedge over it.



(a) Component & no viscosity **(b)** Component & viscosity **(c)** Streamline & no viscosity **(d)** Streamline & viscosity

Figure 6.26: Density solutions for the detached shock problem by CPINNs with different adaptations.

Table 6.10: Errors and viscosity levels of the various CPINN solutions for the detached shock problem.

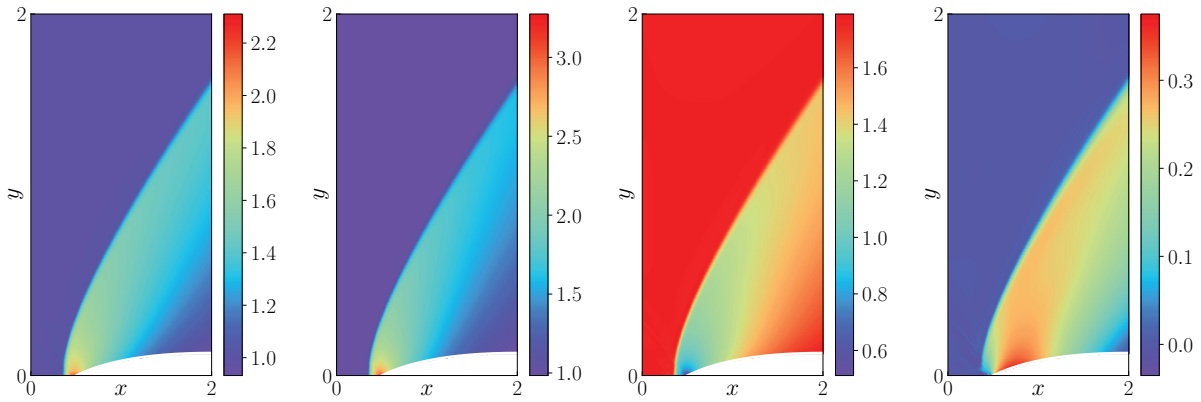
	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$
CPINN	-	2.30e-1	2.72e-1	1.31e-1	6.30e-1
CPINN + viscosity	$\nu =$ 1e-3	1.11e-1	1.44e-1	7.63e-2	3.84e-1
CPINN + streamline	-	2.34e-1	2.95e-1	1.49e-1	7.89e-1
CPINN + streamline + viscosity	$\nu =$ 5e-3	**7.57e-2**	**1.00e-1**	**4.40e-2**	**2.61e-1**

Similar to PINNs, CPINNs are sensitive to the choice of included boundary conditions and output representation. In particular, it is observed that including the Neumann boundary conditions at the bottom boundaries results in a breakdown of the solution similar to the inviscid CPINN solutions. However, it is important to reiterate that the discriminator network needs an additional output for each boundary condition considered, so the discriminator may lack the approximation capacity to correctly involve the additional Neumann boundary conditions. Nevertheless, to show that CPINNs can also nicely resolve the stagnation region, consider the alternative formulation of the streamline representation in Eq. (5.4).

$$\|\mathbf{u}\| : f(x) = \exp(x), \quad \phi : f(x) = \pi \tanh(x) \qquad \text{(5.4 repeated)}$$

This minor change results in a much better solution, as shown in Fig. 6.27. In fact, this is the best solution overall for the detached shock problem as shown in Table 6.11. The reason that this alternative streamline

representation yields better results is likely because it more easily satisfies the Neumann conditions. This assumption is based on the large negative velocities that are produced by the CPINN in Fig. 6.26d below $y = 0$ (not visible), which does not occur in the CPINN solution shown in Fig. 6.27. Note again that this does not mean that this alternative streamline representation is better than the original one, it simply indicates how sensitive (C)PINNs are to even minor changes.



(a) Density **(b)** Pressure **(c)** Horizontal velocity **(d)** Vertical velocity

Figure 6.27: Solution of a CPINN with an alternative streamline formulation, fixed viscosity and Neumann boundary conditions on the detached shock problem for $\nu = 5 \times 10^{-3}$.

Table 6.11: Errors and viscosity levels of the CPINN solution with an alternative streamline formulation, fixed viscosity and Neumann boundary conditions for the detached shock problem.

	Hyperparameter	$L_{2,\rho}$	$L_{2,p}$	$L_{2,u}$	$L_{2,v}$
CPINN + alt. streamline + viscosity	$\nu =$ 5e-3	**5.30e-2**	**7.75e-2**	**3.27e-2**	**1.82e-1**

Closing remarks

While existing adaptations can simulate an oblique shock reasonably, the accuracy is slightly improved when using the local viscosity method. However, the difference compared to the global viscosity method is minimal on all problems considered. While the absence of viscosity in isentropic regions is beneficial, the effects are limited because the problems involve isentropic regions with low gradients. The local viscosity method will likely have a more pronounced impact when considering isentropic discontinuities such as contact discontinuities and shear waves. In contrast, existing adaptations fail on the curved shock problem when not combined with the streamline output representation, which successfully alleviates the formation of zero-velocity regions. However, these regions are less pronounced in the detached shock wave problem and the streamline output representation can actually worsen results due to its strong tendency of attachment. Nevertheless, this effect can be partly mitigated by including the Neumann boundary conditions. Overall, the best results are obtained when combining CPINNs with viscosity and the streamline output representation, although at a much larger computational cost.

In short, the proposed local viscosity and streamline output representation adaptations successfully improve the shock-capturing capabilities of PINNs and for the first time allow accurate simulations of curved and detached steady shocks. While the results are promising, only a single problem is considered for each shock class, so one must be careful about drawing general conclusions from these results.

