Robin Pre-Training for the Deep Ritz Method

Luca Courte and Marius Zeinhofer*[1]

[1]Simula Research Laboratory

Abstract

We analyze the training process of the Deep Ritz Method for elliptic equations with Dirichlet boundary conditions and highlight problems arising from essential boundary values. Typically, one employs a penalty approach to enforce essential boundary conditions, however, the naïve approach to this problem becomes unstable for large penalizations. A novel method to compensate this problem is proposed, using a small penalization strength to pre-train the model before the main training on the target penalization strength is conducted. We present numerical evidence that the proposed method is beneficial.

1 Introduction

After exceptional success in machine learning, neural networks become increasingly popular in numerical analysis. Strategies to approximate solutions of partial differential equations (PDEs) based on neural networks can be traced back to [4] and [6]. Their approaches are currently being revived and extended thanks to increased computational power and ease of implementation in frameworks like Tensorflow and PyTorch, see [12, 5, 11, 8, 1, 10].

Presently, a common drawback of neural network based methods is their poor reliability – a failure to learn even a simple solution to a PDE is not uncommon, see [14, 13]. In this note, we show that using the boundary penalty method known from the finite element literature, see for instance [2], to approximately enforce Dirichlet boundary conditions in the Deep Ritz Method leads to such an unreliability. More precisely, large penalization parameters – albeit mandatory for an accurate solution – lead to highly fluctuating errors and sometimes even to a failure to approximate the solution at all. We present a pre-training strategy to alleviate this issue. The key idea is to pre-train the model using a small penalization parameter and subsequently to shift it by an optimal amount before conducting the training with the desired large penalization. We show that this approach reduces the variance of the Deep Ritz Method and improves the accuracy up to almost one order of magnitude in the best case.

Although our numerical results are conducted for the Poisson equation on model domains, the method is applicable to general domains and general elliptic equations and can easily be combined with more advanced optimization strategies as for example proposed by [3, 7].

*Corresponding Author: mariusz@simula.no

1.1 The Deep Ritz Method

We briefly recall the Deep Ritz Method and the corresponding error estimates for the boundary penalty method. In general, the Deep Ritz Method proposed by [5], transforms the variational formulation of a PDE – if available – into a finite dimensional optimization problem using neural network type functions as an ansatz class. Suppose we want to approximately solve

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where Ω is an open and bounded set in $\mathbb{R}^d$. It is well known that for a function $u \in H_0^1(\Omega)$ and a right-hand side $f \in H_0^1(\Omega)^*$ it is equivalent to solve (1) and to be a minimizer of the following optimization problem

$$u \in \underset{v \in H_0^1(\Omega)}{\arg\min} \frac{1}{2} \int_\Omega |\nabla v|^2 \, \mathrm{d}x - f(v). \tag{2}$$

Now, we want want to minimize (2) over a class of neural network functions. However, it is unfeasible to enforce zero boundary values due to the unconstrained nature of neural networks. A solution is to use the boundary penalty method which allows to approximate (2) by an unconstrained problem. This approach was used by [5]. More precisely, let $\lambda > 0$ be a fixed penalization parameter and denote by Θ a set of neural network parameters and consider the problem of finding a (quasi-)minimizer of the loss function $\mathcal{L}_\lambda : \Theta \to \mathbb{R}$

$$\mathcal{L}_\lambda(\theta) = \frac{1}{2}\int_\Omega |\nabla u_\theta|^2 \mathrm{d}x - f(u_\theta) + \lambda \int_{\partial\Omega} u_\theta^2 \mathrm{d}s \quad (3)$$

This is now an unconstrained optimization problem that enforces zero boundary conditions approximately depending on the size of λ. To gain insight into the nature of this approach, we consider the energy, i.e., the loss function extended to all of $H^1(\Omega)$, namely $E_\lambda : H^1(\Omega) \to \mathbb{R}$ given by

$$E_\lambda(u) = \frac{1}{2}\int_\Omega |\nabla u|^2 \mathrm{d}x - f(u) + \lambda \int_{\partial\Omega} u^2 \mathrm{d}s$$

and its associated Euler Lagrange equation

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ \partial_n u + 2\lambda u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \quad (4)$$

Thus, using (3) to approximate (1) means to use a Robin boundary value problem to approximate a Dirichlet condition. Let us denote by $u_0 \in H^1_0(\Omega)$ the solution to (1) and by $u_\lambda \in H^1(\Omega)$ the solution to (4). The key observation is that using the boundary penalty formulation (3) is only asymptotically exact. More precisely, in [9] it is established that it holds

$$\|u_0 - u_\lambda\|_{H^1(\Omega)} \in \mathcal{O}(\lambda^{-1})$$

and that this rate is sharp. This means that accurately solving essential boundary value problems with the boundary penalty method requires large penalization strengths λ.

