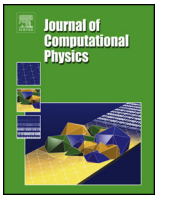
Journal of Computational Physics 481 (2023) 112041

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



Physics informed neural networks: A case study for gas transport problems

Erik Laurin Strelow *, Alf Gerisch, Jens Lang, Marc E. Pfetsch

Technical University Darmstadt, Department of Mathematics, Dolivostraße 15, 64293 Darmstadt, Germany

ARTICLE INFO

Article history:
Received 15 March 2022
Received in revised form 28 December 2022
Accepted 25 February 2023
Available online 5 March 2023

Keywords:
Physics informed neural network
Multi-criteria optimization
Gas flow
Euler equations
Conservation laws

ABSTRACT

Physics informed neural networks have been recently proposed and offer a new promising method to solve differential equations. They have been adapted to many more scenarios and different variations of the original method have been proposed. In this case study we review many of these variations. We focus on variants that can compensate for imbalances in the loss function and perform a comprehensive numerical comparison of these variants with application to gas transport problems. Our case study includes different formulations of the loss function, different algorithmic loss balancing methods, different optimization schemes and different numbers of parameters and sampling points. We conclude that the original PINN approach with specifically chosen constant weights in the loss function gives the best results in our tests. These weights have been obtained by a computationally expensive random-search scheme. We further conclude for our test case that loss balancing methods which were developed for other differential equations have no benefit for gas transport problems, that the control volume physics informed formulation has no benefit against the initial formulation and that the best optimization strategy is the L-BFGS method.

1. Introduction

Natural gas networks [4] transport gas over very large distances from multiple sources to multiple consumers to satisfy the gas demand. There are various elements in the gas network that control the gas flow such as compressors, valves or resistors. Crucially, compressors consume gas itself. Finding the most efficient operation while satisfying the demand is a non-trivial task. This is not only true in practice, but also if one considers the formulation of this task as a mathematical optimization problem. To solve this problem, one needs to bring together different disciplines and techniques such as modeling, simulation and optimization. One important element is the simulation of the gas flow in the network. Here, numerical methods exist to simulate gas flow with a very high accuracy [6,1]. However, the simulation task is one part of the optimization procedure and usually many simulations with varying controls are required to solve the optimization problem. Therefore, the reduction of the simulation cost is key to efficiently solving the optimization problem.

One well understood approach for gas network simulations is to apply methods that are adaptive in space, time and, most importantly, in the models applied to describe the gas flow [1]. Regarding the last point, depending on the dynamics

* Corresponding author.
E-mail addresses: strelow@mathematik.tu-darmstadt.de (E.L. Strelow), gerisch@mathematik.tu-darmstadt.de (A. Gerisch), lang@mathematik.tu-darmstadt.de (J. Lang), pfetsch@mathematik.tu-darmstadt.de (M.E. Pfetsch).

https://doi.org/10.1016/j.jcp.2023.112041

of the gas flow, one can use different models which trade accuracy for simplicity. Gas flow in a single pipe is usually modeled by a system of conservation laws in one space dimension, that is

$$u_t + F(u)_x = 0\,, \quad (x,t) \in [0,L] \times [0,T] \tag{1}$$

for a state vector $u(x,t)$ and flux function $F(u)$ [7, Chapter 14]. The full Euler equations are the most accurate model. Here, we have

$$u_{\mathrm{e}} = \begin{pmatrix} \rho \\ \rho v \\ \rho E \end{pmatrix}, \quad F_{\mathrm{e}}(u_{\mathrm{e}}) = \begin{pmatrix} \rho v \\ \rho v^2 + p \\ v(\rho E + p) \end{pmatrix}, \quad p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho v^2\right), \tag{2}$$

where ρ is the density, v the velocity, E the energy and p the pressure. For natural gas, one usually assumes $\gamma = 1.4$.

However, under certain conditions and especially when considering a pipeline system, one can assume that the gas has constant entropy. Then one can apply the isentropic Euler equations with

$$u_{\mathrm{ise}} = \begin{pmatrix} \rho \\ \rho v \end{pmatrix}, \quad F_{\mathrm{ise}}(u_{\mathrm{ise}}) = \begin{pmatrix} \rho v \\ \rho v^2 + p \end{pmatrix}, \quad p = \kappa \rho^{\gamma}\,, \tag{3}$$

where κ depends on the initial entropy. Advancing this idea further, one derives a hierarchy of gas models. Applying simplified models whenever possible can reduce the simulation time significantly. Still, there is a need for further improvements to handle real large-scale gas networks and more complex scenarios such as optimization.

A further possibility to speed up the simulation is to gear the numerical scheme more closely towards the concrete gas simulation task and utilize the fact that the simulations only differ in the control of the gas network. The result of this procedure is called a reduced method and one prominent example is the reduced basis method. Unfortunately, the reduced basis method is not well suited for the transport nature of the gas flow. In general, it is very challenging to build a reduced method for gas flow tasks since an enabling underlying structure is unknown. For recent developments, see for example [3]. Here, we want to explore a novel route to build a reduced method for gas flow tasks. We will apply physics informed neural networks (PINNs) for the approximation of gas flow models. The application of PINNs in this context has some key advantages which we outline as follows.

1. As briefly mentioned above, finding the right representation of a reduced method to approximate the solution of a transport problem is challenging and an open problem. Since neural networks can approximate continuous functions arbitrarily well, one can expect that the optimization procedure finds a representation that is accurate enough to create gas flow schedules in nearly real-time.
2. By applying machine learning techniques, a physics informed approach has been proposed in [13] and has shown its success in approximating solutions of differential equations. In contrast to the reduced basis methods, the physics informed approach incorporates directly the differential equation itself and specific precomputed solutions are not necessary to obtain the reduced model.
3. Neural networks can easily be extended to realize an additional control vector as input, see for example [8].

In this paper we focus on a specific application which we simulate with PINNs: PDE models of the gas flow in pipes. Our work contributes to the task of designing efficient simulation techniques for these models, which are a crucial building block for the simulation of complex gas networks. Whereas many ideas and techniques presented in this work are transferable to other PDE systems and thus applications, the investigation of such generality is beyond the scope of this article.

PINNs are a quite recent and very active research field and many extensions to the original method have been proposed in the literature. An optimal standard procedure to obtain PINNs for gas flow tasks has not been established yet. For our exploration it is critical to investigate how the different proposed extensions perform for gas flow in a pipe and understand what results in terms of accuracy can be expected. Due to the lack of comparisons between the extensions, this is not yet clear.

The objective of this present study is to perform a comprehensive numerical comparison between many of these extensions to close this knowledge gap. This includes two formulations of the loss function which are minimized over the parameter space of the neural network to obtain the approximation. One formulation is based on the differential form of the conservation law (1) and the other one is based on the integral form of the conservation law. Both formulations differ in the way the initial and boundary data are treated. In the case of the differential form this leads to a multi-objective optimization problem. In our case study we include recently proposed extensions that add weights that are determined during the training process into the loss function to balance the objectives. We further include, as a baseline, the original physics informed formulation as well as a random-search procedure to determine the weights.

We evaluate the performance of these approaches with respect to accuracies achieved for two gas flow benchmark problems varying in the model complexity as well as their initial and boundary conditions.

The loss function formulation, the weights in the loss function and the loss balancing methods constitute a subset of a larger number of hyperparameters required to train a neural network. In addition, we also include more hyperparameters into our case study: different minimization strategies, the number of layers of the network, the number of neurons per

layer, and the number of sampling points in the loss function. Whereas this is not exhaustive it provides important insight into a relevant subset of hyperparameters for our application.

This article is structured as follows. In the next section, we define two test problems as benchmarks. In Section 3, we describe how physics informed neural networks can be used to approximate gas flow by introducing both loss function formulations. In Section 4 we present the loss-balancing extensions. We give details and compare the outcome of the different approaches applied to the two benchmark problems in Section 5. Finally, in Section 6, we state our conclusions.

2. Two test problems for gas transport in a pipe

Throughout this article, we will consider two test problems for one pipe and use the full and the isentropic Euler equations shown above in (2) and (3), respectively.

Problem 1 *(Euler equations).* For this problem, we consider the full Euler equations on the domain $(x,t) \in [0,1] \times [0,2]$. That is, the solution u_e should fulfill (1) with $F = F_e$ and $u = u_e$. To complete this problem, we consider discontinuous initial data and constant Dirichlet boundary data as in [9]. We extract them from the exact solution

$$\rho(x,t) = \begin{cases} 1.4 & \text{if } x < 0.5 + 0.1t\,, \\ 1.0 & \text{if } x > 0.5 + 0.1t\,, \end{cases} \qquad v(x,t) = 0.1\,, \qquad p(x,t) = 1.0\,. \tag{4}$$

The boundary condition should be satisfied at the left ($x = 0$) and right boundary ($x = 1$) by each component of the state vector.

The initial discontinuity in the density is transported in time. As a consequence, the main features of the solution strongly depend on the initial values – a fact which has to be identified by the neural network.

Problem 2 *(Isentropic Euler equations).* For this problem, we consider the isentropic Euler equations. That is, the solution u_{ise} should fulfill (1) with $F = F_{ise}$ and $u = u_{ise}$ on the computational domain $(x,t) \in [0,4] \times [0,2]$. We take an initial state with constant density and no-flow,

$$\rho(x,0) = 2 \qquad \text{and} \qquad \rho v(x,0) = 0 \qquad \text{for } x \in [0,4]\,. \tag{5}$$

Now gas is extracted at the right boundary and thus the flow is increased. In addition, the density remains constant at the left boundary, yielding the boundary conditions

$$\rho(0,t) = 2 \qquad \text{and} \qquad \rho v(4,t) = 0.1t \qquad \text{for } t \in [0,2]\,. \tag{6}$$

We compute a highly resolved reference solution for this problem with the implicit box method [6] on a very fine mesh. Here, the dynamics of the solution depends on the boundary data, which forms another challenge for the neural network.

