Multi-Objective Loss Balancing for Physics-Informed Deep Learning

Rafael Bischof
Swiss Data Science Center
Zürich, Switzerland
rafael.bischof@sdsc.ethz.ch

Michael A. Kraus
Chair of Concrete Structures and Bridge Design (IBK)/ Design++, ETH Zürich
Zürich, Switzerland
kraus@ibk.baug.ethz.ch

Abstract—**Physics-Informed Neural Networks (PINN) are algorithms from deep learning leveraging physical laws by including partial differential equations together with a respective set of boundary and initial conditions as penalty terms into their loss function. In this work, we observe the significant role of correctly weighting the combination of multiple competitive loss functions for training PINNs effectively. To this end, we implement and evaluate different methods aiming at balancing the contributions of multiple terms of the PINNs loss function and their gradients. After reviewing of three existing loss scaling approaches (Learning Rate Annealing, GradNorm and SoftAdapt), we propose a novel self-adaptive loss balancing scheme for PINNs named *ReLoBRaLo* (Relative Loss Balancing with Random Lookback). We extensively evaluate the performance of the aforementioned balancing schemes by solving both forward as well as inverse problems on three benchmark PDEs for PINNs: Burgers' equation, Kirchhoff's plate bending equation and Helmholtz's equation. The results show that ReLoBRaLo is able to consistently outperform the baseline of existing scaling methods in terms of accuracy, while also inducing significantly less computational overhead.**

I. Introduction

The emergence of Physics-Informed Neural Networks (PINNs) [48] has sparked a lot of interest in domains that see themselves regularly confronted with problems in the low data regime. By leveraging well-known physical laws and incorporating them as implicit prior into the deep-learning pipeline, PINNs were shown to require little to no data in order to approximate partial differential equations (PDE) of varying complexity [18, 48].

We consider the case where PINNs are used for finding the unknown, underlying function proper to solve a parameterised PDE. PDEs generally consist of a governing equation and a set of boundary as well as initial conditions.

When trained jointly, i.e. as multi-objective optimisation (MOO), these equations form a set of objective functions that guide the model to approximate a function that satisfies the PDE. While several established and well-studied numerical methods already exist for addressing this problem, such as the Finite Element Method (FEM), Finite Difference Method or Wavelets resp. Laplace transform methods [1, 17, 22, 57], PINNs offer significant advantages, such as being end-to-end differentiable, mesh-free and avoiding the curse of dimensionality [16, 46]. They could therefore prove useful in several engineering applications, such as inversion and surrogate modeling in solid mechanics [19], design optimisation [38] or structural health monitoring and system identification [69]. PINNs have also been successfully applied in computational fluid mechanics and dynamics for surrogate modelling of numerically expensive fluid flow simulations [68], identification of hidden quantities of interest (velocity, pressure) from spatio-temporal visualisations of a passive scaler (dye or smoke) [50] or in an inverse heat transfer application setting in flow past a cylinder without thermal boundaries [4].

However, further research is necessary to tackle current failure modes [39, 66], one of which is the issue of gradient pathologies arising from imbalanced loss terms during training [65]. With the various terms in the objective function stemming from physical laws, they are naturally bound to units of measurements that can vary significantly in magnitude. Consequently, the signal strengths of backpropagated gradients might differ from term to term and lead to pathologies that were shown to impede proper training and cause imbalanced solutions [65], hence posing challenges to global optimisation methods such as Adam, Stochastic Gradient Descent (SGD), or L-BFGS [28, 31, 62, 70]. As a counter-measure, every individual term i may be scaled by a factor λ_i in order to balance its contribution to the total gradient. However, manual tuning of these scaling factors requires laborious grid search and becomes intractable as the number of terms grows.

This work investigates different schemes aiming at adaptively balancing the contributions of multiple terms and

their gradients in the loss function by selecting the optimal scaling factors λ_i in order to improve approximation capabilities of PINNs. To this end, we compare the effectiveness of Learning Rate Annealing (LRAnnealing) [65], proposed in the context of PINNs, to two approaches originating from Computer Vision applications: GradNorm [8] and SoftAdapt [20]. In addition, we derive and present our own variation of an adaptive loss scaling technique, *ReLoBRaLo* (Relative Loss Balancing with Random Lookback), that we found to be more effective at similar efficiency compared to state-of-the-art by testing the algorithms on various benchmark problems for PINNs in the forward and inverse setting: Helmholtz, Burgers and Kirchhoff PDEs.

This paper is organised as follows: we first provide a short introduction to the problem as well as the state-of-the-art of PINNs in sec. III. Further methodical background on multi-objective optimisation (MOO), the framework of Physics-Informed Neural Networks (PINNs) and loss balancing for PINNs training is presented in sec. V. In sec. VI we introduce ReLoBRaLo as a novel self-adaptive loss balancing method. Sec. VIII reports numerical results of the developed approach against state-of-the-art methods for several examples in the forward and inverse setting: Burgers, Kirchhoff and Helmholtz PDEs. Sec. IX presents results of ablation studies as well as a discussion of findings and drawing further conclusions on ReLoBRaLo and its hyperparameter settings across all examples of this paper. Finally, a summary together with an outlook is given in sec. X. All code produced within this publication is freely available and open access here: https://github.com/rbischof/relative_balancing.

II. State-of-the-Art and Related Work

Using neural networks to approximate the solutions of Ordinary Differential Equations (ODEs) and PDEs has been the subject of several studies over the past decade. Initially, Lagaris et al. trained neural networks to solve ODEs and PDEs on a predefined set of grid points [34, 35], while Sirignano proposed a method for solving high-dimensional PDEs through approximation of the solution by a neural network and especially emphasising training efficiency by incorporating mini-batch sampling in high dimensional settings compared to the computationally intractable finite mesh-based schemes [56]. Raissi et al. introduced the term Physics-Informed Neural Networks (PINN) and provided empirical justification by numerical simulations for a variety of nonlinear PDEs, including the Navier–Stokes equation and the Burgers equation [47]. Shin et al. provided first theoretical justification for PINNs by demonstrating convergence of linear elliptic and parabolic PDEs in the L2 sense [54].

However, PINN training efficiency, convergence, and accuracy remain serious challenges [49, 68]. Current research may be ordered into four main approaches: modifying structure of the NN, divide-and-conquer/domain decomposition, parameter initialisation and loss balancing.

Jagtap et al. adapted the typical NN architecture to PINNs by introducing parameters that scale the input

to the activation functions and get updated alongside the network's parameters θ through gradient descent [25, 26]. The authors showed that the adaptive activation function significantly accelerated convergence and also improved solution accuracy. Kim et al. presented a fast and accurate PINN ROM with a nonlinear manifold solution representation, where the NN structure included an encoder and a decoder part [30]. Furthermore, a shallow masked encoder was trained using data from the full-order model simulations in order to use the trained decoder as representation of the nonlinear manifold solution. Peng et al. proposed dictionary-based PINNs to store and retrieve features and speed up convergence by merging prior information into the structure of NNs [45].

Other research focused on decomposing the computational domain in order to accelerate convergence. Jagtap et al. proposed conservative PINNs and extended PINNs that decompose the computational domain into several discrete sub-domains, each one solved independently using a separate, shallow PINN [23, 24]. Inspired by this work, Shukla et al. derived and investigated a distributed training framework for PINNs that used domain decomposition methods in both space and time-space [55]. To accelerate convergence, the distributed framework combined the benefits of conservative and extended PINNs. The time-space domain may become very large when solving PDEs with long time integration, causing the training cost of NNs to become extremely expensive. To that end, Meng et al. proposed a parareal PINN to address the long-standing issue [40]. The authors decomposed the long-time domain into many discrete short-time domains using a fast coarse-grained solver. Training multiple PINNs with many small data sets was much faster than training a single PINN with a large data set. For PDEs with long-time integration, the parareal PINN achieved a significant speedup. Kharazmi et al. introduced hp-variational PINNs to divide the computational space into the trial space and test space by combining domain decomposition and projection onto high-order polynomials [29]. A soft split of the problem domain incorporating variants of the Mixture-of-Experts approach were investigated by [2].

In most works, researchers resort to the Xavier initialisation [14] for selecting the PINN's initial weights and biases. The effects of using more refined initialisation procedures has recently been gaining attention, with Liu et al. showing that a good initialisation can provide PINNs with a head start, allowing them to achieve fast convergence and improved accuracy [37]. Transfer learning for PINNs was introduced by Chakraborty et al. [6] and Goswami et al. [15] to initialise PINNs for dealing with multi-fidelity problems and brittle fracture problems, respectively. After their success in other fields of Deep Learning, meta-learning algorithms have also been implemented in the context of PINNs [12, 13, 51, 58], with Model-Agnostic Meta-Learning (MAML) being amongst the most popular ones [11]. Its second-order objective is to find an initialisation that is sub-optimal in itself, but from where the network requires only few labeled training samples and optimisation steps

in order to specialise on a task and achieve high accuracy (few-shot learning). Subsequently, Nichol et al. proposed the REPTILE algorithm, which turns the second-order optimisation of MAML into a first-order approximation and therefore requires significantly less computation and memory while achieving similar performance [42]. Liu et al. applied the REPTILE algorithm to PINNs by regarding modifications of PDE parameters as separate tasks [37]. The resulting initialisation is such that the PINN converges in just a few optimisation steps for any choice of PDE parameters.

Using derivative information of the target function during training of a neural network was introduced by Czarnecki et al. under the term *Sobolev Training* [9]. Sobolev Training proved to be more efficient in many applicable fields due to lower sample complexity compared to regular training. Son et al. enhance the concept of Sobolev Training in the strict mathematical sense using Sobolev norms in loss functions of neural networks for solving PDEs [61]. It was found that these novel Sobolev loss functions lead to significantly faster convergence on investigated examples compared to traditional L2 loss functions. NNs were used in plain as well as a Sobolev Training manner for constitutive modelling, where it was shown that mechanical relations can be seen as Sobolev training to successfully encapsulate several aspects of the constitutive behavior, such as strain-stress-relationships arising from derivatives of a Helmholtz potential in hyperelasticity [32, 33, 63, 64].

Colby et al. observed that a weighted scalarisation of the multiple loss functions, defined by the sampled data and physical laws for PINNs training, plays a significant role for convergence [67]. Wang et al. recently published a Learning Rate Annealing algorithm that employs back-propagated gradient statistics in the training procedure in order to adaptively balance the terms' contributions to the final loss [65] and investigated the issue of vanishing and exploding gradients that currently limits the applicability of PINNs [66]. To that end, the authors introduced a Neural Tangent Kernel (NTK), which appropriately assigns weights to each loss term at subtle performance improvement, in order to comprehend the training process for PINNs. Shin et al. developed the Lipschitz regularised loss for solving linear second-order elliptic and parabolic type PDEs [54]. McClenny et al. proposed a method for updating the adaptation weights in the loss function in relation to network parameters [39].

III. PHYSICS-INFORMED NEURAL NETWORKS (PINNs)

This section reviews basic Physics-Informed Neural Networks (PINNs) concepts and recent developments.

