POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Towards Adaptive PINNs For PDEs: A Numerical Exploration

Laurea Magistrale in Computational Science & Engineering - Mathematical Engineering

Author: Riccardo Patroni

Advisor: Prof. Marco Verani

Co-advisor: Prof. Edie Miglio

Academic year: 2021-2022

---

1. Introduction

The aim of this work consists in investigating the relevance attached to the architectural parameters, in terms of resulting performance for the underlying models, in the newborn branch of numerical resolution of Partial Differential Equations (PDEs) through the so-called Physics-Informed Neural Networks (PINNs) (see [4, 7]). The inner workings of our architectures adheres to the well-known paradigm of Feed Forward Deep Neural Networks. Employing a general and significative set of scalar PDEs, whose exact solution is known a priori, we will evaluate the relative importance of the following list of structural features: number of hidden layers embedded in the network, amount of neurons exploited per layer and cardinality of the training set, consisting in a collection of uniformly sampled points inside the computational domain, where the strong form of the studied PDE is imposed. The applicative fields of our interest are subdivided in three main categories, whose solutions respectively consist in: single frequency sinusoids, multi-scale target functions or generic analytical expressions. Another important aspect, concerning the proper choice of the neural activation function, will be thoroughly considered. After the presentation of the results coming from the aforementioned tests, where we employ the basic (fixed) structure of PINNs to analyze their general performances in various applicative frameworks, we will propose the implementation of an adaptive scheme that focuses and dynamically acts upon the architectural model's hyper-parameters during the learning phase, along with the related heuristical justifications that led to its elaboration. Such technique jointly employs the so-called Growing Method (see [1]) and a revisited version of the Residual Adaptive Refinement algorithm, formerly introduced in [4]. The essential computational tool exploited for the construction of all models relies on a newly-developed Python library for the resolution of scalar PDEs over elementary hyper-rectangular domains. All learning procedures have been performed by means of a combined ADAM - LBFGS optimizer (see [2, 5]).

2. Technical Background

As we previously anticipated, in this work we have tested the performances of PINNs on a series of scalar PDEs by implicitly imposing their solution in strong form inside the computational domain, coupling the problem with a proper set of boundary conditions. In particular, we will consider a generic scalar Partial Differential Equation, possibly parametrized by a vector of coefficients $\boldsymbol{\lambda}$, for the solution $u(\boldsymbol{x})$ defined on a domain $\Omega \subset R^d$:

$$f\left(\boldsymbol{x}; \frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \dots; \boldsymbol{\lambda}\right) = 0, \qquad \boldsymbol{x} \in \Omega,$$

while all boundary conditions (of Dirichlet, Neumann or periodic type) on the proper edges of $\partial\Omega$ will be grouped under the following notation:

$$\mathcal{B}(u, \boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \partial\Omega.$$

For time-dependent problems we consider the time coordinate as a special component of $\boldsymbol{x}$, so that Ω represents the entire spatio-temporal domain. In this case, the initial condition can be simply treated as a special type of Dirichlet boundary condition. The expression in use for our cost functional reads as follows:

$$\mathcal{L}(\boldsymbol{W}; \mathcal{T}_f, \mathcal{T}_b) = \mathcal{L}_f(\boldsymbol{W}; \mathcal{T}_f) + \mathcal{L}_b(\boldsymbol{W}; \mathcal{T}_b),$$

where the explicit formulation of the written terms is provided as:

$$\mathcal{L}_f(\boldsymbol{W}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\boldsymbol{x} \in \mathcal{T}_f} \left\| f\left(\boldsymbol{x}; \frac{\partial \hat{u}}{\partial x_1}, \dots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \dots; \boldsymbol{\lambda}\right)\right\|_2^2,$$

$$\mathcal{L}_b(\boldsymbol{W}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\boldsymbol{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \boldsymbol{x})\|_2^2.$$

All the networks (or, briefly, models) involved in this exposition, if not specified otherwise, have been trained with the hyperbolic tangent activation function. Moreover, for every set of architectural specifications, a total of three attempts have been performed, each one with a different random seed for the related weights initialization. In all figures presented in this work, the represented model instance (first, second or third) is accordingly specified.

3. Single-Scale & Multi-Scale Tests

Concerning both the single-scale and multi-scale tests performed in this work, we considered the target to be imposed as the exact solution of the one-dimensional Poisson problem with homogeneous boundary conditions:

$$\begin{cases} -u'' = f & x \in (-1, 1) \\ u = 0 & x \in \{-1, 1\} \end{cases} \tag{1}$$

We now present a chosen subset of experimental instances that, in this particular framework, can be considered as the most salient and representative among the studied cases. We start by considering the highest frequencies imposed in the single-scale sensitivity analysis and the stiffest solution belonging to the multi-scale sensitivity analysis, where we aim at understanding the importance of the role associated to the architectural parameters of our PINNs, namely: the number of hidden layers in use, the amount of neurons per layer employed and the cardinality of the training set. We subsequently show the results coming from our single-scale and multi-scale experimental convergence analyses, where we study the related properties of PINNs. In the latter, we fix a proper structure for the network and we observe their predictive power trend by varying the number of training spots, uniformly distributed inside the computational domain.

NPL/HL	1	2
25	0.006956	2.176026
50	0.000391	9.132307
100	0.000318	6.072858

(a) Best error, 80 residuals.

NPL/HL	1	2
25	0.004699	15.495498
50	0.001069	15.223295
100	0.000312	0.000543

(b) Best error, 320 residuals.

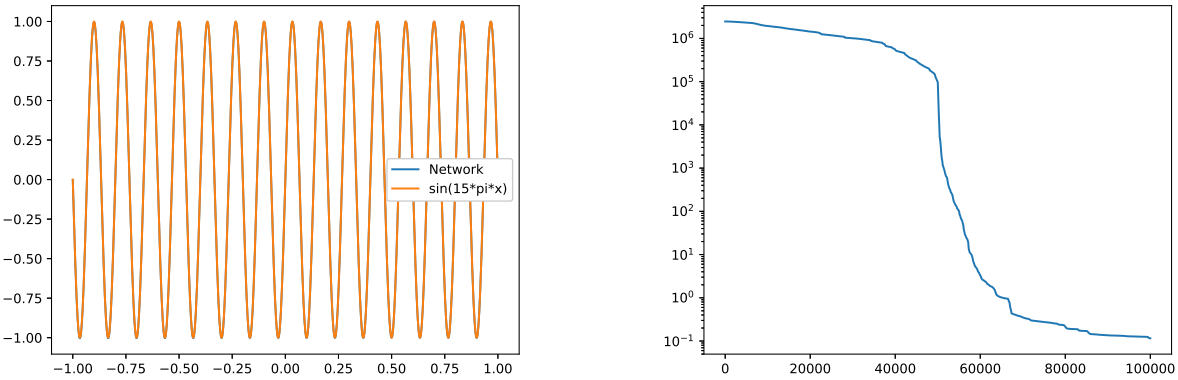
NPL/HL	1	2
25	0.016992	15.584377
50	0.001717	15.975008
100	0.000404	0.029286

(c) Best error, 1280 residuals.