1.2 Main Contributions

The main contributions we provide in this article are

- Using the Deep Ritz Method with boundary penalty, we demonstrate that large values of the penalization parameter λ – albeit being necessary for a reasonably exact method – lead to numerical instabilities and possibly to a failure of the training process.[1]
- To alleviate this problem we propose a pre-training strategy. The training process is started with a low penalization value and the network is subsequently shifted by an optimal amount before fine tuning on the large target penalization strength. We test this approach on different examples and observe increased accuracy and reduced variance in the training outcome.

2 The Pre-Training Approach

Assume we want to solve the problem (1), i.e.,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

by using a neural network ansatz and the boundary penalty method as described in the introduction, i.e., employing the loss function $\mathcal{L}_\lambda$ with penalty parameter λ

$$\mathcal{L}_\lambda(\theta) = \frac{1}{2}\int_\Omega |\nabla u_\theta|^2 \mathrm{d}x - f(u_\theta) + \lambda \int_{\partial\Omega} u_\theta^2 \mathrm{d}s.$$

We propose the following two step pre-training procedure:

(i) We set a pre-training penalization strength λ_P, typically with the value $\lambda_P = 1$, and train the model using the loss function $\mathcal{L}_{\lambda_P}$ for a certain amount of iterations. We denote the neural network that results from this training by u_{θ_P}.

(ii) We choose a (large) target penalization strength λ_T and shift the network u_{θ_P} adding an optimal constant t_{λ_T} namely

$$u_{\theta_T} = u_{\theta_P} + t_{\lambda_T},$$

where

$$t_{\lambda_T} = \frac{\int_\Omega f(x)\,\mathrm{d}x}{2\lambda_T|\partial\Omega|} - \frac{\int_{\partial\Omega} u_{\theta_P}(x)\,\mathrm{d}x}{|\partial\Omega|}.$$

We denote the shifted function by u_{θ_T}. We then continue to train the network u_{θ_T} using the loss function $\mathcal{L}_{\lambda_T}$ corresponding to the target penalization strength λ_T.

[1] With failure we mean in this case that the constant zero function is learned.

The reason to choose the shift parameter t_{λ_T} in this particular way stems from a calculus argument which we provide as the next Lemma.

Lemma 1. Let $\lambda > 0$ and $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ a Lagrangian and $u_0 : \mathbb{R}^n \to \mathbb{R}$ an admissible function, then the energy

$$\begin{aligned} E_\lambda(u) = &\int_\Omega L(x, \nabla u(x))\,\mathrm{d}x - \int_\Omega f(x)u(x)\,\mathrm{d}x \\ &+ \lambda \int_{\partial\Omega} |u(x)|^2\,\mathrm{d}x \end{aligned}$$

is minimized among all translations $u_t := u_0 + t$ for $t \in \mathbb{R}$ if

$$t = \tilde{t} := \frac{1}{2\lambda|\partial\Omega|}\int_\Omega f(x)\,\mathrm{d}x - \frac{1}{|\partial\Omega|}\int_{\partial\Omega} u(x)\,\mathrm{d}x.$$

Proof. The translation u_t for $t \in \mathbb{R}$ that minimizes the energy E_λ has to satisfy

$$\frac{\mathrm{d}}{\mathrm{d}t}E_\lambda(u+t) = 0.$$

Hence, it has to hold

$$0 = -\int_\Omega f(x)u(x)\,\mathrm{d}x + 2\lambda\int_{\partial\Omega} u(x) + t\,\mathrm{d}x$$

which, after rearranging, implies the stated equation for the translation $\tilde{t} \in \mathbb{R}$. Due to the structure of the energy this has to be a minimizer. □

Remark 2. The above motivation is not limited to the Laplace operator. In fact, the derivation of the optimal shifting applies to more general elliptic operators as well.