3. Approximating gas flow with neural networks

3.1. Neural networks

To approximate the gas flow, we design a (deep) neural network

$$h(x,t;\theta) = (h_n \circ \sigma \circ h_{n-1} \circ \ldots \sigma \circ h_2 \circ \sigma \circ h_1)(x,t)\,, \tag{7}$$

where $\sigma = \tanh$ is the activation function, and $h_i(z_i) = A_i z_i + b_i$ with $z_1 = (x,t)^\intercal$ are affine transformations. The neural network is parameterized by the weights A_i and biases b_i of its n layers. Then, let $\theta = (A_1, b_1, A_2, b_2, \ldots, A_n, b_n)$ be all trainable parameters of the neural network.

The number of parameters of the neural network is determined by the number of neurons k in the hidden layers and the number of layers n. We assume that all hidden layers have the same number of neurons k and thus have $A_2, \ldots, A_{n-1} \in \mathbb{R}^{k \times k}$. The sizes of A_1, A_n are accordingly chosen to match the size of the input and output, respectively.

Ideally, the neural network h predicts directly the state vector u_e or u_{ise}, respectively. However, neural networks might not preserve the positivity of the density. This needs to be considered to avoid nonphysical approximations.

For Problem 1, if the neural network predicts ρ, ρv, ρE, we cannot rule out divisions by zero to compute $v = (\rho v)/\rho$. This is not only a theoretical problem, but shows up in the implementation. In the case of the full Euler equations, it is possible to proceed as follows. The neural network predicts the approximations $\overline{\rho}$, $\overline{v}$ and $\overline{p}$. Based on these quantities and using the equation of state, all the remaining quantities can be derived with well defined computations. More formally, we set the output of the neural network to

$$\begin{pmatrix} \overline{\rho} \\ \overline{v} \\ \overline{p} \end{pmatrix} := h(x,t;\theta)\,. \tag{8}$$

Then, based on the output of the neural network, we can define the approximation of the state vector $\overline{u}_{\mathrm{e}}$ and an altered flux function $\overline{F}_{\mathrm{e}}$

$$\overline{u}_{\mathrm{e}}(h) = \begin{pmatrix} \overline{\rho} \\ \overline{\rho} \cdot \overline{v} \\ \overline{\rho E} \end{pmatrix}, \quad \overline{F}_{\mathrm{e}}(h) = \begin{pmatrix} \overline{\rho} \cdot \overline{v} \\ \overline{\rho} \cdot \overline{v}^2 + \overline{p} \\ \overline{v}\left(\overline{\rho E} + \overline{p}\right) \end{pmatrix} \quad \text{with} \quad \overline{\rho E} = \frac{\overline{p}}{\gamma - 1} + \frac{1}{2}\overline{\rho} \cdot \overline{v}^2 . \tag{9}$$

The crucial aspect of this approach is that with the neural network output (8) the state vector and flux of the full Euler equations can be evaluated faithfully (no divisions by zero) even if the neural network approximation of the density is zero or negative.

For Problem 2, as before, we cannot rule out a division by zero to compute v when using the state vector $(\rho, \rho v)^\intercal$ as neural network output. Additionally, the equation of state requires a positive density. Therefore, an analogous procedure as for Problem 1 is not possible. Here, we ensure a positive approximation of the density by considering the following neural network output

$$\begin{pmatrix} \ln(\overline{\rho}) \\ \overline{\rho v} \end{pmatrix} := h(x, t; \theta) \tag{10}$$

and obtain a positive approximation of the density by $\overline{\rho} := \exp(\ln(\overline{\rho}))$. Now we can derive the approximation of the state vector $\overline{u}_{\mathrm{ise}}$ and the altered flux function as follows

$$\overline{u}_{\mathrm{ise}}(h) = \begin{pmatrix} \overline{\rho} \\ \overline{\rho v} \end{pmatrix}, \quad \overline{F}_{\mathrm{ise}}\big(h\big) = \begin{pmatrix} \overline{\rho v} \\ \overline{\rho} \cdot \overline{v}^2 + \overline{p} \end{pmatrix} \quad \text{with} \quad \overline{p} = \kappa \overline{\rho}^{\gamma}, \ \overline{v} = \frac{\overline{\rho v}}{\overline{\rho}} . \tag{11}$$

Eventually, the parameters θ are determined by a nonlinear optimization procedure such that $\overline{u}_{\mathrm{e}}$ and $\overline{u}_{\mathrm{ise}}$ approximate the gas flow as close as possible. That is, we consider $\min_\theta L(\theta)$, where the loss function $L(\theta)$ encodes the conformance to the conservation law (1) as well as the initial and boundary conditions. Crucially, the nonlinear optimization methods require a starting point. We use the Glorot initialization [2] which is based on randomness to compute the starting point for θ.

In the next subsections, we will introduce two different realizations for the loss function L. The derivation is the same for both problems. Therefore, we will drop the subscripts 'e', 'ise' and proceed with $\overline{F}$, u, and $\overline{u}$, which covers both cases.

3.2. Physics informed neural networks

As indicated before, we formulate the search of the neural network that approximates the solution of the differential equation as an optimization problem. Here, we explain the physics informed approach that was introduced in [13]. It is based on the differential form of the conservation law (1). The key building blocks are the squared residuals $\ell_{eq}(x,t)$, $\ell_i(x)$, $\ell_{lb}(t)$ and $\ell_{rb}(t)$, which enforce the differential equation, the initial condition, the left boundary condition and the right boundary condition at certain points (x,t), respectively.

The residual of the conservation law (1) at (x,t) with respect to the network parameters θ is defined by

$$\ell_{eq}(x,t;\theta) = \left\| \overline{u}_t\big(h(x,t;\theta)\big) + \overline{F}_x\big(h(x,t;\theta)\big) \right\|_2^2 . \tag{12}$$

Note that the derivatives are computed by automatic differentiation. The remaining residuals are defined in a in similar way and depend on the considered problem. For example, the residual for the initial condition of Problem 2 is

$$\ell_i(x;\theta) = \Big(\overline{\rho}(x,0;\theta) - 2\Big)^2 + \Big(\overline{\rho v}(x,0;\theta) - 0\Big)^2 . \tag{13}$$

The residuals are used as an indicator for the distance between the approximation and the exact solution. Thus, all should be minimized with respect to their domains. Therefore, we consider integrated residuals (squared L_2-norms)

$$L_{eq} = \frac{1}{\mathrm{vol}(D_{eq})} \int_{D_{eq}} \ell_{eq} \,\mathrm{d}x\,\mathrm{d}t , \qquad L_i = \frac{1}{\mathrm{vol}(D_i)} \int_{D_i} \ell_i \,\mathrm{d}x , \tag{14}$$

$$L_{lb} = \frac{1}{\mathrm{vol}(D_{lb})} \int_{D_{lb}} \ell_{lb} \,\mathrm{d}t , \qquad L_{rb} = \frac{1}{\mathrm{vol}(D_{rb})} \int_{D_{rb}} \ell_{rb} \,\mathrm{d}t , \tag{15}$$

with $D_{eq} = [0, L] \times [0, T]$, $D_i = [0, L]$ and $D_{lb} = D_{rb} = [0, T]$. The integrals are normalized by the volume of the respective domain.

Our solution should minimize all integrals. This yields the multi-objective optimization problem

$$\min_\theta \Big(L_{eq}(\theta), L_i(\theta), L_{lb}(\theta), L_{rb}(\theta)\Big) . \tag{16}$$

This problem is not yet numerically solvable. Before, we need to address two things. First, we need to convert the multi-objective problem into a single-objective problem. For this, we will build the sum of the objectives. Second, we need to discretize the integrals. Here, we use the Monte Carlo integration method and obtain in the case of L_{eq} the approximation

$$\bar{L}_{eq}(\theta) := \frac{1}{|\mathcal{D}_{eq}|} \sum_{(x,t)\in\mathcal{D}_{eq}} \ell_{eq}(x,t;\theta) \approx L_{eq}(\theta) \tag{17}$$

for a finite randomly selected set $\mathcal{D}_{eq} \subset D_{eq}$. From another viewpoint, $\bar{L}_{eq}$ computes the mean of the squared residuals. In addition, we define the approximations $\bar{L}_i(\theta)$, $\bar{L}_{lb}(\theta)$ and $\bar{L}_{rb}(\theta)$ in the same way for randomly selected finite sets $\mathcal{D}_i \subset D_i$, $\mathcal{D}_{lb} \subset D_{lb}$ and $\mathcal{D}_{rb} \subset D_{rb}$. In our implementation, the randomly selected sets are determined by the Latin Hypercube sampling scheme.

The authors in [13] state two advantages by using the Monte Carlo method. First, the computational complexity does not scale with the dimension of the domain. However, this advantage comes at a very high price in terms of a very slow rate of convergence. Second, the choice of random points should prevent that the optimizer learns the solution on a regular grid and thus prevents generalization beyond the grid. This line of thought is carried over from observations that have been made for other machine learning tasks.

