Truly Mesh-free Physics-Informed Neural Networks

Fabricio Arend Torres  
University of Basel  
fabricio.arendtorres@unibas.ch

Marcello Massimo Negri  
University of Basel  
marcellomassimo.negri@unibas.ch

Monika Nagy-Huber  
University of Basel  
monika.nagy@unibas.ch

Maxim Samarin  
University of Basel  
maxim.samarin@unibas.ch

Volker Roth  
University of Basel  
volker.roth@unibas.ch

Abstract

Physics-informed Neural Networks (PINNs) have recently emerged as a principled way to include prior physical knowledge in form of partial differential equations (PDEs) into neural networks. Although generally viewed as being mesh-free, current approaches still rely on collocation points obtained within a bounded region, even in settings with spatially sparse signals. Furthermore, if the boundaries are not known, the selection of such a region may be arbitrary, resulting in a large proportion of collocation points being selected in areas of low relevance. To resolve this, we present a mesh-free and adaptive approach termed particle-density PINN (pdPINN), which is inspired by the microscopic viewpoint of fluid dynamics. Instead of sampling from a bounded region, we propose to sample directly from the distribution over the (fluids) particle positions, eliminating the need to introduce boundaries while adaptively focusing on the most relevant regions. This is achieved by reformulating the modeled fluid density as an unnormalized probability distribution from which we sample with dynamic Monte Carlo methods. We further generalize pdPINNs to different settings that allow interpreting a positive scalar quantity as a particle density, such as the evolution of the temperature in the heat equation. The utility of our approach is demonstrated on experiments for modeling (non-steady) compressible fluids in up to three dimensions and a two-dimensional diffusion problem, illustrating the high flexibility and sample efficiency compared to existing refinement methods for PINNs.

1 Introduction

Many phenomena in physics are commonly described by partial differential equations (PDEs) which give rise to complex dynamical systems but often lack tractable analytical solutions. Important examples can be found for instance in fluid dynamics and in modeling of compressible fluids with typical applications in the design of gas and steam turbines (Oosthuizen & Carscallen, 2013), as well as modeling the collective motion of self-driven particles (Marchetti et al., 2013) such as flocks of birds or bacteria colonies (Szabó et al., 2006). Although much progress has been made in established numerical PDE solvers such as finite element and finite volume methods, the seamless incorporation of data remains an open problem (Freitag, 2020). Classical numerical data assimilation approaches can be divided into two types of methods (Xun et al., 2013), both of which are, however, very time
The underlying idea of PINNs is to use the expressive power of modern neural architectures for solving partial differential equations (PDEs) in a data-driven way, cf. (Raissi et al., 2019). Consider parameterized PDEs of the general form
\[ f(t, x|\lambda) := \partial_t u(t, x) + P(u|\lambda) = 0, \]
where \( P \) is a non-linear operator parameterized by \( \lambda \), and \( \partial_t \) is the partial derivative w.r.t. time \( t \in [0, T] \). The position \( x \in \Omega \) is defined on a spatial domain \( \Omega \subseteq \mathbb{R}^d \). The PDE is subject to initial condition \( g_0 \)
\[ u(0, x) = g_0(x) \]
for \( x \in \Omega \), and boundary conditions \( g_{\partial \Omega} \)
\[ u(t, x) = g_{\partial \Omega}(x) \]
for \( x \in \partial \Omega \) and \( t \in [0, T] \). The main idea of PINNs consists in approximating \( u(t, x) \) (and hence \( f(t, x) \)) with a neural network given a small set of \( N \) noisy observations \( u_{\text{obs}} \)
\[ u(t^{(i)}_i, x^{(i)}_i) + \varepsilon_{i} = u_{\text{obs}}^{(i)} \]
with \( i \in \{0, 1, \ldots, N\} \) and noise \( \varepsilon_i \ll u_i \). This enables us to consider the following two important problem settings: If \( \lambda \) is known, the PDE is fully specified, and we aim to find a solution \( u \) in a data-driven manner by training a neural network. The PDE takes the role of regularizer, where the particular physical laws provide our prior information. A second setting considers the inverse learning of the parameters \( \lambda \) by including them into the optimization process in order to infer physical properties such as the viscosity coefficient of a fluid (Jagtap et al., 2020).

**Loss functions.** Typically, PINNs approximate \( f(t, x) \) by the network \( f_{\Theta}(t, x) \) in which the parameters \( \Theta \) are adjusted by minimizing the combined loss of (i) reconstructing available observations \( (L_{\text{obs}}) \), (ii) softly enforcing the PDE constraints on the domain \( (L_f) \), and (iii) fulfilling the boundary \( (L_{\partial \Omega}) \) and initial conditions \( (L_{\text{init}}) \), i.e.
\[ \Theta = \arg \min_{\Theta} \left[ w_1 L_{\text{obs}}(\bf{X}, t, u_{\text{obs}}, \Theta) + w_2 L_f(\Theta) + w_3 L_{\partial \Omega}(\Theta) + w_4 L_{\text{init}}(\Theta) \right], \]
with loss weights \( w_i \in \mathbb{R}_{>0} \). A common choice for \( L_{\text{obs}} \), \( L_{\partial \Omega} \), and \( L_{\text{init}} \) is the expected \( L^2 \) loss, approximated via the average \( L^2 \) loss over the observations and via sampled boundary and initial conditions, respectively. It should be noted that the formulation of the forward and inverse problem are identical in this setting, as observations and initial conditions are implemented in a similar manner.