3 Numerical Experiments

In this Section we describe the numerical experiments used to illustrate the effectiveness of the proposed pre-training algorithm. We begin with the general setup: The concrete test cases, the used network architectures and training hyperparameters. Then, in Section 3.1 we report the results of the naïve training approach and finally in Section 3.2 we contrast this with the performance of the pre-training approach.

The Test Examples. We test the proposed pre-training method on three different geometries and in cases where we have access to the true solutions. More precisely, we use the following two dimensional domains and right-hand sides

(i) The disk, i.e., $\Omega = B_1(0)$ and $f \equiv 1$, where $B_r(0)$ denotes the ball of radius r around the origin.

(ii) The annulus, i.e., $\Omega = B_2(0)\backslash B_1(0)$ and $f \equiv 1$.

(iii) The square, i.e., $\Omega = [0,1]^2$ and $f(x_1, x_2) := 8\pi^2 \sin(2\pi x_1)\sin(2\pi x_2)$.

In these cases, the analytical solutions are known.

(i) On the disk, the analytical solution is given by $u^D(x) := -\frac{1}{4}|x|^2 + \frac{1}{4}$ for $x \in B_1(0)$.

(ii) On the annulus, the analytical solution is given by $u^A(x) := -\frac{1}{4}|x|^2 + \frac{3}{4\log(2)}\log(|x|) + \frac{1}{4}$ for $x \in B_2(0) \setminus B_1(0)$.

(iii) On the square, the analytical solution is given by $u^S(x_1, x_2) := \sin(2\pi x_1)\sin(2\pi x_2)$ for $(x_1, x_2) \in [0,1]^2$.

The Network Architecture and Training Process. We use small networks with moderate depth to keep the computational cost manageable. Our precise choices that are global for all experiments are reported below.

(i) We use fully connected feed forward networks with four hidden layers and input dimension two and output dimension one. All hidden layers have 14 neurons. We choose the hyperbolic tangent as our activation function.

(ii) We initialize the weights using Glorot uniform initialization and the biases are initialized by zero.

(iii) The numerical integration of the integrals for the training is done using fixed evaluation points. For a number $N \in \mathbb{N}$ we use points in the lattice $\frac{1}{N}\mathbb{Z}^2$ to approximate the integral of a function $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ via

$$\int_\Omega f\,\mathrm{d}x = \frac{1}{N}\sum_{x_i \in \frac{1}{N}\mathbb{Z}^2 \cap \mathrm{int}(\Omega)} f(x_i).$$

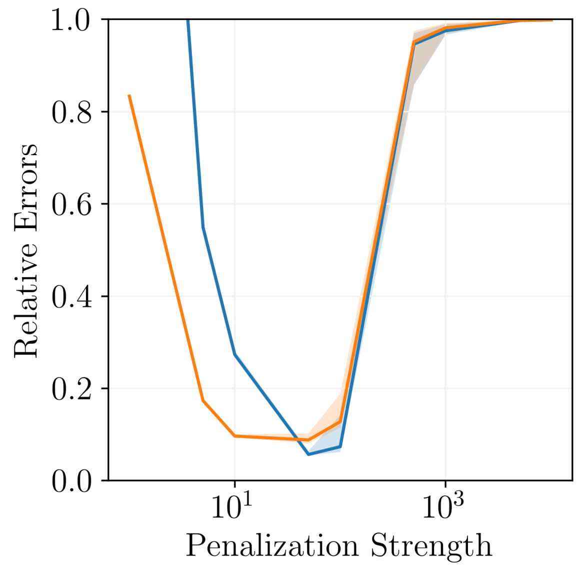

Figure 1: Experiment on the annular domain with zero boundary values, not employing the proposed pre-training. Reported is the median and the first and third quartile (shaded area) over 25 runs for different values of λ.