Replacing the integrals with the approximations results in the optimization problem

$$\min_{\theta} L_{\text{dif}} = \bar{L}_{eq}(\theta) + \bar{L}_i(\theta) + \bar{L}_{lb}(\theta) + \bar{L}_{rb}(\theta)\,. \tag{18}$$

We use the subscript 'dif' to denote the loss function with respect to the differential form of the conservation law. This optimization problem has been originally proposed in [13]. Since then, numerous extensions have been proposed in the literature. Most of them are concerned with the conversion from the multi-objective optimization problem into the single-objective optimization problem. By building the sum of the objectives $\bar{L}_{eq}$, $\bar{L}_i$, $\bar{L}_{lb}$, $\bar{L}_{rb}$ we use constant weights which are determined in the derivation such that the respective volumes of the objectives are neglected. In a more general setting, one does consider the weighted sum of the objectives. Then, the weights are additional degrees of freedom and affect the optimization process. We will discuss this in more detail in Section 4. However, before that we will derive an alternative formulation that circumvents this issue.

3.3. Control-volume physics informed neural networks

While the derivation in the previous subsection is based on the differential form of the conservation law (1), we can derive another objective function based on the integral form of the conservation law. This method was first proposed in [12] and has the advantage that the initial and boundary conditions are naturally included. Consequently, the problem of a multi-objective optimization problem is alleviated.

The starting point is the space-time flux function

$$G(u) = \begin{pmatrix} F(u) & u \end{pmatrix}. \tag{19}$$

Then, by the divergence theorem and for sufficiently smooth u, u fulfills the differential form of the conservation law (1) if and only if u fulfills the integral form of the conservation law

$$0 = \int_{\omega} \operatorname{div}_{(x,t)} G\big(u(x,t)\big)\,\mathrm{d}x\,\mathrm{d}t = \int_{\partial\omega} G\big(u(x,t)\big)\cdot\vec{n}\,\mathrm{d}S \tag{20}$$

for all $\omega \subset [0,L]\times[0,T]$ with piecewise smooth boundary $\partial\omega$, where $\vec{n}$ denotes the outward-facing normal vector on $\partial\omega$.

The idea is to verify that the integral in (20) vanishes on finitely many control volumes ω. To this end, we choose a finite partition $\omega_1,\dots,\omega_{n_\omega}$ of $[0,L]\times[0,T]$. Again, to avoid undefined computations, we need to define a space-time flux which operates directly on the output of the neural network h

$$\overline{G}(h) = \begin{pmatrix} \overline{F}(h) & \overline{u}(h) \end{pmatrix}. \tag{21}$$

In contrast to the original PINN approach, initial and boundary values enter in a natural way. We augment the neural network output by these values and denote the resulting function by $\tilde{h}(x,t;\theta)$. For example, for Problem 1 we define

$$\begin{pmatrix} \tilde{\rho} \\ \tilde{v} \\ \tilde{p} \end{pmatrix} =: \tilde{h}(x,t;\theta) \quad \text{with} \quad \tilde{v}(x,t;\theta) = \begin{cases} 0.1 & \text{if } t = 0\,, \\ 0.1 & \text{if } x = 0 \text{ or } x = 1\,, \\ \overline{v}(x,t;\theta) & \text{otherwise}\,. \end{cases} \tag{22}$$

For the remaining quantities $\tilde{\rho}$, $\tilde{p}$ and for Problem 2, we can proceed in the same way. Now we can formulate the optimization problem

$$\min_{\theta} L_{\text{int}}(\theta) = \sum_{i=1}^{n_\omega} \left(\int_{\partial\omega_i} \overline{G}\left(\tilde{h}(x,t;\theta)\right) \cdot \vec{n}\, \mathrm{d}S \right)^2 . \tag{23}$$

Here, the subscript 'int' denotes the loss function with respect to the integral form of the conservation law. As before, the integrals in the loss function need to be discretized. In contrast to the physics informed approach, the authors in [12] suggest a deterministic integration scheme. This is beneficial since we have lowered the integration domain by one dimension and thus reduced the complexity. In addition, this approach has no derivatives in the loss function. Therefore, the evaluation of L_{int} does not require automatic differentiation and is therefore computationally more efficient than the evaluation of L_{dif} for a similar amount of control points. Further, the mesh $\omega_1, \dots, \omega_{n_\omega}$ is one additional degree of freedom of this approach.

In our numerical tests in Section 5, we will compare two different quadrature strategies: a deterministic Gaussian-Legendre quadrature with three points and a stochastic Monte Carlo quadrature with five points. We will also consider rectangular and triangular meshes.

In summary, the physics informed approach and the control volume physics informed approach are two different approaches for the same problem. In Section 5, we will compare both approaches. We will also consider different optimization schemes – a topic that we have still not discussed in this section. Beforehand, in Section 4, we will take a closer look at the issue of different weights in the physics informed approach.

4. Algorithmic balancing of loss weights

From an optimization standpoint, the physics informed optimization problem (16) is a multi-objective optimization problem. That is, we try to minimize multiple competing objectives $\overline{L}_{eq}$, $\overline{L}_i$, $\overline{L}_{lb}$, $\overline{L}_{rb}$ simultaneously. Building the sum of the competing objectives is just one way to get a single-objective optimization problem. In a more general way, one builds a weighted sum of the competing objectives. That is, we introduce weights $\lambda = (\lambda_{eq}, \lambda_i, \lambda_{lb}, \lambda_{rb}) \geq 0$ for every competing objective and consider the optimization problem

$$\min_{\theta} L_{\text{w-dif}}(\theta) = \lambda_{eq}\overline{L}_{eq}(\theta) + \lambda_i\overline{L}_i(\theta) + \lambda_{lb}\overline{L}_{lb}(\theta) + \lambda_{rb}\overline{L}_{rb}(\theta)\,. \tag{24}$$

The subscript 'w-dif' denotes the loss function with respect to the differential form of the conservation law with additional weights. The weights determine the importance of the individual terms. One can trade off one objective for another objective. The different optimal loss values for every tuple of weights $(\lambda_{eq}, \lambda_i, \lambda_{lb}, \lambda_{rb})$ define a manifold $\mathbb{R}^4 \to \mathbb{R}$ – the so called Pareto front. That is, one does not get one optimal solution, but a family of solutions which are optimal with respect to different tradeoffs. The Pareto front has been analyzed in the context of physics informed neural networks for concrete examples in [14].

Choosing the solution with the lowest error from the Pareto front is challenging. This can be attributed to the fact that the weights affect the minimization of the residuals, but eventually we want to achieve the lowest error. The effect of the weights on the errors depends on the relationship between residuals and errors. It is only poorly understood.

Aside the theoretical challenges, we can observe practical advantages by choosing different weights. For our problems, we can try to make an educated guess to determine which objectives are more important than others. For example, for Problem 1, it is very important to approximate the discontinuous initial data, since the underlying hyperbolic equation transports the information along the characteristic curves and the directions of the curves depend on the initial data. Therefore, one needs to make sure that the source of the characteristics (and thus information) is approximated well. This effect is illustrated in Fig. 1. Here, we train the same initial model using the same sample points with the L-BFGS method but with different weights λ. We observe that increasing the weight of $\overline{L}_i$ decreases the error of v and p.

This small example shows that choosing weights matters for our test problems. But guessing the weights is not generally applicable. Changing the initial or boundary conditions can alter the direction of the characteristics.

The positive effect of the weights has been also described in the literature and multiple procedures have been proposed to algorithmically determine the weights in a beneficial way. We will review them in the following subsections. Further, we will perform numerical tests with these methods in Section 5 to find the most reliable way to approximate our two test problems with a neural network.

4.1. Gradient-based methods

Gradient-based methods for the determination of the weights λ were proposed in [16] and a modification was studied in [5]. While in [5] the authors focused on the incompressible Navier–Stokes equations, in [16] the authors studied a wider range of equations including the Helmholtz equation, the Klein-Gordon equation and also the incompressible Navier-Stokes equations. The authors in [16] take for each objective $\overline{L}_{eq}$, $\overline{L}_i$, $\overline{L}_{lb}$, $\overline{L}_{rb}$ the gradient with respect to the parameters θ into account and consider the distribution of the gradients entries.

The Glorot initialization scheme [2] ensures that the initial mean of these distributions is close to zero and it can be observed to stay close to zero during the optimization procedure [2,16]. On the other hand, the variance of these distribu-

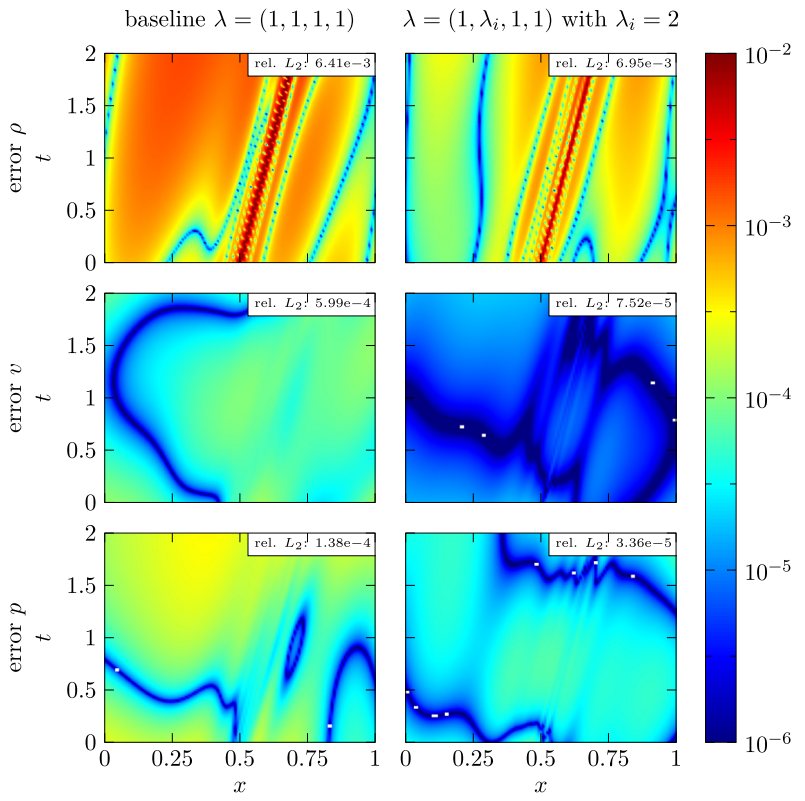


Fig. 1. A comparison between the errors of two models trained to solve Problem 1 with different weights λ. On the left-hand side the standard weights are used and on the right-hand side the weight λ_i is increased. The models are trained with the L-BFGS method, the same initial model and the same sampling points $\mathcal{D}_{eq}, \mathcal{D}_i, \mathcal{D}_{lb}, \mathcal{D}_{rb}$. (For interpretation of the colors in the figure, the reader is referred to the web version of this article.)

tions depends on the concrete objective can vary greatly during the optimization process. This gives rise to the following considerations.

