The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

PARAMETRIC COMPRESSIBLE FLOW PREDICTIONS USING PHYSICS-INFORMED NEURAL NETWORKS

S. WASSING[1], S. LANGER[1] AND P. BEKEMEYER[1]

[1] German Aerospace Center
Institute for Aerodynamics and Flow Technology - $C^2A^2S^2E$
Lilienthalplatz 7, 38108, Braunschweig, Germany
Email: simon.wassing@dlr.de

Key words: Physics-Informed, Partial Differential Equation, Deep Learning

Abstract. The numerical approximation of solutions to the compressible Euler and Navier-stokes equations is a crucial but challenging task with relevance in various fields of science and engineering. Recently, methods from deep learning have been successfully employed for solving partial differential equations by incorporating the equations into a loss function that is minimized during the training of a neural network. This approach yields a so-called physics-informed neural network which does not rely on a classical discretization and can address parametric problems in a straightforward manner. Therefore, it avoids characteristic difficulties of traditional approaches, such as finite volume methods. This has raised the question, whether physics-informed neural networks may be a viable alternative to conventional methods for computational fluid dynamics. Here, we show a new physics-informed neural network training procedure to approximately solve the two-dimensional compressible Euler equations, which makes use of artificial dissipation during the training process. We demonstrate how additional dissipative terms help to avoid unphysical results and how the additional numerical viscosity can be reduced during training while iterating towards a solution. Furthermore, we showcase how this approach can be combined with parametric boundary conditions. Our results highlight the appearance of unphysical results when solving compressible flows with physics-informed neural networks and offer a new approach to overcome this problem. We therefore expect that the presented methods enable the application of physics-informed neural networks for previously difficult to solve problems.

1 Introduction

The motion of compressible fluids gives rise to nonlinear partial differential equations (PDEs) such as the Euler and Navier-Stokes equations. The numerical solution of these equations is for example essential for the development of future aircraft configurations. Classical simulation algorithms employ a spatial and temporal discretization and calculate solutions with iterative algorithms. Despite a non-negligible progress in efficiency in recent years, long term prospects for further acceleration of these codes seem limited. Especially, transferring these methods to potentially advantageous hardware like graphic processing units has shown to be challenging. Hence, our interest is to investigate numerical methods based on machine learning and deep neural networks to solve PDEs modelling compressible flows. With the rising popularity of machine

learning methods and especially deep neural networks, alternative approaches for the solution of differential equations, based on methods from this rapidly advancing field have become a popular alternative to classical solvers.

One of these techniques are Physics-Informed Neural Networks (PINNs). The fundamental idea of PINNs is to use a neural network as a parametric ansatz function for the approximation of the PDE solution and to optimize the networks parameters by minimizing a loss functional which directly incorporates the differential equations as well as the initial and boundary conditions. During the training of the network, the loss functional is evaluated at typically random points inside of the domain. Such approaches for approximating PDE solutions via neural networks have been proposed decades ago in works of Lagaris [1] and Dissanayake [2]. Even though the idea seems natural due to the ability of universal function approximation of neural networks[3], the approach has only recently gained popularity. Nowadays, the networks can be trained more efficiently due to the availability of high performance GPUs. Moreover, the implementation of such algorithms has become straightforward with software libraries for deep learning, such as Tensorflow [4] and PyTorch [5]. The term PINN has been introduced by Raissi et al. [6] who demonstrated the use of this approach on a number of nonlinear PDEs for forward and inverse problems. Subsequently, PINNs have shown to be applicable for solving stochastic PDEs [7], inverse problems [8] and parametric problems [9]. Various alterations of the vanilla PINN formulation have been proposed and improvements for the implementation of boundary conditions [10, 11], training facilitation [12, 13] and training point selection [14, 15] have been developed. Among many other domains, the flow simulation community has readily adapted PINNs for various cases such as blood flow [11, 16], turbulent convection [17] and aerodynamics of airfoils [18]. Even for compressible flows, there has been some success to employ PINNs for simple forward and inverse problems which exhibit contact discontinuities [19, 20]. For a more extensive overview on the usage of PINNs for fluid dynamics the interested reader is referred to [21].

In this work we focus on the use of PINNs for compressible flows for which state of the art solvers typically rely on finite volume or finite element algorithms. These algorithms have been developed and refined for decades and they are in regular use for industrial problems such as aircraft design. These classical algorithms discretize the domain and thus require the construction of carefully crafted computational grids, so-called meshes. Superficially speaking, a finer mesh will result in a more accurate result but will also increase the computational cost needed. Besides, classical algorithms solve a problem for one specific instance of boundary conditions and/or initial conditions. Therefore, for certain tasks, such as design optimization, multiple evaluations of the solver are necessary. The cumulative computational effort of multiple solver evaluations can be significant and is therefore a possibly limiting factor for the usage for commercially relevant problems.

Compared to classical algorithms, PINNs are fundamentally different because they do not rely on a spatial or temporal discretization in a classical sense. Instead a continuous parametric ansatz-function (a neural network) is evaluated and optimized at (oftentimes randomly distributed) points in the domain. Hence, the task of mesh-generation is replaced by the task to find a point distribution in space and time and an appropriate network dimension (i.e. number and width of layers) which, when combined, result in a fast convergence and satisfactory final accuracy. Compared to classical methods, PINNs can directly tackle parametric problems. A

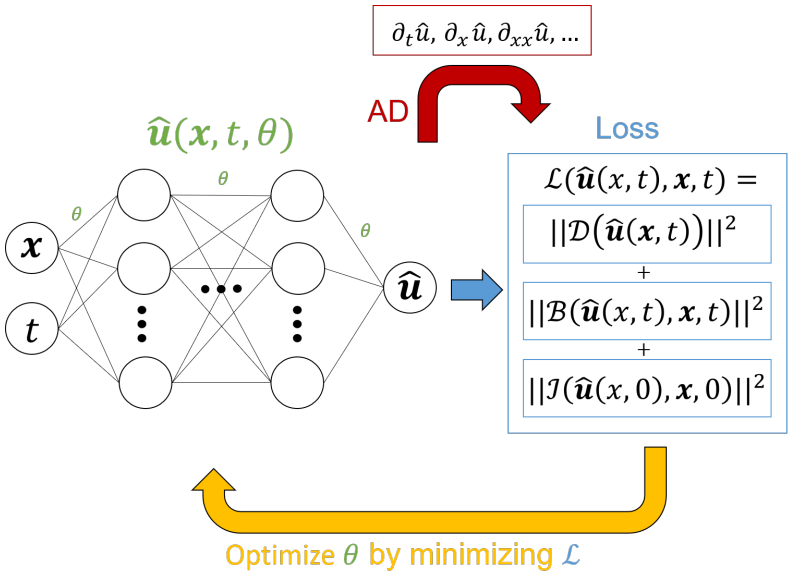

Figure 1: Schematic representation of the physics-informed neural network approach. It consists of a fully connected neural network $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ which is an ansatz for the unknown function $\boldsymbol{u}(\boldsymbol{x}, t)$. The loss for an individual point $\boldsymbol{x}$ in the physical domain and at time t is calculated as the sum of squares of the residual of the PDE, the boundary and the initial condition. The calculation of partial derivatives of $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ with respect to components of $\boldsymbol{x}$ or t can be achieved by using automatic differentiation.

single network can be trained in a continuous parameter space and yield approximate solutions for a whole range of parameter combinations of interest. This ability may have implications for aforementioned usecases like design optimization.