**Enforcing the PDE.** Although PINNs are by nature mesh-free, the PDE loss \( L_f \) in Eq. (5) used for the soft enforcement of Eq. (1) requires a similar discretization step for approximating an integral over the continuous signal domain,
\[ L_f(\Theta) = \frac{1}{|0,T| \times \Omega} \int_{t=0}^{T} \int_{\Omega} ||f_{\Theta}(t, x)||^2 dx dt = E_{p(t,x)} \left[ ||f_{\Theta}(t, x)||^2 \right] \approx \frac{1}{n} \sum_{i=1}^{n} ||f_{\Theta}(t_i, x_i)||^2 \]
with \( p(t, x) \) being supported on \([0, T] \times \Omega \). The points \( \{(t^{(j)}_i, x^{(j)}_i)\}_{j=1}^{n} \subseteq [0, T] \times \Omega \) on which the PDE loss is evaluated are commonly referred to as collocation points. Previous approaches for solving differential equations with neural networks rely on a fixed grid for selecting \( \{(t^{(j)}_i, x^{(j)}_i)\}_{j=1}^{n} \), (Lagaris et al., 1998; Rudd, 2013; Lagaris et al., 2000). Instead, subsequent work proposes stochastic estimates of the integral in each training step via (Quasi-) Monte Carlo methods (Srignano & Spiliopoulos, 2018; Lu et al., 2021; Chen et al., 2019) or Latin Hypercube sampling (Raissi et al., 2019) for a better coverage of the full input domain, i.e. by sampling from \( p(t, x) \propto 1 \). However, these approaches cannot be applied if there are no known boundaries or boundary conditions, e.g. for \( \Omega = \mathbb{R}^d \). Additionally, if the constrained region is large compared to the area of interest (as for example for a shock-wave in a large space), problems may arise as most collocation points will fall into areas of low density. We argue that due to the locality of particle interactions, the regions with higher density are more relevant for the regularization of our network. While manual selection of the
collocation points is possible in some settings, it usually requires some a-priori knowledge about the solution.

To address these short-comings of previous methods, we propose a mesh-free and adaptive approach for sampling collocation points, illustrated on the example of compressible fluids. By changing \( p(t, x) \) to the distribution over the particle positions in the fluid we effectively change the loss functional in Eq. (6). We then generalize to other settings, such as thermodynamics, by interpreting a positive, scalar variable of interest with a finite integral as a particle density. Within this work we specifically focus on PDEs that can be derived based on local particle interactions or can be shown to be equivalent to such a view, as for example is the case for the heat equation with its connection to particle diffusion.

Main contributions. The main contributions of this paper are as follows:

- We demonstrate that PINNs with uniform sampling strategies fail in settings with spatially sparse signals as well as in unbounded signal domains, and these problems can severely degrade the predictive performance of the networks.
- In order to overcome these limitations of existing approaches, we propose a fully mesh-free version of PINNs, in which the collocation points are sampled using physics-motivated MCMC methods. Our model is applicable to a huge range of dynamical systems governed by PDEs that share an underlying microscopic particle description, such as many types of hydrodynamic, electro-dynamic, and thermodynamic problems.
- We critically evaluate and compare our proposed method with existing approaches in high-dimensional settings. Compared to existing mesh refinement methods, significantly fewer collocation points are required to achieve similar or better predictive performances while still being more flexible.

2 Related Work

Adaptive Mesh Refinement. Classical numerical PDE methods require a discretization process based on a computational mesh or grid. The resolution of the mesh sets a fundamental trade-off between computational (and memory) complexity and accuracy. In practice, most computational effort is spent close to boundary regions. But even in settings without complex boundary geometries, local refinements and hierarchical meshes can be preferable over uniform Cartesian grids (Hirsch 2007; Plewa et al. 2005; Pons & Ersoy 2019). Furthermore, a wide range of adaptive mesh refinement methods has been proposed, that rely on heuristics to define regions of interest for the mesh refinement (Plewa et al. 2005; Pons & Ersoy 2019).

Alternative Meshes for PINNs. Recent work proposes local refinement methods for PINNs by adding more samples within regions of high error (Lu et al. 2021; Tadiparthi & Bhattacharya 2021). [Lu et al. 2021] suggest a method called residual adaptive refinement (RAR), where the PDE loss is regularly (after a certain amount of epochs) evaluated on a set of uniformly drawn samples. The locations corresponding to the highest PDE loss are then added to the set of collocation points used in training. [Tadiparthi & Bhattacharya 2021 (preprint)] further enhance RAR by learning a linear map between the uniform distribution and the distribution over the PDE loss by optimizing an optimal transport objective. By sampling uniformly and subsequently transforming these samples, it is attempted to focus on regions of higher error. Due to the conceptual similarity to RAR, we will denote this method as “OT-RAR”.

However, the underlying mechanism for exploring regions of high error is in both cases based on (quasi-) uniform sampling within the boundaries. As such, they do not resolve the issues with unknown boundaries and will furthermore be infeasible in higher dimensions.

Kinetic Theory: From particles to PDEs. Kinetic theory shows that essential conservation laws of fluids can be derived from a microscopic (or molecular) viewpoint (Born & Green 1946). Interactions describing the dynamics of a fluid are described starting from a set of individual particles. The basis of this approach is the so-called molecular distribution function \( \Psi \) over phase space, i.e. \( \Psi(t, x, v) \)}
such that
\[ \int_{\Delta x} \int_{\Delta v} \Psi(t, x, v) dv dx \] (7) is the probability that a molecule with a velocity within \( \Delta v = \Delta v_1 \Delta v_2 \Delta v_3 \) occupies the volume \( \Delta x = \Delta x_1 \Delta x_2 \Delta x_3 \). Based on this distribution function, it is possible to define common quantities as the (mass or particle) density, (local mean) velocity, and macroscopic PDEs by considering the local interactions of individual particles. The one-particle phase space is commonly known from its application in the Boltzmann equation for modelling two-body interactions describing gases (Green, 1956) and active matter (e.g. flocks of birds) (Bertin et al., 2006). The more general form including higher interaction terms is necessary for deriving conservation laws of liquids (Born & Green, 1946).

3 Methodology

In this section we introduce the concept of mesh-free particle-density PINNs (pdPINNs). Firstly, we examine limitations of the common PDE loss in Eq. (6) and, secondly, we present a solution by integrating over the position of fluid particles instead of the full support of the signal domain.

3.1 Modeling Compressible Fluids

We consider the problem of modeling non-steady compressible fluids, i.e. fluids with a spatially and temporally evolving density \( \rho(t, x) \) and velocity \( v(t, x) \). For the sake of notational brevity, we will denote these by \( \rho \) and \( v \) in the following. Our particular interest lies in the prediction of particle movement and thus density \( \rho \). Given noisy observations of the density \( \rho^{(i)} \), and potentially other quantities such as the velocity or pressure, we want to predict the density \( \rho \).