Table 1: Tables for the models trained to approximate the medium-high frequency solution.

Table 1 reports the best L^2 relative errors obtained for the approximation of the second highest frequency solution of our trial pool, $\sin(10\pi x)$. The rows correspond to different amounts of neurons per layer employed (NPL), while on top of every column we indicate the number of hidden layers in use (HL). For the easiest two targets considered (low and medium frequency, not illustrated here) we appreciate extremely low error values for all architectural combinations. Instead, the represented table is significant because it shows some trends that typically emerge in our study, through which we ascertain the importance of the role played by the structural hyper-parameters of our models. In this case, for instance, shallow networks perform better than their double-layered counterparts. Furthermore, in a seemingly counter-intuitive manner, we see that the performance of our models is not necessarily positively correlated to the total number of residual points exploited. Concerning

the single-scale sensitivity framework, we have always managed to train at least one successful model for each different target function. The experimental proof of this statement is contained in Figure 1, where we show an almost perfect match of our best model for the approximation of the stiffest frequency target ($\sin(15\pi x)$).

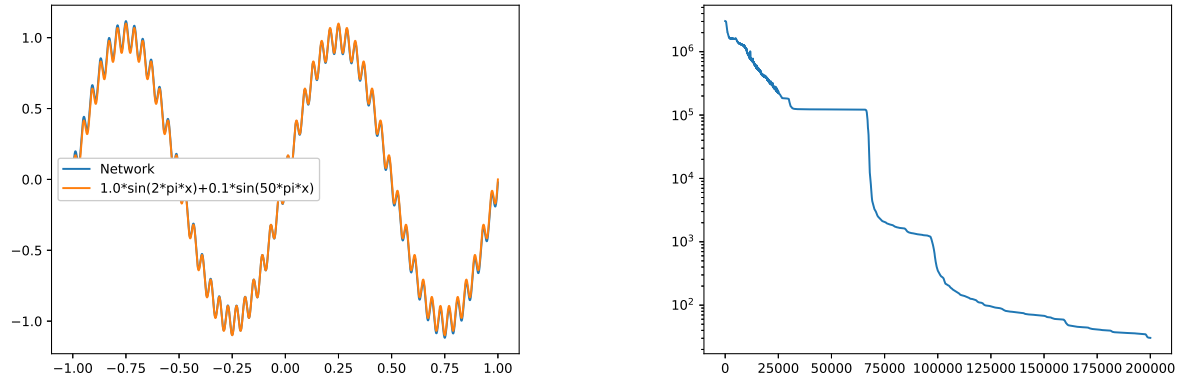


(a) Target solution and model approximation.

(b) Loss function evolution.

Figure 1: Best approximation of the high frequency solution relative to the first model trained with 1 hidden layer, 100 neurons and 80 residuals. On the left, we can fully appreciate the perfectly overlapping plots of the network and the exact solution, performing 15 complete oscillations in the computational domain. On the right, we see the loss function evolution in semi-logarithmic scale.

Figure 2 illustrates the best model instance regarding the simulation of our stiffest multi-scale test, formerly introduced in [7]. Exploiting the linearity embedded in the one-dimensional Poisson operator with homogeneous boundary conditions (1), we implicitly impose the following weighted sum of sinusoidal functions as the exact solution of our differential problem: $\sin(2\pi x) + 0.1\sin(50\pi x)$. The visualized multi-layered network presents, at the end of its learning phase, a relative L^2 error whose magnitude stays below the 2% threshold.



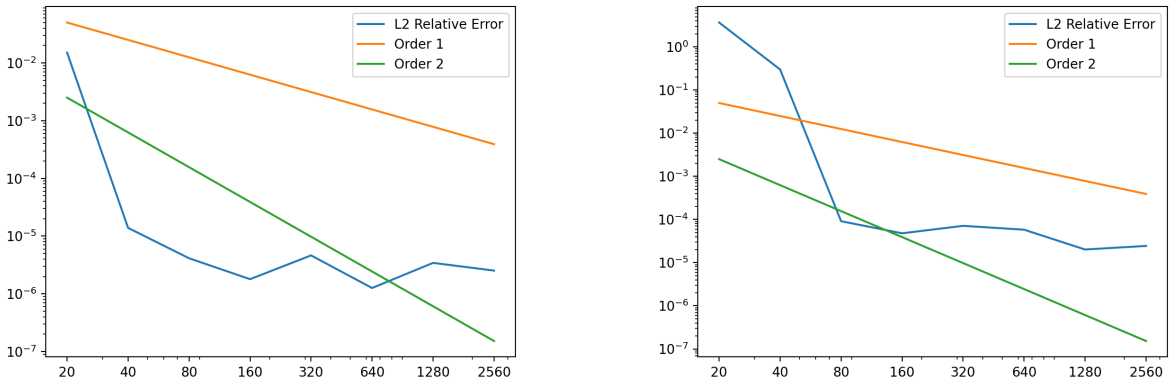
(a) Target solution and model approximation.

(b) Loss function evolution.

Figure 2: Best approximation of the multi-frequency solution by the third model trained with 2 hidden layers, 400 neurons per layer and 320 residuals. On the left, we can fully appreciate the almost perfectly overlapping plots of the network and the exact solution, while on the right we see the entire loss function evolution in semi-logarithmic scale, passing from nearly $1e6$ to a final value close to $1e1$.

In addition to the networks produced for the aforementioned multi-scale experiment, we eventually trained a new tranche of shallow networks embedded with all the other settings tuned as for the successful model that we presented just above (400 neurons per layer and 320 residuals), augmenting by a factor of ten the total number

of learning iterations to be performed for the approximation of the exact same solution target by means of system (1). These additional networks have been trained in order to make a fair comparison, in terms of total learning time exploited, between shallow and deep models. The latter, which were the only ones that obtained satisfactory approximation results for the mentioned test case, actually took an overall computational cost of about ten times larger than the former (given the same number of optimization iterations). The best model resulting from these newly-constructed trials shows a relative L^2 error that is very similar to the one presented by the candidate visualized in Figure 2, with a value of 2.4%.
Overall, we can confidently ascertain that shallow networks almost always provide a higher reliability in emulating single-scale or multi-frequency solution targets, whether this holds with an exact equal number of learning iterations or with the a similar computational time at their disposal. In particular, we observe a generally higher stability for the models endowed with a single hidden layer.



(a) Single-scale best trend plot.

(b) Multi-scale best trend plot.

Figure 3: Best error plots for the single-scale (left) and the multi-scale (right) convergence analyses (with respect to the number of uniformly distributed residual points employed) visualized in semi-logarithmic scale. The linear and quadratic orders lines are also represented.

Figure 3 above illustrates the best L^2 error trend for the experimental convergence analyses performed, respectively, over the low frequency target ($\sin(\pi x)$) coming from the single-scale sensitivity study and the easiest multi-scale profile encountered in the homonym analysis, which consists in a weighted sum of all the sinusoidal functions belonging to the former. In this framework we fixed, for each of these two cases, a proper architecture that proved to be successful for the approximation of the respective target in the related sensitivity study, and we varied the total number of uniformly distributed residual points exploited inside the computational domain. The shown performances of PINNs tend to stabilize at saturation levels that should depend on the solution expression and on the chosen architectural features.