If the variance of a distribution is closer to zero, the elements of the gradient are more insensitive with respect to the parameters θ and the objective has less influence in the optimization step. On the other hand, if the variance is larger, there are more derivatives that are very sensitive to the parameters θ and the objective has a strong influence in the optimization step. Therefore, the idea is to use the weights λ to scale the distributions such that every objective is equally treated by the optimization procedure. This prevents that one of the objectives is prioritized during the optimization process. Such uneven distributions are seen in [16] as an indication of an imbalanced loss function. This interpretation is closely related to how initialization schemes for neural networks work.

In the resulting methods, the weights are updated throughout the optimization process to adapt to the changing distributions. The distribution of $\nabla_\theta \bar{L}_{eq}(\theta)$ is the reference distribution and the other distributions are scaled to be similar. Thus $\lambda_{eq} = 1$ remains constant. For the remaining weights, the authors in [5] consider $\mu(|\nabla_\theta \bar{L}(\theta)|)$, where $\mu(v)$ denotes the mean of a vector v, for every objective $\bar{L} = \bar{L}_{eq}, \bar{L}_i, \bar{L}_{lb}, \bar{L}_{rb}$ as the defining attribute for the respective distribution and define the intermediate weights

$$\hat{\lambda}_i = \frac{\mu(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_i(\theta)|)}, \quad \hat{\lambda}_{lb} = \frac{\mu(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_{lb}(\theta)|)}, \quad \hat{\lambda}_{rb} = \frac{\mu(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_{rb}(\theta)|)}. \tag{25}$$

These values change rapidly during the optimization process and are thus smoothed out. This is done through a convex combination of the old and intermediate weights $\hat{\lambda}$ with respect to an $\alpha \in (0, 1)$. Thus, the weights are updated according to

$$\lambda_i \leftarrow (1-\alpha)\lambda_i + \alpha\hat{\lambda}_i, \quad \lambda_{lb} \leftarrow (1-\alpha)\lambda_{lb} + \alpha\hat{\lambda}_{lb}, \quad \lambda_{rb} \leftarrow (1-\alpha)\lambda_{rb} + \alpha\hat{\lambda}_{rb}. \tag{26}$$

This update can take place in every iteration of the optimization process. However, it is also possible to do it at a lower frequency.

The mean of the absolute value $\mu(|\cdot|)$ is the mean absolute deviation of a distribution with zero mean, which is closely related to the variance of the distribution. In [16] the authors move away from this viewpoint and propose a different strategy to define the intermediate weights. Namely, they consider the maximum absolute value of $\nabla_\theta \bar{L}_{eq}(\theta)$ and define

$$\hat{\lambda}_i = \frac{\max(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_i(\theta)|)}, \quad \hat{\lambda}_{lb} = \frac{\max(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_{lb}(\theta)|)}, \quad \hat{\lambda}_{rb} = \frac{\max(|\nabla_\theta \bar{L}_{eq}(\theta)|)}{\mu(|\nabla_\theta \bar{L}_{rb}(\theta)|)}. \tag{27}$$

For further reference, we call the definitions in (25) the avg-avg weights and the definitions in (27) the max-avg weights. Usually, the weights $\lambda_{eq}, \lambda_i, \lambda_{lb}, \lambda_{lb}$ are initialized with 1. In [16] $\alpha = 0.9$ and in [5] $\alpha = 0.1$ is set. We will test both values, which results in four strategies in total, in our numerical examples.

4.2. Magnitude-based method

A magnitude-based method is introduced in [15] and has been developed for linear elliptic PDEs. The authors are concerned with inherent scaling issues of the underlying differential equation and propose a method called magnitude normalization. The strategy is summarized by the authors as follows: "Each loss functional is normalized by the magnitude of the terms that comprise it". It was developed for linear boundary values problems. However, we seek to adapt the strategy for our nonlinear initial boundary value problem. The main components are the magnitudes $m_{eq}(x,t)$, $m_i(x)$, $m_{lb}(t)$ and $m_{rb}(t)$ that are analogously defined to $\ell_{eq}(x,t)$, $\ell_i(x)$, $\ell_{lb}(t)$ and $\ell_{rb}(t)$. The goal is that the fraction of objective and magnitude, e.g., ℓ_{eq}/m_{eq}, stays roughly the same. For example, we define for the isentropic Euler equations

$$m_{eq}(x,t;\theta) = \left(|\overline{\rho}_t(x,t)| + |(\overline{\rho v})_x(x,t)|\right)^2 + \left(|(\overline{\rho v})_t(x,t)| + |\overline{p}_x(x,t)| + |(\overline{\rho}\overline{v}^2)_x(x,t)|\right)^2 \tag{28}$$

and for the left boundary condition of Problem 1

$$m_{lb}(x;\theta) = \left(|1|\right)^2 + \left(|0.1|\right)^2 + \left(|1|\right)^2. \tag{29}$$

The other magnitudes can be derived in a similar way. Then we can integrate the magnitudes over their respective domains, resulting in $M_{eq}(\theta)$, $M_i(\theta)$, $M_{lb}(\theta)$ and $M_{rb}(\theta)$ analogously to (14). Next, we discretize these integrals with the Monte Carlo quadrature rule in the same way as in (17), and arrive at $\overline{M}_{eq}(\theta)$, $\overline{M}_i(\theta)$, $\overline{M}_{lb}(\theta)$ and $\overline{M}_{rb}(\theta)$. Here, we use the same sample points as before.

Eventually, we define the weights as the inverse of the magnitudes,

$$\lambda_{eq} = \frac{1}{\overline{M}_{eq}(\theta)}, \quad \lambda_i = \frac{1}{\overline{M}_i(\theta)}, \quad \lambda_{lb} = \frac{1}{\overline{M}_{lb}(\theta)}, \quad \lambda_{rb} = \frac{1}{\overline{M}_{rb}(\theta)}. \tag{30}$$

The magnitudes depend on the neural network parameters θ and therefore the weights λ change throughout the optimization process. In contrast to the gradient-based methods, the authors do not recommend smoothing of the weights. In our implementation, we will update the weights at a specific frequency. This is not entirely consistent with the proposed approach of [15]. Here, the weights are updated only at certain events.

4.3. Attention-based method

The next method we consider is the attention-based mechanism that was introduced in [10] and tested there with the Allen-Cahn equation. In contrast to the previous methods, this method integrates the weights into the optimization process and does not determine the weights based on the solution. For the original method, the authors propose to weight every sampling point in $\mathcal{D}_{eq}$, $\mathcal{D}_i$, $\mathcal{D}_{lb}$, $\mathcal{D}_{rb}$ and thus changing their importance. However, this significantly increases the computational complexity and, for the sake of comparison, we adapt this method for problem (24).

We consider $L_{\text{w-dif}}(\theta,\lambda)$, the objective function of the problem (24), with $\lambda = (\lambda_{eq}, \lambda_i, \lambda_{lb}, \lambda_{rb})$. Now the weights λ should penalize the objectives with the highest values. That is achieved by maximizing the loss function $L_{\text{w-dif}}$ with respect to λ while minimizing the loss function with respect to θ. Consequently, we consider the saddle point problem

$$\min_\theta \max_\lambda \; L_{\text{w-dif}}(\theta,\lambda). \tag{31}$$

The solution of this problem can be approximated by a gradient descent method with the update

$$\theta \leftarrow \theta - \nabla_\theta L_{\text{w-dif}}(\theta,\lambda) \qquad \text{and} \qquad \lambda \leftarrow \lambda + \nabla_\lambda L_{\text{w-dif}}(\theta,\lambda). \tag{32}$$

At the start, we initialize the weights with $\lambda = (1,1,1,1)$.

4.4. Random-search

In order to put the previous methods into perspective, we also perform an additional random-search. That is, we draw a fixed number of random tuples $(\lambda_{eq}, \lambda_i, \lambda_{lb}, \lambda_{rb})$ and run the training process for each weight tuple. For both test problems we draw 20 random tuples from the cube $[1,5]\times[1,5]\times[1,5]\times[1,5]$ using the Latin-Hypercube sampling strategy.

After the training process, we choose the weights and parameters with the minimal loss value L_{dif} and refer to this result as rnd-search-min. For comparison, we refer to the result with the highest loss value L_{dif} as rnd-search-max.

5. Numerical tests

In this section, we want to compare the different methods. In particular, we consider the physics informed approach and the different weighting methods in Subsection 5.1. Further, we investigate the control volume physics informed approach in Subsection 5.2. In the last subsection, we report on results for different sizes of neural networks (measured in the number of parameters) and different numbers of sampling points.