Consider the following abstract parameterised and non-

linear PDE problem:

$$
\begin{aligned}
\text{PDE} : \mathcal{F}\left(\hat{\mathbf{u}}, \frac{\partial \hat{\mathbf{u}}}{\partial t}, \frac{\partial \hat{\mathbf{u}}}{\partial \mathbf{x}}, \cdots ; \mu\right) &= 0, \quad \mathbf{x} \in \Omega, \, t \in \Upsilon \\
\text{B.C.} : \mathcal{B}\left(\hat{\mathbf{u}}, \frac{\partial \hat{\mathbf{u}}}{\partial \mathbf{x}}, \frac{\partial^2 \hat{\mathbf{u}}}{\partial \mathbf{x}^2}, \cdots\right) &= 0, \quad \mathbf{x} \in \Gamma \\
\text{I.C.} : \mathcal{C}\left(\hat{\mathbf{u}}, \frac{\partial \hat{\mathbf{u}}}{\partial t}, \frac{\partial^2 \hat{\mathbf{u}}}{\partial t^2}, \cdots\right) &= 0, \quad t \in \Upsilon
\end{aligned}
\tag{1}
$$

where $\mathbf{x} \in \mathbb{R}^d$ is the spatial coordinate and t is the time; $\mathcal{F}$ denotes the residual of the PDE, containing the differential operators (i.e. $\partial_{\mathbf{x}}\hat{\mathbf{u}}, \partial_t\hat{\mathbf{u}}, ...$); $\mu = [\mu_1, \mu_2, ...]$ are the PDE parameters; $\hat{\mathbf{u}}(\mathbf{x}, t)$ is the solution of the PDE with initial condition $\mathcal{C}$ and boundary condition $\mathcal{B}$ (which can be Dirichlet, Neumann or mixed); Ω, Γ and Υ represent the spatial domain resp. boundary. A special example considered in this paper is the Burgers equation (given in Eq. 12): $\partial_t\hat{\mathbf{u}} + \hat{\mathbf{u}}\partial_{\mathbf{x}}\hat{\mathbf{u}} - \nu\partial_{\mathbf{x}}^2\hat{\mathbf{u}} = 0$ with PDE parameter μ as viscosity coefficient ν. This paper is concerned with solving forward as well as inverse problems from different fields of application. For the forward problem, solutions of PDEs are to be inferred with fixed parameters μ, while for the inverse problem setting, μ is unknown and has to be learned from observed data together with the PDE solution.

Following the "vanilla" implementation of PINNs [48], a fully-connected feed-forward neural network (FCNN) $U(\mathbf{x}, t; \theta)$ is used to approximate the function $\hat{\mathbf{u}}(\mathbf{x}, t)$ which solves the PDE. A FCNN consists of multiple hidden layers with trainable parameters (weights and biases; denoted by θ) and takes as inputs the space and time coordinates $(\mathbf{x}, t)$, cf. fig. 1. The losses are then defined as follows:

$$
\begin{aligned}
\mathcal{L}_{\Omega} &= \frac{1}{|\hat{\Omega}|} \sum_{\mathbf{x}, t \in \hat{\Omega}} \left\| \mathcal{F}\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial t}, \frac{\partial \mathbf{u}}{\partial \mathbf{x}}, \frac{\partial^2 \mathbf{u}}{\partial \mathbf{x}^2}, \cdots, \mu\right) \right\|_2^2 \\
\mathcal{L}_{\Gamma_i} &= \frac{1}{|\hat{\Gamma}_i|} \sum_{\mathbf{x}, t \in \hat{\Gamma}_i} \left\| \mathcal{B}_i\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial \mathbf{x}}, \frac{\partial^2 \mathbf{u}}{\partial \mathbf{x}^2}, \cdots\right) \right\|_2^2 \\
\mathcal{L}_{\Upsilon_i} &= \frac{1}{|\hat{\Upsilon}_i|} \sum_{\mathbf{x}, t \in \hat{\Upsilon}_i} \left\| \mathcal{C}_i\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial t}, \frac{\partial^2 \mathbf{u}}{\partial t^2}, \cdots\right) \right\|_2^2 \\
\mathcal{L}_{\Psi} &= \frac{1}{|\hat{\Psi}|} \sum_{\mathbf{x}, t \in \hat{\Psi}} \| \mathbf{u} - d(\mathbf{x}, t) \|_2^2
\end{aligned}
\tag{2}
$$

where $\hat{\Omega}$ is a set of collocation points on the physical domain, $\hat{\Gamma}_i$ for the boundary conditions (BC), $\hat{\Upsilon}_i$ for the initial conditions (IC) and $\hat{\Psi}$ represents a set of measurements (data); the function d maps $(\mathbf{x}, t)$ to measurements at those coordinates; $\mathbf{u}$ is the output from the neural network $U(\mathbf{x}, t; \theta)$. PINNs are generally trained using the L2-norm (mean squared error / MSE) on uniformly sampled collocation points defined as a data set $\{\mathbf{x}_i, t_i\}_{i=1}^N$ prior to training. Note that the number of points N (denoted by $|\cdot|$ in Eq. 2) may vary for different loss terms.

The objectives in Eq. 2 are trained jointly and hence fall into the class of multi-objective optimisation (MOO) (cf. Eq. 4)
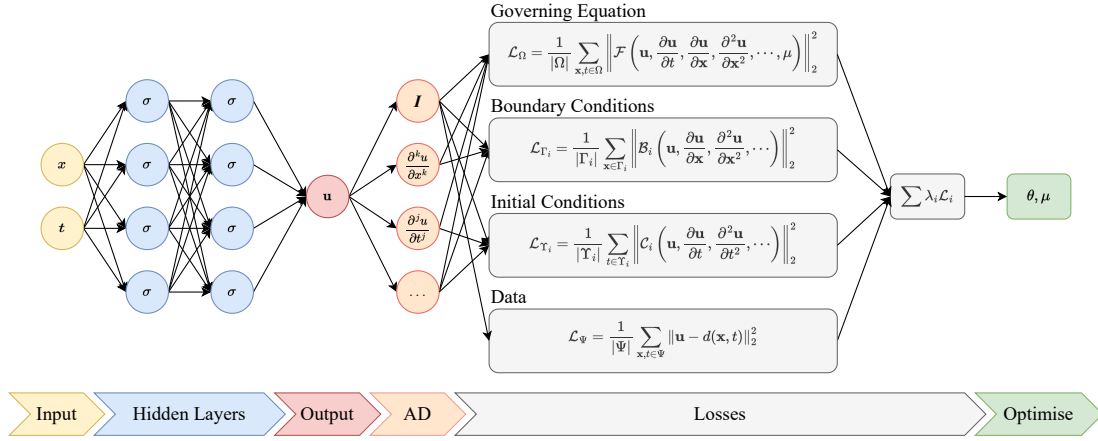
Figure 1: Schematic of a Physics-Informed Neural Network (PINN): A fully-connected feed-forward neural network with space and time coordinates $(\mathbf{x}, t)$ as inputs, approximating a solution $\hat{\mathbf{u}}(\mathbf{x}, t)$. Derivatives of $\mathbf{u}$ w.r.t. inputs are computed by automatic differentiation (AD) and then incorporated into residuals of the governing equations as the loss function, which is composed of multiple terms weighted by different coefficients. Parameters of the FCNN θ and the unknown PDE parameters μ may be optimised simultaneously by minimising the loss function.

$$\mathcal{L}(\theta, \mu) = (\mathcal{L}_\Omega, \mathcal{L}_{\Gamma_1}, \ldots, \mathcal{L}_{\Gamma_n}, \mathcal{L}_{\Upsilon_1}, \ldots, \mathcal{L}_{\Upsilon_m}, \mathcal{L}_\Psi)^T \quad (3)$$

where the individual terms can be interpreted in the following way:

- the first term $\mathcal{L}_\Omega$ penalises the residual of the governing equations (PDEs), included in both the forward and inverse problem.
- the following n terms $\mathcal{L}_{\Gamma_i}$ enforce the boundary conditions (BCs), included only in the forward problem.
- the following m terms $\mathcal{L}_{\Upsilon_i}$ enforce the initial conditions (ICs), included only in the forward problem.
- the last term $\mathcal{L}_\Psi$ makes the network approximate the measurements, included both in the forward (albeit not strictly necessary) and inverse problem.

A common approach for handling MOO is through linear scalarisation described in more detail in the following Section IV.

IV. MULTI-OBJECTIVE OPTIMISATION

Multi-objective optimisation (MOO) is concerned with simultaneously optimising a set of $k > 1$, potentially conflicting objectives [5, 27].

$$\mathcal{L}(\theta) = (\mathcal{L}_1(\theta), \ldots, \mathcal{L}_k(\theta))^T \quad (4)$$

which can be turned into a single objective through linear scalarisation:

$$\mathcal{L}(\theta) = \sum_{i=1}^{k} \lambda_i \mathcal{L}_i(\theta), \quad \lambda_i \in \mathbb{R}_{>0} \quad (5)$$

Many problems in engineering, natural sciences, or economics can be formulated as multi-objective optimisations and generally require trade-offs to simultaneously satisfy all objectives to a certain degree [7]. The solution of MOO models is usually expressed as a set of Pareto optima,

representing these optimal trade-offs between given criteria according to the following definitions [53]:

Definition IV.1. A solution $\hat{\theta} \in \Omega$ Pareto dominates solution θ (denoted $\hat{\theta} \prec \theta$) if and only if $\mathcal{L}_i(\hat{\theta}) \leq \mathcal{L}_i(\theta), \forall i \in \{1, \ldots, m\}$ and $\exists j \in \{1, \ldots, m\}$ such that $\mathcal{L}_j(\hat{\theta}) < \mathcal{L}_j(\theta)$.

Definition IV.2. A solution $\hat{\theta} \in \Omega$ is said to be Pareto optimal if $\forall \theta \in \Omega, \hat{\theta} \preceq \theta$. The set of all Pareto optimal points is called the Pareto set and the image of the Pareto set in the loss space is called the Pareto front.

In theory, a Pareto optimal solution θ is independent of the scalarisation [52]. However, when using neural networks for MOO, the solution space becomes highly non-convex. Thus, although neural networks are universal function approximators [21], they are not guaranteed to find the globally optimal solution through gradient-based optimisation. Scaling the loss space therefore provides the option of guiding the gradients into having an a priori deemed desirable property. However, manually finding optimal λ_i requires laborious grid search and becomes intractable as k gets large. Furthermore, one might want to let λ_i evolve over time. This raises the need for an automated scheme to dynamically choose the scalings λ_i.

V. ADAPTIVE LOSS BALANCING SCHEMES

This section reviews different methods aiming at balancing the various terms within multi-objective optimisation. To this end, we compare the effectiveness of Learning Rate Annealing [65], proposed in the context of PINNs as well as two approaches originating from Computer Vision applications: GradNorm [8] and SoftAdapt [20]. This forms the basis for deriving and presenting our own loss balancing method as given in sec. VI.