Commonly, the PDE serves as a physics-based regularizer of the network by enforcing PDE loss \( L_f \) in Eq. (6) during standard PINN training. For this, \( L_f \) is evaluated on a set of collocation points that are, for example, uniformly distributed on a bounded region. However, the limitations of this approach become apparent when considering an advection problem defined by the following PDE:

\[ \partial_t \rho + v \cdot (\nabla \rho) = 0. \] (8)

Figure 1 illustrates a one-dimensional case on the domain \([0, T] \times \Omega\), with \( \Omega = \mathbb{R} \), and a known constant velocity \( v \propto 1 \). We measure the density \( \rho^{(i)} \) at different (spatially fixed) points in time and space \( \{(t^{(i)}, x^{(i)})\} \), on which a neural network \( \rho_{\theta}(t, x) \) is trained. For optimizing the standard PDE loss \( L_f \) as given in Eq. (6), we would require a bounded region \( \Omega_B := [a, b] \subset \Omega \) with \( a < b \) and \( a, b \in \mathbb{R} \). This, in turn, leads to two issues:

1. Our region of interest, the moving density, is concentrated on a small subset of \( \Omega \). Enforcing Eq. (8) in areas with low-density has little regularizing effect on \( \rho_{\theta} \).

2. In order to define appropriate boundaries of \( \Omega_B \), we require prior knowledge about the solution which depends on the time frame, velocity, and the underlying PDE. Otherwise, the main bulk of density might move out of the considered area \( \Omega_B \).

We note that for issue (1.), the dynamics of fluids are assumed to be mainly caused by local interactions of particles, which is the case for commonly considered dynamics of gases, liquids or active particles (Hoover & Hoover, 2003; Toner & Tu, 1995). Furthermore, the naive solution of selecting an arbitrary large area with \( a \ll b \) for resolving (2.) would amplify the sparsity in (1.).

Therefore, a proper resolution to these issues should be able to (i) focus on areas that have a relevant regularizing effect on the prediction of \( \rho_{\theta} \) and (ii) adapt to the fluid movements without being restricted to a predefined mesh. Let us assume that we had access to the positions of each of the \( N \) total particles via the function \( x_j(t) \), with \( j \in \{0, 1, \ldots, N\} \). We would then propose to use the following loss over these positions within the considered time frame \([0, T]\):

\[ L_{pd}(\Theta) = \frac{1}{T} \int_{t=0}^{T} \frac{1}{N} \sum_{j=0}^{N} \left\| f_{\theta} \left( t, x_j(t) \right) \right\|_2^2 dt. \] (9)

It can be seen that the average in Eq. (9) is a sample-based estimator of an expected PDE loss. Namely, the expectation over the occupation probability of particles in the whole time frame, denoted
by the molecular distribution $\Psi(t, x) = \Psi(t)\Psi(x|t)$ introduced in section 2:

$$L_{pd}(\Theta) \approx \int_t^T \Psi(t) \int_\Omega \Psi(x|t) \left[ ||f_{\Theta}(t, x_j(t))||_2^2 \right] dt,$$

(10)

where $\Psi(t) \propto 1$ due to the conservation of mass. As the particle density corresponds directly to the occupation probability of a molecule $\Psi(t, x)$ with a changed normalization constant, we can estimate $L_{pd}$ via samples drawn from the normalized particle density or, similarly, from the normalized mass density when assuming a homogeneous fluid.

### 3.2 Particle-Density PINNs

In summary, we propose to draw collocation points from the normalized density:

$$(t_i, x_i) \sim \rho_N(t, x) = \frac{1}{Z} \rho(t, x).$$

(11)

However, as the true density is unknown, we instead rely on the learned density $\rho_{\Theta}(t, x)$ as a proxy provided by the neural network. We denote the associated normalized PDF by $q_{\Theta}(t, x)$ with support on $[0, T] \times \Omega$. The PDE loss is then defined as the expectation w.r.t. $q$:

$$L_{pd}(\Theta) = E_{q_{\Theta}(t, x)} \left[ ||f_{\Theta}(t, x)||_2^2 \right] = \int_t^T q_{\Theta}(t) \int_\Omega q_{\Theta}(x|t) ||f_{\Theta}(x, t)||_2^2 dx \ dt.$$

(12)

In order to approximate this integral, samples need to be drawn from $q_{\Theta}(t, x)$. This can be done in a principled way by using dynamic Monte Carlo methods, despite the fact that the normalization constant $Z$ is unknown. Such a sampling procedure additionally eliminates the need for $\Omega$ to be a bounded domain in settings where such boundaries are not available. In contrast, methods based on the mesh-based loss in Eq. (6) are not directly applicable to unbounded domains such as $\Omega = \mathbb{R}^d$, and potentially arbitrary boundaries have to be introduced.

Although motivated in the context of an advection problem, the proposed approach is generally applicable to compressible fluid dynamics settings and more general settings as we elaborate on below.

### Further particle-based PDEs

The advection equation (8) can be seen as a special case of mass conservation (assuming $\nabla \cdot v = 0$) which is one of the fundamental physical principles expressed as a continuity equation. The continuity equation relates temporal changes of the fluid density $\rho$ to spatial changes of the flux density $\rho v$ through

$$\partial_t \rho + \nabla \cdot (\rho v) = 0.$$

(13)

Most famously, it belongs to a broader set of problems considering viscous flows, which are described by the Navier-Stokes equations. An even more general setting, for which conservation laws such
as the conservation of momentum do not hold, considers the modeling of self-propelled, active particles, for which Toner and Tu (Toner & Tu 1995, Tu et al. 1998, Toner & Tu 1998) introduced hydrodynamic equations. Our method is generally applicable in settings where (i) a non-negative scalar field (with a finite integral) of interest can be interpreted as a particle density, and (ii) the local interactions of these particles give rise to the considered PDEs. In the heat equation, for instance, it is well-established that the diffusion of the temperature coincides with the diffusion of particles according to Fick’s second law. The diffusion of the temperature $T$ within a homogeneous medium is described by a parabolic PDE defined as

$$\partial_t T - \alpha \nabla^2 T = 0,$$  

where $\alpha$ is the thermal diffusivity and $\nabla^2$ is the Laplacian operator. For $T := \rho$, the diffusion equation of particles is obtained, substituting the thermal diffusivity with the mass diffusivity. Other possible applications involve Maxwell’s equations for conservation of charge in electrodynamics.