Part III

Closure

7

Conclusion

In recent years, PINNs have been shown to tackle a wide range of fluid dynamics problems despite suffering from many failure modes. This is because the failure modes have been well-documented and many adaptations have been proposed to alleviate them. However, the simulation of highly compressible flows involves unique failure modes that have not yet been properly identified. Therefore, the goal of this thesis was to provide a unifying theory on why PINNs fail on highly compressible problems and to design new adaptations that can alleviate these failure modes to allow for the simulation of more complex shocks such as curved and detached shocks. For this purpose, the following research questions are answered. Note that the Euler equations are considered since there is little reference literature using the Navier-Stokes equations.

> Why do PINNs fail on highly compressible problems?

Essentially, two failure modes have been identified that prohibit PINNs from effectively simulating highly compressible flows. The first, more fundamental failure mode is related to the differential nature of PINNs. In particular, analysis of the differential Euler equations shows that entropy is conserved along streamlines, even though shocks are not isentropic phenomena. Hence, PINNs suffer from the *entropy failure mode* because they try to find an isentropic solution to a non-isentropic problem. As a result, the simulation of problems involving shocks can lead to nonphysical phenomena that bear little resemblance to the physical solution. Note that the simulation of other highly compressible features such as expansion waves and even contact discontinuities is no issue to PINNs, since they are isentropic. Interestingly, CPINNs can temporarily cope with the entropy failure mode by not satisfying the Euler equations at shocks, producing locally high residuals. While this results in an accurate solution initially, the failure mode resurfaces once the residuals decrease further on in the training process.

The second, more applied failure mode is related to the ambiguous nature of the slip boundary conditions. While these boundary conditions aim to constrain the flow direction in a tangential direction along the surface, they do so by penalizing the normal component of the velocity. As a result, a trivial zero velocity is also a solution to the boundary conditions even though this is usually only valid for stagnation points. The simplicity of this trivial solution compared to learning the dependency between the velocity components creates a strong local minimum, which can cause PINNs to initially produce zero-velocity regions over the surfaces of objects. Interestingly, this *slip failure mode* leads to the creation of fictive objects that are delineated by a combination of shear waves and contact discontinuities. The flow around this fictive object is physical, although it is drastically simpler than the flow around the original object. Overall, this failure mode prevents PINNs from being able to simulate highly compressible flows over curved objects, even in the absence of shocks.

> How do existing adaptations relate to these failure modes?

The symptoms of both failure modes have certainly been observed in literature, although they have only been treated with limited success. This is mainly because the underlying rationales of the proposed adaptations do not sufficiently identify the above failure modes, so some of the successful results are only obtained by luck. A prime example is the domain extension method by Papados (2021), which promises

to eliminate the nonphysical phenomena arising from the entropy failure mode by extending the domain to suppress spurious oscillations. Although it improves the results, it is actually because the collocation point density is reduced which makes it less likely that points are located near the shock so that the entropy failure mode cannot manifest. A different adaptation by Liu (2022) suppresses the high residuals to allow for sharp discontinuities, which led to the first PINN simulation of an unsteady curved shock. However, this method is more of a remedy, as it makes no notion of entropy and stimulates overfitting. In addition, it also affects other discontinuities such as contact discontinuities and shear waves.

Another class of adaptations is based on the addition of viscosity to the PDEs, which mimics the numerical viscosity that manifests in traditional solvers allowing them to capture shocks. Since viscosity allows for dissipation, it also provides a mean of entropy increase and thus the avoidance of the entropy failure mode. However, when the viscosity is too small it has no effect and when the viscosity is too large the solution is diffused too much. The former also gives reason to believe that the entropy failure mode can manifest in the Navier-Stokes equations too, despite the presence of a dissipation term. In addition, it not only affects shocks but also other regions of the flow that would otherwise be isentropic. Nevertheless, the results indicate that a fixed viscosity allows for the accurate capture of shocks. To avoid selecting the viscosity level manually, both Wassing et al. (2023) and Coutinho et al. (2023) propose methods that automatically reduce the viscosity. While these vanishing viscosity methods are effective, the authors do not mention that viscosity below a certain threshold leads to the resurfacing of an entropy failure mode. Lastly, the addition of viscosity is also used to alleviate symptoms of the slip failure mode by Wassing et al. (2023), although this is only effective on subsonic flows without shocks.

> What adaptations can alleviate these failure modes?

Current viscosity methods are effective at suppressing the entropy failure mode, but they are indiscriminate in the sense that viscosity also affects isentropic regions of the domain. While Coutinho et al. (2023) have proposed local viscosity methods, they either rely on foreknowledge of the solution or on a discretized mesh. Therefore, a truly local viscosity method is proposed that includes the viscosity as an additional output of the PINN. By adding a viscous loss term to the loss function, large diffusive viscosities are prevented. The results show that this method can accurately localize the viscosity at shocks, leading to a substantially lower average viscosity. Nevertheless, its results are often only marginally better than an existing similar global viscosity method. This is likely because global viscosity only has a limited negative effect on isentropic regions since the gradients there are substantially lower than at shocks. However, problems involving other discontinuities would likely benefit significantly from a local viscosity method because global viscosity cannot differentiate between isentropic and non-isentropic discontinuities.

To counteract the slip failure mode, a streamline output representation is proposed that allows for a more natural formulation of the slip boundary conditions. This new formulation does not penalize normal velocities, but rather the misdirection of the flow. As a result, the adapted PINNs are able to accurately capture steady attached curved shocks which is otherwise not possible. In addition, while the results show that CPINNs are able to capture curved shocks without this new output representation, their performance significantly improves when using it. Nevertheless, the streamline output representation is so effective at producing attached shocks that it fails to detach shocks when necessary. This strong tendency of attachment can be alleviated by including Neumann boundary conditions, which results in reasonably accurate detached shocks. However, the best results are obtained by combining the streamline output representation with CPINNs.