These properties raise the question whether PINNs are a viable alternative to classical algorithms in computational fluid dynamics. In this paper, we take a step towards answering this question, by presenting a new training procedure for PINNs to solve compressible, inviscid flow problems by incorporating artificial viscosity during the training phase. Similarly to classical methods, the use of artificial viscosity helps to avoid unphysical solutions of the Euler equations and stabilizes the training procedure. In Sec. 2 we give a general introduction to the standard PINN approach for solving (parametric) initial and boundary value problems. In Sec. 2.2 we discuss the application of the approach to the compressible Euler equations and explain how the incorporation of artificial dissipation during the training can facilitate convergence when looking at the aerodynamic problem of calculating the flow around a solid object of arbitrary shape.

In Sec. 3 we demonstrate the difficulties which arise when trying to solve compressible flows with PINNs by calculating the flow around a cylinder in the subsonic regime. Furthermore, it is showcased how the use of artificial dissipation can help to overcome these problems. In Sec. 4, we demonstrate how this approach can be extended towards parametric boundary conditions such as variable Mach numbers. In Sec. 5 the presented results are evaluated in a more general context and future implications as well as newly arising questions are discussed.

2 Methods

2.1 Physics-Informed Neural Networks

Physics-informed neural networks are deep neural networks, which are employed to solve a differential equation. A general initial-boundary value problem for the unknown function $\boldsymbol{u}$ on the spatial domain $\Omega \subset \mathbb{R}^d$ and in the time interval $(0, T) \subset \mathbb{R}$ can be defined as:

$$\begin{aligned} \mathcal{D}(\boldsymbol{u}(\boldsymbol{x}, t), \boldsymbol{x}, t) &= 0 \quad && (\boldsymbol{x}, t) \in (\Omega \times (0, T)) \\ \mathcal{I}(\boldsymbol{u}(\boldsymbol{x}, 0), \boldsymbol{x}) &= 0 \quad && \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{u}(\boldsymbol{x}, t), \boldsymbol{x}, t) &= 0 \quad && (\boldsymbol{x}, t) \in (\partial\Omega \times (0, T)), \end{aligned} \tag{1}$$

where $\mathcal{D}$ is a general differential operator and $\mathcal{I}$ and $\mathcal{B}$ are the initial and boundary condition, respectively. The operator $\mathcal{D}$ may include multiple nonlinear differential terms of different order. A neural network $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ is now used to approximate the unknown solution $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t) \approx \boldsymbol{u}(\boldsymbol{x}, t)$. The vector $\boldsymbol{\theta}$ includes neural networks parameters (weights and biases) which have to be adapted during an optimization/training process to find an accurate approximation of the solution. For this, an objective or loss functional $\mathcal{L}$ is defined. The left hand side of Eqs. (1) vanishes on the entire domain $x \in \Omega$ and for all times $t \in (0, T)$ if $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ is a solution to Eqs. 1. Therefore, a simple loss functional could take the form

$$\begin{aligned} \mathcal{L}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)) = &\iint_{\Omega \times (0,T)} \mathcal{D}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t))^2 \ \mathrm{d}\boldsymbol{x}\, \mathrm{d}t + \alpha \int_{\Omega} \mathcal{I}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, 0), \boldsymbol{x})^2 \ \mathrm{d}\boldsymbol{x} + \\ &\beta \iint_{\partial\Omega \times (0,T)} \mathcal{B}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t), \boldsymbol{x}, t)^2 \ \mathrm{d}\boldsymbol{x}\, \mathrm{d}t \end{aligned} \tag{2}$$

which is 0 and thus minimal if $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ is a solution to Eqs. 1. The coefficients α and β scale the importance of the boundary and initial loss term with respect to the residual term. However, since the network $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)$ can only be evaluated at discrete input values, the loss is instead calculated by taking the sum over a representative point distribution inside of the spatial and temporal domain

$$\begin{aligned} \mathcal{L}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t)) &= \mathcal{L}_{\mathrm{Res}} \ + \ \alpha\mathcal{L}_{\mathrm{I}} \ + \ \beta\mathcal{L}_{\mathrm{B}} \\ \mathcal{L}_{\mathrm{Res}} &= \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i, t_i))^2 && \boldsymbol{x}_i \in \Omega \, ; t_i \in (0, T) \\ \mathcal{L}_{\mathrm{I}} &= \alpha \frac{1}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \mathcal{I}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i, 0))^2 && \boldsymbol{x}_i \in \Omega \\ \mathcal{L}_{\mathrm{B}} &= \beta \frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \mathcal{B}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i, t_i))^2 && \boldsymbol{x}_i \in \partial\Omega \, ; t_i \in (0, T) \end{aligned} \tag{3}$$

where N_{Res}, N_{I} and N_{B} are the number of points that evaluate the residual, the initial condition and the boundary condition, respectively. Since for a well defined problem, the bounds of

the domain are known, the generation of these training points can be achieved with quasi-random low discrepancy sequences such as Sobol [22] or Halton [23] or with other methods like Latin Hypercube sampling [24] at little additional cost. For a uniform point distribution, the calculation of the loss functional can be interpreted as a Monte-Carlo integration of Eq. 2 where the normalization by the volume has been dropped. As highlighted in various publications [15, 14, 19], a non uniform distribution of training points or an adaptive training point selection, based on the local residual may accelerate training and improve the final accuracy for certain problems.

The partial derivatives in $\mathcal{D}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i, t_i))$ are calculated, using automatic differentiation. The network parameters are then tuned using an iterative optimization algorithm. The most popular algorithms are variants of stochastic gradient descend such as Adam [25]. During the optimization, the gradient of the loss with respect to the network parameters $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i, t_i))$ has to be calculated. This can again be achieved, using automatic differentiation and the backpropagation algorithm [26]. A schematic representation of a typical PINN is shown in Fig. 1. As usual for neural networks, the optimization problem is generally non-convex and the optimizer can therefore converge to local minima or regions of vanishing gradients. Furthermore, for partial differential equations such as the compressible Euler equations which may not have a unique solution, a minimum of the loss may not always correspond to a physically reasonable solution.

2.1.1 Parametric Problems

One prospect of physics-informed neural networks is the possibility to solve parametric problems. Let τ be a general parameter of the initial and boundary value problem. A PINN approximation $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}, t, \tau)$ of the unknown solution $u(\boldsymbol{x}, t, \tau)$ simply receives τ as an additional input to the neural network alongside $\boldsymbol{x}$ and t. The parameter adds an additional dimension to the input space and thus the training points. The training points are now sampled in a $d+2$-dimensional domain $(\boldsymbol{x}, t, \tau) \in \Omega \times (0, T) \times (\tau_1, \tau_2)$, where τ_1 and τ_2 are the lower and upper bound of the parameter space. The parameter τ can then be incorporated into the calculation of any of the loss terms in Eq. 3. Similarly, this approach can be extended to more than one parameter.