A. Learning Rate Annealing

Wang et al. [65] conducted a study on gradients in PINNs and identified pathologies that explained some failure modes. One pathology is gradient stiffness in the boundary conditions caused by the imbalance amongst the different loss terms. As a remedy, it is proposed to adaptively scale the loss using gradient statistics, thus reducing the laborious tuning of these hyperparameters.

$$\hat{\lambda}_i(t) = \frac{\max\{|\nabla_\theta \mathcal{L}_\Omega(t)|\}}{|\nabla_\theta \mathcal{L}_{\{\Gamma,\Upsilon\}_i}(t)|}, \ i \in \{1,\dots,k\}$$

$$\lambda_i(t) = \alpha\lambda_i(t-1) + (1-\alpha)\hat{\lambda}_i(t) \qquad (6)$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta\nabla_\theta\left(\mathcal{L}_\Omega(t) + \sum_{i=1}^{k} \lambda_i(t)\mathcal{L}_{\{\Gamma,\Upsilon\}_i}(t)\right)$$

where $\overline{|\nabla_\theta \mathcal{L}_{\Gamma_i}(t)|}$ is the mean of the gradient w.r.t. the parameters θ; α is a hyperparameter with a value $\alpha = 0.9$ recommended by the authors.

With this method, whenever the maximum value of $|\nabla_\theta \mathcal{L}_\Omega(t)|$ grows considerably larger than the average value in $|\nabla_\theta \mathcal{L}_{\{\Gamma,\Upsilon\}}(t)|$, the scalings $\hat{\lambda}_i(t)$ correct for this discrepancy such that all gradients have similar magnitudes. Additionally, exponential decay is used in order to smoothen the balancing and avoid drastic changes of the loss space between optimisation steps.

This procedure induces a few drawbacks. Its unboundedness potentially involves up- or down-scaling of terms by means of several orders of magnitude. The up-scaling in particular can cause problems similar to the effect of choosing a learning rate that is too large and therefore leads to repeatedly overshooting the objective. Furthermore, scaling all terms to have the same magnitude throughout training can incite the network to optimise for the "low-hanging fruit". A term, whose loss decreased considerably in the last optimisation step, will see its contribution to the total gradient scaled back up to the same magnitude to match the other terms. Therefore, the network might focus on the objectives that are easiest to optimise for.

B. GradNorm

Chen et al. [8] take a different approach and make the scalings λ_i trainable. The updates on these trainable scalings are chosen such that all terms improve at the same relative rate w.r.t. their initial loss and performed by a separate optimiser. A term that improved at a higher rate since the beginning of training compared to the other terms, gets a weaker scaling until all terms have made the same relative progress. Therefore, one could argue that they weakly enforce each optimisation step to Pareto dominate (cf. definition IV.1) its predecessor. The loss for updating the scalings within GradNorm is computed as follows:

$$\mathcal{L}(t;\lambda) = \sum_{i=1}^{k} \left| G_\theta^{(i)}(t) - \overline{G}_\theta(t) \times [r_i(t)]^\alpha \right|_1 \qquad (7)$$

where $G_\theta^{(i)}(t) = \|\nabla_\theta \lambda_i \mathcal{L}_i(t)\|_2$ is the L_2 norm of the gradient w.r.t. the network parameters θ for the scaled loss of objective $i \in \{1,\dots,k\}$; $\overline{G}_\theta(t) = \frac{1}{k}\sum_{i=1}^{k} G_\theta^{(i)}(t)$ is the average of all gradient norms; $r_i(t) = \mathcal{L}_i(t)/(\mathcal{L}_i(0)\cdot\overline{\mathcal{L}}(t))$ defines the rate at which term i improved so far; α is a hyperparameter representing the strength of the restoring force which pulls tasks back to a common training rate. Note that $\overline{G}_\theta(t) \times [r_i(t)]^\alpha$ is the desirable value that $G_\theta^{(i)}(t)$ should take on, so gradients must be prevented from flowing through this expression. The final loss for updating the networks parameters is then simply a linear scalarisation with the scalings that were previously updated:

$$\mathcal{L}(t;\theta) = \sum_{i=1}^{k} \lambda_i(t)\mathcal{L}_i(t) \qquad (8)$$

This algorithm is fairly evolved and, despite solving some of Learning Rate Annealing's issues, it still requires a separate backward-pass for each task, which becomes prohibitively expensive as k gets large. Furthermore, it relies on two separate optimisation rounds at each step: one for adapting the scalings λ_i and another for updating the weights θ. By means of Eq. 4, GradNorm can thus be formulated as a scalarised MOO via:

$$\mathcal{L}(t) = (\mathcal{L}(t;\theta), \mathcal{L}(t;\lambda))^T \qquad (9)$$

which in turn requires empirical hyperparameter tuning (learning rate, initialisation, etc.) to keep the system balanced - exactly the problem we are actually trying to solve through the use of adaptive loss balancing schemes.

C. SoftAdapt

Similar to GradNorm, SoftAdapt [20] leverages the ansatz of relative progress in order to balance the loss terms. However, the authors relax it by only considering the previous time-step $\mathcal{L}_i(t-1)$ and taking the difference between time steps instead of the division. The scalings are then normalised by using a softmax function:

$$\lambda_i(t) = \frac{\exp\left(\mathcal{T}(\mathcal{L}_i(t) - \mathcal{L}_i(t-1))\right)}{\sum_{j=1}^{k} \exp\left(\mathcal{T}(\mathcal{L}_j(t) - \mathcal{L}_j(t-1))\right)}, \ i \in \{1,\dots,k\}$$

$$(10)$$

where $\mathcal{L}_i^{(t)}$ is the loss of term i at optimisation step t.

SoftAdapt also differs from GradNorm in the sense that it does not require gradient statistics and thus eliminates the need of performing separate backward passes for each objective. Instead, it makes use of the fact that magnitudes in the gradients directly depend on the magnitudes of the terms in the loss function and therefore aims at achieving the balance solely through loss statistics. This is obviously true only if the same loss function is used for every objective (e.g. the L_2 loss). However, this setting generalises to a vast majority of applications involving PINNs.

VI. Relative Loss Balancing with Random Lookback (ReLoBRaLo)

Drawing inspiration from existing balancing techniques as outlined in sec. V, we propose a novel method and

implementation for balancing the multiple terms in the scalarised MOO loss function for training of PINNs upon:

- SoftAdapt's concept of operating on loss statistics as opposed to gradient statistics is employed. A computationally inexpensive softmax ensures the sum of scalings is bounded.
- Inspired by GradNorm, the progress is calculated by dividing the loss at the current iteration $\mathcal{L}_i(t)$ by the loss at the previous iteration $\mathcal{L}_i(t-1)$.
- Similarly to Learning Rate Annealing, the scalings are updated using an exponential decay in order to utilise loss statistics from more than just one training step in the past.
- In addition, a random lookback (called *saudade* ρ) is introduced into the exponential decay, which decides whether to use the previous steps' loss statistics to compute the scalings, or whether to look all the way back until the start of training $\mathcal{L}_i^{(0)}$.

$$
\begin{aligned}
\lambda_i^{bal}(t,t') &= m \cdot \frac{\exp\left(\frac{\mathcal{L}_i(t)}{\mathcal{T}\mathcal{L}_i(t')}\right)}{\sum_{j=1}^{m} \exp\left(\frac{\mathcal{L}_j(t)}{\mathcal{T}\mathcal{L}_j(t')}\right)}, \; i \in \{1,\dots,m\} \\
\lambda_i^{hist}(t) &= \rho\lambda_i(t-1) + (1-\rho)\lambda_i^{bal}(t,0)) \\
\lambda_i(t) &= \alpha\lambda_i^{hist} + (1-\alpha)\lambda_i^{bal}(t,t-1)
\end{aligned}
$$
$$(11)$$

where α is the exponential decay rate, ρ is a Bernoulli random variable and $\mathbb{E}[\rho]$ should be chosen close to 1. The intermediate step $\lambda_i^{bal}(t,t')$ calculates scalings based on the relative improvements of each term between time steps t' and t. The following step $\lambda_i^{hist}(t)$ defines, whether the scalings calculated in the previous time step (ρ evaluates to 1) or the relative improvements since the beginning of training (ρ evaluates to 0) should be carried forward. Note that this concept of randomly retaining or discarding the history of scalings is what we denote as "random lookbacks". Finally, the scaling $\lambda_i(t)$ for term i is obtained by means of an exponential decay, where α controls the weight given to past scalings versus the scalings calculated in the current time step.

This method is an attempt at combining the best attributes of the aforementioned approaches into a new scheme for scalarised MOO objective functions. First and foremost, it still weakly enforces every training step to Pareto dominate its predecessor, which is an important property in physical applications. It also avoids using gradient statistics, making it considerably more efficient than Learning Rate Annealing and GradNorm. Furthermore, it reduces drastic changes in the loss space by using exponential decay and can easily be adapted to use more or fewer information of past optimisation steps by tuning the hyperparameter α. One can think of α as the model's ability to remember the past, with a high alpha giving lots of weight to past loss statistics, while a lower alpha increases stochasticity. Setting $\alpha = 1$ results in each term's relative progress being computed w.r.t. the initial loss $\mathcal{L}_i^{(0)}$. However, we found this to be too restrictive, since

it causes the model to stop making progress as soon as one term reaches a local minimum. We chose values α between 0.9 and 0.999 and report the effects of varying this hyperparameter in sec. IX.

Choosing the value of α also requires to make a trade-off: a high value means the model will remember potential deterioration of certain terms for longer and therefore leave a longer time frame in order to compensate them. However, it also induces a latency between a term starting to deteriorate and the scalings λ_i reacting accordingly. We therefore study the effect of introducing the saudade Bernoulli random variable ρ that causes the model to occasionally look back until the start of training. $\mathbb{E}[\rho] = 0$ is maximum saudade as it always takes the loss value of the initial training step, while $\mathbb{E}[\rho] = 1$ corresponds to minimum saudade, taking only into account the last value from the history of the i-th scaling factor. Selecting $\mathbb{E}[\rho]$ somewhere between 0 and 1 allows to set a lower value for α, thus making the model more flexible while still occasionally "reminding" it of the progress made since the start of training. Furthermore, the random lookback can give episodic new impulses and let the model escape local minima by changing the loss space, as well as inciting it to explore more of the parameter space. In case the impulse would turn out to have a negative effect on the accuracy, one can still choose to roll back and reset the network's parameters θ to the previous state.

The last hyperparameter is the so-called temperature $\mathcal{T}$. Setting $\mathcal{T} \to \infty$ re-calibrates the softmax to output uniform values and thus all $\hat{\lambda}_i^{(t)} = 1$. On the other hand, $\mathcal{T} \to 0$ essentially turns the softmax into an argmax function, with the scaling $\hat{\lambda}_i^{(t)} = k$ resulting for the term with the lowest relative progress and $\hat{\lambda}_i^{(t)} = 0$ for all others. A pedagogical example with interpretation of expected behaviours and how to draw conclusions from the histories of the scalings is given for Burgers' equation in sec. VIII-A.

Note that the network should be prevented from optimising this expression. This can be achieved by stopping the gradients from flowing through the calculation of the scalings. Also, depending on the problem at hand, $\exp(\mathcal{L}_i(t)/(\mathcal{T}\mathcal{L}_i(t')))$ could evaluate to a very large number, thus leading to overflows. This issue can be preemptively tackled by subtracting a large number (e.g. 10^{-9}) from the input to the softmax.