Interpretation

All in all, a theory on highly compressible failure modes is presented that leads to the introduction of new adaptations that significantly improve the shock-capturing capabilities of PINNs. In fact, to the author's best knowledge, the results provide the first successful simulations of steady curved and detached shocks by PINNs. Nevertheless, it is important to state that the adaptions still do not provide a consistent method of capturing complex shocks, as a slight change in the boundary conditions or network outputs can lead to drastically different results. As John Anderson said, *"Like all computational fluid dynamics applications, solutions are frequently more of an art than a science."*. It is therefore not wise to draw strong conclusions on the general performance of these adaptations, but they are certainly a step in the right direction.

8

Recommendations

While the results are promising, they perhaps provide more questions than answers. In particular, the introduced adaptations are based on empirical observations and there are certainly more possible definitions or even entirely new adaptations that can achieve the same effects. This chapter provides an overview of possible future research directions that could further enhance the usage of PINNs for highly compressible flows. However, this list is certainly not exhaustive.

8.1 Design of viscosity

8.1.1 Relation viscosity and collocation points

The results show that the minimum viscosity level that admits physical shocks is fairly consistent. Since the same density of collocation points was used across the considered problems, it may indicate that there is a relation between the two. Therefore, a future point of study could be the investigation of how the minimum viscosity level depends on the density and placement of collocation points.

8.1.2 Other forms of viscosity

All the viscosity methods considered in this thesis have the form $\nu\Delta u$, with u the conserved variable. However, there might be more suitable definitions. For example, it may be possible to add a viscosity term to the energy equation alone, as the conservation of entropy originates from it. A starting point is a review of discretization schemes in traditional solvers since they are carefully designed to result in numerical viscosity that satisfies certain criteria. The complete freedom that PINNs offer in the modification of the solved equations could potentially allow even better forms of artificial viscosity.

8.2 Boundary conditions

8.2.1 Importance of boundary conditions

For most problems considered in the thesis, certain boundary conditions were omitted because they can complexify the loss landscape and prevent suitable convergence. This is conflicting, as they should be included to make the problem well-posed and guarantee the existence of a unique solution. Therefore, it is valuable to investigate what the effect is of including or excluding certain boundary conditions. Perhaps it is possible to include a weaker form of certain boundary conditions, where they are only included if they are violated significantly.

8.2.2 Alternative definitions

Similar to the streamline output representation for the slip boundary condition, there could be alternative definitions of the outputs that lead to more natural formulations of other boundary conditions. An example could be using the characteristic form and variables of the Euler equations, allowing a more natural definition of outlet boundary conditions by means of characteristic boundary conditions.

8.3 Integral solvers

8.3.1 Finite volume PINNs

Although Section 4.3 mentions that traditional solvers can capture shocks due to the presence of numerical artificial viscosity, this is only true for finite difference solvers. Finite volume solvers use the integral conservation laws instead, which means the Rankine-Hugoniot jump relations are satisfied naturally. While PINNs are fundamentally based on a differential approach, there is active research on the design of finite volume PINNs (Praditia et al., 2021; Patel et al., 2022; Papados, 2022). These methods can completely mitigate the entropy failure mode, although they likely still suffer from the zero-velocity failure mode.

8.4 Applications

8.4.1 Full body simulations

The considered oblique, curved and detached shock problems are all symmetric and concern only the frontal parts of objects. It is interesting to investigate if the proposed framework and adaptations are also effective at simulating full bodies, which would include for example shocks at trailing edges. Such simulations were not considered for this thesis as they require more computational resources and come with their own peculiarities.

8.4.2 Shock wave interactions

Only single shock waves were considered in this thesis, but it is also possible that shock waves intersect and interact. These points of intersection could be more challenging to PINNs as they occur at a singular point, possibly far away from collocation points. Furthermore, they create contact discontinuities as some streamlines pass through two shocks while some only through one. A simple problem that features such an interaction is the double wedge problem.

References

Anderson, John D. (2006). *Hypersonic and high-temperature gas dynamics*. 2nd ed. AIAA education series. OCLC: ocm68262944. Reston, Va: American Institute of Aeronautics and Astronautics.

Anderson, John David (2021). *Modern compressible flow: with historical perspective*. eng. Fourth edition. New York, NY: McGraw Hill.

Anderson, John David, Lorenzo M. Albacete, and Allen Edward Winkelmann (1968). *On Hypersonic Blunt Body Flow Fields Obtained with a Time-dependent Technique*. en. United States Naval Ordnance Laboratory.

Ang, Elijah and Bing Feng Ng (Jan. 2022). "Physics-Informed Neural Networks for Flow Around Airfoil". In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2022-0187. URL: https://doi.org/10.2514/6.2022-0187.

Baydin, Atılım Günes et al. (Jan. 2017). "Automatic differentiation in machine learning: a survey". In: *The Journal of Machine Learning Research* 18.1, pp. 5595–5637.

Berrone, S. et al. (2022). "Enforcing Dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks". In: DOI: 10.48550/ARXIV.2210.14795. URL: https://arxiv.org/abs/2210.14795 (visited on 07/09/2023).

Billig, Frederick S. (June 1967). "Shock-wave shapes around spherical-and cylindrical-nosed bodies." en. In: *Journal of Spacecraft and Rockets* 4.6, pp. 822–823. DOI: 10.2514/3.28969. URL: https://arc.aiaa.org/doi/10.2514/3.28969 (visited on 07/04/2023).