2.2 Approximation of Compressible Flows with PINNs

The inviscid flow of compressible fluids is governed by the Euler equations which describe the conservation of mass, momentum and energy in a continuous fluid. The two-dimensional Euler equations in their differential conservative form are given by

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} = 0,$$
$$\boldsymbol{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \; \boldsymbol{F}_x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho H u \end{pmatrix}, \; \boldsymbol{F}_y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho H v \end{pmatrix} \tag{4}$$

where $\boldsymbol{W}$ is the vector of the conservative variables with the density ρ, the local fluid velocity $\boldsymbol{q} = (u, v)$, and the total specific Energy E. The total enthalpy H is defined as $H = E + \frac{p}{\rho}$. This

system of partial differential equations can be closed by the equations of state of ideal gases, which yield $p = \rho(\kappa - 1)(E - \rho \boldsymbol{q}^2/2)$, with κ being the ratio of specific heats. The Mach number $M = \|\boldsymbol{q}\|_2/a$ is the ration between the velocity and the speed of sound $a = \sqrt{\kappa p/\rho}$.

Compared to the incompressible equations, the more general compressible Euler equations model the flow of an inviscid fluid when the elastic forces are not negligible compared to the inertial forces. The equations are therefore relevant for aerodynamic flow conditions where the Mach number is not close to 0. The compressible Euler equations are a quasilinear system of conservation laws and exhibit weak solutions. However, not any weak solution is physically reasonable. A general principal of systems of conversation laws is that a physically correct solution should arise for a modified system with an additional viscous term at the limit of vanishing viscosity [27].

For aerodynamic problems one is oftentimes interested in the stationary solution. Therefore, the time derivative in Eq. (1) is dropped. When solving this system of equations with a physics-informed neural network, the network outputs can be chosen to approximate the primitive variables $\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}) \approx (\rho(x), u(x), v(x), E(x))$. The residual loss term becomes

$$\mathcal{L}_{\text{Res}} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{F}_x(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i))}{\partial x} + \frac{\partial \boldsymbol{F}_y(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i))}{\partial y} \right)^2 . \tag{5}$$

The partial derivatives are calculated using automatic differentiation. In the following, we focus on the calculation of an inviscid flow around an object with a piecewise smooth surface, such as a cylinder or an airfoil (see Fig. 2). Far from the object the flow field should reach a constant state of $\boldsymbol{W}_\infty = (\rho_\infty, \rho_\infty u_\infty, \rho_\infty v_\infty, \rho_\infty E_\infty)$ which is determined by the Mach number M_∞ and often called far field. Also, the flow should be tangential to the objects surface. This results in a boundary loss term of

$$\mathcal{L}_{\text{B}} = \alpha \left(\frac{1}{N_\infty} \sum_{i=1}^{N_\infty} [\boldsymbol{W}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_i)) - \boldsymbol{W}_\infty]^2 + \frac{1}{N_{\text{obj}}} \sum_{j=1}^{N_{\text{obj}}} [\boldsymbol{q}(\hat{\boldsymbol{u}}(\boldsymbol{\theta}, \boldsymbol{x}_j)) \cdot \boldsymbol{n}_j]^2 \right) \tag{6}$$

with N_∞ points $\boldsymbol{x}_i$ on the boundary of the physical domain and N_{obj} points $\boldsymbol{x}_j$ on the boundary of the object. The surface normals at positions $\boldsymbol{x}_j$ are given by $\boldsymbol{n}_j$.

2.2.1 Artificial Dissipation

A fundamental challenge for PINNs in the search of solutions to the compressible Euler equations lies in finding a physically reasonable minimum of the loss (i.e. an entropy solution). By definition of the loss function (3), any weak solution of Eqs. (4) that fulfills a given set of boundary conditions is a (global) minimum of the loss and thus a possible final state of the network to converge to. Furthermore, the optimizer can always converge to local minima, which correspond to unphysical solutions.

In classical computational methods for the solution of compressible flows, artificial dissipation is introduced in the form of upwind schemes or through a combination of central schemes and explicit dissipative terms [28]. The dissipation smears out discontinuities and has a stabilizing

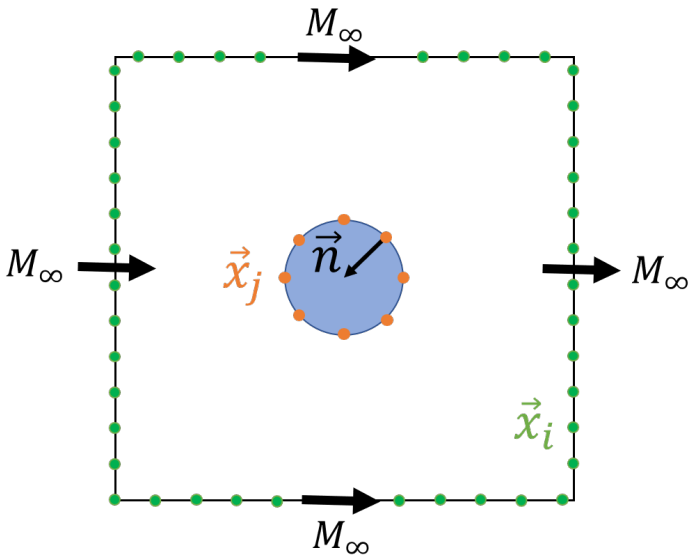

Figure 2: Schematic representation of boundary training point distribution and boundary conditions for flow around an object. The flow in the far field is constrained at the points $\boldsymbol{x}_i$ with $i = 1 \dots N_\infty$. The no-flux boundary condition at the objects surface is enforced at the points $\boldsymbol{x}_j$ with $j = 1 \dots N_{obj}$. The far field is determined by the Mach number M_∞.

effect. A straightforward approach is to add a dissipative term to Eqs. (4) to obtain a similar effect during the PINN training

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} = \eta \Delta \boldsymbol{W} \tag{7}$$

where $\Delta = (\partial_x^2 + \partial_y^2)$ is the Laplacian and η can be interpreted as an artificial viscosity. This can be interpreted as a regularization of the optimization/training problem. Similarly to classical scalar dissipation schemes like the JST-scheme [29], we scale the dissipation locally, based on the spectral radius of the flux jacobians

$$\eta = \nu(a + |\boldsymbol{q}|) \tag{8}$$

where ν is a constant viscosity factor. This artificial viscosity factor ν is chosen empirically. For the evaluation of the additional term in Eq. (7), 14 additional partial derivatives have to be calculated which increases the computational effort significantly. Also, the dissipation changes the physical problem and the resulting solution will disagree with the fully inviscid solution. Essentially, one is no longer solving for an inviscid solution and therefore the fluid is decelerated by shear stresses. Scalar dissipation schemes employ 4th order differences in smooth flow regions which only dampens higher frequency modes of the flow. Second order differences are only used near discontinuities, where a scheme of first order accuracy is required due to Godunov's theorem. An additional fourth order differential term could theoretically be added to Eq. (7). This would however introduce fourth order derivatives which would require additional expensive evaluations of the computational graph. Instead, we follow a different approach and adjust the viscosity

during the training process, to limit the dissipative effect on the final solution. Therefore, we propose a 3 phase training routine:

1. Train with constant viscosity ν until there is no significant change in residual loss
2. Linearly reduce the viscosity ν until a minimal viscosity of ν_0 has been reached
3. Continue the training with constant viscosity ν_0 to fine-tune the result.

The idea of this procedure is to guide the network towards a physical solution (the entropy solution) during the initial training phase. Once the network has converged to a state that resembles a physical but viscous solution, the viscosity can be reduced, since from this point on, the network should be in a state near the entropy solution. It is crucial to no longer train with a very large learning rate during step 2, since large learning rates may allow the network to step away from the neighborhood of the entropy solution once again. Finally, a reasonable final viscosity ν_0 has to be chosen, which is sufficient to stabilize the solution, but which has little dissipative effects on the final result. For certain examples it may even be possible to completely drop the viscous term (i.e. set $\nu = 0$). However, generally it may be sufficient if the viscosity is reduced to a value where the dissipative term becomes very small compared to the other terms.