## 4 Model and Implementation

Within the literature, a wide range of different network architectures and optimization strategies for PINNs have emerged. They either emphasize well-behaved derivatives with respect to the input domain (Sitzmann et al., 2020), allow higher expressivity for modelling high frequency data (Tancik et al., 2020; Wang et al., 2021b), or resolve gradient pathologies within PINNs (Wang et al., 2021a). In our work, the particular choice of the network architecture is independent of our proposed method. For the experiments in section 5 we mostly choose fully-connected networks with either sinusoidal or tanh activations.

### Finite total density

For reformulating the density $\rho_{\theta}$ as a probability, we have to ensure non-negativity as well as a finite integral over the input domain $\Omega$. Non-negativity can be achieved via a squared activation function after the last layer (or alternatively with an exponential or softplus activation). An additional bounded activation function $g$ is added, which guarantees the output to be within a pre-specified range $[0, c_{\text{max}}]$. By further multiplying the bounded output of the network with a Gaussian kernel, we enforce the integral over $\mathbb{R}^d$ to be finite. Summarizing these three steps, let $\tilde{\rho}_\theta$ denote the output of the last layer of our fully connected neural network and $p_{\text{gauss}}(x) = \mathcal{N}(x; \mu, \Sigma)$, then we predict the density $\rho_{\theta}$ as

$$\rho_{\theta}(t, x) = p_{\text{gauss}}(x) g(\tilde{\rho}_\theta(t, x)^2) \leq c_{\text{max}} p_{\text{gauss}}(x).$$  

The used mean $\mu$ and covariance $\Sigma$ are maximum likelihood estimates based on the observations $x$, i.e., the sample mean $\bar{x}$ and covariance $\Sigma$ of the sensor locations. In this way we enforce a zero-prior in regions far away from the observed data. To avoid being too restrictive for the network, we add a scaled identity matrix to the covariance $\Sigma = \Sigma + c \cdot I$, which may be manually adjusted depending on the problem at hand.

### Markov chain Monte Carlo sampling

Finally, for drawing samples from the unnormalized density $\rho_{\theta}(t, x)$, we use Markov chain Monte Carlo methods. For the sake of simplicity, we used a random walk Metropolis-Hastings sampler, with the step size of the random walk fine-tuned for each problem setting. The chains are initialized on an estimated maximum of the density within the convex hull of our data. Although not explored within this work, we highlight that gradient-based samplers, such as the Metropolis-adjusted Langevin algorithm (Xifara et al., 2014) or Hamiltonian Monte Carlo (Duane et al., 1987; Betancourt, 2017), are particularly suited for our setting, as the gradients of $\rho_{\theta}$ with respect to the input space are readily available. For problems where boundaries are known and we have to sample from a constrained region, reflective MCMC samplers can be used (Mohasel Afshar & Domke, 2015). Alternative approaches for sampling from an unnormalized probability distribution such as modern variational inference schemes (Rezende & Mohamed, 2015) may pose attractive alternatives to MCMC, providing a quick sampling mechanism with the drawback of only approximating the target distribution.

Although the predicted density $\rho_{\theta}(t, x)$ approximates $\rho(t, x)$ once trained, it is purely random in its initial, untrained stage. As such, we start training the pdPINNs with a warm-up phase in which collocation points are obtained from a pre-specified background distribution of choice, slowly increasing the fraction of samples obtain from $\rho_{\theta}(t, x)$. For details we refer to Appendix section A.1.
5 Experiments

In this section we demonstrate the advantages of pdPINNs compared to uniform sampling and the adaptive refinement methods RAR and OT-RAR proposed by Lu et al. (2021) and Tadiparthi & Bhattacharya (2021), respectively. Although we refer to uniform sampling, we rely in all our experiments on quasi-random Sobol sequences for more stable behavior in the low samples regime. We considered slight variations of their proposed implementations to allow a fair comparison with a limited number of admissible samples. Specifically, instead of adding to an ever increasing set of collocation points, we limit them to a specified budget. The changes are elaborated in more details in the Appendix section A.2 and provided as code in the supplementary material.

![Figure 2: Visualization of the 2D compressible fluid experiment. (a) bird-eye view of the groundtruth particle density. (b) z-projection of the data over time, obtained by summing over the xy grid cells.](image)

5.1 Particle Simulation of a Compressible Fluid

As a challenging prediction task, a non-steady compressible fluid in three dimensions is simulated by propagating fluid parcels through a pre-defined velocity field. In other words, we simulate a fluid from the Lagrangian perspective with the conservation of mass as the underlying PDE (see Eq. (13)). The paths of the roughly 240’000 parcels are solved for with a basic backward Euler scheme. The particle density and velocity are obtained by counting the number of parcels within a voxel, and by averaging the individual velocities. For the training observations provided to the network, we introduce a set of spatially fixed sensors (or radars) which count the number of fluid parcels that are present within a radius $r$ at a given time. The birds-eye view of the setting is shown in Figure 2a, where circles indicate the area covered by the radars. Along the $z$-axis, the simulated sensor distinguishes over 21 contiguous altitude layers. Another disjoint set of sensors is provided for the validation set while the test performance is evaluated on a grid. Figure 2b shows the 3D simulated data projected along the $z$-axis and over time, which gives a complementary illustration with respect to the birds-eye view. For details regarding the data set and the velocity field we refer to the Appendix section A.3. The full code for the data generation, as well as the data sets themselves, is provided in the supplementary material.

We consider two different types of (common) application settings with increasing difficulty and dimensionality: (i) the 2D setting obtained by projecting onto the $xy$-plane, and (ii) the full 3D setting. In both settings we look at the dynamics over the time frame $t \in [0, 3]$ and learn both the density and the velocity using two SIREN (Sitzmann et al., 2020) networks $ρ_Θ_1(t, x)$ and $v_Θ_2(t, x)$. Within the experiments, the compared settings always correspond to similar architectures, hyperparameters, and optimizers. In all settings, the PDE used for regularizing the networks is the continuity equation for the conservation of mass (see Eq. (13)).