Our main measure for the comparison will be the relative L_2-error of the different methods. We measure the error with respect to the output of the neural network and the reference solution. In case of Problem 1 we consider the error of ρ, v, p and in case of Problem 2 the error of ρ, ρv. For example, the error of ρ is defined by

$$\text{error } \rho = \frac{\sqrt{\sum_{(x,t)\in\mathcal{D}_{err}} \left(\rho(x,t) - \overline{\rho}(x,t;\theta)\right)^2}}{\sqrt{\sum_{(x,t)\in\mathcal{D}_{err}} \left(\rho(x,t)\right)^2}} \tag{33}$$

for a finite set $\mathcal{D}_{err} \subset [0, L] \times [0, T]$. The remaining errors are defined in the same way. For Problem 1, the set $\mathcal{D}_{err}$ contains 50 000 random points, and for Problem 2, $\mathcal{D}_{err}$ contains the grid points of the numerically computed reference solution.

For the first two subsections, we will use a neural network with $n = 4$ layers and $k = 27$ neurons. Further, we use fixed amounts of sampling points, namely for

$$\begin{aligned} &\text{Problem 1}, \quad |\mathcal{D}_{eq}| = 6\,400, \quad |\mathcal{D}_i| = 320, \quad |\mathcal{D}_{lb}| = 160, \quad |\mathcal{D}_{rb}| = 160 \quad \text{and} \\ &\text{Problem 2}, \quad |\mathcal{D}_{eq}| = 6\,400, \quad |\mathcal{D}_i| = 160, \quad |\mathcal{D}_{lb}| = 160, \quad |\mathcal{D}_{rb}| = 320. \end{aligned} \tag{34}$$

The choices are justified by the numerical results in Subsection 5.3 and will be further investigated there. Also note the emphasis on the challenging initial data for Problem 1 and the right boundary data for Problem 2.

As mentioned above, the initial parameters of the neural network as well as the sampling points are randomly chosen. Therefore, different runs of the same test will produce different results. We take this into account in our experiments and run every test five times. Then we report the mean and the standard deviation of the relative L_2 errors. Crucially, for the same run, we use the same (randomly chosen) initial neural network and the same (randomly chosen) sampling points across all tests. This eliminates the influence of the randomness between the different tests and ensures a fair comparison.

5.1. Comparison of different weighting methods

In this subsection, we want to compare the different weighting methods. That is, the methods reviewed in Section 4 to algorithmically determine the weights λ_{eq}, λ_i, λ_{lb} and λ_{rb} for problem (24). For the gradient- and magnitude-based methods, the weights will be updated every 10th iteration. We will optimize the parameter of the neural network (or in the case of the attention method, the parameter and the weights λ) with the standard Adam method. Further, we use an initial learning rate (step size) of 0.01 and, to ensure convergence, we use an exponentially decaying learning rate that decays the learning rate by a factor of 0.9 every 1000 steps. We run the method for 30 000 steps.

Table 1 shows the results for Problem 1 and Table 2 for Problem 2.

The results of the rnd-search-min method show that there is a clear advantage in using specific weights in the loss function. This is strictly the case for Problem 2. For Problem 1, we see a notable decrease in the errors of ρ and v by using the rnd-search-min weights instead of the baseline weights. However, for the error of p this is not the case.

Among the dynamic weighting methods, which are the gradient-, magnitude-, and attention-based methods, the attention-based approach has the best results for both problems. But, there is a noticeable gap between these results and the rnd-search-min results. In the case of Problem 2, the gradient- and magnitude-based methods produce worse results. In the case of Problem 1, these methods perform better, but not as good as the baseline weights. Here, the good results of the avg-avg method with $\alpha = 0.1$ are noteworthy. In summary, the dynamic weighting methods are not able to achieve results as good as the rnd-search-min method.

Fig. 2 shows the selected weights of the different methods of the first run. Since the loss function can be scaled arbitrarily, the weights of the different methods cannot be compared directly. However, the weights determined by the gradient-based methods should be emphasized since they are orders of magnitudes larger than the weights determined by the other methods. Here, only the avg-avg weights for Problem 1 have a reasonable range and thus may explain the favorable results. Also, for the attention-based approach for Problem 1 the weights of the initial data and for Problem 2 the weights of the right boundary data are the largest. Thus, the approach realizes the aforementioned importance of the initial data for Problem 1 and the right boundary data for Problem 2. However, this does not lead to better results than the usage of the baseline weights. Lastly, the weights of the magnitude-based method are almost constant throughout the optimization procedure. However, this fixed choice is not beneficial.

5.1.1. L-BFGS optimization

In addition to the Adam method, the L-BFGS method is very popular to optimize physics informed neural networks [13,9,15,10]. In this subsection, we want to take a closer look at the benefit of the L-BFGS method.

Table 1
Results of the different weighting methods for Problem 1. The baseline test uses $\lambda = (1, 1, 1, 1)$. Bold numbers refer to the two smallest values in each column.

	loss L_{dif}	error ρ	error v	error p
baseline	**1.6e−5 ±1.4e−5**	**6.6e−3 ±7.2e−4**	**1.2e−3 ±3.2e−4**	**5.4e−5 ±1.6e−5**
rnd-search-min	**1.6e−6 ±8.7e−7**	**5.7e−3 ±7.2e−4**	**6.0e−4** ±5.6e−4	**8.1e−5 ±2.3e−5**
rnd-search-max	1.1e−4 ±4.4e−5	8.1e−3 ±1.0e−3	3.0e−3 ±2.8e−3	1.5e−4 ±8.9e−5
max-avg $\alpha = 0.1$	4.0e−1 ±1.9e−1	8.3e−2 ±4.0e−3	8.0e−2 ±2.5e−2	3.6e−2 ±9.2e−3
max-avg $\alpha = 0.9$	3.7e−2 ±1.1e−2	7.3e−2 ±7.6e−3	3.1e−2 ±1.0e−2	3.7e−2 ±6.2e−3
avg-avg $\alpha = 0.1$	5.5e−4 ±1.7e−4	1.8e−2 ±2.5e−3	1.8e−3 ±4.5e−4	2.9e−4 ±1.2e−4
avg-avg $\alpha = 0.9$	4.4e−3 ±1.9e−3	5.3e−2 ±1.4e−2	8.8e−3 ±6.4e−3	1.2e−3 ±8.2e−4
magnitude	7.6e−3 ±8.3e−4	6.7e−2 ±3.8e−3	3.2e−2 ±7.8e−3	4.2e−3 ±1.2e−3
attention	2.7e−4 ±1.2e−4	1.3e−2 ±2.3e−3	2.4e−3 ±**4.3e−4**	2.6e−4 ±9.4e−5

Table 2
Results of the different weighting methods for Problem 2. The baseline test uses $\lambda = (1, 1, 1, 1)$. Bold numbers refer to the two smallest values in each column.

	loss L_{dif}	error ρ	error ρv
baseline	**2.5e−6** ±1.4e−6	**4.3e−4** ±2.1e−4	**2.4e−2** ±1.2e−2
rnd-search-min	**1.0e−6 ±3.3e−7**	**2.2e−4 ±4.6e−5**	**1.2e−2 ±2.1e−3**
rnd-search-max	5.2e−6 ±7.6e−7	7.5e−4 ±1.2e−4	3.9e−2 ±1.5e−2
max-avg $\alpha = 0.1$	1.8e−2 ±2.7e−3	3.0e−2 ±5.1e−3	2.0e0 ±1.8e−1
max-avg $\alpha = 0.9$	2.3e−2 ±4.7e−3	3.1e−2 ±6.3e−3	2.0e0 ±2.0e−1
avg-avg $\alpha = 0.1$	2.5e−3 ±2.0e−3	5.2e−3 ±1.6e−3	9.1e−1 ±5.9e−1
avg-avg $\alpha = 0.9$	7.0e−3 ±1.3e−3	1.6e−2 ±6.6e−3	1.7e0 ±1.2e−1
magnitude	1.2e−4 ±1.7e−5	3.9e−3 ±3.3e−4	2.1e−1 ±1.9e−2
attention	1.0e−5 ±**3.2e−7**	9.5e−4 ±**8.1e−5**	5.4e−2 ±**5.3e−3**

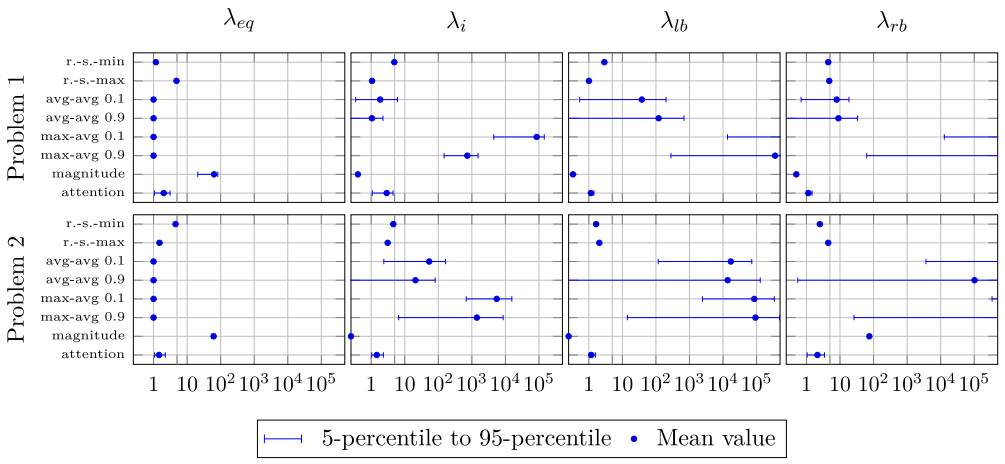


Fig. 2. Visualization of the weights of all methods for both problems from the first (of five) run as used during the Adam optimization algorithm (with learning rate decay). For the dynamic weight balancing methods the weights change during the course of the optimization. The figure shows the range in which 90 percent of the weights are located. It further shows the mean value of the weights. For the other methods the mean value is the only value used during the optimization procedure. Note that due to the different magnitudes of the weights, some data are outside of the axis.