3 2D Flow around Cylinder

As a testcase, we consider a subsonic two-dimensional inviscid flow around a cylinder. The center of the cylinder is positioned in the origin and has a radius of $r = 0.1$. The physical domain is chosen to be $\boldsymbol{x} = (x, y) \in \Omega = (-1, 1) \times (-1, 1)$. For this kind of problem, a continuous, spatially symmetric solution is expected. Figs. 3 (a)-(b) show a numerical reference solution of the problem for a Mach number of $\mathrm{M} = 0.2$ with far field boundary conditions. The solution was obtained using the DLR in-house finite volume solver TAU [30, 31].

To calculate an approximation of this solution with a PINN, no-flux boundary conditions at the cylinder wall and far-field boundary conditions at the external domain boundary are enforced using Eq. 6.

The weighting factor α is set to $\alpha = 1$. As shown in various publications, such as [32, 12, 13], the (adaptive) weighting of loss terms is an effective technique to accelerate the convergence and improve the accuracy because it can compensate imbalances in the gradients between the different loss terms. However, we have observed that for certain problems, adaptive loss term weighting may lead to instabilities during the early stages of the training. For the presented problems we do not see significant imbalances during the training and are able to achieve satisfying results with a constant weighting factor. Therefore, loss term weighting is not considered herein.

A summary of the parameters within each training phase is shown in the Appendix in Tab.1. This includes the utilized optimizer, the learning rate lr and the viscosity factor ν. The training points for the residual loss are randomly sampled in the entire domain, using a Latin Hypercube sampling. We choose a fully connected neural network of constant layer width with hyperbolic tangent activation functions. For the majority of the training, Adam [25] is used as the optimizer (phase 1 to phase 3.1 in Tab. 1). A brief final training period with the quasi-Newton LBFGS

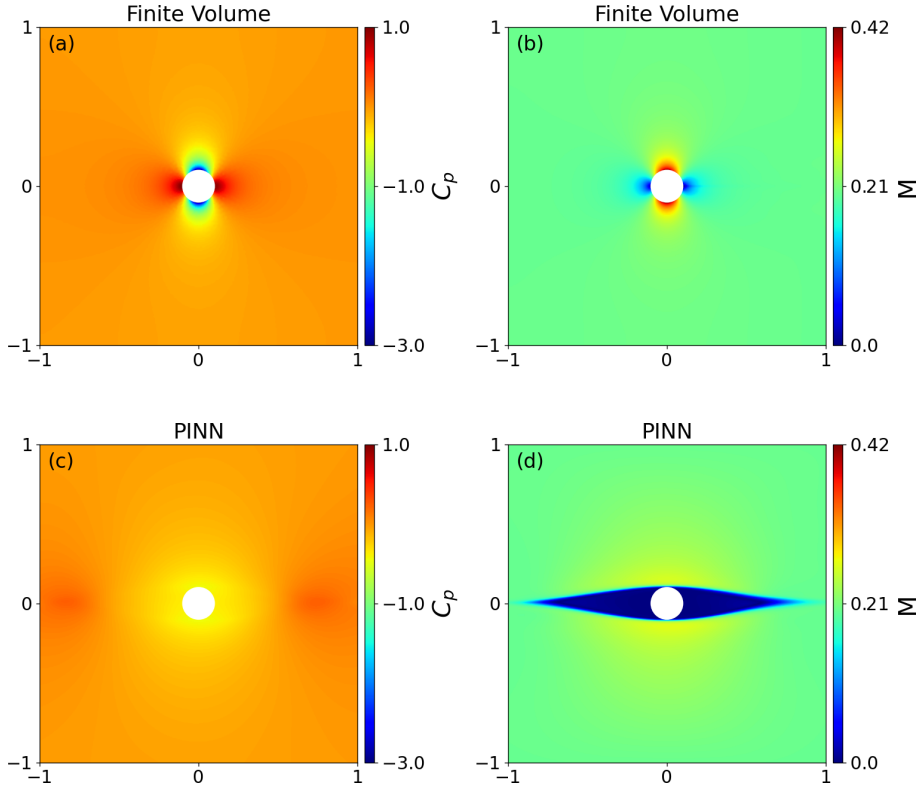

Figure 3: Results for pressure coefficient C_p and local Mach number M for the flow around a cylinder at $M_\infty = 0.2$. Fig. (a)-(b) show the reference solution which was calculated with the DLR in-house finite volume solver TAU. Fig. (c)-(d) show the result that has been achieved with a standard PINN approach without additional dissipative terms. The velocity field shows extreme unphysical changes which extend from the left to the right boundary of the domain and enclose an area of low velocity which contains the cylinder. This results in relatively small pressure gradients, compared to the physical solution.

optimizer has shown to be an efficient way for improving the final accuracy (phase 3.2 in Tab. 1). Since this algorithm requires comparatively more memory and is (in PyTorch) unsuitable for minibatch training, a second set of training points with a reduced number of points N is used. For both point sets, the same number of points are used to represent the boundary of the physical domain and the cylinder ($N = N_\infty = N_\mathrm{obj}$). For the implementation, the python package SMARTy [33] was used, which is a toolbox for surrogate modeling and other data driven tasks. SMARTy supports Tensorflow and PyTorch as backends for the creation and training of neural networks. The presented results were obtained with PyTorch.

Figs. 3 (c)-(d) show the result of a PINN which solves the non-dissipative Eqs. (4). The training parameters correspond to phase 1 in Tab. 1 with a viscosity factor of $\nu = 0$. One can clearly see steep unphysical changes in the velocity field and a flat pressure field. To avoid such results, Eqs. (7) are considered instead. To determine a reasonable viscosity factor ν for this specific testcase, the error of the resulting solution is compared in Fig. 7 for various values of

ν. For very high ν, the dissipative term covers the other terms in Eqs. (7) and the inflow is slowed down significantly, even at large distances to the cylinder. For very low ν, similarly to Fig. 3 (b), the result becomes unphysical again, even though the velocity changes are smeared out due to the remaining dissipation. Lower errors are observed for a wide viscosity range $\nu \in (5, 100) \cdot 10^{-4}$. For all following simulations, a value of $\nu = 7.5 \cdot 10^{-4}$ is selected because the effect of the dissipative term should be minimal but still strong enough to ensure a sufficient level of stabilization. The result at $\nu = 7.5 \cdot 10^{-3}$ is shown in Fig. 5 after the initial training phase 1 (cf. Tab. 1). It resembles the reference solution. However, due to the dissipative loss of energy, the velocity downstream is reduced and the result is non-symmetric.

As proposed in Sec. 2.2, this result can be improved by reducing the viscosity factor ν in a subsequent training phase. After a linear reduction of ν to a value of $\nu_0 = 0$ over 1000 epochs, the training is continued without the viscous term for 10000 epochs. Empiric investigations of different reduction phase lengths, show that it can be relatively short, compared to the initial training phase and that a less steep reduction of ν has little impact on the final accuracy. We choose to fully reduce the viscosity to 0, because it is possible for this specific problem. Based on our experience with different geometries, this is not always the case and a final viscosity factor of $\nu_0 > 0$ is required for other geometries. Fig. 4 shows the development of the total loss and the error of the PINN in comparison to the reference solution during the training.