Brunton, Steven L., Bernd R. Noack, and Petros Koumoutsakos (2020). "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52.1, pp. 477–508. DOI: 10.1146/annurev-fluid-010719-060214. eprint: https://doi.org/10.1146/annurev-fluid-010719-060214. URL: https://doi.org/10.1146/annurev-fluid-010719-060214.

Butcher, J.C. (June 2003). *Numerical methods for ordinary differential equations*. en. Nashville, TN: John Wiley & Sons.

Cai, Shengze et al. (Dec. 2021). "Physics-informed neural networks (PINNs) for fluid mechanics: a review". en. In: *Acta Mechanica Sinica* 37.12, pp. 1727–1738. DOI: 10.1007/s10409-021-01148-1. URL: https://doi.org/10.1007/s10409-021-01148-1.

Canuto, Claudio et al. (2012). *Spectral methods in fluid dynamics*. Springer Science & Business Media.

Choromanska, Anna et al. (Jan. 2015). *The Loss Surfaces of Multilayer Networks*. arXiv:1412.0233 [cs]. DOI: 10.48550/arXiv.1412.0233. URL: http://arxiv.org/abs/1412.0233.

Coutinho, Emilio Jose Rocha et al. (Sept. 2023). "Physics-informed neural networks with adaptive localized artificial viscosity". In: *Journal of Computational Physics* 489, p. 112265. DOI: 10.1016/j.jcp.2023.112265. URL: https://doi.org/10.1016/j.jcp.2023.112265.

Cybenko, G. (Dec. 1989). "Approximation by superpositions of a sigmoidal function". en. In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

Daw, Arka et al. (Oct. 2022). *Mitigating Propagation Failures in PINNs using Evolutionary Sampling*. arXiv:2207.02338 [cs]. DOI: 10.48550/arXiv.2207.02338. URL: http://arxiv.org/abs/2207.02338.

Deng, Li (2012). "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142.

DHPC (2022). *DelftBlue Supercomputer (Phase 1)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

Eivazi, Hamidreza et al. (July 2022). "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations". In: *Physics of Fluids* 34.7, p. 075117. DOI: 10.1063/5.0095270. URL: https://doi.org/10.1063/5.0095270.

Evans, Lawrence C. (2022). *Partial differential equations*. eng. Second edition. Graduate studies in mathematics 19. Providence, Rhode Island: American Mathematical Society.

Ferrer-Sánchez, Antonio et al. (2023). *Gradient-Annihilated PINNs for Solving Riemann Problems: Application to Relativistic Hydrodynamics*. DOI: 10.48550/ARXIV.2305.08448. URL: https://arxiv.org/abs/2305.08448.

Ferziger, Joel H and Milovan Peric (Dec. 2012). *Computational methods for fluid dynamics*. en. 3rd ed. Berlin, Germany: Springer.

Geuzaine, Christophe and Jean-François Remacle (May 2009). "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering* 79.11, pp. 1309–1331. DOI: 10.1002/nme.2579. URL: https://doi.org/10.1002/nme.2579.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

Gu, Yiqi, Haizhao Yang, and Chao Zhou (Sept. 2021). "SelectNet: Self-paced learning for high-dimensional partial differential equations". In: *Journal of Computational Physics* 441, p. 110444. DOI: 10.1016/j.jcp.2021.110444. URL: https://doi.org/10.1016/j.jcp.2021.110444.

Haberman, Richard (2014). *Applied partial differential equations with fourier series and boundary value problems*. eng. 5th edition, Pearson new international edition. OCLC: 973285331. Harlow: Pearson.

Han, Jihun and Yoonsang Lee (2023). "Hierarchical Learning to Solve PDEs Using Physics-Informed Neural Networks". In: *Computational Science - ICCS 2023*. Springer Nature Switzerland, pp. 548–562. DOI: 10.1007/978-3-031-36024-4_42. URL: https://doi.org/10.1007/978-3-031-36024-4_42.

Hennigh, Oliver et al. (2021). "NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework". In: *Computational Science – ICCS 2021*. Springer International Publishing, pp. 447–461. DOI: 10.1007/978-3-030-77977-1_36. URL: https://doi.org/10.1007/978-3-030-77977-1_36.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (Jan. 1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8. URL: https://doi.org/10.1016/0893-6080(89)90020-8.

Jagtap, Ameya D., Kenji Kawaguchi, and George Em Karniadakis (Mar. 2020a). "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks". In: *Journal of Computational Physics* 404, p. 109136. DOI: 10.1016/j.jcp.2019.109136. URL: https://doi.org/10.1016/j.jcp.2019.109136.

Jagtap, Ameya D., Kenji Kawaguchi, and George Em Karniadakis (July 2020b). "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2239, p. 20200334. DOI: 10.1098/rspa.2020.0334. URL: https://doi.org/10.1098/rspa.2020.0334.

Jagtap, Ameya D. et al. (Oct. 2022). "Physics-informed neural networks for inverse problems in supersonic flows". In: *Journal of Computational Physics* 466, p. 111402. DOI: 10.1016/j.jcp.2022.111402. URL: https://doi.org/10.1016/j.jcp.2022.111402.

Jiang, Chiyu "Max" et al. (Nov. 2020). "MeshfreeFlowNet: a physics-constrained deep continuous space-time super-resolution framework". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '20. Atlanta, Georgia: IEEE Press, pp. 1–15.

Jin, Xiaowei et al. (Feb. 2021). "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations". en. In: *Journal of Computational Physics* 426, p. 109951. DOI: 10.1016/j.jcp.2020.109951. URL: https://www.sciencedirect.com/science/article/pii/S0021999120307257.