5.1.1 Projection on $xy$-plane

In the 2D setting, the density and velocity are measured by $15^2$ sensors on the $xy$-plane within the area $[-4, 4]^2$ at 21 equidistant timesteps. For each method, we perform a hyper-parameter search over different PDE-weights $ω_2$ (see Eq. (5)) and select the one with the highest average validation $R^2$. Figure 3a shows the test $R^2$ performance for different numbers of collocation points and 11 different seeds, the distribution of which are visualized with the box plots. Results clearly show that the proposed pdPINN outperforms competing methods across all number of collocation
points. Furthermore, we observe that the performance gap shrinks as the collocation points increase, eventually converging to the same limiting value. This is expected, since in the high-sample limit the sample strategy becomes irrelevant and all methods should provide equivalent results. However, what is relevant is that the proposed pdPINN model achieves significantly better results with significantly fewer collocation points. We further took into consideration how the PDE-weight $w_2$ affects the performance as a function of the number of collocation points and verified we still obtain results consistent with Figure 3a. We point the reader to the Appendix section A.3 for a more detailed view of these results.

Figure 3: Explained variance of $\sqrt{\rho}$ evaluated on the test set, for different number of collocation points in the (a) 2D and (b) 3D experiment.

Figure 4: 3D setting predictions (obtained with 2048 collocation points) are summed over $xy$ grid cells to obtain $z$-axis projection over time. The OT-RAR method shows disconnected density predictions that violate mass conservation.

5.1.2 Full Three-Dimensional Setting

In the 3D setting, the density and velocity are again measured by $15^2$ sensors on the $xy$-plane, but within the smaller area $[-3, 3]^2$ at 11 equidistant timesteps and 21 equidistant altitude layers. Compared to the 2D setting, we have a spatially denser set of sensors, but fewer time steps.

Qualitative results are reported in Figure 4 displaying the density predictions over time and projected over the $z$-axis. This is obtained by summing over the $xy$ grid cells and allows us to visualize that the OT-RAR predictions clearly violate mass conservation. Conversely, the pdPINN model accurately predicts the density while also faithfully enforcing the continuity equation. In Figure 3b we quantitatively evaluate the performance of the different methods for different numbers of collocation points, with the method-specific PDE-weight $w_2$ being selected as in the 2D setting. Also in the 3D setting, quantitative results show that the proposed pdPINN approach outperforms baseline methods across all number of collocation points.

As expected, in this higher-dimensional (3D and time) setting the difference in performances is even more pronounced. Already with very few collocation points (512) pdPINNs achieve results comparable with those obtained with orders of magnitude more points (32768). We verified that, even when getting close to the memory limit of a NVIDIA Titan X GPU, other sampling strategies at best achieve comparable results with pdPINNs. Lastly, results from the 2D and 3D experiments suggest
that the strength of the proposed method would be even more considerable for higher-dimensional settings.

5.2 Heat Equation

We further consider a 2D diffusion problem, namely the heat equation introduced in section 5.2 where randomly distributed sensors provide measurements of the temperature. We focus on a general setting with the initial conditions being zero temperature everywhere except for a specified region, as shown in Figure 5a, and we let the system evolve in the time range \( t \in [0, 0.2] \). The networks are only provided sensor measurements of the temperature. For details regarding the training and test data we refer to the Appendix section A.4. During the warm-up phase of the pdPINN, collocation points are sampled uniformly, and afterwards 90\% of the samples are drawn from the particle density distribution, which is proportional to the modeled temperature. Collocation points are re-sampled every 500 epochs. Differently from previous experiments, the employed architecture is a fully-connected two-layer neural network with 32 hidden units and tanh activations.

Density predictions for PINNs with uniform sampling and pdPINNs are illustrated in Figure 5b and 5c respectively, and compared with the ground truth in Figure 5a. We can clearly observe that the uniform sampling strategy does not allow putting enough focus on the relevant parts of the domain, i.e. regions with high temperature, and that it visibly fails to reconstruct the temperature profile. In contrast, the pdPINN approach allows sampling in the region of higher density and reliably predicts the true temperature at all times. We also evaluate quantitatively the performance of the two approaches in terms of the R\(^2\) test error over the predicted temperature. Consistent with previous 2D and 3D compressible fluids experiments, we show that the proposed pdPINNs outperform PINNs with uniform sampling strategy. As in previous settings, we show that with few samples the regularization enforced by the PDE loss is not strong enough, leading to comparable results in both approaches. However, as the number of sample increases, the PDE loss enforced with MCMC samples quickly and steadily outperforms uniform sampling. We also verified that in the limit of high samples the two sampling strategies converge, as expected. Further details regarding the quantitative results are discussed in the Appendix section A.4.

![Figure 5: Temperature predictions of the heat equation experiment (trained with 128 collocation points) at time \( t \sim 0.044 \). (a) ground truth (b) uniform sampling, and (c) pdPINN.](image)

6 Conclusion

In this work, we introduced a general extension to PINNs applicable to a great variety of problem settings involving physics-based regularization of neural networks. In order to overcome the limitations of classical mesh-based PINNs, we introduce a novel PDE loss that is defined with respect to the particle density in rather general types of PDEs. By employing MCMC methods to sample collocation points from the density approximated by the network, we derive an efficient and easy-to-implement improvement for providing a more appropriate regularization objective in PINNs. In particular, our new pdPINNs are completely mesh-free, thereby overcoming severe efficiency problems of classical PINNs in high-dimensional and sparse settings. Further, the absence of a mesh allows us to elegantly handle settings with uncertain or unknown domain boundaries.
These advantages of the pdPINN model increase the relevance of PINNs in higher dimensional real-world problems. Limitations of the proposed approach are settings where the physical variable of interest does not align with the normalized particle density we sample from. An example would be modelling incompressible fluids where the focus is on the dynamics of the velocity field rather than on the density, which is constant by assumption.