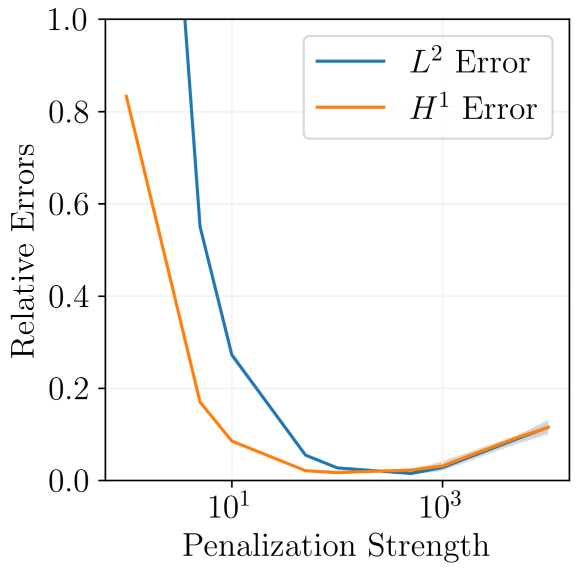

Figure 2: Experiment on the annular domain with zero boundary values, employing the proposed pre-training strategy. Reported is the median and the first and third quartile (shaded area) over 25 runs for different values of λ.

Integrals over the boundary of a domain are approximated using arclength parametrization with $N \cdot |\partial\Omega|$ many equi-spaced evaluation points. Here $|\partial\Omega|$ denotes the length of $\partial\Omega$. For the square and the annulus we use roughly $250,000$ evaluation points, i.e., the lattice constants are $N = 500$ and $N = 160$ respectively. For the disk we use roughly 90000 evaluation points, i.e, $N = 160$. We chose these numbers such that no further improvement of the method was noticeable upon refining.

(iv) For the optimization process we use the Adam optimizer with 10000 iterations, either for the use on one penalization strength or distributed between pre-training and target penalization. Depending on the experiment we vary slightly with the learning rate. Details can be found in the corresponding sections.

(v) The relative $L^2(\Omega)$ and $H^1(\Omega)$ errors are computed using 10^6 uniformly sampled evaluation points in Ω for every integral appearing in the respective norms.

(vi) We use the boundary penalization strengths $\lambda = 1, 5, 10, 50, 100, 500, 1000, 5000$ and 10000 and every experiment is conducted 25 times to capture the stochastic influence of the initialization.

3.1 Naïve Approach

We begin our experiments by directly training on the target penalization strength, where we vary the penalization parameter in a range between $\lambda = 1$ and $\lambda = 10,000$ having in mind that a penalization strength λ introduces an error of $\mathcal{O}(\lambda^{-1})$. We train for $10,000$ iterations with a fixed learning rate of 0.001.

We observe an optimal penalization for values of λ between 50 and 100, both for relative L^2 and relative H^1 errors, we refer to Table 1 and Table 2. Furthermore, as clearly visible in Figure 1 and Figure 3 for the example of the annulus and the square, both the error & the variance in the errors across different runs increases dramatically beyond penalization strengths of 100 to 1000.

In particular, for all three equations we fre-

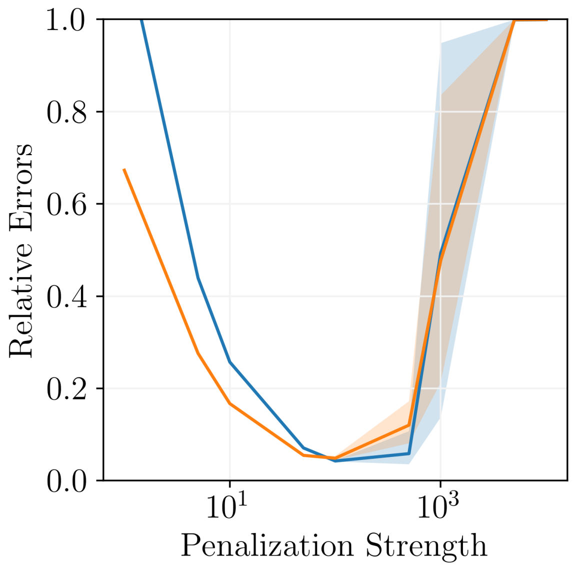

Figure 3: Experiment on the square with zero boundary values, directly training on the target penalization strength. Reported is the median and the first and third quartile (shaded area) over 25 runs for different values of λ.