One can clearly see, how the error is further reduced during the reduction phase. After the reduction of the viscosity, the error continues to decrease, although significantly slower. A final training phase employing the quasi Newton optimizer LBFGS further improves the accuracy. The final result is shown in Fig. 5 (d). Compared to the result after the first training phase, the loss of velocity downstream, caused by the dissipation, is almost completely revoked.

For a quantitative assessment of the errors, we consider the pressure coefficient C_p, the local Mach number M and the density ρ. The mean absolute difference to the reference solution was calculated for a small square that contains the cylinder ($-0.5 < x < 0.5; -0.5 < y < 0.5$). This difference was then normalized with the range of values of the reference solution for each individual quantity. The resulting, normalized errors are shown in Tab. 2. The final errors are highest for C_p and on the order of 2% with the most significant errors observed close to the cylinder. This is expected, since the largest gradients of the solution occur in close proximity to the surface. In comparison to classical methods, the accuracy of the PINN is orders of magnitude worse than traditional solvers. Note that the hyperparameters of the training and the network shape are not optimized, improved accuracy may therefore be possible with different parameters. However, the cost of hyperparameter optimization is further increased by the fact that one has to additionally choose appropriate values for ν, ν_0 and the duration of the different phases. Nevertheless, it is highly unlikely that the presented approach can even come close to the accuracy of a classical solver, even with optimized hyperparameters.

With respect to training times, it is difficult to compare traditional methods and PINNs because PINNs are best trained on graphic chips, while classical algorithms mostly use conventional processors. The presented PINNs were trained with NVIDIA Quadro P2200 and NVIDIA RTX 2080 cards using the PyTorch deep learning library. The resulting training times are on the order of 20 hours. For reference, the reference solutions were calculated on a desktop Intel i7-9700 CPU with convergence times on the order of one hour. The PINN training time is mostly limited by the clock time of the GPU and by the available video memory.

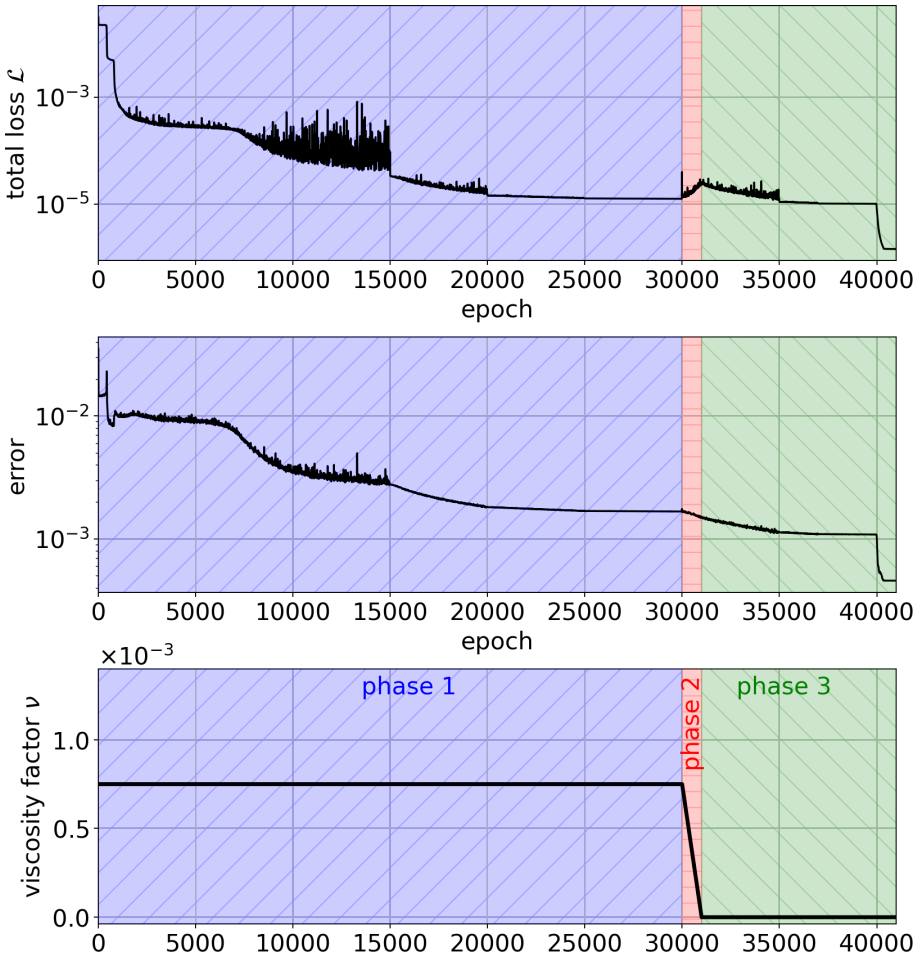

Figure 4: Loss and error history as well as artificial viscosity factor ν during training of PINN for flow around cylinder. The three phases of the applied training procedure are highlighted. Note that during the last 1000 epochs, the LBFGS optimizer was used instead of ADAM (see Tab. 1 for more details).