First, we optimize the neural network only by the L-BFGS method. Second, we optimize the neural network by a hybrid optimization scheme of the Adam and the L-BFGS method. This combination is often proposed in the literature and we test it as well. Here, we proceed as follows. We start by optimizing the neural network with the Adam method for (again) 30 000 steps and change the weights λ according to the used weighting method. Next, we only optimize the parameters θ by the L-BFGS method. The weights λ are constant throughout the L-BFGS method and keep their last assigned value. Importantly, we use no learning rate decay for the Adam method, since then the subsequent L-BFGS optimization is inhibited and barely improves the results. The L-BFGS method has a no fixed step size and stops if the method has converged. For the stopping criterion, we require a gradient tolerance of 10^{-7}.

Table 3
Results of the different weighting methods for Problem 1 optimized first by the Adam method and then by the L-BFGS method. The results with the suffix 'only lbfgs' are only optimized by the L-BFGS method. The last column contains the average number of iterations the L-BFGS method needed to converge. Bold numbers refer to the two smallest values in each column.

	loss L_{dif}	error ρ	error v	error p	Iterations
baseline only lbfgs	1.0e−5 ±1.0e−5	6.2e−3 ±5.9e−4	6.7e−4 ±3.7e−4	1.1e−4 ±6.2e−5	8095.6 ±2018.6
baseline	9.6e−5 ±1.1e−4	1.0e−2 ±4.7e−3	1.1e−3 ±7.4e−4	5.0e−5 ±3.2e−5	378.8 ±**415.5**
rnd-search-min only lbfgs	**1.4e−6** ±9.0e−7	**5.8e−3** ±**3.0e−4**	**3.1e−4** ±**1.2e−4**	5.3e−5 ±2.6e−5	8515.6 ±1017.7
rnd-search-min	**1.1e−6** ±**4.0e−7**	6.9e−3 ±3.6e−3	9.4e−4 ±6.9e−4	**4.0e−5** ±3.4e−5	1120.6 ±1188.7
rnd-search-max only lbfgs	3.8e−5 ±1.8e−5	7.6e−3 ±9.4e−4	1.3e−3 ±6.0e−4	1.3e−4 ±3.8e−5	7078.2 ±1575.3
rnd-search-max	1.1e−4 ±6.8e−5	1.1e−2 ±3.5e−3	1.5e−3 ±9.9e−4	4.8e−5 ±**1.8e−5**	**276.4** ±**220.3**
max-avg $\alpha = 0.1$	3.5e−1 ±1.9e−1	8.7e−2 ±5.2e−3	6.8e−2 ±3.3e−2	1.5e−2 ±3.5e−3	**277.2** ±478.1
max-avg $\alpha = 0.9$	8.4e−3 ±1.3e−2	3.3e−2 ±1.2e−2	1.9e−2 ±2.5e−2	2.8e−3 ±2.3e−3	11411.4 ±2935.1
avg-avg $\alpha = 0.1$	4.5e−5 ±8.6e−5	6.2e−3 ±2.0e−3	4.8e−4 ±4.3e−4	1.1e−4 ±1.4e−4	3201.2 ±1658.2
avg-avg $\alpha = 0.9$	3.0e−6 ±**3.5e−7**	**5.6e−3** ±**2.9e−4**	**2.2e−4** ±**6.0e−5**	**4.2e−5** ±**7.8e−6**	8083.2 ±1179.1
magnitude	4.0e−4 ±1.4e−4	1.6e−2 ±2.5e−3	3.2e−3 ±1.5e−3	1.5e−4 ±7.2e−5	9920.4 ±1213.7
attention	5.9e−6 ±4.4e−6	6.0e−3 ±3.7e−4	1.0e−3 ±4.3e−4	7.8e−5 ±2.9e−5	5734.2 ±2257.8

Table 4
Results of the different weighting methods for Problem 2 optimized first by the Adam method and then by the L-BFGS method. The results with the suffix 'only lbfgs' are only optimized by the L-BFGS method. The last column contains the average number of iterations the L-BFGS method needed to converge. Bold numbers refer to the two smallest values in each column.

	loss L_{dif}	error ρ	error ρv	Iterations
baseline only lbfgs	3.9e−7 ±**8.1e−8**	1.5e−4 ±**7.3e−6**	8.5e−3 ±**5.5e−4**	4734.8 ±1448.0
baseline	7.4e−7 ±5.9e−7	1.7e−4 ±6.1e−5	9.5e−3 ±3.5e−3	1162.2 ±1188.3
rnd-search-min only lbfgs	**2.4e−7** ±8.5e−8	**1.3e−4** ±1.7e−5	**7.4e−3** ±9.6e−4	5560.6 ±2194.0
rnd-search-min	**1.4e−7** ±**3.8e−8**	**1.3e−4** ±**1.0e−5**	**7.4e−3** ±**5.8e−4**	2524.6 ±972.0
rnd-search-max only lbfgs	1.2e−6 ±1.4e−7	2.7e−4 ±2.8e−5	1.5e−2 ±1.6e−3	2772.8 ±452.5
rnd-search-max	1.3e−5 ±1.5e−5	9.7e−4 ±8.4e−4	5.6e−2 ±5.0e−2	**130.0** ±**127.6**
max-avg $\alpha = 0.1$	8.8e−3 ±2.0e−3	2.1e−2 ±9.6e−3	1.9e0 ±1.2e−1	**30.6** ±**38.3**
max-avg $\alpha = 0.9$	3.9e−3 ±4.5e−3	7.5e−3 ±5.7e−3	1.0e0 ±8.3e−1	916.2 ±1136.5
avg-avg $\alpha = 0.1$	4.8e−3 ±6.2e−3	1.7e−2 ±1.9e−2	8.2e−1 ±6.1e−1	346.0 ±528.4
avg-avg $\alpha = 0.9$	1.5e−4 ±9.6e−5	3.7e−3 ±1.2e−3	2.1e−1 ±6.6e−2	1396.8 ±730.6
magnitude	2.6e−5 ±1.3e−5	1.7e−3 ±3.7e−4	9.6e−2 ±2.0e−2	3808.8 ±1959.0
attention	8.5e−7 ±1.2e−6	2.2e−4 ±1.5e−4	1.2e−2 ±8.5e−3	3429.2 ±1650.5

The results for Problem 1 and 2 are given in Table 3 and 4, respectively.

For the second test problem, the best results are obtained by the rnd-search-min method. Again, we can conclude that different weights can decrease the overall errors. For this problem we cannot observe an advantage in first using the Adam procedure. In our test setup, the L-BFGS method usually required less time than the hybrid approach despite taking more L-BFGS iterations to converge. The hybrid optimization scheme improves the errors of the dynamic weighting methods in comparison to only using the Adam routine. However, none of the methods show an advantage against the best weights determined by the rnd-search-min scheme.

These findings can be applied to the first test problem with some restrictions. Again, the lowest errors are achieved by the rnd-search-min method. However, the lowest approximation error of the pressure p is obtained by running the Adam method first. But then, the errors of the density ρ and the speed v are larger. There seems to be a trade-off between the different quantities and the gradient-based avg-avg method with $\alpha = 0.9$ handles this challenge very well.

In summary, for our test problems, it seems to be a reasonable choice to only use the L-BFGS method with some fixed weights. This method is easier to use and produces very good as well as reliable results very fast. Finding these fixed weights is expensive. The random-search has the greatest computational cost, but the effort can be justified if the best results are required and can be controlled by the number of samples. There might be an advantage in using the gradient-based avg-avg instead. But this comes with an increased computational cost of running the Adam method first and this has only worked for one of two test problems.

Also see Fig. 3 that shows the best and worst weights found by the random-search method. It illustrates the different choices made by the random procedure in every run, highlights the complexity of the underlying search problem and thus the challenges the algorithmic balancing methods have to overcome.

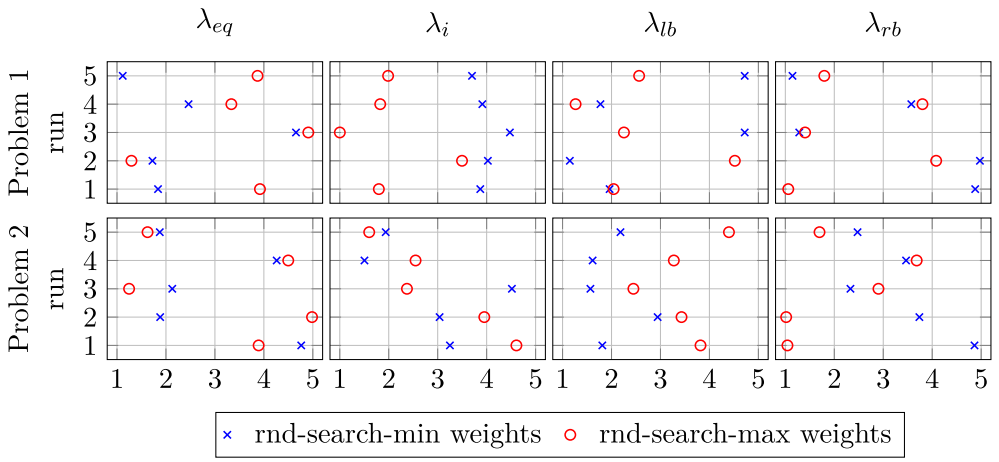


Fig. 3. The figure shows the best (rnd-search-min) and worst (rnd-search-max) weights found by the random-search method for every run optimized only by the L-BFGS method. Note that every run uses a different list of randomly selected weights. Therefore, a direct comparison between the different runs is impossible.