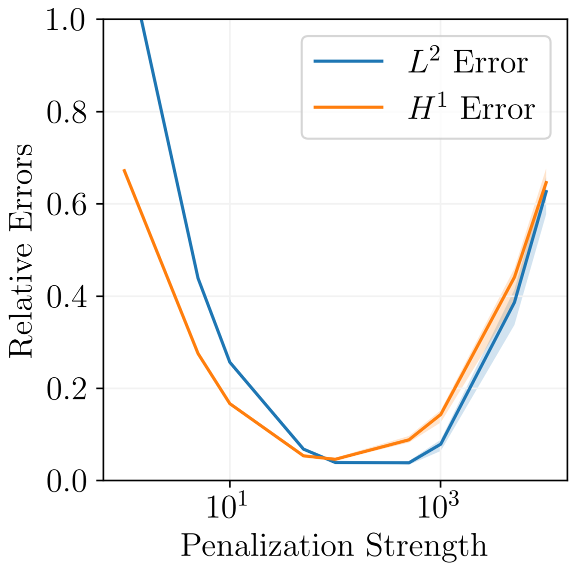

Figure 4: Experiment on the square with zero boundary values, employing the proposed pre-training strategy. Reported is the median and the first and third quartile (shaded area) over 25 runs for different values of λ.

quently observe a total failure to train in the large penalization regime, i.e., relative errors of around 100%. In this case the zero function is learned. Heuristically, we attribute the deteriorating accuracy to a growing number of increasingly attractive, poor local minima in the loss landscape for large penalization strengths until for very large penalization parameters even the zero function can result from training. A large value of λ forces the gradient descent dynamics violently towards zero boundary conditions at the expense of other features of the PDEs solution, hence good minima become less attractive and difficult to find.

Table 1: Best relative L^2 errors without pre-training.

Domain	Best relative L^2 error
Disk	$(3.14 \pm 0.82) \cdot 10^{-2}$ with $\lambda = 100$
Annulus	$(6.19 \pm 1.55) \cdot 10^{-2}$ with $\lambda = 50$
Square	$(3.17 \pm 0.12) \cdot 10^{-2}$ with $\lambda = 100$

Table 2: Best relative H^1 errors without pre-training.

Domain	Best relative H^1 error
Disk	$(4.61 \pm 1.61) \cdot 10^{-2}$ with $\lambda = 50$
Annulus	$(9.33 \pm 1.68) \cdot 10^{-2}$ with $\lambda = 50$
Square	$(4.62 \pm 1.14) \cdot 10^{-2}$ with $\lambda = 100$

3.2 Pre-Training Approach

In the previous Section, we have seen that the naïve approach using the boundary penalty method introduces large errors and unstable training dynamics. To mitigate this effect, we propose the pre-training strategy detailed in Section 2. We find that the proposed pre-training significantly reduces both the errors and the errors' variance.

For the experiments in this Section we set the pre-training penalization strength to $\lambda_P = 1$ and pre-train for 4000 iterations. Here, the first 1000 iterations the learning rate is set to 0.01 and for the remaining 3000 iterations we use 0.001. We proceed by shifting the neural network by the optimal amount t_{λ_T} and train 6000 iterations with learning rate set to 0.001 on the target penalization strength λ_T.

We report the relative $L^2(\Omega)$ and $H^1(\Omega)$ errors obtained through using the proposed pre-training strategy in Figure 2 and the best results we obtained in Table 3 and Table 4.

We clearly see beneficial effects for both the accuracy and the variance of the errors for all three equations. We refer also to Figure 2 and Figure 4 that should be contrasted with their counterparts without pre-training, i.e., Figure 1 and Figure 3.

In case of the disk, the relative error is almost reduced by a full magnitude. Also on the annulus, we see a drastic improvement of accuracy. For the square, the improvement in accuracy is smaller, however, the variance in the reported errors is decreased drastically as well. This is no surprise as the example of the square possesses an oscillating solution in the interior and we believe that the main error is due to this comparatively complicated solution. In the following, we list the advantages of our proposed pre-training strategy.

(i) Improved accuracy and increased reliability as discussed above.