As we have demonstrated, our method is applicable to a great range of PDEs, ranging from hydrodynamic flow problems to electro- and thermodynamic problems. Extensions of pdPINNs are possible in more refined sampling procedures.

References

Bertin, E., Droz, M., and Grégoire, G. Boltzmann and hydrodynamic description for self-propelled particles. *Physical Review E*, 74(2):022101, 2006.

Betancourt, M. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Born, M. and Green, H. S. A general kinetic theory of liquids i. the molecular distribution functions. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 188(1012):10–18, 1946.

Chen, J., Du, R., Li, P., and Lyu, L. Quasi-monte carlo sampling for machine-learning partial differential equations. *arXiv preprint arXiv:1911.01612*, 2019.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rollet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL [http://jmlr.org/papers/v22/20-451.html](http://jmlr.org/papers/v22/20-451.html).

Freitag, M. A. Numerical linear algebra in data assimilation. *GAMM-Mitteilungen*, 43(3):e202000014, 2020.

Green, M. S. Boltzmann equation from the statistical mechanical point of view. *The Journal of Chemical Physics*, 25(5):836–855, 1956.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL [https://doi.org/10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2).

Hirsch, C. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

Hoover, W. G. and Hoover, C. Links between microscopic and macroscopic fluid mechanics. *Molecular Physics*, 101(11):1559–1573, 2003.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL [https://arxiv.org/abs/1412.6980](https://arxiv.org/abs/1412.6980).

Knott, M. and Smith, C. S. On the optimal mapping of distributions. *Journal of Optimization Theory and Applications*, 43(1):39–49, 1984.
Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Lagaris, I. E., Likas, A. C., and Papageorgiou, D. G. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

Lam, S. K., Pitrou, A., and Seibert, S. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

Marchetti, M. C., Joanny, J.-F., Ramaswamy, S., Liverpool, T. B., Prost, J., Rao, M., and Simha, R. A. Hydrodynamics of soft active matter. *Reviews of modern physics*, 85(3):1143, 2013.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

Mohasel Afshar, H. and Domke, J. Reflection, refraction, and hamiltonian monte carlo. *Advances in neural information processing systems*, 28, 2015.

Oosthuizen, P. H. and Carscallen, W. E. *Introduction to compressible fluid flow*. CRC press, 2013.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Plewa, T., Linde, T., Weirs, V. G., et al. Adaptive mesh refinement-theory and applications. 2005.

Pons, K. and Ersoy, M. Adaptive mesh refinement method. part 1: Automatic thresholding based on a distribution function. 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Recktenwald, G. Finite-difference approximations to the heat equation. 2004.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rudd, K. *Solving partial differential equations using artificial neural networks*. PhD thesis, Duke University, 2013.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Szabó, B., Szöllösi, G., Gönci, B., Jurányi, Z., Selmecei, D., and Vicsek, T. Phase transition in the collective migration of tissue cells: Experiment and model. *Physical Review E*, 74(6):061908, 2006.
Tadiparthi, V. and Bhattacharya, R. Optimal transport based refinement of physics-informed neural networks. *arXiv preprint arXiv:2105.12307*, 2021.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Toner, J. Reanalysis of the hydrodynamic theory of fluid, polar-ordered flocks. *Physical Review E*, 86(3):031918, 2012.

Toner, J. and Tu, Y. Long-range order in a two-dimensional dynamical XY model: how birds fly together. *Physical review letters*, 75(23):4326, 1995.

Toner, J. and Tu, Y. Flocks, herds, and schools: A quantitative theory of flocking. *Physical review E*, 58(4):4828, 1998.

Tu, Y., Toner, J., and Ulm, M. Sound waves and the absence of Galilean invariance in flocks. *Physical review letters*, 80(21):4819, 1998.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Wang, S., Wang, H., and Perdikaris, P. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Xifara, T., Sherlock, C., Livingstone, S., Byrne, S., and Girolami, M. Langevin diffusions and the metropolis-adjusted langevin algorithm. *Statistics & Probability Letters*, 91:14–19, 2014.

Xun, X., Cao, J., Mallick, B., Maity, A., and Carroll, R. J. Parameter estimation of partial differential equation models. *Journal of the American Statistical Association*, 108(503):1009–1020, 2013.
A Appendix

A.1 Background Sampling for pdPINNs

At initialization, the network prediction $\rho_{\Theta}$ is random and thus does not carry any useful information, i.e. sampling from this density would be meaningless. Therefore, we start training the pdPINNs with a warm-up phase in which samples are obtained from a pre-specified background distribution:

$$x \sim p_{bg}(t, x) = p(t)p_{bg}(x|t)$$ (16)

with $p(t) = \mathcal{U}(0, T)$. To avoid introducing a mesh, we could rely on the previously estimated Gaussian distribution introduced in Section 4, i.e. $p_{bg}(x|t) = \mathcal{N}(x)$. As a second alternative, we consider random linear combinations of the convex hull of $\{x^{(i)}\}_{i=1}^{N}$ spanned by $c$ data points summarized as rows of matrix $Z \in \mathbb{R}^{c \times d}$. This leads to $x = mZ$ with weight $m \in \mathbb{R}^{c}$ which can be drawn from a Dirichlet distribution, i.e. $m \sim \text{Dir}(\alpha = 1)$. Of course, a uniform sampling mechanism on a defined region is also suitable and the definitive choice strongly depends on the data and PDE at hand.

We initially draw all samples from the background distribution, and then slowly increase the proportion of samples obtained from the particle density, as we found that leaving some background samples slightly helps in the training.

A.2 Implementation of RAR and OT-RAR

For our comparison, we considered the adaptive refinement methods RAR and OT-RAR, proposed by [Lu et al., 2021] and [Tadiparthi & Bhattacharya, 2021, preprint]. Both methods rely on consecutive refinements of a fixed grid in the initial proposal. The number of collocation points is steadily increased and collocation points once added will not be removed. To allow for a more fair comparison, we adapt both methods to use a limited budget of points, and in addition we regularly resample them. This leads to a slightly modified version of the methods which is similar in spirit. For learning the linear mapping proposed by [Tadiparthi & Bhattacharya, 2021], we rely on the PyOT (Flamary et al., 2021) implementation of [Knott & Smith, 1984]. The pseudo-code for sampling a set of collocation points is given in Algorithm 1 and Algorithm 2. The required input $f_{\Theta}$ refers to the PDE approximated by the network, as discussed in Section 4. For more specific details on the methods we refer to the original papers.