VII. Hyperparameter Tuning and Meta Learning

This paper uses grid search in combination with Bayesian Optimisation (BO) [36, 43, 59] for hyperparameter tuning. This study uses hyperparameters for defining the NN architecture (d_K hidden layers and w_K neurons per layer) and training settings (learning rate l_r, exponential decay rate α and saudade ρ). Tab. I contains the ranges and distributions for the hyperparameters. Bayesian Optimisation reduces the empiricism of selecting the PINNs hyperparameters to learn an optimal NN structure. First, 20 random points in the hyperparameter space are sampled and evaluated. The model's performance at those points serves as evidence for

fitting prior Gaussian Processes in order to estimate the unknown loss function w.r.t. the hyperparameters. Using Expected Improvement (EI) [41], further 80 points are then sampled and evaluated to refine the prior. This procedure provides an educated guess as to which are the optimal hyperparameters for the task at hand. Finally, we fine-tune the results by performing fine-grained grid search around the hyperparameters returned by Bayesian Optimisation (BO).

Hyperparameter	Range	Log-scaling
Learning Rate l_r	$[10^{-6}, 10^{-2}]$	yes
Layers d_K	$[2, 4]$	no
Neurons per Layer w_K	$[32, 512]$	no
Exponential Decay Rate α	$[0, 1]$	no
Temperature $\mathcal{T} = 10^t$	$[10^{-6}, 10^2]$	yes
Expected Saudade $\mathbb{E}[\rho]$	$[0, 1]$	no
Activation function σ	$\{tanh,\ sigmoid\}$	no

Table I: Hyperparameters for architecture and training settings together with ranges as used for Bayesian Optimisation.

Within this study, the exact same Bayesian Optimisation configuration was used for all examples presented in sec. VIII, hence it is sufficient to only display tab. I. Respective results of the Grid Search and Bayesian Optimisation can also be found in sec. VIII.

VIII. Results

We evaluate the different balancing schemes on three problems (Burgers equation, Kirchhoff plate bending and Helmholtz equation) originating from physics-informed deep learning, where the objective function consists of various terms of potentially considerably different magnitudes and compare their performances, as well as their computational efficiency. Training was done on networks of varying depth and width (acc. to tab. I) and limited to 10^5 steps of gradient descent (GD) using the Adam optimiser [31]. Additionally, we reduced the learning rate by a multiplicative factor of 0.1 whenever the optimisation stopped making progress for over 3'000 optimisation steps and finally used early stopping in case of 9'000 steps without improvement. When addressing the inverse problem, i.e. approximating a set of measurements while subjecting the network to PDE constraints for finding unknown PDE parameters μ, we further investigated the payoff of using two separate optimisers: one for updating network weights θ, and a separate one for updating PDE parameters μ. Further details on hyperparameter tuning and meta learning is given in sec. IX.

A. Burgers' Equation

Burgers' equation is a one-dimensional PDE describing the main properties of the Navier-Stokes equations [44] used i.a. to model shock waves, gas dynamics, or traffic

flow [3]. Using Dirichlet boundary conditions, the PDE takes the following form:

$$
\begin{aligned}
&\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, \quad x \in [-1,1], \quad t \in [0,1] \\
&u(0, x) = -sin(\pi x) \\
&u(t, -1) = u(t, 1) = 0
\end{aligned}
\tag{12}
$$

At first, we investigate the solution of the forward problem, where we set the PDE parameter $\mu := \nu = \frac{1}{100\pi}$. In order to find the latent function $\hat{\mathbf{u}}(\mathbf{x}, t)$, we can parameterise it with a neural network $U(x, t; \theta)$ and turn the set of equations into a linear scalarised objective (cf. Eq. 5) of Mean Squared Errors (MSE). This loss function will weakly enforce the network to approximate the PDE solution $\hat{\mathbf{u}}(\mathbf{x}, t)$.

$$
\begin{aligned}
\mathcal{L}_\Omega &= \frac{1}{|\hat{\Omega}|}\sum_{(x,t)\in\hat{\Omega}}\left\|\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} - \nu\frac{\partial^2 U}{\partial x^2}\right\|_2^2 \\
\mathcal{L}_{\Gamma_1} &= \frac{1}{|\hat{\Gamma}_1|}\sum_{t\in\hat{\Gamma}_1}\|U(-1, t; \theta)\|_2^2 \\
\mathcal{L}_{\Gamma_2} &= \frac{1}{|\hat{\Gamma}_2|}\sum_{t\in\hat{\Gamma}_2}\|U(1, t; \theta)\|_2^2 \\
\mathcal{L}_\Upsilon &= \frac{1}{|\hat{\Upsilon}|}\sum_{x\in\hat{\Upsilon}}\|U(x, 0; \theta) + \sin(\pi x)\|_2^2
\end{aligned}
\tag{13}
$$

For the forward problem, PINNs training induces the following loss function employed during training:

$$
\mathcal{L} = \lambda_0\mathcal{L}_\Omega + \lambda_1\mathcal{L}_{\Gamma_1} + \lambda_2\mathcal{L}_{\Gamma_2} + \lambda_3\mathcal{L}_\Upsilon
\tag{14}
$$

After a successful convergence of the PINNs training using ReLoBRaLo, we obtain the results displayed in fig. 2(b), whereas the final algorithm settings are reported in tab. VIII. As there is no analytical solution available for the Burgers equation, we compared the results to a reference solution calculated using the finite element method (FEM), displayed in fig. 2(a). A plot of the squared difference in u as given by the FEM and PINNs is shown in fig. 2(c) and delivers a relative max error of below 5%.

However, Burgers' equation can also be turned into an *inverse problem* by regarding the PDE parameter ν as an unknown to be estimated from a set of observations (i.e. data) over the spatial and temporal domain. In this setting, the PINNs induced loss function to be deployed reads:

$$
\begin{aligned}
\mathcal{L} = \lambda_0\frac{1}{|\hat{\Omega}|}&\sum_{(x,t)\in\hat{\Omega}}\left\|\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} - \nu\frac{\partial^2 U}{\partial x^2}\right\|_2^2 \\
&+ \lambda_1\frac{1}{|\hat{\Psi}|}\sum_{(x,t)\in\hat{\Psi}}\|U - u\|_2^2
\end{aligned}
\tag{15}
$$

Similar to the network's weights and biases, the additional trainable PDE variable μ (here viscosity ν) is now also updated through gradient descent:

$$
\begin{aligned}
\theta^{(t)} &= \theta^{(t-1)} - \eta\nabla_\theta\mathcal{L} \\
\mu^{(t)} &= \mu^{(t-1)} - \eta\nabla_\mu\mathcal{L}
\end{aligned}
\tag{16}
$$

Measurement data Ψ for the inverse problem setting were obtained from our reference solution computed using the FEM without addition of noise. At every iteration, we sample from the available data in order to generate a batch of collocation points.
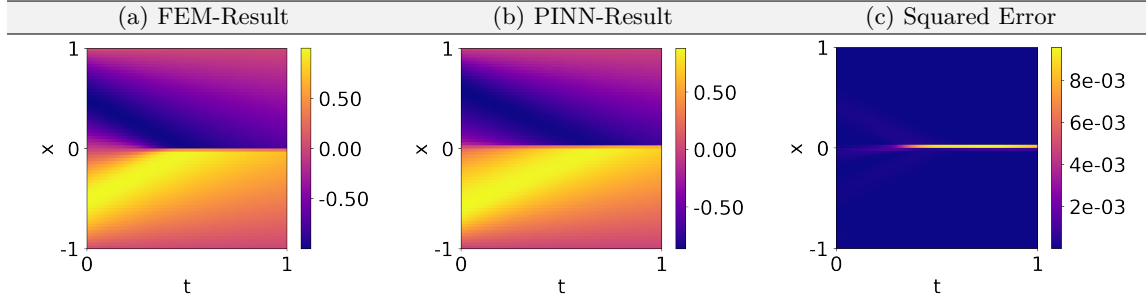
Figure 2: Burgers' equation problem: (a) FEM reference solution, (b) PINNs results predicted with a fully-connected network consisting of two layers and 128 nodes each, and (c) squared error.
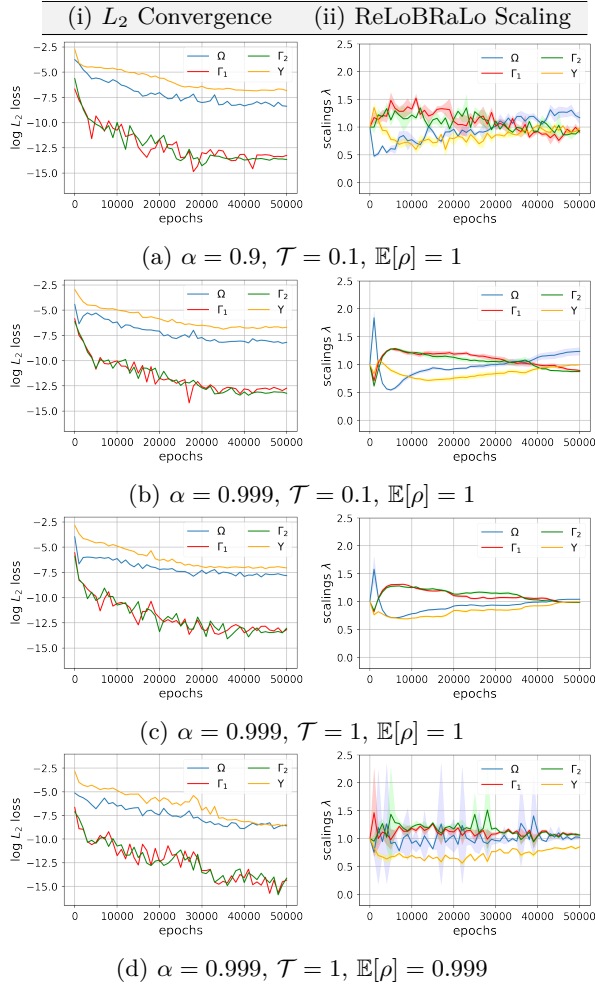

Figure 3: Median of the log L_2 loss over multiple training runs (i/left) of Burgers' equation and the mean and variance of the corresponding scaling factors λ (ii/right) computed with ReLoBRaLo.

Fig. 3 shows the scaling factors λ_i of our ReLoBRaLo method with varying hyperparameters for Burgers' equation. As can be expected, a larger value for α leads to a smoother curve because past loss statistics are dragged on longer, therefore countering the stochasticity that might arise at every optimisation step. On the other hand, the temperature $\mathcal{T}$ influences the magnitude of the scalings. Another general tendency in the plots of the scaling factors is the fact that the relatively lower loss contributions (here: BC 1 and BC 2) correspond to higher scaling values (potentially greater than 1), while the larger loss contributions (here: PDE and IC) correspond to lower scaling values (potentially less than 1). Note that, whenever "log" is used in this and all subsequent figures, we refer to the natural logarithm of that quantity.