4. Saw-Teeth & Generic Tests

The present section can be essentially subdivided in two parts: the first focuses upon the approximation of a specific class of functions, the so-called saw-teeth waves, while the second is dedicated to the study of another generic set of targets, solutions to particularly renowned PDEs as the Heat equation or the Burger's differential problem. The former set of tests is composed by three trials, each dedicated to the emulation of three saw-teeth solutions for, respectively: the pure approximation problem (2), an ODE where we impose the first order derivative (3) and, finally, the one-dimensional Poisson equation with homogeneous boundary conditions (1). Still concerning the first three tests of this section, we understand that the functional expression of their exact solutions can be attained by exploiting proper compositions of ReLU Neural Networks, as shown in [6]. For this reason, in this framework we have decided to discuss also the results related to the approximations obtained by embedding our architectures with the ReLU activation function. Moreover, in light of the following theorem (from [6]), we necessarily expect shallow networks to behave poorly whenever they do not possess a minimum amount of neurons. For this particular class of solutions, deep architectures should provide the most accurate and reliable results, at least in principle (as it is shown in [6]).

Theorem 4.1. *Assume $\mathcal{N}$ to be a shallow network with one-dimensional input and output layers, embedded with the ReLU activation function and N neurons:*

$$\mathcal{N} \in \mathcal{S}(\sigma) := \left\{ \sum_{i=1}^{N} z_i \cdot \sigma\left(w_i \cdot x + b_i\right) + q : \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{b} \in \mathbb{R}^N, q \in \mathbb{R} \right\}.$$

Then, it holds that $\mathcal{N} : \mathbb{R} \to \mathbb{R}$ is a piece-wise linear map characterized by 2N linear pieces at most.

Consider the salient results of the first experimental test, where we impose the three saw-teeth waves illustrated below in Figures 4, 5 (ordered by stiffness, respectively characterized by 4, 8 and 16 teeth) by means of the pure approximation problem:

$$u = f \qquad x \in [0, 1] \tag{2}$$

Table 2 presents a series of expected trends for the approximation of the easiest saw-teeth target, similarly observed in the analogous board containing the results coming from the Tanh models. The first consideration concerns the number of training points: we clearly see that, in this case, the performance increases as soon as the former grows. With a few exceptions, we also appreciate a positive correlation between the accuracy and the number of neurons per layer employed (keeping the rest fixed). Deeper networks are able to provide the best results, also needing fewer neurons to achieve satisfactory performances (in agreement with Theorem 4.1).

NPL/HL	1	2	3
2	0.591705	0.527785	0.591710
4	0.527785	0.465181	0.466103
8	0.465180	0.415417	0.293268
16	0.465199	0.416296	0.293730
32	0.591730	0.327118	0.060990
64	0.439188	0.043996	0.058448
128	0.326080	0.029619	0.056192

(a) Best error, 20 residuals.

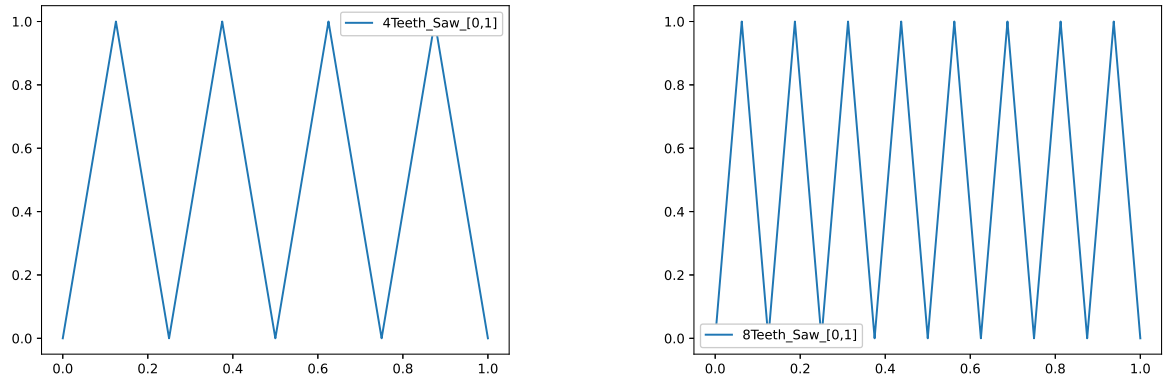
NPL/HL	1	2	3
2	0.592457	0.569203	0.661452
4	0.592457	0.569203	0.592457
8	0.465466	0.324929	0.389396
16	0.414214	0.007199	0.293054
32	0.418693	0.293176	0.018639
64	0.292910	0.022210	0.025200
128	0.009365	0.015527	0.026813

(b) Best error, 40 residuals.

NPL/HL	1	2	3
2	0.592484	0.569111	0.661452
4	0.569111	0.592484	0.437429
8	0.437422	0.464102	0.389281
16	0.389270	0.324899	0.005755
32	0.325007	0.292926	0.005905
64	0.292894	0.008450	0.007756
128	0.004502	0.004558	0.009267

(c) Best error, 80 residuals.

Table 2: Tables for the ReLU models trained to approximate the 4-saw-teeth solution.

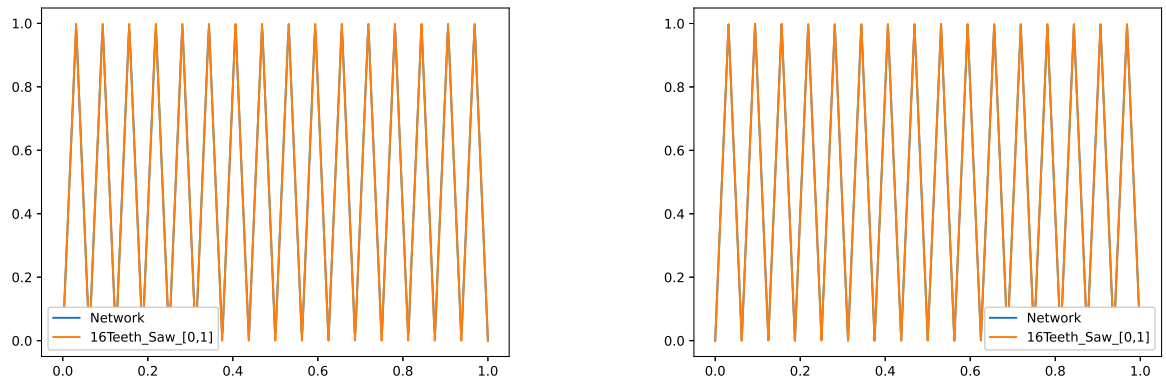


(a) 4-saw-teeth solution target.

(b) 8-saw-teeth solution target.

Figure 4: Representation of the 4-saw-teeth (left) and 8-saw-teeth (right) functional profiles over $[0, 1]$.