Table 5
The results of the control volume physics informed approach for Problem 1. The iterations column contains the average number of iterations the L-BFGS method needed to converge. Bold numbers refer to the two smallest values in each column.

mesh	quadrature	optimizer	loss L_{int}	error ρ	error v	error p	Iterations
rect	Gaussian	Adam	1.3e−6 ±1.8e−6	1.8e−2 ±4.0e−4	1.3e−3 ±8.4e−4	3.3e−4 ±1.5e−4	
		Hybrid	3.5e−7 ±2.4e−7	**1.7e−2** ±**2.2e−4**	**6.6e−4** ±3.0e−4	1.7e−4 ±8.2e−5	964.8 ±1566.7
		L-BFGS	**1.8e−7** ±**4.8e−8**	**1.8e−2** ±5.2e−4	**4.0e−4** ±**1.1e−4**	**8.3e−5** ±**1.8e−5**	2695.4 ±999.7
	random	Adam	1.9e−5 ±4.0e−6	3.6e−2 ±3.5e−3	3.4e−3 ±8.6e−4	6.2e−4 ±1.4e−4	
		Hybrid	1.4e−5 ±2.7e−6	3.5e−2 ±1.5e−3	2.2e−3 ±5.7e−4	3.8e−4 ±1.2e−4	**434.4** ±844.8
		L-BFGS	6.8e−6 ±1.5e−6	3.7e−2 ±1.3e−3	1.7e−3 ±3.1e−4	2.1e−4 ±5.2e−5	7446.0 ±1895.0
trig	Gaussian	Adam	1.5e−6 ±1.8e−6	2.0e−2 ±3.7e−4	1.3e−3 ±5.6e−4	3.5e−4 ±9.5e−5	
		Hybrid	1.4e−6 ±1.9e−6	2.0e−2 ±6.1e−4	1.0e−3 ±2.6e−4	2.5e−4 ±8.3e−5	**85.8** ±109.7
		L-BFGS	**2.8e−7** ±**4.3e−8**	2.0e−2 ±**2.4e−4**	7.0e−4 ±**1.5e−4**	**1.3e−4** ±**4.0e−5**	2330.0 ±879.0
	random	Adam	3.2e−5 ±8.1e−6	4.8e−2 ±2.8e−3	4.6e−3 ±5.9e−4	6.7e−4 ±1.0e−4	
		Hybrid	2.3e−5 ±5.7e−6	4.8e−2 ±2.5e−3	4.0e−3 ±7.1e−4	5.1e−4 ±1.2e−4	994.8 ±1804.5
		L-BFGS	1.7e−5 ±2.8e−6	4.8e−2 ±4.2e−3	3.4e−3 ±4.8e−4	3.6e−4 ±5.2e−5	5782.8 ±2028.2

5.2. Control volume approach

In this subsection, we look into the results of the control volume physics informed approach. To discretize the objective of the optimization problem (23), we use two integrations methods: the deterministic Gauss-Legendre rule with three points and a random Monte Carlo quadrature with five points. Further, we use a rectangular and a triangular mesh to partition the computational domains into control volumes w_i used in (23). To enable a comparison with the physics informed approach, we choose the sizes of the meshes such that the number of quadrature points approximately equals the number of sampling points.

Again, we optimize with the Adam method alone (with a learning rate decay), the hybrid approach (Adam without learning rate decay first and then L-BFGS), and just the L-BFGS method. The optimizers use the same configurations as before.

The results are shown in Table 5 for Problem 1 and in Table 6 for Problem 2. Interestingly, for both problems the deterministic quadrature rule produces the best results. This is in contrast to the general belief that random methods work better in the machine learning context. Unfortunately, the results of the control volume approach are worse than those of the standard physics informed approach. The method is only able to achieve competitive results for the first test problem and the variables v and p. In the remaining cases, the errors are significantly larger.

An explanation could be that the physics informed approach uses derivative information and the control volume approach does not. This makes the computational complexity of the control volume approach cheaper. However, the results suggest that this is accompanied by a loss of accuracy. Consequently, the restriction that we use roughly the same amount of quadrature points as the amount of sample points might favor the physics informed approach.

Table 6
The results of the control volume physics informed approach for Problem 2. The iterations column contains the average number of iterations the L-BFGS method needed to converge. Bold numbers refer to the two smallest values in each column.

mesh	quadrature	optimizer	loss L_{int}	error ρ	error ρv	Iterations
rect	Gaussian	Adam	3.2e−6 ±3.4e−7	9.2e−4 ±8.3e−5	5.3e−2 ±**4.0e−3**	
		Hybrid	3.2e−6 ±**3.2e−7**	1.0e−3 ±7.3e−5	6.1e−2 ±5.5e−3	35.2 ±35.0
		L-BFGS	**1.4e−6** ±9.1e−7	**5.1e−4** ±2.0e−4	**3.0e−2** ±1.1e−2	1532.0 ±730.5
	random	Adam	4.6e−6 ±3.8e−7	9.5e−4 ±6.8e−5	5.4e−2 ±**3.6e−3**	
		Hybrid	4.6e−6 ±3.5e−7	1.0e−3 ±**5.9e−5**	6.1e−2 ±4.4e−3	**16.6** ±21.7
		L-BFGS	3.3e−6 ±1.0e−6	6.7e−4 ±2.1e−4	3.9e−2 ±1.3e−2	1103.6 ±403.9
trig	Gaussian	Adam	3.6e−6 ±3.5e−7	1.1e−3 ±1.0e−4	6.3e−2 ±5.6e−3	
		Hybrid	3.4e−6 ±**2.1e−7**	1.1e−3 ±**5.2e−5**	6.6e−2 ±4.9e−3	58.8 ±46.3
		L-BFGS	**9.7e−7** ±4.8e−7	**5.0e−4** ±1.3e−4	**3.0e−2** ±7.2e−3	1946.0 ±548.6
	random	Adam	9.6e−6 ±4.0e−7	1.3e−3 ±9.0e−5	7.2e−2 ±5.1e−3	
		Hybrid	9.5e−6 ±3.6e−7	1.4e−3 ±8.0e−5	7.8e−2 ±6.0e−3	**6.4** ±3.0
		L-BFGS	8.8e−6 ±1.5e−6	1.1e−3 ±1.5e−4	5.7e−2 ±8.9e−3	926.6 ±479.7

Table 7
Number of sampling points and sizes of the neural networks, n layers and k neurons, used in the scaling test for Problem 1 and Problem 2.

Test case	Problem 1 / Problem 2					Problem 1		Problem 2	
	$\lvert\mathcal{D}_{eq}\rvert$	$\lvert\mathcal{D}_{lb}\rvert$	n	k	# Parameter	$\lvert\mathcal{D}_i\rvert$	$\lvert\mathcal{D}_{rb}\rvert$	$\lvert\mathcal{D}_i\rvert$	$\lvert\mathcal{D}_{rb}\rvert$
1	400	10	2	20	≈ 100	20	10	10	20
2	800	20	2	40	≈ 200	40	20	20	40
3	1600	40	3	18	≈ 400	80	40	40	80
4	3200	80	3	26	≈ 800	160	80	80	160
5	6400	160	4	27	≈ 1600	320	160	160	320
6	12800	320	5	32	≈ 3200	640	320	320	640

5.3. Scaling model and parameter sizes

While for traditional numerical PDE methods there is a one-to-one match between the conditions and the degrees of freedom, this is not the case for physics informed neural networks. The number of sampling points as well as the number of neural network parameters should be increased simultaneously, but it is very much unclear how this relationship should be exactly. This is partly attributed to the fact that there is always an optimization error and the conditions, which are encoded in the loss function, are not satisfied at any sampling point.

Despite the lack of theoretical evidence, there is still a decision to be made. Until now we have used fixed numbers of parameters and sampling points for the neural network. In this subsection, we will validate this choice and compare different possibilities.

Our comparison is based on a simple scaling scheme. That is, we consider six test cases and every test case corresponds to one refinement level. From one to the next test case we double the amount of parameters of the neural network as well as the sampling points. Then, we study the relative error improvement between the test cases. To increase the number of parameters of a neural network, we increase the number of layers, i.e., make the neural network deeper, as well as the number of the neurons, i.e., make the neural network wider. In Table 7 the used sizes are shown. The network size and the number of sample points that are used in the previous subsections correspond to the fifth test case. The error is again measured by the averaged relative L_2 error of five runs. The neural networks are optimized by using the physics informed approach (18) and the L-BFGS method. The results are shown in Fig. 4.

For both problems the error decreases over the first five test cases. In the sixth test case the error increases. Hence, the choice we have made in (34), which corresponds to the fifth test case, is justified by these results. Therefore, the results in Subsection 5.1 and 5.2 are most likely the best we can expect.

In addition, we can make the following observations. In every level, we double the number of parameters and sampling points, but the error is not halved in consequence. This suggests a sub-linear convergence rate and leads to the unfortunate situation that disproportionately more computing effort has to be invested for more accurate results. Further, the errors of Problem 2 decrease more consistently than the errors of Problem 1. This might be due to the fact that the underlying conservation law of Problem 2 is simpler or the discontinuous initial data of Problem 1 poses special problems.