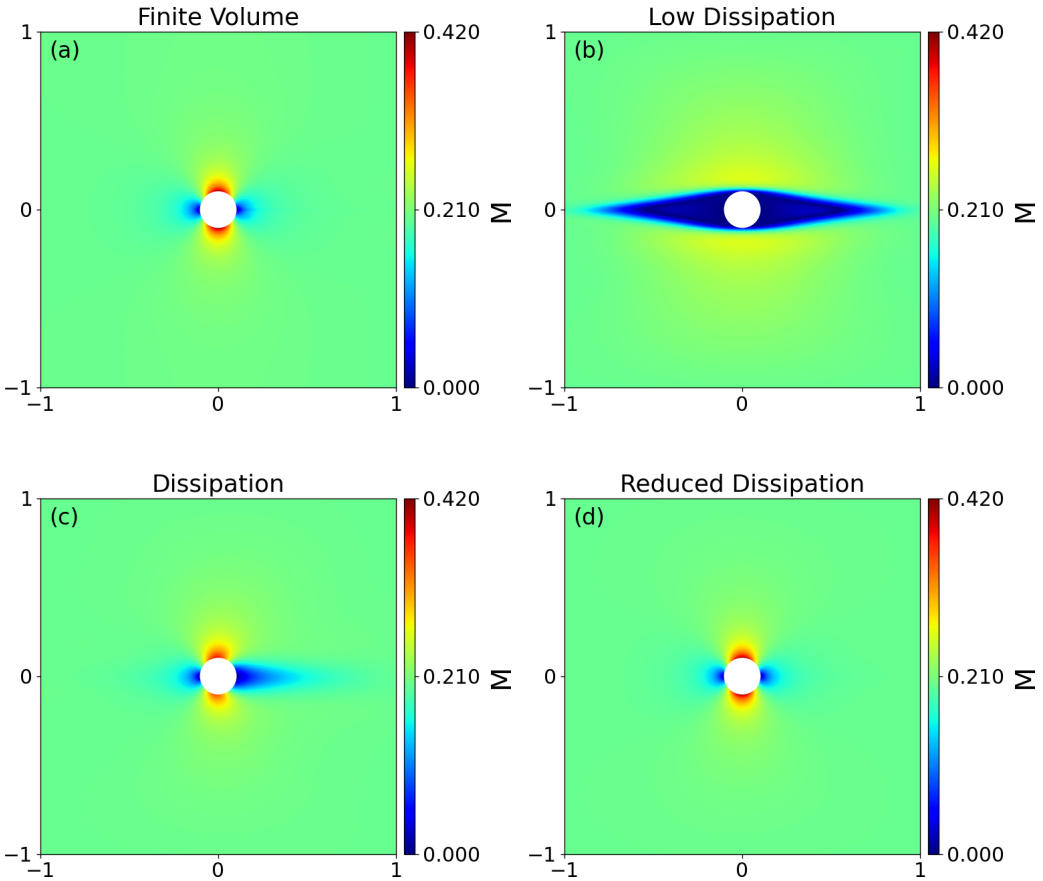

Figure 5: Comparison of the local Mach number M for different artificial viscosity factors in comparison to the reference finite volume result (a). Fig. (b) shows the PINN result for an artificial viscosity factor of $\nu = 10^{-4}$. One can see that the result resembles the unphysical inviscid solution in Fig. 3 (b). However, due to the additional dissipation, the extreme velocity gradients are smeared out. Fig (c) shows the result for a viscosity factor of $\nu = 7.5 \cdot 10^{-4}$. Even though the result is physically reasonable, the artificial dissipation causes a loss of energy downstream of the cylinder. Therefore, the symmetry in flow direction is lost and the downstream velocity is reduced. Fig (d) shows the result of the same training run as Fig. (c), after gradually reducing the viscosity. One can see that the effect of the dissipative term is mostly revoked while the unphysical result of Fig. (b) is avoided.

4 Parametric 2D Flow

As a parametric problem we again consider the flow around a cylinder, but with variable parameterized boundary conditions. Similarly to before, a cylinder of radius $r = 0.1$ is positioned with its center in the origin of the domain $\Omega = (-1, 1) \times (-1, 1)$. The network prediction receives the Mach number M_∞ as an additional input. The training points are therefore sampled in a three-dimensional domain where the third dimension corresponds to the parameter space. We choose a parameter space of $M_\infty \in (0.2, 0.4)$. The upper limit of the Mach number of $M_\infty = 0.4$ is slightly below the critical Mach number of the cylinder, at which the velocity locally exceeds the speed of sound. The artificial viscosity factor of $\nu = 7.5 \cdot 10^{-4}$ is used based on the previous investigations. Since the actual artificial viscosity in Eq. (8) is scaled with the local velocity, the strength of the dissipation naturally increases at higher Mach numbers. The neural network dimensions are increased, compared to Problem I, to account for the increased complexity of the problem. A summary of the adapted hyperparameters is shown in Tab. 1. The resulting predictions for the pressure field for two different Mach numbers are shown in Fig. 6. One can see that even at $M_\infty = 0.4$, close to the critical Mach number, the results are visually indistinguishable to the reference solution. The plot of the absolute error reveals that inaccuracies are most prominent on the cylinder surface. Overall the errors at $M_\infty = 0.2$ (Fig.6 (c)) are lower and one can observe a chessboard like distribution. This indicates, that a higher training point density may have further improved the prediction because the solution is not fitted properly at locations where only a few training points are nearby. Furthermore, one can observe how an inaccurate local prediction of the flow field on the upper right of the cylinder and on the left propagates into the flow field further away from the cylinder. For $M_\infty = 0.4$, the result is particularly inaccurate at the top and bottom of the cylinder, where the local flow velocity comes close to the speed of sound. Results of similar accuracy are achieved for the entire parameter space. Compared to the non-parametric problem, the training effort is only marginally increased primarily due to the increased network dimensions and the training times are on the order of 20 hours. The accuracy of the results is still acceptable although slightly higher than for the non-parametric PINN in Sec. 3. A comparison of the normalized errors at $M_\infty = 0.2$ is shown in Tab. 2. The errors were calculated as explained in Sec. 4. Again, the hyperparameters are not optimized and higher accuracies may be possible when employing hyperparameter optimization.

5 Conclusion

Physics-informed neural networks are a method which uses deep learning to solve partial differential equations by incorporating the residuals of the equations into the objective function. The Euler equations are a system of conservation laws, which describe the dynamics of inviscid fluids and are naturally difficult to solve numerically. Here we showed that, similarly to classical numerical methods, the use of artificial dissipation in a PINN's loss function can help to find physically correct solutions and avoid unphysical results. The artificial viscosity can be reduced during the training to circumvent the effect of the dissipative term on the final solution. The presented method may open up new possibilities for the use of PINNs for similar inviscid problems, which were previously unsolvable using vanilla PINNs. Furthermore, we showed how

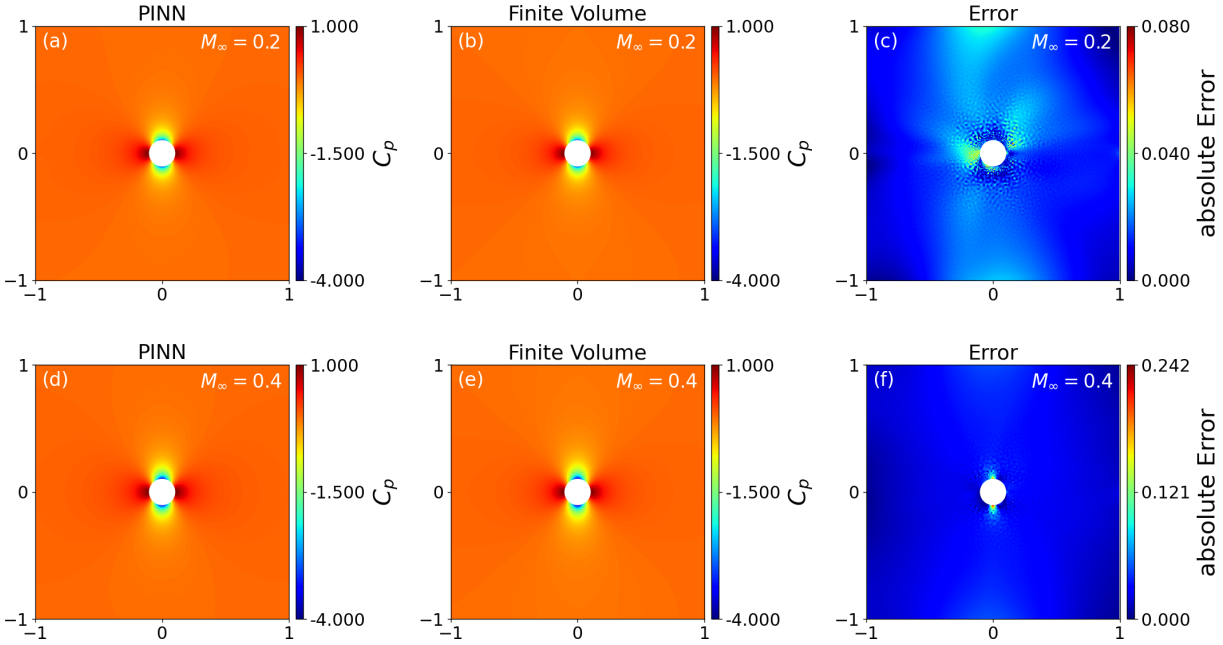

Figure 6: Comparison of parametric PINN solution with a reference finite volume result for different Mach numbers. The absolute errors between the reference and the PINN solution is shown in Figs. (c) and (f). Note the difference in the error scales.

the concept of artificial dissipation can also be applied to problems with parametric boundary conditions.