Karniadakis, Ameya D. Jagtap & George Em (June 2020). "Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations". In: *Communications in Computational Physics* 28.5, pp. 2002–2041. DOI: 10.4208/cicp.OA-2020-0164. URL: http://global-sci.org/intro/article_detail/cicp/18403.html.

Karras, Tero et al. (2017). *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. DOI: 10.48550/ARXIV.1710.10196. URL: https://arxiv.org/abs/1710.10196.

Khan, Arbaaz and David A. Lowther (Sept. 2022). "Physics Informed Neural Networks for Electromagnetic Analysis". In: *IEEE Transactions on Magnetics* 58.9, pp. 1–4. DOI: 10.1109/tmag.2022.3161814. URL: https://doi.org/10.1109/tmag.2022.3161814.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Korpelevich, G. M. (1977). “The extragradient method for finding saddle points and other problems”. In: vol. 13. 4, pp. 35–49.

Krishnapriyan, Aditi S. et al. (Nov. 2021). *Characterizing possible failure modes in physics-informed neural networks*. arXiv:2109.01050 [physics]. DOI: 10.48550/arXiv.2109.01050. URL: http://arxiv.org/abs/2109.01050.

Kumar, M. Pawan, Benjamin Packer, and Daphne Koller (2010). “Self-Paced Learning for Latent Variable Models”. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*. NIPS’10. Vancouver, British Columbia, Canada: Curran Associates Inc., pp. 1189–1197.

Kármán, Th. von (1911). “Ueber den Mechanismus des Widerstandes, den ein bewegter Körper in einer Flüssigkeit erfährt”. In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* 1911, pp. 509–517. URL: http://eudml.org/doc/58812.

Lagaris, I.E., A. Likas, and D.I. Fotiadis (1998). “Artificial neural networks for solving ordinary and partial differential equations”. In: *IEEE Transactions on Neural Networks* 9.5, pp. 987–1000. DOI: 10.1109/72.712178. URL: https://doi.org/10.1109/72.712178.

Laubscher, Ryno, Pieter Rousseau, and Chris Meyer (June 2022). “Modeling of Inviscid Flow Shock Formation in a Wedge-Shaped Domain Using a Physics-Informed Neural Network-Based Partial Differential Equation Solver”. In: *Volume 10C: Turbomachinery - Design Methods and CFD Modeling for Turbomachinery; Ducts, Noise, and Component Interactions*. American Society of Mechanical Engineers. DOI: 10.1115/gt2022-81768. URL: https://doi.org/10.1115/gt2022-81768.

Lederer, Johannes (2021). *Activation Functions in Artificial Neural Networks: A Systematic Overview*. DOI: 10.48550/ARXIV.2101.09957. URL: https://arxiv.org/abs/2101.09957.

LeVeque, Randall J. (1992). *Numerical Methods for Conservation Laws*. Birkhäuser Basel. DOI: 10.1007/978-3-0348-8629-1. URL: https://doi.org/10.1007/978-3-0348-8629-1.

Liang, Tengyuan and James Stokes (2018). “Interaction Matters: A Note on Non-asymptotic Local Convergence of Generative Adversarial Networks”. In: DOI: 10.48550/ARXIV.1802.06132. URL: https://arxiv.org/abs/1802.06132.

Liepmann, H. W. and A. Roshko (2001). *Elements of gasdynamics*. Mineola, N.Y: Dover Publications.

Liu, Dehao and Yan Wang (Apr. 2021). “A Dual-Dimer method for training physics-constrained neural networks with minimax architecture”. In: *Neural Networks* 136, pp. 112–125. DOI: 10.1016/j.neunet.2020.12.028. URL: https://doi.org/10.1016/j.neunet.2020.12.028.

Liu, Dong C. and Jorge Nocedal (Aug. 1989). “On the limited memory BFGS method for large scale optimization”. en. In: *Mathematical Programming* 45.1, pp. 503–528. DOI: 10.1007/BF01589116. URL: https://doi.org/10.1007/BF01589116.

Liu, Li (June 2022). “Discontinuity Computing with Physics-Informed Neural Network”. In: DOI: 10.36227/techrxiv.19391279. URL: https://doi.org/10.36227/techrxiv.19391279.

Liu, Xu et al. (May 2022). “A novel meta-learning initialization method for physics-informed neural networks”. In: *Neural Computing and Applications* 34.17, pp. 14511–14534. DOI: 10.1007/s00521-022-07294-2. URL: https://doi.org/10.1007/s00521-022-07294-2.

Lu, Lu et al. (Jan. 2021). “DeepXDE: A Deep Learning Library for Solving Differential Equations”. In: *SIAM Review* 63.1, pp. 208–228. DOI: 10.1137/19M1274067. URL: https://epubs.siam.org/doi/10.1137/19M1274067.

Mao, Zhiping, Ameya D. Jagtap, and George Em Karniadakis (Mar. 2020). “Physics-informed neural networks for high-speed flows”. en. In: *Computer Methods in Applied Mechanics and Engineering* 360, p. 112789. DOI: 10.1016/j.cma.2019.112789. URL: https://www.sciencedirect.com/science/article/pii/S0045782519306814.

Markidis, Stefano (2021). “The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?” In: *Frontiers in Big Data* 4. URL: https://www.frontiersin.org/articles/10.3389/fdata.2021.669097.

Martens, James (2010). “Deep Learning via Hessian-Free Optimization”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, pp. 735–742.

Martin, John and Hanspeter Schaub (Mar. 2022). “Physics-informed neural networks for gravity field modeling of the Earth and Moon”. In: *Celestial Mechanics and Dynamical Astronomy* 134.2. DOI: 10.1007/s10569-022-10069-5. URL: https://doi.org/10.1007/s10569-022-10069-5.