The relatively small variances across training runs with $\mathbb{E}[\rho] = 0$ suggest that the optimisation progress follows similar patterns, even when varying depth and width of the network. Therefore, these values can provide valuable insight into the training and help to identify possibilities of improving the model. E.g. the fact that the scaling for the governing equation has the largest value after 50,000 epochs indicates that it was the first term to stop making progress. We will see in the following sections that the opposite holds true for Helmholtz' and Kirchhoff's equations, where the boundary conditions have more difficulties making progress (cf. fig. 6). This knowledge can help taking informed decisions to improve the framework, e.g. by adapting the activation functions, the loss function or the model's architecture accordingly.

Tab. II summarises the performances of the different balancing techniques against a baseline for the forward and inverse problem setting, where we manually chose the optimal scalings λ_i through grid search. As can be observed, the adaptive scaling techniques perform similarly well to the baseline, with Learning Rate Annealing and ReLoBRaLo reaching a considerably lower validation error. The results show that either one of these methods greatly reduces the amount of work required for hyperparameter search, while still achieving great results with high probability.

Besides accuracy, computational efficiency is another important metric for evaluating adaptive loss balancing methods. By designing our ReLoBRaLo method such that

Burgers		Baseline	GradNorm	LR anneal.	SoftAdapt	ReLoBRaLo
Forward	train f	$5.5 \cdot 10^{-4}$	$6.6 \cdot 10^{-4}$	$9.9 \cdot 10^{-4}$	$2.0 \cdot 10^{-4}$	$5.6 \cdot 10^{-5}$
	val u	$1.2 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$1.6 \cdot 10^{-4}$	$8.1 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$
	std val u	$5.7 \cdot 10^{-4}$	$2.1 \cdot 10^{-3}$	$2.3 \cdot 10^{-4}$	$9.5 \cdot 10^{-3}$	$6.8 \cdot 10^{-4}$
Inverse	val μ	$1.9 \cdot 10^{-10}$	$6.8 \cdot 10^{-5}$	$2.5 \cdot 10^{-11}$	$1.1 \cdot 10^{-7}$	$2.2 \cdot 10^{-10}$
$\nu = \frac{1}{100\pi}$	std μ	$1.2 \cdot 10^{-9}$	$5.1 \cdot 10^{-5}$	$3.4 \cdot 10^{-11}$	$5.7 \cdot 10^{-7}$	$2.1 \cdot 10^{-10}$

Table II: Comparison of the median L_2 training and validation loss on Burgers' equation against a baseline of manually chosen scalings. The reported values are the median over four independent runs with identical settings. Additionally, we report the standard deviation over the runs of the best performing model on the validation loss.
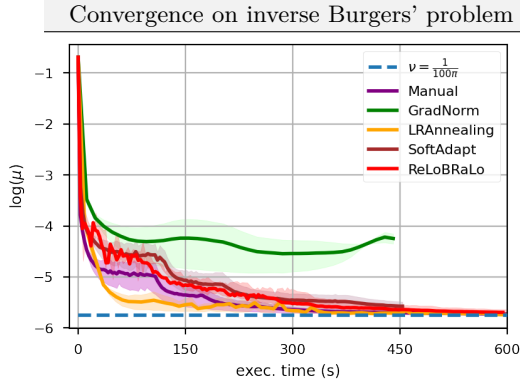

Figure 4: Approximation of the true PDE parameter value ν (dashed line) for the inverse problem setting of Burgers' equation. Reported values are the mean (solid line) and standard deviation (shaded area) of four independent computational runs.

it requires only one backward pass, its computational overhead can be expected to be relatively small compared to GradNorm and Learning Rate Annealing, which both utilise gradient statistics and hence separate backward passes for each term. Indeed, tab. III shows that Burgers' equation with its four terms in the loss function can be solved by ReLoBRaLo about 40% faster than Learning Rate Annealing and 70% faster than GradNorm and thus adds to efficiency and sustainability of PINNs training. Note that the reported values in tab. III stem from tasks where the balancing operation was performed at every optimisation step. Both GradNorm and Learning Rate Annealing can be made more efficient by updating the scaling terms once every arbitrary number of iterations. However, this introduces a trade-off between flexibility and efficiency and therefore an additional, very sensitive hyperparameter with a high impact on the method's accuracy and efficiency. On the other hand, ReLoBRaLo adapts its scalings at every iteration and very low computational cost.

	GradNorm	LR ann.	SoftAdapt	ReLoBRaLo
ΔT_{co} [s]	+10.6	+3.5	+0.4	+0.6

Table III: Median computational overhead ΔT_{co} (in s) per 1'000 optimisation steps compared to using no balancing scheme (3.7s) on Burgers' equation.

It is noteworthy that, while the forward problem induced

a loss function consisting of four terms, the inverse problem requires only two terms (Eq. 15). Hence, selecting the scalings manually is significantly less time-consuming than it is for the forward problem. Consequently, tab. II shows that the baseline was harder to outperform, with Learning Rate Annealing (LR Annealing) and ReLoBRaLo being the only methods yielding better results. It is worth noting however that Learning Rate Annealing approximates the true value of ν significantly faster than ReLoBRaLo and is therefore the optimal choice for this particular problem setting, cf. fig. 4. Further conclusions and comparisons across different loss balancing methods are made in sec. IX.

B. Kirchhoff Plate Bending Equation

The Kirchhoff–Love theory of plates arose from civil and mechanical engineering and consists of a two-dimensional mathematical model used to determine stresses and deformations in thin plates subjected to forces and moments [1]. The Kirchhoff plate bending problem assumes that a mid-surface plane can be used to represent a three-dimensional plate in two-dimensional form and together with a linear elastic material a fourth-order PDE can be derived to describe its mechanical behaviour:

$$\nabla^4 u(x,y) - \frac{p(x,y)}{D} = 0, \quad (x,y) \in \mathbb{R}^2_{>0}$$
$$D = \frac{Eh^3}{12(1-\nu^2)} \tag{17}$$

where $p(x,y)$ is the load acting on the plate at coordinates (x,y); D is the plate's flexural stiffness computed with Young's modulus E, the plate's thickness h and Poisson's ratio ν. The Kirchhoff plate bending problem poses several severe problems to FEM solutions [1], yet analytical solutions can be inferred e.g. using Fourier series for special cases such as an applied sinusoidal load:

$$p(x,y) = p_0 \sin\left(\frac{x\pi}{a}\right) \sin\left(\frac{y\pi}{b}\right)$$
$$u(x,y) = \frac{p_0}{\pi^4 D(\frac{1}{a^2} + \frac{1}{b^2})^2} \sin\left(\frac{x\pi}{a}\right) \sin\left(\frac{y\pi}{b}\right) \tag{18}$$

In this paper we consider a concrete plate of width $a = 10\,\text{m}$, length $b = 10\,\text{m}$, base load $p_0 = 0.015\,\text{MN}\,\text{m}^{-2}$, Young's modulus $E = 30.000\,\text{MN}\,\text{m}^{-2}$, plate height $h = 0.2\,\text{m}$ and Poisson's ratio of $\nu = 0.2$, as well as simply supported edge boundary conditions as it arises in typical

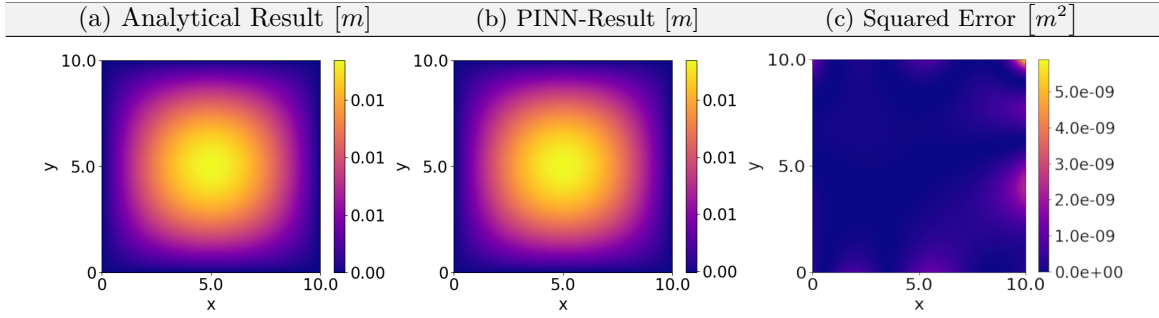| (a) Analytical Result $[m]$	(b) PINN-Result $[m]$	(c) Squared Error $[m^2]$

Figure 5: Kirchhoff plate bending problem: (a) analytical reference solution $[m]$, (b) PINNs results $[m]$ predicted with a fully-connected network consisting of three layers and 128 nodes each, and (c) squared error $[m^2]$.

civil engineering structures such as slabs [10]. We hence consider the following boundary conditions (BC):

$$u(0,y) = u(a,y) = u(x,0) = u(x,b) = 0$$
$$m_x(0,y) = m_x(a,y) = m_y(x,0) = m_y(x,b) = 0 \qquad (19)$$

where m_x and m_y are bending moments computed as follows:

$$m_x(x,y) = -D\left(\partial_x^2 u + \nu \partial_y^2 u\right)$$
$$m_y(x,y) = -D\left(\nu \partial_x^2 u + \partial_y^2 u\right) \qquad (20)$$
$$m_{xy}(x,y) = -D(1-\nu)\partial_{xy}^2 u$$

In total, we obtain 8 boundary conditions and therefore 9 terms in the PINNs loss function, making this a challenging task for balancing the contributions of the various objectives:

$$\mathcal{L}^{(t)} = \frac{\lambda_0}{|\hat{\Omega}|} \sum_{x,y \in \hat{\Omega}} \left\| \nabla^4 U(x,y;\theta) - \frac{p}{D} \right\|_2^2$$
$$+ \sum_{i=1}^{4} \frac{\lambda_i}{|\hat{\Gamma}_i|} \sum_{x,y \in \hat{\Gamma}_i} \|U(x,y;\theta)\|_2^2$$
$$+ \sum_{i=5}^{6} \frac{\lambda_i}{|\hat{\Gamma}_i|} \sum_{x,y \in \hat{\Gamma}_i} \|m_{ux}(x,y)\|_2^2 \qquad (21)$$
$$+ \sum_{i=7}^{8} \frac{\lambda_i}{|\hat{\Gamma}_i|} \sum_{x,y \in \hat{\Gamma}_i} \|m_{uy}(x,y)\|_2^2$$

After the successful convergence of the PINNs training, we obtain the results displayed in fig. 5(b) and compare it to the analytically available solution displayed in fig. 5(a). A plot of the squared difference in u as given by the analytical and PINNs results is shown in fig. 5(c) and delivers a negligible maximum error. The final algorithm settings are reported in tab. VIII.