Algorithm 1 Adapted RAR

**Input:** $f_{\Theta}$, uniform distribution $\mathcal{U}_B$, number of col. points $k$, previous col. points $X_{prev}$.

- $X_{prop} \leftarrow [x_1, x_2, \ldots, x_k]^T$ with $x_i \sim \mathcal{U}_B$ \hspace{1cm} $\triangleright$ Sample proposals
- $X_{comb} \leftarrow \text{concat}(X_{prev}, X_{prop})$ \hspace{1cm} $\triangleright$ Concatenate old and new points
- $X_{new} \leftarrow \text{topk}(X_{comb}, ||f_{\Theta}(X_{comb})||^2, k)$ \hspace{1cm} $\triangleright$ Keep top $k$ proposed points based on $f_{\Theta}$

**Output:** $X_{new}$

A.3 Experiments: Compressible Fluid

In the supplement, we provide code in Python for the data generation and for the pdPINN model. Below we provide the details for all the experiments we conducted. Furthermore, we provide short videos showing the predicted density movements for each different approach. More details on this can be found in the README.html provided in the supplementary files. The model is implemented using PyTorch [Paszke et al., 2019], with a custom Python implementation of the Metropolis-Hastings MCMC sampler relying on NumPy, SciPy, and Numba [Harris et al., 2020; Virtanen et al., 2020; Lam et al., 2015].

All experiments were run on Nvidia GeForce GTX Titan X GPUs with 12 GB VRAM. A total of 6 GPUs were available, which were used to parallelize multiple runs. In most settings, training in each experiment took less than 10 minutes. The only exceptions were settings with a large amount of
Algorithm 2 Adapted OT-RAR

**Input:** $f_\Theta$, uniform distribution $U_B$, number of col. points $k$, number of points for empirical distribution $j < 2k$, previous col. points $X_{\text{prev}}$.

\[
X_{\text{prop}} \leftarrow [x_1, x_2, \ldots, x_k]^T \text{ with } x_i \sim U_B \quad \triangleright \text{Sample proposals}
\]

\[
X_{\text{comb}} \leftarrow \text{concat}(X_{\text{prev}}, X_{\text{prop}}) \quad \triangleright \text{Concatenate old and new points}
\]

\[
X_{\text{target}} \leftarrow \text{topk}(X_{\text{comb}}, ||f_\Theta(X_{\text{comb}})||^2_2, j) \quad \triangleright j \text{ samples for target empirical distribution}
\]

\[
X_{\text{source}} \leftarrow [x_1, x_2, \ldots, x_j]^T \text{ with } x_i \sim U_B \quad \triangleright j \text{ samples for source empirical distribution}
\]

\[
M_{\text{OT}} \leftarrow \text{LinOT}(X_{\text{source}}, X_{\text{target}}) \quad \triangleright \text{Obtain linear operator that maps to target distribution}
\]

\[
X_{\text{new}} \leftarrow [x_1, x_2, \ldots, x_k]^T \text{ with } x_i \sim U_B \quad \triangleright \text{Sample uniformly}
\]

\[
X_{\text{map}} \leftarrow M_{\text{OT}}(X_{\text{new}}) \quad \triangleright \text{Map samples to target distribution}
\]

**Output:** $X_{\text{map}}$

collocation points, with the bottleneck being the non-optimized bare-bones implementation of the Metropolis-Hastings sampler.

### A.3.1 Data Generation

The velocity field in the $xy$-plane was generated from a scalar potential field $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}$ and the $z$-component of a vector potential $a : \mathbb{R}^2 \rightarrow \mathbb{R}$. Through the Helmholtz decomposition\(^1\), we can construct the velocity field $v_{xy} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$:

\[
v_{xy} \begin{bmatrix} x \\ y \end{bmatrix} = -\nabla \Phi + \begin{bmatrix} \delta a/\delta y \\ -\delta a/\delta x \end{bmatrix}. \tag{17}
\]

For both experiments the following fields were used:

\[
\Phi \begin{bmatrix} x \\ y \end{bmatrix} = -\frac{1}{2} (x - 2) \cdot (y - 2), \tag{18}
\]

\[
a \begin{bmatrix} x \\ y \end{bmatrix} = -\frac{1}{5} \exp \left( -\frac{2}{3} x^2 - \frac{2}{5} y^2 \right). \quad \tag{19}
\]

The derivatives were obtained using the symbolic differentiation library SymPy \cite{Meurer_2017}. To add a nonsteady component, the resulting velocity field is modulated in amplitude as a function of time $t \in [0, 3]$:

\[
v_{xy}(t, x, y) = v_{xy} \left( \frac{x}{y} \right) \left( 3 \cdot \sin \left( \frac{2}{3} \pi t \right) + 0.05 \right). \tag{20}
\]

The $z$ (altitude) component of the velocity only depends on time and is given by:

\[
v_z(t) = 1.6 \cdot \sin \left( \frac{4}{3} \pi t \right). \tag{21}
\]

For the initial distribution of the fluid, the parcel positions were drawn from Gaussian mixtures. For $t \in [0, 3]$, these parcels were simulated using the above constructed velocity field.

The measurements at the sensors were obtained by counting the number of parcels within a given radius over multiple timesteps. The density corresponds to the mass divided by the sensor area, and the velocity is an average over all the parcel velocities. For the training data additional zero-mean isotropic Gaussian noise is added to all measurements.

---

\(^1\)This is the 2D formulation of the Helmholtz decomposition, where the vector potential has non-zero components only along the $z$-axis as in $a_{3d} = [0, 0, a]^T$. The full decomposition is commonly written as $v_{3d} = -\nabla \Phi_{3d} + \nabla \times a_{3d}$. 