The proposed implementation of artificial viscosity is very simplistic and one can think of many ways to improve this approach. Similarly to artificial matrix-dissipation schemes in classical CFD methods [34, 35, 36, 37], a yet to be developed matrix-dissipation model for η might reduce the influence of the dissipative term during training, while maintaining its stabilizing properties. Also, recently an alternative regularization scheme based on an entropy criteria has been proposed for solving hyperbolic problems with PINNs which might be an additional measure to avoid unphysical results, when solving the compressible Euler equations, especially at higher Mach numbers [38, 39].

The presented approach of parametric boundary conditions can, in theory, be extended to multiple parameters (e.g. with variable Mach number and angle of attack for airfoils) or even for parameterized geometries. Therefore, it could be applicable for tasks such as design optimization which traditionally require many solver evaluations. Whether a sufficient level of accuracy can be reached for more complex multi-parameter problems remains to be seen.

The presented results for the flow around the cylinder raise the question if the proposed training procedure can deliver similar results for more complex geometries of the object. Naturally, airfoils are the next logical step. An additional point of future interest is the behavior of the presented approach in the transsonic regime, where the local velocity exceeds the speed of sound. Mao et al. [19] have demonstrated that PINNs have the ability to find solutions of the compressible Euler equations which exhibit contact discontinuities and shock waves. However,

since the simplistic artificial dissipation in the proposed procedure will smoothen discontinuities, it remains to be seen whether a reasonably accurate representation of compression shocks in the transsonic regime is still possible.

An aspect of PINNs that has not been considered so far is that higher fidelity solution data can be incorporated into the objective function as an additional loss term. This opens the possibility to directly combine PINNs with classical solvers or to even incorporate experimental data into the loss. For parametric problems, such a hybrid-data-driven approach may improve accuracy and even speed up the convergence during training. This flexibility may open up new possibilities for physics-informed reduced order modeling for higher dimensional parametric flows. We have seen that a very simple parametric boundary condition is straightforward to implement and satisfying results can be achieved without a significant increase in model complexity or training effort. While this may be the case for a single parameter, it remains to be seen if this is still the case for multiple parameters (e.g. parametrization of the object shape, Mach number and angle of attack). The combination of PINNs with pointwise higher fidelity data in the parameter space may be able to alleviate the additional effort that is required for the training of PINNs for such high-dimensional problems.

Overall, the presented results and other work indicate that the precision and speed of state-of-the-art finite volume solvers for compressible flows will not be reached with PINNs in the near future. One has to consider, that computational fluid dynamics solvers have been actively developed for decades and are highly optimized in terms of speed, accuracy and robustness. Right now it is difficult to evaluate the maximum potential performance of PINNs as PDE solvers, since many aspects, like best practices for training parameters and training point distributions, are simply not established yet. Even though PINNs may not outperform classical algorithms for simple forward problems, the described features of PINNs make them applicable where classical methods are lacking.

Appendix

Table 1: Summary of neural network and training parameters.

Sec.	**3. 2D Flow around Cylinder**	**4. Parametric 2D Flow**
hidden layers	8	8
neurons/layer	20	30
activation	tanh	tanh
phase 1	Adam $\nu = 7.5 \cdot 10^{-4}$ $N = 20000$ batchsize = 2500 15000 epochs w. $lr = 10^{-3}$ 5000 epochs w. $lr = 10^{-4}$ 5000 epochs w. $lr = 10^{-5}$ 5000 epochs w. $lr = 10^{-6}$	Adam $\nu = 7.5 \cdot 10^{-4}$ $N = 40000$ batchsize = 8000 15000 epochs w. $lr = 10^{-3}$ 5000 epochs w. $lr = 10^{-4}$ 5000 epochs w. $lr = 10^{-5}$
phase 2	Adam reduce $\nu = 7.5 \cdot 10^{-4}$ to $\nu_0 = 0$ $N = 20000$ batchsize = 2500 1000 epochs w. $lr = 10^{-4}$	Adam reduce $\nu = 7.5 \cdot 10^{-4}$ to $\nu_0 = 0$ $N = 40000$ batchsize = 8000 1500 epochs w. $lr = 10^{-4}$
phase 3.1	Adam $\nu = \nu_0 = 0$ $N = 20000$ batchsize = 2500 4000 epochs w. $lr = 10^{-4}$ 2000 epochs w. $lr = 10^{-5}$ 3000 epochs w. $lr = 10^{-6}$	Adam $\nu = \nu_0 = 0$ $N = 40000$ batchsize = 8000 3500 epochs w. $lr = 10^{-4}$ 5000 epochs w. $lr = 10^{-5}$
phase 3.2	LBFGS $\nu = \nu_0 = 0$ $N = 5000$ 1000 epochs	LBFGS $\nu = \nu_0 = 0$ $N = 5000$ 1500 epochs

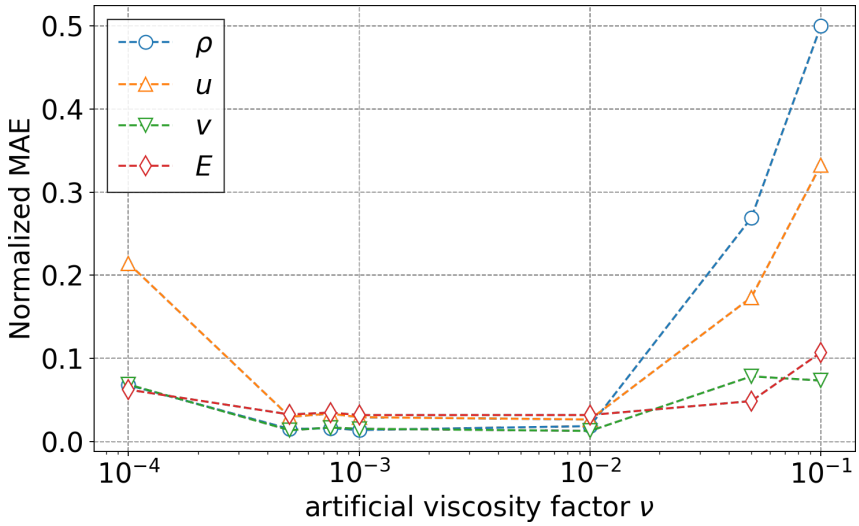

Figure 7: Normalized mean absolute error for the primitive variables of the Euler equations. The error was calculated in comparison to the reference solution for a small square that encloses the cylinder $(-0.5 < x < 0.5, -0.5 < y < 0.5)$. The error was then normalized, based on the maximum deviation of the reference solution from the average value of the primitive variables in the entire domain. The errors are similarly low for a wide range of viscosities $\nu \in (5, 100) \cdot 10^{-4}$. A more accurate identification of the minimum is not necessary because the viscosity is adapted during the training process.

Table 2: Percentage errors at $M_\infty = 0.2$.

Sec.	**3. 2D Flow around Cylinder**	**4. Parametric 2D Flow**
C_p	2.24%	2.24%
M	0.21%	0.26%
ρ	0.33%	0.39%

References

[1] I. E. Lagaris, A. Likas, and D. I. Fotiadis. "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations". In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000. ISSN: 10459227. DOI: 10.1109/72.712178. URL: http://arxiv.org/pdf/physics/9705023v1.

[2] M. W. M. G. Dissanayake and N. Phan-Thien. "Neural-network-based approximations for solving partial differential equations". In: *Communications in Numerical Methods in Engineering* 10.3 (1994), pp. 195–201. ISSN: 10698299. DOI: 10.1002/cnm.1640100303.