Fig. 6 shows an example of ReLoBRaLo's training progress on Kirchhoff's equation. In this particular example, one can notice the larger variance of scaling values towards the end of training. Also, the scalings did not converge towards the value 1, thus suggesting that the training finished without all terms having stopped making progress, i.e. the scalings for the boundary conditions on the moments (yellow) were increasing at the end of training, while the boundary conditions on the displacements (red) were
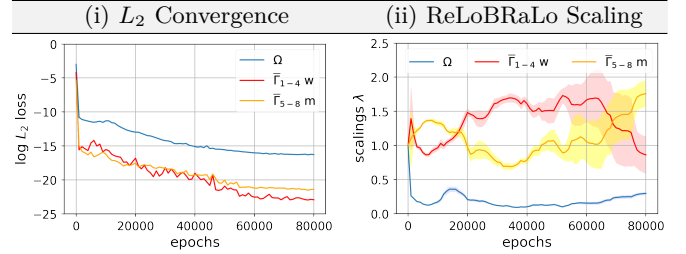
(i) L_2 Convergence	(ii) ReLoBRaLo Scaling

Figure 6: Median of the log L_2 loss over multiple training runs (a) and the mean and variance of the corresponding scaling factors λ (b) computed with ReLoBRaLo on Kirchhoff's equation with $\alpha = 0.999$, $\mathcal{T} = 0.1$, $\mathbb{E}[\rho] = 1$. For the sake of readability, the boundary conditions $1-4$ and $5-8$ were aggregated by taking the mean value.

decreasing. This gives a strong indication as to where the model's limitations lie. In this case, additional attention should be paid to the moments, e.g. by selecting an activation function which is better behaved in the second derivative than tanh. Note how ReLoBRaLo in combination with early stopping weakly imposes Pareto optimal updates, as it gradually increases the weights of underperforming terms and eventually leads to a stop of the training process due to a lack of global progress. This is an important property in the context of PINNs, because optimising only a subset of terms in the loss can lead to unsatisfactory solutions from a physical perspective.

Concerning performance, ReLoBRaLo outperforms the baseline and other algorithms by almost an order of magnitude in accuracy, while also yielding a very small standard deviation and hence being very consistent across training runs (cf. tab. IV). The results show its effectiveness, even on Kirchhoff's challenging problem with a total of 9 terms (cf. Eq. 21). Furthermore, the execution times in tab V underline the efficiency benefit (up to sixfold speedup) of balancing the loss without gradient statistics, as separate backwards passes for each term become increasingly computationally expensive as the number of terms in the loss function grows. Further conclusions and comparisons across different loss balancing methods are made in sec. IX.

For the inverse Kirchhoff problem setting, we select

Kirchhoff		Baseline	GradNorm	LR anneal.	SoftAdapt	ReLoBRaLo
Forward	train f	$1.2 \cdot 10^{-7}$	$5.3 \cdot 10^{-7}$	$9.1 \cdot 10^{-9}$	$1.8 \cdot 10^{-8}$	$6.0 \cdot 10^{-9}$
	val u	$1.3 \cdot 10^{-8}$	$1.7 \cdot 10^{-8}$	$2.7 \cdot 10^{-9}$	$2.5 \cdot 10^{-9}$	$4.0 \cdot 10^{-10}$
	std val u	$3.9 \cdot 10^{-8}$	$2.2 \cdot 10^{-7}$	$1.0 \cdot 10^{-6}$	$1.9 \cdot 10^{-9}$	$7.7 \cdot 10^{-10}$
Inverse	val μ	2.1	3.6	6.0	9.5	$3.2 \cdot 10^{-2}$
$D = 20.8\overline{3}$	std μ	1.6	4.7	0.8	4.9	$2.9 \cdot 10^{-2}$

Table IV: Comparison of the median L_2 training and validation loss on Kirchhoff's equation against a baseline of manually chosen scalings. The reported values are the median over four independent runs with identical settings. Additionally, we report the standard deviation over the runs of the best performing model on the validation loss.
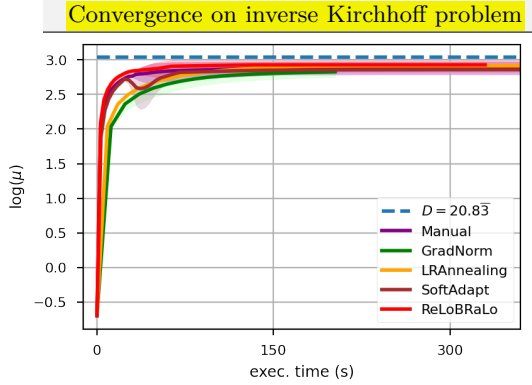

Figure 7: Approximation of the true PDE parameter value D (dashed line) for the inverse problem setting of Kirchhoff plate bending. Reported values are the mean (solid line) and standard deviation (shaded area) of four independent runs.

	GradNorm	LR ann.	SoftAdapt	ReLoBRaLo
ΔT_{co} [s]	128.6	139.7	20.2	22.5

Table V: Median computational overhead ΔT_{co} (in s) per 1'000 optimisation steps compared to using no balancing scheme (17.3s) on Kirchhoff's equation.

the PDE parameter $\mu := D$ (i.e. flexural stiffness) to be learned for given data, which we obtained by sampling from the analytically known solution. More specifically, we initialised $\mu = 0.5$ and tasked the network with approximating $D = 20.8\overline{3}$. Given the large disparity between the initialisation and the target, we empirically found the use of two separate optimisers beneficial in this case, where one optimiser is used for updating the network's parameters θ and a different one for updating the PDE parameter μ. Differently from Burgers' equation, ReLoBRaLo also sets a new benchmark in Kirchhoff's inverse problem, both in accuracy as well as convergence speed, cf. fig. 7 and tab. V.

C. Helmholtz equation

The Helmholtz equation represents a time-independent form of the wave equation and arises in many physical and engineering problems such as acoustics and electromagnetism [60]. The equation has the form:

$$\Delta u(x,y) + k^2 u(x,y) = f(x,y), \quad x,y \in [-1,1]^2 \quad (22)$$

where k is the wave number. This represents a common problem to benchmark PINNs and possesses an analytical solution in combination with Dirichlet boundaries:

$$f(x,y) = (-\pi^2 - (4\pi)^2 + k^2)sin(\pi x)sin(4\pi y)$$
$$u(x,y) = sin(\pi x)sin(4\pi y) \quad (23)$$
$$u(-1,y) = u(1,y) = u(x,-1) = u(x,1) = 0$$

Both, the x_1 and x_2 input variables, are bounded below by -1 and bounded above by 1. Therefore, the boundary conditions add four terms to the loss function of the *forward problem*, resulting in a 5-term total physics-informed loss:

$$\mathcal{L}^{(t)} = \frac{\lambda_0}{|\hat{\Omega}|} \sum_{x,y \in \hat{\Omega}} \left\| \Delta U(x,y;\theta) + k^2 U(x,y;\theta) - f(x,y) \right\|_2^2$$
$$+ \sum_{i=1}^{4} \frac{\lambda_i}{|\hat{\Gamma}_i|} \sum_{x,y \in \hat{\Gamma}_i} \|U(x,y;\theta)\|_2^2$$

$$(24)$$

where $U(x,y;\theta)$ is the parameterisation of the latent function $\hat{\mathbf{u}}(x,y)$ using a neural network with parameters θ.

After a successful convergence of the PINNs training, we obtain the results displayed in fig. 8(b) and compare it to the analytically available solution displayed in fig. 8(a). A plot of the squared difference in u as given by the analytical and PINNs results is shown in fig. 8(c) and delivers a negligible max error. The final algorithm settings are reported in tab. VIII.

The Helmholtz equation reveals a limitation of our basic loss balancing approach and motivates the introduction of the random lookback. GradNorm and Learning Rate Annealing both achieve impressive results and substantially outperform the baseline as well as ReLoBRaLo with $\mathbb{E}[\rho] = 1$ in terms of $L2$ accuracy for the BC terms, cf. fig. 9. This is likely due to the considerable initial difference in magnitudes between the governing equation and the boundary conditions. Furthermore, the high values of α, necessary for "remembering" the deteriorations longer, induce a latency between the increase of a term's loss until the scaling λ reacts accordingly (cf. fig. 9(d)). On the other hand, GradNorm and Learning Rate Annealing do not succeed in decreasing the $L2$ error as much as ReLoBRaLo for the governing equation term. Fig. 9 shows that both GradNorm and Learning Rate Annealing focus on improving the boundary conditions right from the beginning of training, whereas ReLoBRaLo with $\mathbb{E}[\rho] = 1$ counters the initial deterioration, but eventually "forgets"
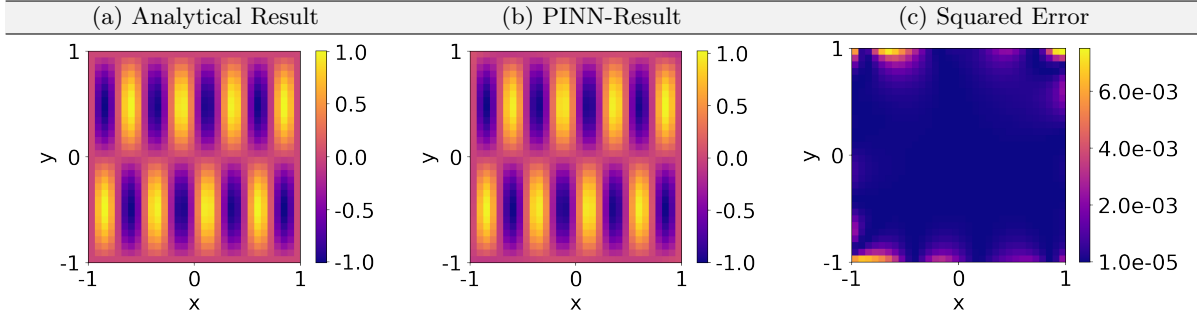
Figure 8: Helmholtz's problem: (a) analytical reference solution, (b) PINNs results predicted with a fully-connected network consisting of two layers and 128 nodes each, and (c) squared error.
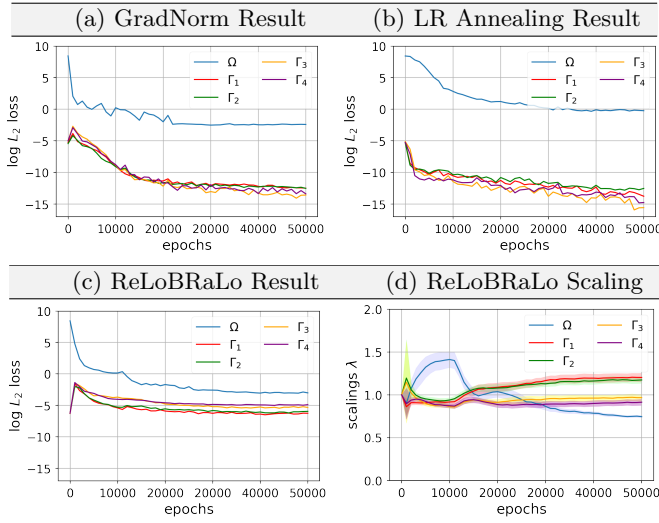

Figure 9: Median of the log L_2 loss over multiple training runs on the Helmholtz's equation using (a) GradNorm, (b) Learning Rate Annealing and (c) ReLoBRaLo, as well as the mean and variance of scalings calculated by ReLoBRaLo (d).

and instead focuses on the more dominant governing equation. This is also reflected in the discrepancy between the training and validation loss: GradNorm and Learning Rate Annealing have a higher training loss than ReLoBRaLo, but still exceed at approximating the underlying function (cf. tab. VI). This triggered further investigation on the saudade and temperature parameters as described in the remainder of the next section.