McClenny, Levi and Ulisses Braga-Neto (Apr. 2022). *Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism*. arXiv:2009.04544 [cs, stat]. DOI: 10.48550/arXiv.2009.04544. URL: http://arxiv.org/abs/2009.04544.

Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger (2017). *The Numerics of GANs*. DOI: 10.48550/ARXIV.1705.10461. URL: https://arxiv.org/abs/1705.10461.

Metz, Luke et al. (2016). *Unrolled Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1611.02163. URL: https://arxiv.org/abs/1611.02163.

Nabian, Mohammad Amin, Rini Jasmine Gladstone, and Hadi Meidani (Apr. 2021). "Efficient training of physics-informed neural networks via importance sampling". In: *Computer-Aided Civil and Infrastructure Engineering* 36.8, pp. 962–977. DOI: 10.1111/mice.12685. URL: https://doi.org/10.1111/mice.12685.

Nieuwstadt, F.T.M., Jerry Westerweel, and B.J. Boersma (2018). *Turbulence: introduction to theory and applications of turbulent flows*. eng. OCLC: 1042117849. Place of publication not identified: Springer.

O'Shea, Keiron and Ryan Nash (2015). *An Introduction to Convolutional Neural Networks*. DOI: 10.48550/ARXIV.1511.08458. URL: https://arxiv.org/abs/1511.08458.

Papados, Alexandros (Aug. 2021). "Physics-Informed Deep Learning and its Application in Computational Solid and Fluid Mechanics". In: URL: https://github.com/alexpapados/Physics-Informed-Deep-Learning-Solid-and-Fluid-Mechanics.

Papados, Alexandros (2022). "Finite-Volume Physics-Informed Neural Networks". en. In: DOI: 10.13140/RG.2.2.19087.05286. URL: https://rgdoi.net/10.13140/RG.2.2.19087.05286.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the Difficulty of Training Recurrent Neural Networks". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, pp. III–1310–III–1318.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Patel, Ravi G. et al. (Jan. 2022). "Thermodynamically consistent physics-informed neural networks for hyperbolic systems". In: *Journal of Computational Physics* 449, p. 110754. DOI: 10.1016/j.jcp.2021.110754. URL: https://doi.org/10.1016/j.jcp.2021.110754.

Praditia, Timothy et al. (2021). *Finite Volume Neural Network: Modeling Subsurface Contaminant Transport*. DOI: 10.48550/ARXIV.2104.06010. URL: https://arxiv.org/abs/2104.06010.

Pulliam, T. H. (Oct. 1981). "Characteristic Boundary Conditions for the Euler Equations". en. In: *Numerical Boundary Condition Procedures*. URL: https://ntrs.nasa.gov/citations/19810025322 (visited on 07/09/2023).

Rahaman, Nasim et al. (2019). "On the spectral bias of neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 5301–5310.

Raissi, M., P. Perdikaris, and G.E. Karniadakis (Feb. 2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045. URL: https://doi.org/10.1016/j.jcp.2018.10.045.

Raissi, Maziar, Alireza Yazdani, and George Em Karniadakis (2018). *Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data*. DOI: 10.48550/ARXIV.1808.04327. URL: https://arxiv.org/abs/1808.04327.

Rao, Chengping, Hao Sun, and Yang Liu (Mar. 2020). "Physics-informed deep learning for incompressible laminar flows". en. In: *Theoretical and Applied Mechanics Letters* 10.3, pp. 207–212. DOI: 10.1016/j.taml.2020.01.039. URL: https://www.sciencedirect.com/science/article/pii/S2095034920300350.

Reddy, J. N. (2019). *Introduction to the finite element method*. Fourth edition. Mechanical engineering. OCLC: on1065524414. New York, NY: McGraw Hill Education.

Ruder, Sebastian (2016). *An overview of gradient descent optimization algorithms*. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747.

Saad, Youcef and Martin H. Schultz (July 1986). "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3, pp. 856–869. DOI: 10.1137/0907058. URL: https://doi.org/10.1137/0907058.

Saxena, Divya and Jiannong Cao (2020). *Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions*. DOI: 10.48550/ARXIV.2005.00065. URL: https://arxiv.org/abs/2005.00065.

Schäfer, Florian and Anima Anandkumar (June 2020a). *Competitive Gradient Descent*. arXiv:1905.12103 [cs, math]. DOI: 10.48550/arXiv.1905.12103. URL: http://arxiv.org/abs/1905.12103.

Schäfer, Florian, Hongkai Zheng, and Anima Anandkumar (Oct. 2020b). *Implicit competitive regularization in GANs*. arXiv:1910.05852 [cs, stat]. DOI: 10.48550/arXiv.1910.05852. URL: http://arxiv.org/abs/1910.05852.

Shapiro, Ascher H. (1976). *The dynamics and thermodynamics of compressible fluid flow. 1*. eng. 23. print. New York: Wiley.

Shewchuk, Jonathan R (1994). *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. USA.

Shu, Chi-Wang (1998). "Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws". In: *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, pp. 325–432. DOI: 10.1007/bfb0096355. URL: https://doi.org/10.1007/bfb0096355.

Sohl-Dickstein, Jascha et al. (2015). *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. DOI: 10.48550/ARXIV.1503.03585. URL: https://arxiv.org/abs/1503.03585.

Sukumar, N. and Ankit Srivastava (Feb. 2022). "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks". en. In: *Computer Methods in Applied Mechanics and Engineering* 389, p. 114333. DOI: 10.1016/j.cma.2021.114333. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045782521006186 (visited on 07/09/2023).