Figure 5 shows the best models obtained for the stiffest solution target of our single-scale sensitivity analysis ($\sin(15\pi x)$). Also for this case we appreciate an extremely high reliability for the represented networks.



(a) Target solution and Tanh model approximation.

(b) Target solution and ReLU model approximation.

Figure 5: Best approximations of the 16-saw-teeth solution by the first Tanh network trained with 3 hidden layers, 320 neurons per layer and 1000 residuals (left) and by the first ReLU network trained with the exact same architecture (right). Both show a very high level of accuracy, well below 1%.

Consider now the second of these three linked experimental tests, where instead of passing the exact point-wise values of the target function to learn it directly, we implicitly impose its derivative by means of the simple ODE:

$$\begin{cases} u' = f & x \in (0,1) \\ u = 0 & x = 0 \end{cases} \tag{3}$$

Analyzing the outcomes of this experiment we can immediately ascertain that the models embedded with the hyperbolic tangent function always succeed in reliably reproducing all three target profiles in at least one case, while ReLU architectures thoroughly fail. The notorious difficulties shown by the latter are seemingly related to the appearance of an effective differential term inside the PDE under study, as it happened for all ReLU networks belonging to the single-scale and multi-scale tests. All these situations share the modality in which the presented issue manifests itself: all the mentioned ReLU models, in fact, immediately fall inside the zero-solution local minimum of the cost functional at the beginning of their learning procedure.

NPL/HL	1	2	3
2	0.920594	0.727893	0.504348
4	0.514314	0.692048	1.290939
8	0.641434	1.708846	0.435716
16	0.674161	0.460413	0.403932
32	0.522378	0.459221	0.430089
64	0.510950	0.429403	0.398886
128	0.770445	0.448640	0.513728

(a) Best error, 20 residuals.

NPL/HL	1	2	3
2	0.610074	0.416505	0.665898
4	0.396115	0.049787	0.047099
8	0.059533	0.250497	0.184904
16	0.062870	3.073549	0.021354
32	0.044272	0.021754	0.016591
64	0.049279	0.023817	0.021972
128	0.051243	0.186686	0.040941

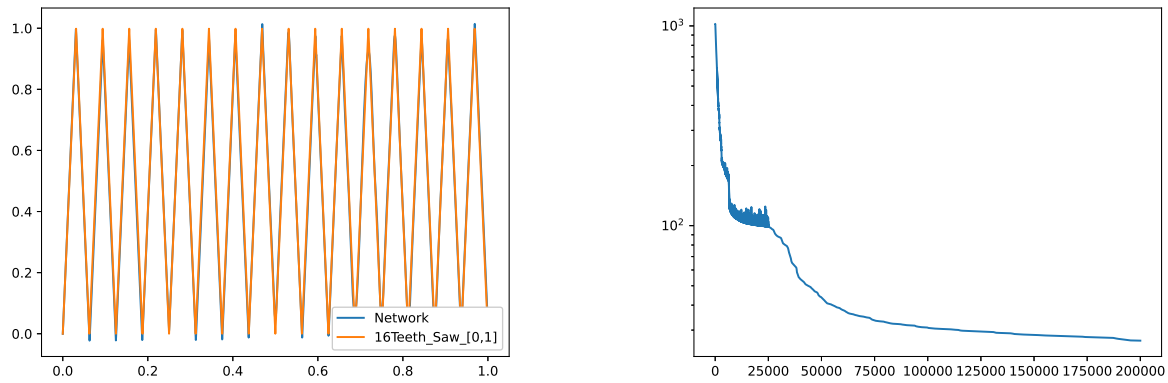
(b) Best error, 40 residuals.

NPL/HL	1	2	3
2	0.687386	0.597666	0.431485
4	0.364336	0.048274	0.091481
8	0.058471	0.036431	0.014043
16	0.047344	0.064007	0.077940
32	0.008358	0.010178	0.051681
64	0.007155	0.017330	0.074364
128	0.006418	0.145762	0.018758

(c) Best error, 80 residuals.

Table 3: Tables for the Tanh models trained to approximate the 4-saw-teeth solution to (3).

Table 3 reports the best relative L^2 errors for the Tanh architectures that have been trained to learn the 4-saw-teeth wave. We immediately recognize that 20 residual points are definitely not enough to obtain a reliable approximation of such target. By employing 40 training spots we appreciate better results for the deepest architectures, while this trend is inverted when we exploit 80 of them. The most stable and accurate networks of this test have been endowed with the maximum number of training points and they are characterized by a unique hidden layer. Under the described configuration we appreciate an increasing performance as the number of neurons is augmented, reaching an accuracy level that overcomes the 99% threshold. Figure 6 below illustrates the graphical representation of the best model obtained for the approximation of our stiffest saw-teeth wave (characterized by 16 teeth), alongside the related loss function evolution.



(a) Target solution and Tanh model approximation.

(b) Loss function evolution.

Figure 6: Best approximation plot of the 16-saw-teeth solution to (3) by the third Tanh network trained with 3 hidden layer, 160 neurons per layer and 1000 residuals (left) and its related cost functional evolution (right) in semi-logarithmic scale. The network shows lacks of accuracy in correspondence to several sharp corners of the exact solution, as we can clearly notice from the left graph. The cost function is still slowly decreasing at the end of the learning procedure.

We eventually consider the last experimental trial belonging to the first tranche of generic tests considered in our work, consisting in the implicit imposition of a smooth target by means of the one-dimensional Poisson equation coupled with homogeneous boundary conditions (1). Here we actually seek for an inexact approximation of the saw-teeth solutions with a regularization of their second order derivative. The latter, in fact, formally exist only in a distributional sense and cannot be alternatively imposed in strong form. The accuracy of such regularizations depend on the value of a positive parameter, ϵ: in its limit to zero, the exact solution of these approximated problems tends, in a distributional sense, to the original saw-teeth target profile. We should therefore expect to observe, in the related test, a non-disposable component of the error.
As in the previous test all ReLU networks suffer from the same issues that we have already explained, while some instances among the Tanh models are once again capable of grasping the essential features of the two simplest targets, the 4-saw-teeth and 8-saw-teeth waves, reaching a satisfactory level of performance for them. Table 4 gathers all the related results: we immediately see that, concerning the most difficult solution profile, we have not been able to train any successful network. Regarding the 4-saw-teeth wave function, instead, we appreciate an accuracy level that improves whenever we increment one of the following hyper-parameters (keeping the others fixed): number of hidden layers or amount of neurons per layer. We also notice that, in order to obtain satisfactory results for this target, we need fewer units inside the hidden layers whenever we augment the depth of the network. We finally highlight that only the widest (and deepest) architectural configuration is capable of reaching an acceptable level of precision for the 8-saw-teeth solution profile.

NPL/HL	1	2	3
2	1.208059	1.009891	2.350983
4	2.010388	2.356984	1.000177
8	3.824559	2.410723	3.362979
16	6.120269	3.605160	3.583601
32	0.497861	3.788019	0.025933
64	0.158427	0.026248	0.028666
128	0.039368	0.026866	0.030798
\	\	\	\

(a) 4-saw-teeth, 400 residuals.