For the inverse Helmholtz problem setting, we select the wave number k to be learned for given data, which we obtained by sampling from the analytically known solution. Furthermore, we initialised $\mu = 0.5$ and tasked the network with approximating $k = 1$. In the Helmholtz inverse problem setting similarly to the inverse Burgers problem, also just one optimiser was chosen for updating the network's parameters θ together with the PDE parameter μ. ReLoBRaLo also sets a new benchmark for Helmholtz's inverse problem, both in accuracy as well as convergence speed, cf. fig. 10 and tab. VII.
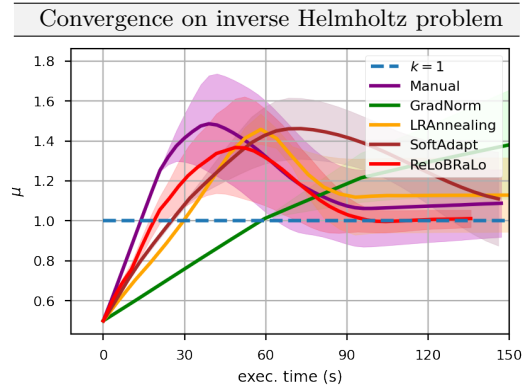

Figure 10: Approximation of the true PDE parameter value k (dashed line) in the inverse problem setting of Helmholtz's equation. Reported values are the mean (solid line) and standard deviation (shaded area) of four independent runs.

IX. ABLATION AND SENSITIVITY STUDY

The proposed ReLoBRaLo loss balancing scheme together with the PINNs architecture introduce many hyperparameters that have major influence on performance, efficiency and accuracy. In order to investigate the relations between hyperparameters and find their optimal combinations, we conduct an ablation and sensitivity study w.r.t. the temperature T, exponential decay rate α, and expected saudade $\mathbb{E}[\rho]$ and report its results in this section.

Fig. 11 visualises the models' sensitivity to the exponential decay rate α and the temperature T. A larger α causes the network to "remember" longer, while T controls how much the scalings "sheer out". Fig. 11(c) shows that Helmholtz's equation benefits most from small values for T, which turn the balancing more aggressive. This is in line with the findings in the previous section (cf. sec. VIII-C), where we noted that the large difference in magnitudes between the terms in the loss function caused issues to ReLoBRaLo and that resolute balancing was necessary to avoid the boundary conditions to be neglected. In fact, we found the optimal T to be 10^{-5} (cf. tab. VIII). On the other hand, Burgers and Kirchhoff require smoother scalings with a tendency towards higher T and α. It is worth noting that all three tasks benefit from the relaxation

Helmholtz		Baseline	GradNorm	LR anneal.	SoftAdapt	ReLoBRaLo
Forward	train f	$1.4{\cdot}10^{-2}$	$7.1{\cdot}10^{-2}$	$2.7{\cdot}10^{-1}$	$9.5{\cdot}10^{-3}$	$4.7{\cdot}10^{-3}$
	val u	$7.1{\cdot}10^{-2}$	$5.6{\cdot}10^{-6}$	$1.4{\cdot}10^{-5}$	$1.6{\cdot}10^{-3}$	$2.6{\cdot}10^{-5}$
	val std u	$8.1{\cdot}10^{-3}$	$1.9{\cdot}10^{-5}$	$7.6{\cdot}10^{-5}$	$1.5{\cdot}10^{-3}$	$8.2{\cdot}10^{-5}$
Inverse	val μ	$2.7{\cdot}10^{-3}$	$1.5{\cdot}10^{-1}$	$5.1{\cdot}10^{-2}$	$9.1{\cdot}10^{-2}$	$3.7{\cdot}10^{-4}$
$k=1$	std μ	$5.0{\cdot}10^{-2}$	$3.6{\cdot}10^{-1}$	$7.2{\cdot}10^{-2}$	$2.1{\cdot}10^{-2}$	$2.5{\cdot}10^{-4}$

Table VI: Comparison of the median L_2 training and validation loss on Helmholtz's equation against a baseline of manually chosen scalings. The reported values are the median over four independent runs with identical settings. Additionally, we report the standard deviation over the runs of the best performing model on the validation loss.
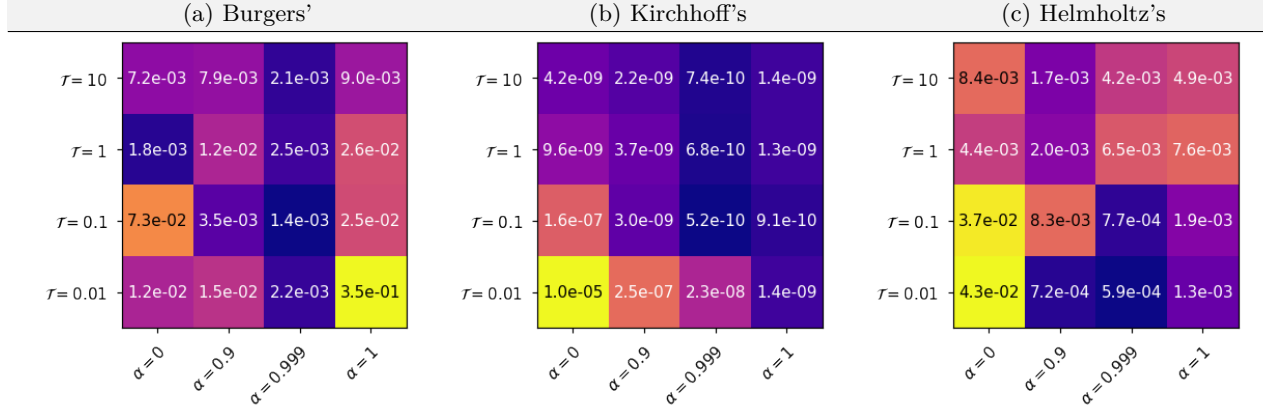

Figure 11: Ablation of the model's performance when varying $\mathcal{T}$ and α with $\mathbb{E}[\rho] = 1$. The reported values are the median of the log L_2 loss over multiple training runs.

	GradNorm	LR ann.	SoftAdapt	ReLoBRaLo
$\Delta T_{co}\,[s]$	10.4	6.7	5.0	5.2

Table VII: Median computational overhead ΔT_{co} (in s) per 1'000 optimisation steps compared to using no balancing scheme (4.8s) on Helmholtz's equation.

Hyperparameter	Burgers	Kirchhoff	Helmholtz
Learning Rate l_r	10^{-3}	10^{-3}	10^{-3}
Layers d_K	4	4	2
Neurons per Layer w_K	256	360	256
Exponential Decay Rate α	0.999	0.999	0.99
Temperature $\mathcal{T}$	10^{-1}	10^{-2}	10^{-5}
Expected Saudade $\mathbb{E}[\rho]$	0.9999	0.9999	0.99
Activation function σ	$tanh$	$tanh$	$tanh$

Table VIII: Final choices of hyperparameters for architecture and training settings.

$\mathbb{E}[\rho]$	Helmholtz	Burgers	Kirchhoff
0.0	$2.0{\cdot}10^{-3}$	$1.0{\cdot}10^{-3}$	$1.5{\cdot}10^{-09}$
0.5	$5.2{\cdot}10^{-5}$	$9.5{\cdot}10^{-3}$	$2.7{\cdot}10^{-09}$
0.9	$4.0{\cdot}10^{-5}$	$1.3{\cdot}10^{-3}$	$2.1{\cdot}10^{-09}$
0.99	$2.6{\cdot}10^{-5}$	$4.9{\cdot}10^{-4}$	$6.9{\cdot}10^{-10}$
0.999	$4.1{\cdot}10^{-5}$	$3.8{\cdot}10^{-4}$	$5.6{\cdot}10^{-10}$
0.9999	$1.2{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$	$4.0{\cdot}10^{-10}$
1	$8.1{\cdot}10^{-4}$	$4.7{\cdot}10^{-4}$	$7.4{\cdot}10^{-10}$

Table IX: Validation loss when varying the expected value of ρ. The reported values are the median over three independent runs.

through the exponential decay, as setting $\alpha = 1$ always causes a deterioration of the model's performance.

However, the relaxation through the exponential decay induces a new trade-off between making the model remember longer and letting it adapt quickly to changes during training. We therefore study the effects of a random lookback through a Bernoulli random variable ρ (*saudade*). It allows setting a lower value for α, thus making the model more flexible, while occasionally "reminding" it of its progress since the start of training $\mathcal{L}_i^{(0)}$.

Tab. IX summarises the change in performance when varying the expected saudade on all three experiments as a comparison. It is apparent that Helmholtz benefits more from frequent lookbacks, as it hits its best performance at $\mathbb{E}[\rho] = 0.99$, whereas Burgers and Kirchhoff only require an expected lookback every 10'000 optimisation steps. Figs. 12 and 3 illustrate the effect of random lookbacks. While the stochasticity in the scaling factor increases and therefore makes them less interpretable, it increases the weight on the boundary conditions. The scaled contribution of the boundary conditions consequently leads to a better approximation of the underlying function $\hat{\mathbf{u}}$. It is worth noting that the addition of the random lookback improves the accuracy on Helmholtz' equation by more than an order of magnitude while having a lesser, albeit still significant effect on Burgers' and Kirchhoff's equations.

X. Synopsis and Outlook

From previous work we observe, that a competitive relationship between physics loss items in the training of PINNs exists and potentially spoils training success,
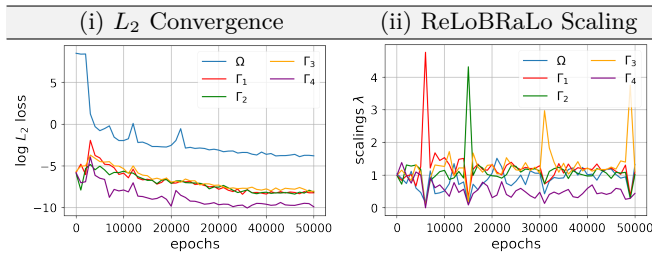
Figure 12: Example of a single training process of ReLo-BRaLo on Helmholtz's equation with $\alpha = 0.999$, $\mathcal{T} = 10^{-4}$, $\mathbb{E}[\rho] = 0.9999$.

performance or efficiency. This paper investigated different methods aiming at adaptively balancing a loss function consisting of various, potentially conflicting objectives as it may arise in scalarised MOO in PINNs. We proposed a novel adaptive loss balancing method by (i) combining the best attributes of existing approaches, and (ii) introducing a saudade parameter ρ to occasionally incorporate historic loss contribution. This forms a new scheme called **Relative Loss Balancing with Random Lookback** (ReLoBRaLo) for selecting bespoke weights in order to combine multiple loss terms for the training of PINNs. The effectiveness and merits of using ReLoBRaLo was then demonstrated empirically by investigating several standard PDEs, including solving Helmholtz equation, Burgers' equation and Kirchhoff plate bending equation, and considering both forward problems as well as inverse problems. Our computations showed that ReLoBRaLo is able to consistently outperform the baseline of existing scaling methods (GradNorm, Learning Rate Annealing, SoftAdapt) in terms of accuracy, while also being up to six times more computationally efficient (training epochs or wall-clock time). Finally, we showed that the adaptively chosen scalings λ can be inspected to learn about the PINNs training process and identify weak points. This allows to take informed decisions in order to improve the framework.