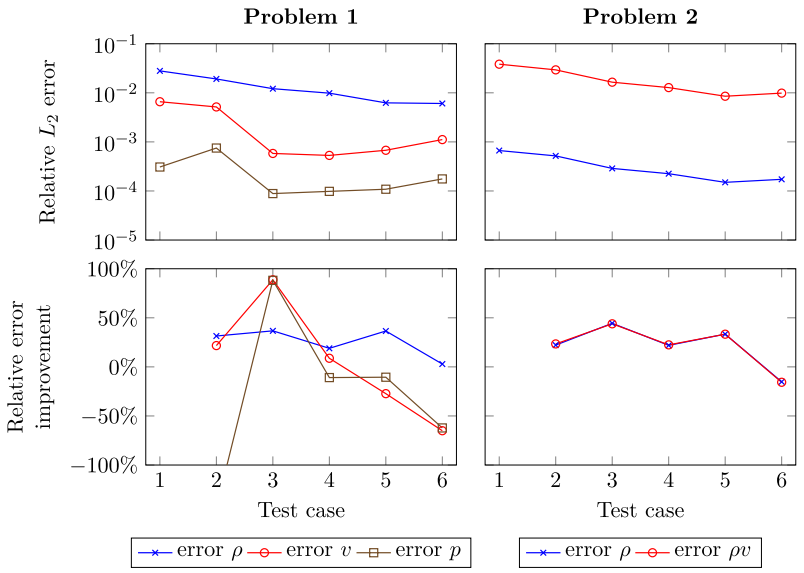


Fig. 4. Results of the scaling test for Problem 1 (left) and Problem 2 (right).

We were not able to decrease the error of this method arbitrarily low. This is certainly desirable but has not been described in the literature yet. First promising theoretical results for certain kinds of differential equations in [11] show that the error is composed of the quadrature error, the error that occurs by considering the approximations $\bar{L}_{eq}$, $\bar{L}_i$, $\bar{L}_{lb}$, $\bar{L}_{rb}$ instead of L_{eq}, L_i, L_{lb}, L_{rb}, as well as the training error, the value of the residuals L_{eq}, L_i, L_{lb}, L_{rb}. Consequently, if both errors can be decreased arbitrarily low, then the resulting approximation error can be arbitrarily low, too. While the quadrature error can be easily controlled by adding more sampling points, this is not true for the training error that stems from the optimization error. Therefore, a reasonable explanation for Fig. 4 is that the error is dominated by the quadrature error for the first few refinement levels and can therefore be decreased by increasing the number of sampling points. However, once the optimization error dominates the overall error, there is no further improvement possible. Hence, there is further research necessary to transfer the promising theoretical findings into practical results.

Finally, we have also evaluated the errors of all numerical results in this section with the maximum norm. Although the errors were slightly higher, we could not find any significant differences in the results.

6. Conclusion

In this article we have carried out a comprehensive case study on physics informed neural networks for gas transport problems. We have provided an objective comparison between different approaches to obtain a neural network that approximates the gas flow for two test problems. Our goal is to provide knowledge on how to effectively obtain these neural networks and what results can be expected in this specific use case. We want to use this knowledge to develop reduced order methods, i.e. PINNs, that avoid redundant computations, are fast for similar simulations and can maintain a high accuracy. This is important because these models and their approximate solutions form the building blocks for gas network simulations and, as a consequence, for gas network optimization and optimal control. Highly efficient approximation methods of suitable accuracy for gas flow in a pipe are therefore crucial to further the state-of-the-art in these more complex gas network settings.

One main difficulty of the original physics informed approach is the multi-objective training problem that needs to be solved to obtain the approximation. The individual objectives arise from separately enforcing the differential equation, the initial and boundary data. The single objective loss function for the training problem is obtained by forming a weighted sum of the individual terms. This translates the difficulty to the selection of these weights such that the individual terms in the loss function are suitably balanced. Dealing with this issue has been one central point of our case study. Here, our tests show that choosing appropriate weights in the physics informed loss function is very important to obtain a neural network with the lowest error. Our best weights have been determined by a random-search scheme that randomly chooses a list of weights, trains one neural networks for every chosen weight and then selects the neural network with the lowest training error. This procedure, however, is very computationally expensive and since the weights are problem dependent, it must be repeated when the problem changes.

A number of strategies (loss balancing methods) have been proposed in the literature to dynamically determine these weights during the training process. Such strategies have been applied successfully, with improvements over the original physics informed approach, but for different differential equations than in our case study. These methods usually have a small computational overhead compared to the original method. We have tested several of these strategies, but none of the

obtained solutions of our test problems were better and most of them were significantly worse than the solution determined by the random-search scheme and the original physics informed method. This leads to the important conclusion that these methods do not work on a general class of differential equations and that gas transport problems may pose own unique challenges which are not recognized by these methods. We also suggest that future work on loss balancing methods should include a comparison with a random-search scheme.

Additionally, we have tested the control volume physics informed approach – a formulation based on the integral form of the conservation law. This approach has some theoretical advantages over the original physics informed approach as well as the loss balancing extensions. Here, by considering the integral form, one can lower the integration dimension and also avoid automatic differentiation in the loss function as well as the aforementioned problem of determining weights. These advantages come along with an increased implementation overhead. However, and this is another important conclusion, in our test cases this method is outperformed by the original physics informed formulation based on the differential form. Additionally, in these tests deterministic quadrature rules have performed better than the random Monte Carlo quadrature rule.

We have also considered three different training procedures including the Adam method, the L-BFGS method and a hybrid of both. We conclude in our tests that the L-BFGS alone has produced the best results with the fewest iterations. However, in combination with algorithmic weighting methods we have observed a few cases which benefit from the hybrid scheme.

If we take all conclusions of our case study into account, we recommend the original physics informed approach trained by the L-BFGS method to obtain the best approximation of gas flow problems in a pipe by a neural network. If the computational budget allows, we strongly suggest to perform a random procedure to determine optimal loss balancing weights.

One of the biggest strength of physics informed neural networks is the flexibility to adapt the method easily for different use cases or exchange building blocks in the implementation such as the optimization method or the quadrature rule. Here, case studies like this one are very important to give rise to the most promising paths forward. Our tests show that to obtain highly accurate approximations we need to solve the optimization problem in the training phase with a very high accuracy and also that more accurate quadrature rules provide better results. However, in traditional machine-learning tasks one avoids to solve the training problem with a high accuracy to avoid an overfit. Our observations therefore show new research directions for PINNs that divert from traditional machine-learning.

We see two further research directions for PINNs which can lead to progress for our particular application of gas transport in a pipe but also for a more general problem setting. First, are there specific properties of hyperbolic conservation laws which limit the applicability of the considered loss-balancing methods and if so how to overcome these barriers? Second, can we identify which PINN extensions work particularly well with which PDE problems and understand why? Both of these directions aim at a deeper understanding of the connection between the underlying physics, as encoded in the PDE, and its representation within the neural network.

CRediT authorship contribution statement

Erik Laurin Strelow: Formal analysis, Software, Visualization, Writing – original draft. **Alf Gerisch:** Conceptualization, Software, Writing – review & editing. **Jens Lang:** Conceptualization, Supervision, Writing – review & editing. **Marc E. Pfetsch:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

The authors are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the collaborative research center TRR154 *"Mathematical modeling, simulation and optimisation using the example of gas networks"* (Project-ID 239904186, TRR154/3-2022, TP B01; Project-ID 39904186, TRR154/3-2022, TP A01) as well as the Graduate School CE within the Centre for Computational Engineering at TU Darmstadt.

The authors gratefully acknowledge the computing time provided to them on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities.

References

[1] P. Domschke, A. Dua, J.J. Stolwijk, J. Lang, V. Mehrmann, Adaptive refinement strategies for the simulation of gas flow in networks using a model hierarchy, Electron. Trans. Numer. Anal. 48 (2018) 97–113.

[2] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y.W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 9, PMLR, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.

[3] C. Himpe, S. Grundel, P. Benner, Model order reduction for gas and energy networks, J. Math. Ind. 11 (13) (2021).

[4] I.E.A. The, Role of Gas in Today's Energy Transitions, Technical report, Paris, 2019, https://www.iea.org/reports/the-role-of-gas-in-todays-energy-transitions.

[5] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.

[6] O. Kolb, J. Lang, P. Bales, An implicit box scheme for subsonic compressible flow with dissipative source term, Numer. Algorithms 53 (2010) 293–307.

[7] Randall J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002.

[8] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229.

[9] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Eng. 360 (2020) 112789.

[10] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, in: J. Lee, E. Darve, P. Kitanidis, M.W. Mahoney, A. Karpatne, M.W. Farthing, T. Hesser (Eds.), Proceedings of the AAAI 2021 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences, Aachen, in: CEUR Workshop Proceedings, vol. 2964, CEUR-WS, 2021.

[11] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, IMA J. Numer. Anal. (2022).

[12] R.G. Patel, I. Manickam, N.A. Trask, M.A. Wood, M. Lee, I. Tomas, E.C. Cyr, Thermodynamically consistent physics-informed neural networks for hyperbolic systems, J. Comput. Phys. 449 (2021) 110754.

[13] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[14] F.M. Rohrhofer, S. Posch, B.C. Geiger, On the Pareto front of physics-informed neural networks, arXiv:2105.00862, 2021.

[15] R. van der Meer, C. Oosterlee, A. Borovykh, Optimally weighted loss functions for solving pdes with neural networks, J. Comput. Appl. Math. 405 (2022).

[16] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (2021) A3055–A3081.