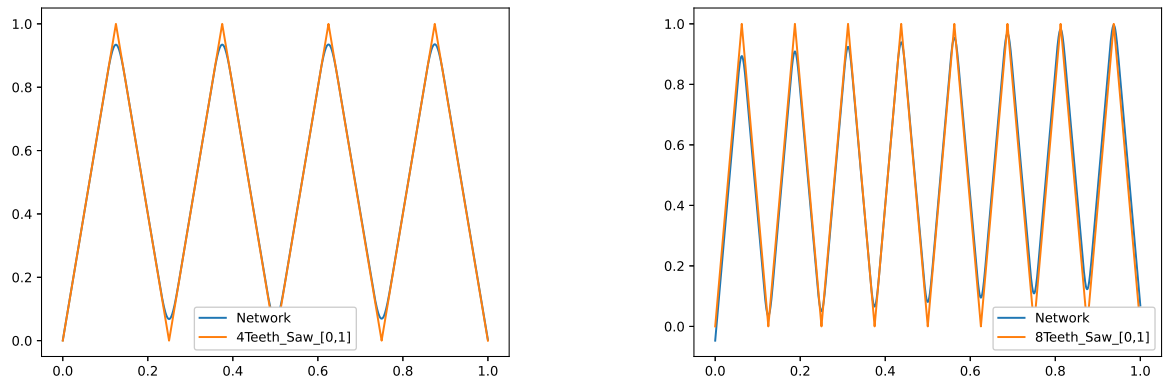
NPL/HL	1	2	3
2	1.214229	1.012062	2.414002
4	1.491775	2.523375	2.366880
8	2.952532	2.466374	4.157850
16	3.400309	6.170633	6.203755
32	5.977035	7.325827	2.065365
64	2.289498	8.055386	1.102825
128	0.870293	0.079352	6.258685
256	0.226520	0.175243	0.067705

(b) 8-saw-teeth, 800 residuals.

NPL/HL	1	2	3
5	0.541032	2.508024	2.605327
10	4.250449	3.224238	3.684910
20	5.817959	6.211009	13.806236
40	9.138324	3.032845	10.321249
80	7.907577	10.335821	10.610042
160	14.255289	23.221757	3.093698
320	6.860933	21.305189	14.751557
\	\	\	\

(c) 16-saw-teeth, 1600 residuals.

Table 4: Tables for the Tanh models trained to approximate the 4-saw-teeth (left), 8-saw-teeth (middle) and 16-saw-teeth (right) solutions to (3) with the relative number of internal points employed.



(a) 4-saw-teeth solution and model approximation.

(b) 8-saw-teeth solution and model approximation.

Figure 7: Best approximation plots of the 4-saw-teeth solution by the second Tanh network trained with 3 hidden layer, 32 neurons per layer and 400 residuals (left) and of the 8-saw-teeth solution by the first Tanh network trained with 3 hidden layers, 256 neurons per layer and 800 residuals (right). Both architectures present lacks of accuracy near all the sharp corners of the exact solution, where the regularization effectively takes place. Their entire loss evolution, not represented here, suggest that they expressed all the available potential during the relative learning phases.

Figure 7 shows the graphical illustration of the most successful networks that we managed to produce for the approximation of the 4-saw-teeth and 8-saw-teeth targets. We can distinctly notice that, in proximity of the sharp corners of the exact solution, namely where the inexact regularization takes place, the indefinitely differentiable functions represented by our Tanh networks show difficulties in emulating such a sudden change of derivative. We can confidently presume that the dominant component of the resulting errors is actually connected to this specific feature in these regions of the domain. The aforementioned inexact approximation of the differential problem consists in exploiting a proper weighted sum of Dirac deltas as the source term for the Poisson system (1), where we ideally impose the desired (up to regularization) second order derivative for the Neural Network to be trained. For a more complete investigation we have performed another attempt, in which we have decreased the absolute value of the regularization parameter ϵ by one order of magnitude to evaluate whether we can actually improve the approximation to the original solution in the limit formulation of the problem (for ϵ that tends to zero), where we should be able to retrieve, in principle, the correct saw-teeth profile. It actually turns out that the performance worsens as ϵ diminishes, leading us to believe that for small values of ϵ the source term appearing in the approximated formulation becomes too stiff to be interpreted.
As a general comment on the presented first tranche of generic tests, where the same solution targets have been imposed through three different differential systems, we can confidently affirm that we have been able to appreciate the role played by the depth of the exploited architectures, as we should have expected from the theoretical result that we previously reported (see Theorem 4.1). These ad-hoc experiments have also definitively disproved the capability of ReLU networks to understand even a set of particularly suited target solutions, whenever an effective differential term enters in the expression of the problem under investigation. As a matter of fact, we dedicated another significant generic test to the comprehension of such a pathological phenomenon. For such an experiment we still make use of the one-dimensional Poisson equation coupled with homogeneous boundary conditions (1), but this time we impose a bell-like profile as the exact solution of the problem. We conjecture two possible explanations for the failure linked to the ReLU-embedded architectures:

- Since this activation function identically evaluates to zero for the negative half-line of the real numbers, it might introduce sparsity in the gradient back-propagation inside the network.
- Differently from the hyperbolic tangent function, it has a discontinuous derivative in correspondence to the origin that may be the root of the unexpected behavior seen throughout our experiments.

In order to discover which (if any) of these motivations concur in manifesting the mentioned issue, we employed two additional activation functions for our models (other than ReLU and Tanh): the so-called LeakyReLU and SiLU. The former should be able to avoid the possible problem consisting in the phenomenon commonly known as dying gradient, related to the first point made above. The latter, instead, is an indefinitely regularized version

of ReLU, expected to overcome the possible issue connected to the discontinuous derivative in the axis origin. Tables 5 and 6 suggest that the correct hypothesis is actually the second one. The results concerning LeakyReLU and SiLU turn out to be very similar to the outcomes associated to, respectively, ReLU models and Tanh networks. We are therefore led to the conclusion that a discontinuity in the first derivative of the neural activation function is the main cause for the unexpected behavior that we have observed in this work whenever ReLU has been employed. If the reason for such failure had been related to the first hypothesis, we should have seen the opposite behavior in the outcomes that we gathered in the tables below. Figure 8 represents the best among all the architectures that have been trained for this experimental test, which also turned out to be extremely useful to understand the origin behind the aforementioned issue.

NPL/HL	1	2
10	0.799493	0.792292
20	0.818724	1.004341
40	0.828104	0.874943
80	0.916210	0.988886

(a) 10 residuals.

NPL/HL	1	2
10	0.688966	0.821222
20	1.020829	0.922211
40	0.899339	0.991628
80	0.968637	0.989566

(b) 20 residuals.

NPL/HL	1	2
10	0.781606	0.937788
20	0.774002	0.833073
40	0.969570	0.966451
80	0.988512	0.992599

(c) 40 residuals.

NPL/HL	1	2
10	0.992159	0.901548
20	0.854928	0.938832
40	0.862462	0.944870
80	0.865678	0.992672

(d) 80 residuals.