14
A.3.2 Architecture and Training

In both experiments, the networks for density $\rho_{\Theta_1}$ and velocity $v_{\Theta_2}$ prediction (parameterized by $\Theta_1$ and $\Theta_2$, respectively) are fully-connected layers with sinusoidal activation functions, as proposed by Sitzmann et al. (2020). The number of layers and units for each setting is shown in Table A.1.

The sine frequency hyperparameter required in the SIREN architecture was tuned by hand according to the validation loss of the baseline model (i.e. without a PDE loss), leading to a sine-frequency of 12 for the 2D setting, and 5 for the 3D setting. We note that the proposed default value of 30 in Sitzmann et al. (2020) heavily overfits our relatively low-frequency data and we thus recommend an adjustment of this hyperparameter for usage in PINNs.

For training the network, the ADAM optimizer (Kingma & Ba, 2014) with a learning rate of $8 \times 10^{-4}$ (2D Setting) or $10^{-4}$ (3D Setting) was used. The learning rate was multiplied by a factor of 0.99 each epoch. All models were trained for 300 (3D setting) or 500 (2D setting) epochs. The 2D setting was trained using full-batch gradient descent, whereas for the 3D setting we used a mini-batch size of 6031. In all experiments we trained and evaluated on 10 different random seeds.

Table A.1: Architecture for Particle Simulation Experiments.

| Experiment | Input | Output Variable | # Hidden Layers | # Hidden Units |
|------------|-------|-----------------|-----------------|----------------|
| 2D         | $[0, T] \times \mathbb{R}^2$ | Density $\rho_{\Theta_1} \in \mathbb{R}^+$ | 2               | 256            |
|            |       | Velocity $v_{\Theta_2} \in \mathbb{R}^2$ | 1               | 64             |
| 3D         | $[0, T] \times \mathbb{R}^3$ | Density $\rho_{\Theta_1} \in \mathbb{R}^+$ | 6               | 256            |
|            |       | Velocity $v_{\Theta_2} \in \mathbb{R}^3$ | 3               | 256            |

A.3.3 Results for varying PDE Weights

As mentioned in Section 5.1, the weight of the PDE loss term has to be adjusted depending on the number of collocation points and the chosen loss term. For a low number of collocation points, a high weight for the PDE loss will disturb the training, resulting in worse performance than the baseline without additional regularization. The more collocation points are used, the higher the weight can be without leading to negative effects. Consequently, one would ideally have to optimize the weight depending on the number of collocation points. Figures A.6 and A.7 show this behaviour for the 2D and 3D setting of the compressible fluid experiment. The colors indicate different weights for the respective method, with the color gradient (blue → red) indicating the increase of the weight. It is important to note, that the colors between different methods should not be interpreted as similar weights. Each method relies on a different loss, making a comparison of the weights for this loss meaningless. We can see in both settings, that pdPINN performs favourably, especially when using only few collocation points. To emphasize the advantage provided by pdPINNs, Figure A.8 compares the proposed pdPINN with only 8192 collocation points with the uniform sampler using 65536 (i.e. a factor of 8), which is for our implementation close to the hardware limit of the used GPU with 12GB VRAM (depending on the used network architecture and mini-batch size). In both cases, we selected the PDE weight $w_2$ based on the maximum validation $R^2$. As a sidenote, the deviation of pdPINN at 128 collocation points is caused by a single outlier out of the 10 different random seeds.

A.4 Experiments: Heat Equation

The dataset for the heat equation experiment was generated by numerically solving the heat equation through the finite difference method, precisely the Forward Time, Centered Space (FTCS) approximation (Recktenwald, 2004). We used Dirichlet boundary conditions in form of zero temperature around a squared shape far away from the relevant domain. These boundary conditions are not provided to the PINNs for a slightly more difficult setting. Overall, the dataset is composed of 1000 training points, 1’971’120 test points and 492’780 validation points. We made sure training points contained enough information about the initial condition, i.e. we selected a sufficient amount of points around the initial source of non-zero temperature. In contrast, validation and test points are taken uniformly in time and space. The implementation is in PyTorch (Paszke et al., 2019), using the ADAM optimizer (Kingma & Ba, 2014) combined with an exponential learning rate scheduler which multiplies the learning rate by a factor of 0.9999 at each epoch, starting with a rate of $10^{-4}$ and decreasing it until
Figure A.6: 2D compressible fluid experiment: Average test $R^2$ of predicted $\sqrt{\rho}$. The colors indicate different PDE weights $w_2$ within each method. The bars correspond to 95% confidence intervals for the mean estimation based on 1000 bootstrap samples.

Figure A.7: 3D compressible fluid experiment: Average test $R^2$ of predicted $\sqrt{\rho}$. The colors indicate different PDE weights $w_2$ within each method. The bars correspond to 95% confidence intervals for the mean estimation based on 1000 bootstrap samples.

Figure A.8: 3D compressible fluid experiment: Boxplot of Test $R^2$ of predicted $\sqrt{\rho}$ comparing pdPINN and uniform sampling with a factor 8 difference for the number of collocation points. For each method, the used PDE weight was selected based on the highest validation $R^2$.

reaching a minimum value of $10^{-5}$. Training was terminated through early-stopping, as soon as the validation $R^2$ didn’t improve for more than 3000 epochs.

Figure A.9 illustrates the test $R^2$ of the predicted $T$ averaged over 20 different seeds. Error bars correspond to 95% confidence interval for the mean estimation, based on 1000 bootstrap samples, while colors indicate the different PDE weights $w_2$ explored. When the PDE loss is enforced with few samples (16) the regularization is not strong enough, which is to be expected. Hence PINNs and pdPINNs show similar results in this regime. However, as the number of samples increases (32-64-128-256), the PDE loss enforced by the proposed pdPINNs quickly outperforms uniform sampling. Lastly, for a sufficiently large number of samples (512-1024) we see no significant difference within
the two sampling procedures, as in such a low-dimensional domain the uniform sampling fully explores the considered area. This, again, is in line with the observed results of the other experiments.

![Graph](image)

Figure A.9: Test R² of predicted T in the heat equation experiment as a function of different number of collocation points. Results are averaged over 20 different seeds and the resulting error bars correspond to 95% confidence interval for the mean estimation, based on 1000 bootstrap samples. Different colors indicate different PDE weights w₂.