(ii) Potential computational savings. Even though our method requires to pre-train the model we believe that the overall computational cost can be reduced. In fact, using larger learning rates is possible as the loss landscape is less rough for small values of λ. We have already exploited this in our experiment setup. Also in the main training, potentially larger learning rates can be used as the model is already in a reasonable state. We leave the fine-tuning of learning rate schedules for future research.

(iii) Decreased sensitivity to the choice of penalization parameter. Particularly, the experiments on the disk and the annulus suggest that the accuracy of the method does not depend as strongly on the penalization strength as in the case without pre-training. This facilitates choosing a reasonable penalization strength.

Table 3: Best relative L^2 errors with pre-training.

Domain	Best relative L^2 error
Disk	$(0.63 \pm 0.48) \cdot 10^{-2}$ with $\lambda = 500$
Annulus	$(1.81 \pm 0.80) \cdot 10^{-2}$ with for $\lambda = 500$
Square	$(3.05 \pm 0.21) \cdot 10^{-2}$ with $\lambda = 100$

Table 4: Best relative H^1 errors with pre-training.

Domain	Best relative H^1 error
Disk	$(1.17 \pm 0.21) \cdot 10^{-2}$ with $\lambda = 500$
Annulus	$(1.97 \pm 0.47) \cdot 10^{-2}$ with $\lambda = 100$
Square	$(4.54 \pm 0.48) \cdot 10^{-2}$ with $\lambda = 100$

4 Conclusions and Future Research

We observe that approximately enforcing essential boundary conditions in the Deep Ritz Method leads to an unstable training process and reduced accuracy. To mitigate this effect, we employ a pre-training approach and show that the proposed method stabilizes the training and increases accuracy.

Future research can concentrate on fine-tuning the interplay of the learning rate and the penalization parameter, as we observed that smaller penalization parameters potentially allow larger learning rates and hence an accelerated training procedure.

Author Contributions

All authors contributed equally to all parts of the manuscript.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] I. Babuška. The finite element method with penalty. *Mathematics of computation*, 27(122): 221–228, 1973.

[3] E. C. Cyr, M. A. Gulian, R. G. Patel, M. Perego, and N. A. Trask. Robust Training and Initialization of Deep Neural Networks: An adaptive Basis Viewpoint. In *Mathematical and Scientific Machine Learning*, pages 512–536. PMLR, 2020.

[4] M. Dissanayake and N. Phan-Thien. Neural-Network-based Approximations for solving Partial Differential Equations. *Communications in Numerical Methods in Engineering*, 10 (3):195–201, 1994.

[5] W. E and B. Yu. The Deep Ritz method: a Deep Learning-based numerical Algorithm for solving Variational Problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018. doi: 10.1007/s40304-018-0127-z. URL `https://doi.org/10.1007/s40304-018-0127-z`.

[6] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial Neural Networks for solving Ordinary and Partial Differential Equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178. URL `https://doi.org/10.1109/72.712178`.

[7] K. Lee, N. A. Trask, R. G. Patel, M. A. Gulian, and E. C. Cyr. Partition of unity networks: Deep hp-approximation. In *CEUR Workshop Proceedings*, volume 2964, page 180. CEUR-WS, 2021.

[8] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations*, 2021.

[9] J. Müller and M. Zeinhofer. Error Estimates for the Deep Ritz Method with Boundary Penalty. In *3rd Conference on Mathematical and Scientific Machine Learning (MSML2022)*, 2022.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[11] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of non-linear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018. doi: 10.1016/j.jcp.2017.11.039. URL `https://doi.org/10.1016/j.jcp.2017.11.039`.

[12] J. Sirignano and K. Spiliopoulos. DGM: A Deep Learning Algorithm for solving Partial Differential Equations. *Journal of computational physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029. URL `https://doi.org/10.1016/j.jcp.2018.08.029`.

[13] R. van der Meer, C. W. Oosterlee, and A. Borovykh. Optimally weighted loss functions for solving pdes with neural networks. *Journal of Computational and Applied Mathematics*, 405:113887, 2022. doi: 10.1016/j.cam.2021.113887. URL `https://doi.org/10.1016/j.cam.2021.113887`.

[14] S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021. doi: 10.1137/20M1318043. URL `https://doi.org/10.1137/20M1318043`.