Table 5: Best error tables for the models trained with the LeakyReLU activation function.

NPL/HL	1	2
10	0.258267	0.379755
20	0.245813	0.119700
40	0.312816	0.175850
80	0.453008	0.109568

(a) 10 residuals.

NPL/HL	1	2
10	0.000147	0.000098
20	0.000206	0.000078
40	0.000454	0.000115
80	0.000299	0.000314

(b) 20 residuals.

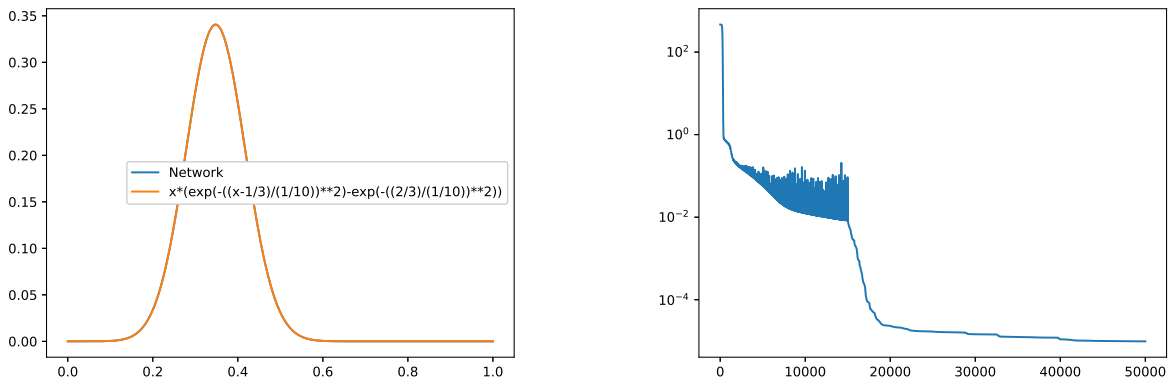
NPL/HL	1	2
10	0.000066	0.000103
20	0.000029	0.000020
40	0.000088	0.000043
80	0.000026	0.000030

(c) 40 residuals.

NPL/HL	1	2
10	0.000144	0.000046
20	0.000116	0.000023
40	0.000049	0.000010
80	0.000016	0.000030

(d) 80 residuals.

Table 6: Best error tables for the models trained with the SiLU activation function.



(a) Target solution and SiLU model approximation.

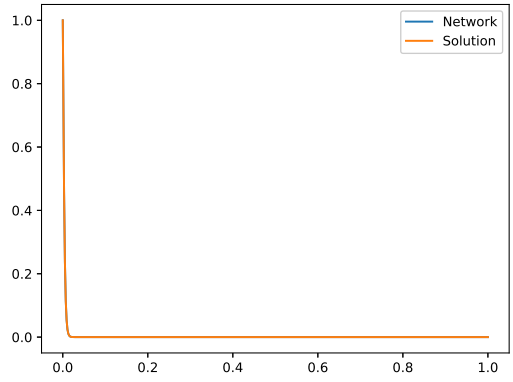
(b) Loss function evolution.

Figure 8: Best approximation plot of the target solution to (1) by the first SiLU network trained with 2 hidden layers of 40 neurons each and 80 residuals (left) and its related cost functional evolution (right) in semi-logarithmic scale. Given the final flat shape of the latter after all the available LBFGS learning iterations were performed, we can confidently ascertain that convergence has been reached by this model. From the former, we see that the network and the exact solution plots basically overlap.

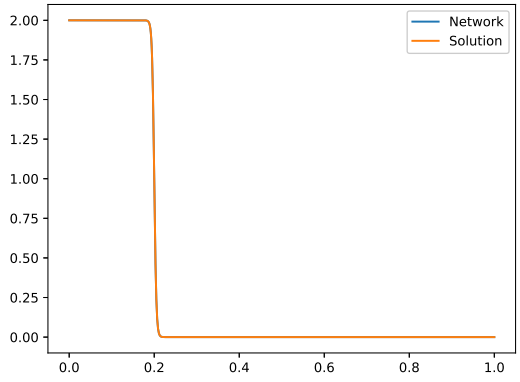
We have selected, among all the other generic experimental tests that have been performed in this framework, the one that presents the most difficult target, corresponding to the stiff solution to the following system:

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} - \nu\dfrac{\partial^2 u}{\partial x^2} = 0 & (x,t) \in \Omega = [0,1] \times [0,1] \\ u = g & (x,t) \in \Gamma \subset \partial\Omega \end{cases} \tag{4}$$

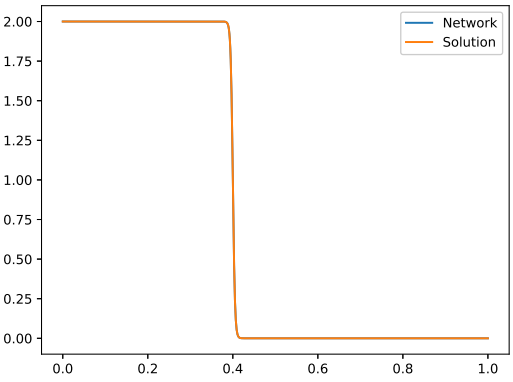
This differential problem (inspired by [4]) is also known as the regularized Burger's equation, and appears in different fields of physical interest. Our Dirichlet condition involves the analytical expression for the solution of our system corresponding to a value for ν which is set equal to $2.5e-3$, namely $g(x,t) = 1 - \tanh(200(x-t))$. Since we are dealing with a parabolic problem, the boundary conditions are only imposed over the homonym frontier of the computational domain: $\Gamma = \{(x_b, t)\ \forall t \in [0,1] : x_b \in \{0,1\}\} \cup \{(x,0)\ \forall x \in [0,1]\}$.
As we can see from Figure 9 below, representing the best network obtained for this trial alongside a six time-frame evolution of the target profile, the exact solution is mainly characterized by an extremely sharp interface that lies in proximity of the spacetime line $t = x$. Despite the stiffness that characterizes the exact solution for this differential problem, we appreciate a stunning precision for our best approximation, which presents a relative L^2 error that stays below the 0.01% threshold. It is worth remarking that, for this experiment, we have been able to obtain accurate outcomes with the two largest shallow architectures employed (embedded with 50 or 100 neurons) and with the smallest double-layered models (characterized by only 25 neurons per layer). Moreover, we crucially notice that a minimum number of uniformly distributed point is necessary in order to obtain satisfactory performances. Considering the stiff nature characterizing the solution of this problem, such fact should not surprise our reader: including more training points close to the sharp interface of the target profile should in fact lead to better overall approximations.



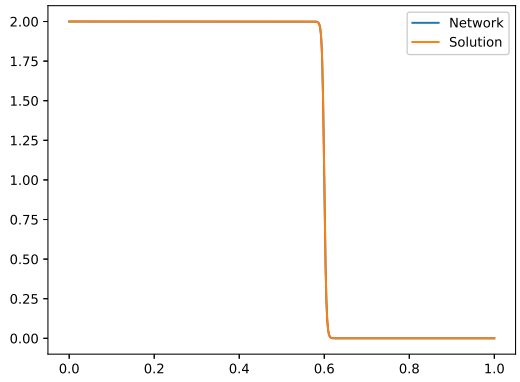
(a) Target solution and network approximation, $t = 0$.