[3] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8.

[4] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[5] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 3.12.2019. URL: http://arxiv.org/pdf/1912.01703v1.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "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 (2019), pp. 686–707. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.10.045.

[7] Dongkun Zhang, Ling Guo, and George Em Karniadakis. *Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks*. 3.05.2019. URL: http://arxiv.org/pdf/1905.01205v2.

[8] Maziar Raissi et al. "Deep learning of vortex-induced vibrations". In: *Journal of Fluid Mechanics* 861 (2019), pp. 119–137. ISSN: 0022-1120. DOI: 10.1017/jfm.2018.872.

[9] Oliver Hennigh et al. "NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework". In: *Computational Science – ICCS 2021*. Ed. by Maciej Paszynski et al. Cham: Springer International Publishing, 2021, pp. 447–461. ISBN: 978-3-030-77977-1.

[10] Hengjie Wang et al. "Mosaic Flows: A Transferable Deep Learning Framework for Solving PDEs on Unseen Domains". In: *Computer Methods in Applied Mechanics and Engineering* 389.1 (2022), p. 114424. ISSN: 00457825. DOI: 10.1016/j.cma.2021.114424. URL: http://arxiv.org/pdf/2104.10873v2.

[11] Luning Sun et al. "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data". In: *Computer Methods in Applied Mechanics and Engineering* 361 (2020), p. 112732. ISSN: 00457825. DOI: 10.1016/j.cma.2019.112732.

[12] Suryanarayana Maddu et al. *Inverse-Dirichlet Weighting Enables Reliable Training of Physics Informed Neural Networks*. 2.07.2021. URL: http://arxiv.org/pdf/2107.00940v1.

[13] Sifan Wang, Yujun Teng, and Paris Perdikaris. *Understanding and mitigating gradient pathologies in physics-informed neural networks*. 13.01.2020. URL: http://arxiv.org/pdf/2001.04536v1.

[14] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. "Efficient training of physics–informed neural networks via importance sampling". In: *Computer-Aided Civil*

and Infrastructure Engineering 36.8 (2021), pp. 962–977. ISSN: 1093-9687. DOI: 10.1111/mice.12685.

[15] Lu Lu et al. *DeepXDE: A deep learning library for solving differential equations*. 10.07.2019. URL: http://arxiv.org/pdf/1907.04502v2.

[16] Amirhossein Arzani, Jian-Xun Wang, and Roshan M. D'Souza. "Uncovering near-wall blood flow from sparse data with physics-informed neural networks". In: *Physics of Fluids* 33.7 (2021), p. 071905. ISSN: 1070-6631. DOI: 10.1063/5.0055600.

[17] Didier Lucor, Atul Agrawal, and Anne Sergent. *Physics-aware deep neural networks for surrogate modeling of turbulent natural convection*. 5.03.2021. URL: http://arxiv.org/pdf/2103.03565v1.

[18] Hamidreza Eivazi et al. *Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations*. 22.07.2021. URL: http://arxiv.org/pdf/2107.10711v1.

[19] Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. "Physics-informed neural networks for high-speed flows". In: *Computer Methods in Applied Mechanics and Engineering* 360 (2020), p. 112789. ISSN: 00457825. DOI: 10.1016/j.cma.2019.112789.

[20] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems". In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028. ISSN: 00457825. DOI: 10.1016/j.cma.2020.113028.

[21] Shengze Cai et al. *Physics-informed neural networks (PINNs) for fluid mechanics: A review*. 20.05.2021. URL: http://arxiv.org/pdf/2105.09506v1.

[22] I. M. Sobol'. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *USSR Computational Mathematics and Mathematical Physics* 7.4 (1967), pp. 86–112. ISSN: 0041-5553. DOI: 10.1016/0041-5553(67)90144-9. URL: https://www.sciencedirect.com/science/article/pii/0041555367901449.

[23] J. H. Halton. "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals". In: *Numerische Mathematik* 2.1 (1960), pp. 84–90. ISSN: 0945-3245. DOI: 10.1007/BF01386213.

[24] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), p. 239. ISSN: 00401706. DOI: 10.2307/1268522.

[25] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 22.12.2014. URL: http://arxiv.org/pdf/1412.6980v9.

[26] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 0028-0836. DOI: 10.1038/323533a0.

[27] Lawrence C. Evans. *Partial differential equations*. Reprinted with corr. Vol. 19. Graduate studies in mathematics. Providence, RI: American Mathematical Society, 2008. ISBN: 0821807722.

[28] Jiri Blazek. *Computational fluid dynamics: Principles and applications*. Third edition. Amsterdam, Boston, and Heidelberg: Butterworth-Heinemann an imprint of Elsevier, 2015. ISBN: 9780080999951. URL: https://www.loc.gov/catdir/enhancements/fy1701/2015932012-d.html.

[29] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes". In: *14th Fluid and Plasma Dynamics Conference*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 1981. DOI: 10.2514/6.1981-1259.

[30] Stefan Langer, Axel Schwöppe, and Norbert Kroll. "Investigation and Comparison of Implicit Smoothers Applied in Agglomeration Multigrid". In: *AIAA Journal* 53.8 (2015), pp. 2080–2096. ISSN: 0001-1452. DOI: 10.2514/1.J053367.

[31] Stefan Langer, Axel Schwöppe, and Norbert Kroll. "The DLR Flow Solver TAU - Status and Recent Algorithmic Developments". In: *Proc. 52nd Aerospace Sciences Meeting, January 2014*. Conference Proceeding Series. AIAA, 2014. DOI: 10.2514/6.2014-0080.

[32] Xiaowei Jin et al. "NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations". In: *Journal of Computational Physics* 426 (2021), p. 109951. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109951. URL: http://arxiv.org/pdf/2003.06496v1.

[33] Philipp Bekemeyer et al. "Data-Driven Aerodynamic Modeling Using the DLR SMARTy Toolbox". In: *AIAA Aviation Conference 2022 (Paper accepted for publication)* (2022).

[34] E. Turkel. "Improving the accuracy of central difference schemes". In: *11th International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics*. Ed. by D. L. Dwoyer and M. Y. Hussaini. Vol. 323. Springer-Verlag, 1989.

[35] R. C. Swanson and E. Turkel. "Artificial dissipation and central difference schemes for the Euler and Navier-Stokes equations". In: *AIAA Paper, AIAA-1987-1107* (1987).

[36] Stefan Langer. "Agglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids". In: *Journal of Computational Physics* 277.0 (2014), pp. 72–100. ISSN: 00219991. DOI: 10.1016/j.jcp.2014.07.050. URL: http://www.sciencedirect.com/science/article/pii/S0021999114005439.

[37] Stefan Langer. "Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems". In: *International Journal for Numerical Methods in Fluids* 69.2 (2012), pp. 332–352. ISSN: 1097-0363. DOI: 10.1002/fld.2561.

[38] Ravi G. Patel et al. "Thermodynamically consistent physics-informed neural networks for hyperbolic systems". In: *Journal of Computational Physics* 449 (2022), p. 110754. ISSN: 00219991. DOI: 10.1016/j.jcp.2021.110754.

[39] Ameya D. Jagtap et al. *Physics-informed neural networks for inverse problems in supersonic flows*. 24.02.2022. URL: http://arxiv.org/pdf/2202.11821v1.