Sun, Yubiao, Ushnish Sengupta, and Matthew Juniper (June 2023). "Physics-informed deep learning for simultaneous surrogate modeling and PDE-constrained optimization of an airfoil geometry". In: *Computer Methods in Applied Mechanics and Engineering* 411, p. 116042. DOI: 10.1016/j.cma.2023.116042. URL: https://doi.org/10.1016/j.cma.2023.116042.

Tancik, Matthew et al. (Dec. 2020). "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., pp. 7537–7547.

Tang, Kejun, Xiaoliang Wan, and Chao Yang (Mar. 2023). "DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations". In: *Journal of Computational Physics* 476, p. 111868. DOI: 10.1016/j.jcp.2022.111868. URL: https://doi.org/10.1016/j.jcp.2022.111868.

Taylor, Jamie M. et al. (2023). *Deep Fourier Residual method for solving time-harmonic Maxwell's equations*. DOI: 10.48550/ARXIV.2305.09578. URL: https://arxiv.org/abs/2305.09578.

Tieleman, T. and G. Hinton (2012). *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*.

Toro, Eleuterio F. (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin Heidelberg. DOI: 10.1007/b79761. URL: https://doi.org/10.1007/b79761.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

Versteeg, H. K. and W. Malalasekera (2007). *An introduction to computational fluid dynamics: the finite volume method*. 2nd ed. OCLC: ocm76821177. Harlow, England ; New York: Pearson Education Ltd.

Wang, Chulin et al. (Dec. 2020). *Physics-Informed Neural Network Super Resolution for Advection-Diffusion Models*. arXiv:2011.02519 [physics]. DOI: 10.48550/arXiv.2011.02519. URL: http://arxiv.org/abs/2011.02519.

Wang, Sifan, Shyam Sankaran, and Paris Perdikaris (Mar. 2022a). *Respecting causality is all you need for training physics-informed neural networks*. arXiv:2203.07404 [nlin, physics:physics, stat]. DOI: 10.48550/arXiv.2203.07404. URL: http://arxiv.org/abs/2203.07404.

Wang, Sifan, Yujun Teng, and Paris Perdikaris (Jan. 2021a). "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks". In: *SIAM Journal on Scientific Computing* 43.5, A3055–A3081. DOI: 10.1137/20M1318043. URL: https://epubs.siam.org/doi/10.1137/20M1318043.

Wang, Sifan, Hanwen Wang, and Paris Perdikaris (Oct. 2021b). "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 384, p. 113938. DOI: 10.1016/j.cma.2021.113938. URL: https://doi.org/10.1016/j.cma.2021.113938.

Wang, Sifan, Xinling Yu, and Paris Perdikaris (Jan. 2022b). “When and why PINNs fail to train: A neural tangent kernel perspective”. en. In: *Journal of Computational Physics* 449, p. 110768. DOI: 10.1016/j.jcp.2021.110768. URL: https://www.sciencedirect.com/science/article/pii/S002199912100663X.

Wang, Z. et al. (Feb. 2018). “Model identification of reduced order fluid dynamics systems using deep learning”. en. In: *International Journal for Numerical Methods in Fluids* 86.4, pp. 255–268. DOI: 10.1002/fld.4416. URL: https://onlinelibrary.wiley.com/doi/10.1002/fld.4416.

Wassing, Simon, Stefan Langer, and Philipp Bekemeyer (2023). “Artificial Viscosity in Physics-Informed Neural Networks for Parametric Compressible Flows”. In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.4353534. URL: https://doi.org/10.2139/ssrn.4353534.

Weller, H. G. et al. (Nov. 1998). “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12.6, pp. 620–631. DOI: 10.1063/1.168744. URL: https://doi.org/10.1063/1.168744.

Wiatrak, Maciej, Stefano V. Albrecht, and Andrew Nystrom (2019). *Stabilizing Generative Adversarial Networks: A Survey*. DOI: 10.48550/ARXIV.1910.00927. URL: https://arxiv.org/abs/1910.00927.

Wight, Colby L. and Jia Zhao (June 2021). “Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks”. In: *Communications in Computational Physics* 29.3, pp. 930–954. DOI: 10.4208/cicp.OA-2020-0086. URL: http://global-sci.org/intro/article_detail/cicp/18571.html.

Wu, Chenxi et al. (Jan. 2023). “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks”. en. In: *Computer Methods in Applied Mechanics and Engineering* 403, p. 115671. DOI: 10.1016/j.cma.2022.115671. URL: https://www.sciencedirect.com/science/article/pii/S0045782522006260.

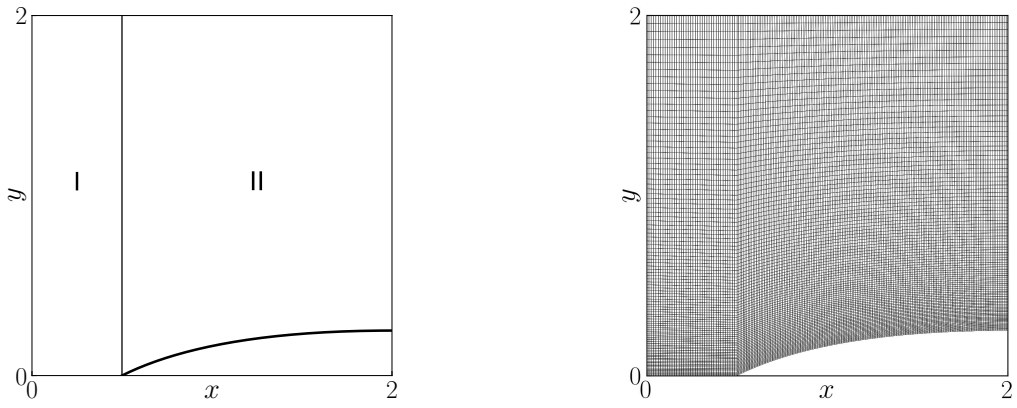
Yadav, Abhay et al. (2017). *Stabilizing Adversarial Nets With Prediction Methods*. DOI: 10.48550/ARXIV.1705.07364. URL: https://arxiv.org/abs/1705.07364.