Future research is concerned with inspection of performance, efficiency, robustness, and scalability of ReLoBRaLo to further PDE classes such as Navier-Stokes equations etc. The adoption of Sobolev Training with Sobolev norms or the incorporation of the Mixture-of-Experts approach [2] together with ReLoBRaLo may solve the drawback associated with the high costs involved in estimating the neural network solutions of PDEs.

REFERENCES

[1] BATHE, K. *Finite Element Procedures.* No. pt. 2 in Finite Element Procedures. Prentice Hall, 1996.

[2] BISCHOF, R., AND KRAUS, M. A. Mixture-of-experts-ensemble meta-learning for physics-informed neural networks. In *Proceedings of 33. Forum Bauinformatik* (2022).

[3] BONKILE, M. P., AWASTHI, A., LAKSHMI, C., MUKUNDAN, V., AND ASWIN, V. S. A systematic literature review of burgers' equation with recent advances. *Pramana 90*, 6 (Apr 2018), 69.

[4] CAI, S., WANG, Z., CHRYSSOSTOMIDIS, C., AND KARNIADAKIS, G. E. Heat transfer prediction with unknown thermal boundary conditions using physics-informed neural networks. In *Fluids Engineering Division Summer Meeting* (2020), vol. 83730, American Society of Mechanical Engineers, p. V003T05A054.

[5] CARUANA, R. Multitask learning. *Machine Learning 28* (07 1997).

[6] CHAKRABORTY, S. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics 426* (2021), 109942.

[7] CHANG, K.-H. Chapter 17 - design optimization. In *e-Design*, K.-H. Chang, Ed. Academic Press, Boston, 2015, pp. 907–1000.

[8] CHEN, Z., BADRINARAYANAN, V., LEE, C.-Y., AND RABINOVICH, A. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv e-prints* (Nov. 2017), arXiv:1711.02257.

[9] CZARNECKI, W. M., OSINDERO, S., JADERBERG, M., SWIRSZCZ, G., AND PASCANU, R. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems* (2017), vol. 2017-Decem, pp. 4279–4288.

[10] EN. *EN 1992-1-1 Eurocode 2: Design of concrete structures - Part 1-1: General ruels and rules for buildings* (Brussels, 2005), CEN.

[11] FINN, C., ABBEEL, P., AND LEVINE, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, JMLR.org, p. 1126–1135.

[12] FINN, C., AND LEVINE, S. Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm. *arXiv e-prints* (Oct. 2017), arXiv:1710.11622.

[13] FINN, C., XU, K., AND LEVINE, S. Probabilistic Model-Agnostic Meta-Learning. *arXiv e-prints* (June 2018), arXiv:1806.02817.

[14] GLOROT, X., AND BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010), Y. W. Teh and M. Titterington, Eds., vol. 9 of *Proceedings of Machine Learning Research*, PMLR, pp. 249–256.

[15] GOSWAMI, S., ANITESCU, C., CHAKRABORTY, S., AND RABCZUK, T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics 106* (2020), 102447.

[16] GROHS, P., HORNUNG, F., JENTZEN, A., AND VON WURSTEMBERGER, P. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362* (2018).

[17] GROSSMANN, C., ROOS, H.-G., AND STYNES, M. Numerical treatment of partial differential equations.

[18] Guo, H., Zhuang, X., and Rabczuk, T. A Deep Collocation Method for the Bending Analysis of Kirchhoff Plate. *arXiv e-prints* (Feb. 2021), arXiv:2102.02617.

[19] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering 379* (2021), 113741.

[20] Heydari, A. A., Thompson, C. A., and Mehmood, A. SoftAdapt: Techniques for Adaptive Loss Weighting of Neural Networks with Multi-Part Loss Functions. *arXiv e-prints* (Dec. 2019), arXiv:1912.12355.

[21] Hornik, K., Stinchcombe, M., and White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks 3*, 5 (1990), 551–560.

[22] Hughes, T. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Dover Civil and Mechanical Engineering. Dover Publications, 2012.

[23] Jagtap, A., and Karniadakis, G. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics 28* (11 2020), 2002–2041.

[24] Jagtap, A., Kharazmi, E., and Karniadakis, G. 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* (06 2020), 113028.

[25] Jagtap, A. D., Kawaguchi, K., and Em Karniadakis, G. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society of London Series A 476*, 2239 (July 2020), 20200334.

[26] Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics 404* (Mar. 2020), 109136.

[27] Jones, D. F., Mirrazavi, S. K., and Tamiz, M. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European journal of operational research 137*, 1 (2002), 1–9.

[28] Kendall, A., Gal, Y., and Cipolla, R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *arXiv e-prints* (May 2017), arXiv:1705.07115.

[29] Kharazmi, E., Zhang, Z., and Karniadakis, G. E. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering 374* (2021), 113547.

[30] Kim, Y., Choi, Y., Widemann, D., and Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *arXiv e-prints* (Sept. 2020), arXiv:2009.11990.

[31] Kingma, D. P., and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv e-prints* (Dec. 2014), arXiv:1412.6980.

[32] Kraus, M. A. *Machine Learning Techniques for the Material Parameter Identification of Laminated Glass in the Intact and Post-Fracture State.* PhD thesis, Universität der Bundeswehr München, 2019.

[33] Kraus, M. A., and Drass, M. Artificial intelligence for structural glass engineering applications—overview, case studies and future potentials. *Glass Structures & Engineering 5*, 3 (2020), 247–285.

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

[35] 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 (1998), 987–1000.

[36] Liashchynskyi, P., and Liashchynskyi, P. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv e-prints* (Dec. 2019), arXiv:1912.06059.

[37] Liu, X., Zhang, X., Peng, W., Zhou, W., and Yao, W. A novel meta-learning initialization method for physics-informed neural networks. *arXiv preprint arXiv:2107.10991* (2021).

[38] Martins, J. R. R. A., and Ning, A. *Engineering Design Optimization.* Cambridge University Press, 2021.

[39] McClenny, L., and Braga-Neto, U. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism. *arXiv e-prints* (Sept. 2020), arXiv:2009.04544.

[40] Meng, X., Li, Z., Zhang, D., and Karniadakis, G. E. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering 370* (2020), 113250.

[41] Mockus, J., Tiesis, V., and Zilinskas, A. *The application of Bayesian methods for seeking the extremum*, vol. 2. 09 2014, pp. 117–129.

[42] Nichol, A., Achiam, J., and Schulman, J. On First-Order Meta-Learning Algorithms. *arXiv e-prints* (Mar. 2018), arXiv:1803.02999.

[43] Nogueira, F. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–.

[44] Orlandi, P. *The Burgers equation.* Springer Netherlands, Dordrecht, 2000, pp. 40–50.

[45] Peng, W., Zhou, W., Zhang, J., and Yao, W. Accelerating Physics-Informed Neural Network Training with Prior Dictionaries. *arXiv e-prints* (Apr. 2020), arXiv:2004.08151.

[46] Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not

shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing 14*, 5 (2017), 503–519.

[47] RAISSI, M. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research 19* (2018), 1–24.

[48] RAISSI, M., PERDIKARIS, P., AND KARNIADAKIS, G. E. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arxiv.org* (2017).

[49] 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* (2019), 686–707.

[50] RAISSI, M., YAZDANI, A., AND KARNIADAKIS, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science 367*, 6481 (2020), 1026–1030.

[51] RAJESWARAN, A., FINN, C., KAKADE, S., AND LEVINE, S. Meta-Learning with Implicit Gradients. *arXiv e-prints* (Sept. 2019), arXiv:1909.04630.

[52] RUCHTE, M., AND GRABOCKA, J. Efficient multi-objective optimization for deep learning. *ArXiv abs/2103.13392* (2021).

[53] SENER, O., AND KOLTUN, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems* (2018), S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc.

[54] SHIN, Y. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics 28*, 5 (June 2020), 2042–2074.

[55] SHUKLA, K., JAGTAP, A. D., AND KARNIADAKIS, G. E. Parallel Physics-Informed Neural Networks via Domain Decomposition. *arXiv e-prints* (Apr. 2021), arXiv:2104.10013.

[56] SIRIGNANO, J., AND SPILIOPOULOS, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics 375* (2018), 1339–1364.

[57] SMITH, G. *Numerical Solutions of Partial Differential Equations: Finite Difference Methods*, 3rd ed. Oxford University Press, New York.

[58] SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv. 41*, 1 (Jan. 2009).

[59] SNOEK, J., LAROCHELLE, H., AND ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2* (Red Hook, NY, USA, 2012), NIPS'12, Curran Associates Inc., p. 2951–2959.

[60] SOMMERFELD, A. *Partial differential equations in physics*. Academic press, 1949.

[61] SON, H., JANG, J. W., HAN, W. J., AND HWANG, H. J. Sobolev training for the neural network solutions of pdes. *arXiv preprint arXiv:2101.08932* (2021).

[62] THEODORIDIS, S. Chapter 5 - online learning: the stochastic gradient descent family of algorithms. In *Machine Learning (Second Edition)*, S. Theodoridis, Ed., second edition ed. Academic Press, 2020, pp. 179–251.

[63] VLASSIS, N. N., MA, R., AND SUN, W. Geometric deep learning for computational mechanics part i: Anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering 371* (2020), 113299.

[64] VLASSIS, N. N., AND SUN, W. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering 377* (2021), 113695.

[65] WANG, S., TENG, Y., AND PERDIKARIS, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv e-prints* (Jan. 2020), arXiv:2001.04536.

[66] WANG, S., YU, X., AND PERDIKARIS, P. When and why PINNs fail to train: A neural tangent kernel perspective. *arXiv e-prints* (July 2020), arXiv:2007.14527.

[67] WIGHT, C. L., AND ZHAO, J. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *arXiv e-prints* (July 2020), arXiv:2007.04542.

[68] XIANG, Z., PENG, W., ZHENG, X., ZHAO, X., AND YAO, W. Self-adaptive loss balanced physics-informed neural networks for the incompressible navier-stokes equations. *arXiv preprint arXiv:2104.06217* (2021).

[69] YUAN, F.-G., ZARGAR, S. A., CHEN, Q., AND WANG, S. Machine learning for structural health monitoring: challenges and opportunities. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2020* (2020), vol. 11379, International Society for Optics and Photonics, p. 1137903.

[70] ZHANG, C., LIAO, Q., RAKHLIN, A., MIRANDA, B., GOLOWICH, N., AND POGGIO, T. Theory of Deep Learning IIb: Optimization Properties of SGD. *arXiv e-prints* (Jan. 2018), arXiv:1801.02254.