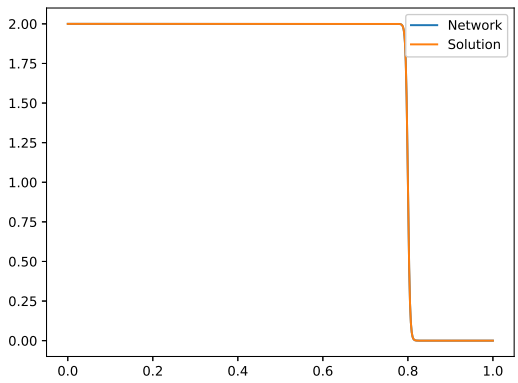
(b) Target solution and network approximation, $t = 0.2$.



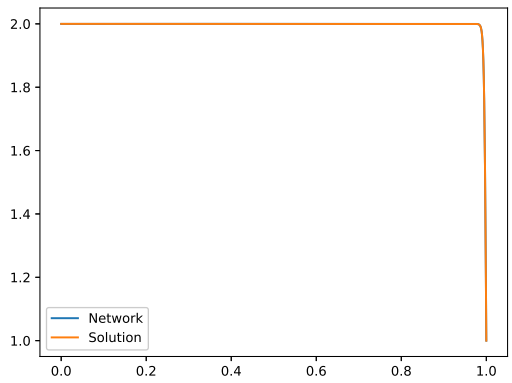
(c) Target solution and network approximation, $t = 0.4$.



(d) Target solution and network approximation, $t = 0.6$.



(e) Target solution and network approximation, $t = 0.8$.



(f) Target solution and network approximation, $t = 1$.

Figure 9: Best approximation plots, at six uniformly-spaced time steps, of the target solution to (4) by the second Tanh network trained with 2 hidden layers of 25 neurons each and 800 residuals.

Concerning the other experiments carried out in this work, as for example the one-dimensional Heat equation and a particular case of the monodimensional Advection-Diffusion-Reaction system, it is worth mentioning that in all cases we have managed to reach an extremely high level of precision for at least one model trained with the hyperbolic tangent activation function. ReLU networks, on the other hand, failed substantially everywhere. Inspecting the overall results that we have obtained throughout all the sensitivity analyses performed for the basic version of the PINN, where no hyper-parameter is changed during the learning phase of the models, we can consider the Tanh networks as a generally reliable and successful tool that has the potential, if endowed with a proper set of architectural features and initialized with an adequate structure, to resolve Partial Differential Equations in the explored numerical framework.
We shall eventually linger on the outcomes concerning our generic experimental convergence test, performed in the same fashion as for the analogous studies seen for the single-scale and multi-scale contexts by fixing a reliable structure for the networks to be trained and accordingly varying in a specific selected range the number of residual points in use. The underlying PDE is the one-dimensional Poisson equation with homogeneous boundary conditions (1), where we impose the bell-like profile that we encountered before as the exact solution of our differential problem. As we can appreciate from Figure 10, also for this test we observe the formation of a plateau for the best performances reached by these models. This trend, indeed, had already emerged in our previous convergence analyses: the experimental conclusion that we infer on the basis of these outcomes consists in the fact that such phenomenon appears to be a characteristic intrinsic feature of PINNs.

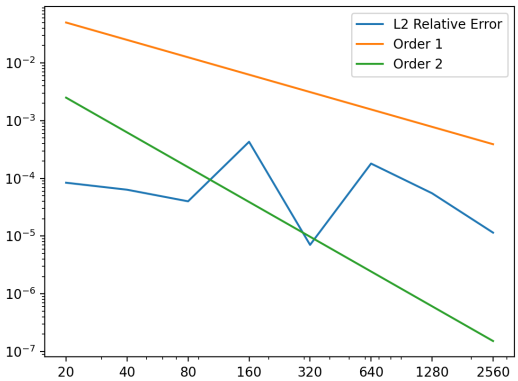


Figure 10: Best error trend for the basic PINN generic convergence analysis (with respect to the number of uniformly distributed residual points employed) for (1) visualized in semi-logarithmic scale. The linear and quadratic orders lines are also represented.

5. Adaptive Tests

The original contribution of this work consists in the development and testing of an innovative adaptive version of the PINN structure, where the hyper-parameters of the networks might be modified during the execution of their learning procedure. The main aim related to this new numerical tool is twofold: first of all having at our disposal an algorithm which is able to understand the correct architectural configurations for the problem at hand, pinpointing the right number of hidden layers, the proper amount of neurons for each of these and a convenient number of residual points to be employed, as well as their location inside the computational domain. Secondly, it is needless to say that we would like to keep the original utility associated to the basic version of the PINN, which consists in finding a reliable functional expression for the solution underlying the studied PDE. Many proposals have already been advanced in literature on this subject: here, however, we focus on the mere architectural properties of our networks, concerning the features that we mentioned above. With this goal in mind, we employ the renowned Growing Method (see [1]), coming from the general mathematical branch of Machine Learning, alongside a revisited version of the so-called Residual Adaptive Refinement technique introduced in [4]. The former acts on the number of hidden layers and the related amount of neurons (i.e. the network's structure), while the latter is responsible for the selection of the most appropriate location of the residual spots that should be employed during the learning phase of our models. It is worth noticing that all the tests performed with our basic PINNs have been carried out by employing a uniformly sampled training set: this has been done to ensure consistency and reproducibility of our experiments. The two previously mentioned techniques are jointly exploited because they are supposed to intervene for the same fundamental reason: they are, indeed, both useful whenever the current architecture appears to be unable in grasping the essential nature of the solution target, a situation that is encountered when its complexity is excessively high for the potential capacity of our model. Put simply, we expect to need more training points and larger structures in order to be able to interpret stiffer solution profiles, but at the same time we should not overestimate the required architectures in order to avoid possible negative effects such as overfitting or larger generalization errors. In some of our basic experiments, indeed, we observed better performances related to shallow networks, while in other tests we saw that multi-layered models were able to provide more accurate results.