Zeng, Qi et al. (Oct. 2022). *Competitive Physics Informed Networks*. arXiv:2204.11144 [cs, math]. DOI: 10.48550/arXiv.2204.11144. URL: http://arxiv.org/abs/2204.11144.

A

OpenFOAM simulations

The reference solutions required to calculate the L_2 errors for the problems in Section 6.2 and Section 6.3 are both obtained via OpenFOAM (Weller et al., 1998). For the curved shock, the blocking structure in Fig. A.1a is used to generate a structured mesh via `gmsh`, a tool developed by Geuzaine et al. (2009). Progression is used to refine the cells near the bottom boundaries. A coarse version of the resulting mesh is shown in Fig. A.1b, which only features approximately 20,000 cells for visual purposes. The actual mesh consists of approximately 80,000 cells.



(a) Blocks **(b)** Coarse mesh

Figure A.1: The blocking used to create a structured mesh for the curved shock problem and the resulting mesh.

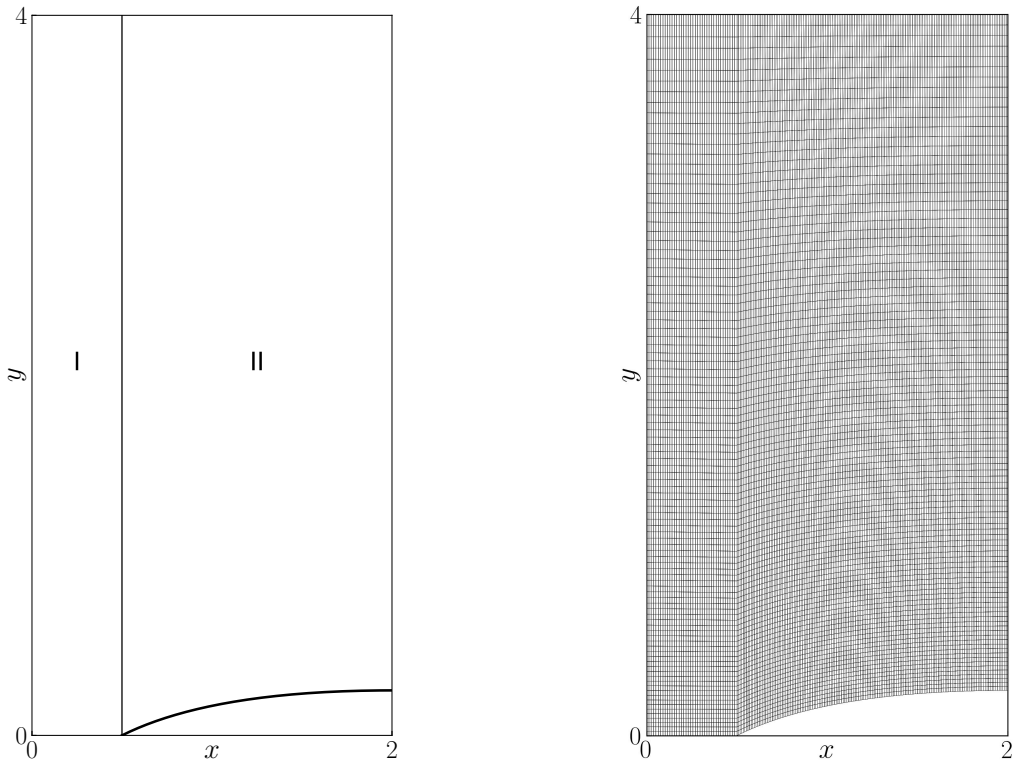
The corresponding boundary conditions are given in Table A.1. The simulation is performed with the `rhoCentralFoam` solver, an unsteady compressible flow solver. Although it can also be used to simulate viscous effects and turbulence, the viscosity has been set to zero. Furthermore, the Courant–Friedrichs–Lewy (CFL) number is set to 0.1, although this is only for consistency with the detached simulation so it can certainly be increased. The simulation is terminated once a steady solution is reasonably obtained, which is approximately at $t = 6$. For more information about the used numerical schemes, transport model and other settings the reader is referred to the source code mentioned in Chapter 5.

Table A.1: Boundary conditions for the fields in the OpenFOAM solver.

	Inlet	Outlet	Bottom	Top	Object
p	fixedValue: 1.000	zeroGradient	symmetryPlane	zeroGradient	zeroGradient
T	fixedValue: 0.003	zeroGradient	symmetryPlane	zeroGradient	zeroGradient
U	fixedValue: (2.958, 0, 0)	zeroGradient	symmetryPlane	zeroGradient	slip

The setup for the detached shock wave problem is identical, except for an inlet velocity decrease and a mesh extension to $y = 4$ as is shown in Fig. A.2. Note that the number of cells is also kept the same,

so the progression is modified to maintain reasonable aspect ratios near the bottom boundaries. The simulation is again terminated once a steady solution is obtained, this time at $t = 30$ since it takes some time before the detachment distance has converged.



(a) Blocks

(b) Coarse mesh

Figure A.2: The blocking used to create a structured mesh for the detached shock problem and the resulting mesh.