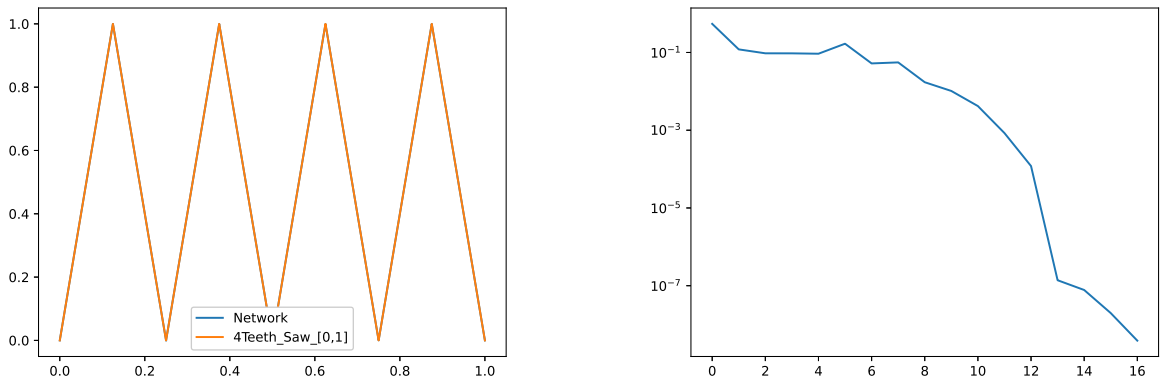
In relation to our Growing algorithm, it is important to mention that our adaptive models base their structure on a newly conceived type of Feed Forward Neural Network, characterized by special neurons, endowed with the identity activation function, on top of its hidden layers. This feature might be actually exploited in all branches of Machine Learning that can benefit from the Growing technique. Without the need of entering into the related details, the deployment of the mentioned architecture combined with any random zero-mean initialization of the additional weights introduced by the Growing algorithm allows to retain, on average, the functional map of the model even after we have enlarged the network by adding a new hidden layer at the back of its structure.

Regarding the application of the Residual Adaptive Refinement Method, we specify that the selection of the supplementary points is preceded by an evaluation of the PDE residual over the so-called residuals pool, an extremely numerous and dense set of internal spots that is also involved for the computation of the general performance of the network at each adaptive cycle. The residual points that are effectively added clearly present the worst (highest) values for the mentioned numerical indicator.

The formal criterion upon which we base the intervention of the Growing Method and the Residual Adaptive Refinement technique is guided by the evaluation of the ratio between the values of the overall PDE residual over the training and test sets. The philosophy underpinning our algorithm consists in prioritizing the achievement of a satisfactory level of accuracy with the fewer possible amount of hidden layers, starting from a shallow model and growing vertically (increasing the number of neurons) before passing to multi-layered structures. This design choice finds its roots in the experimental evidence that shallow networks prove to be much faster and computationally cheaper than wider architectures during the optimization phase.

We shall now briefly illustrate the results obtained for two among the tests conducted with the newly introduced adaptive PINN. For each we trained three networks, each prompted with a different random initialization seed and all the other specifications kept fixed.

Our first adaptive trial concerns the already encountered pure approximation problem 2, where we impose the 4-saw-teeth solution target. The obtained relative L^2 errors are all very accurate, with two of them that are particularly worthy (with a value of about 0.1% and 0.01%). Figure 11 shows the overlapping plots for the exact solution and the best ReLU model that we achieved alongside the corresponding loss function evolution. The cost functional is evaluated by exploiting the dense residual pool (in addition to the usual fixed boundary points) at the end of every adaptive cycle. For the illustrated model, a total of 16 learning periods were actually executed. Comparing the best networks observed in the counterpart test performed during the basic PINN sensitivity analysis with our best adaptive architectures, we appreciate a better accuracy here. Notice that the final number of residuals for our best network amounts to 2560, much more than the ones used for the basic PINN counterparts. A minimum of 10 learning cycles were forced for this experiment.



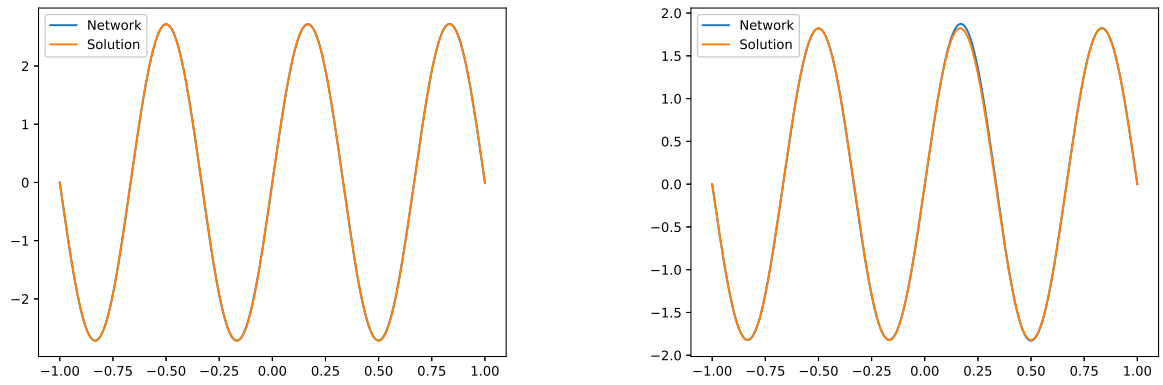
(a) Target solution and model approximation. (b) Loss function evolution.

Figure 11: Best approximation plot of the target solution by the second trial adaptive PINN embedded with ReLU (left) and its related cost functional evolution (right) in semi-logarithmic scale.

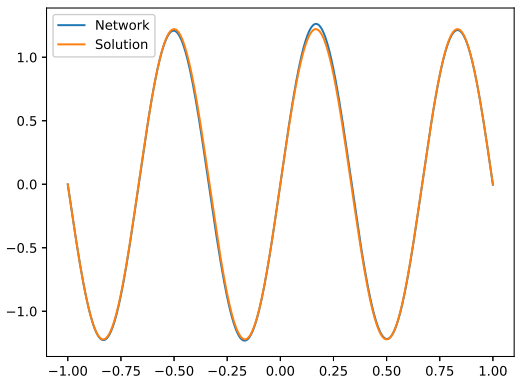
The last adaptive experimental test that we present is inspired to one of the trials performed for the basic PINN sensitivity analysis, where we imposed a sinusoidal target solution with exponentially decaying magnitude for the Heat differential problem reported just below (where $\Gamma = \{(x_b, t)\ \forall t \in [0,1] : x_b \in \{-1,1\}\} \cup \{(x,0)\ \forall x \in [-1,1]\}$ is the parabolic frontier of the spacetime domain $\Omega = [-1,1] \times [0,1]$):

$$\begin{cases} \dfrac{\partial u}{\partial t} - \nu \dfrac{\partial^2 u}{\partial x^2} = 0 & (x,t) \in \Omega \\ u = g & (x,t) \in \Gamma \subset \partial\Omega \end{cases} \tag{5}$$

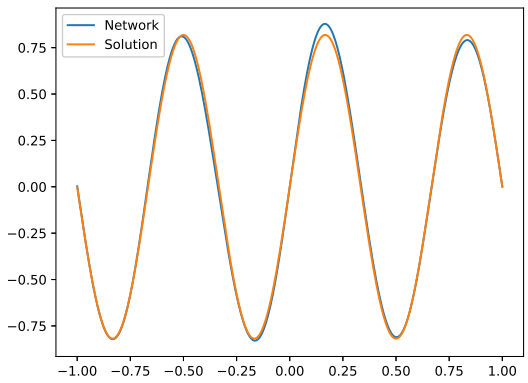
The performances shown by all the three trained instances are very similar to one another, with a relative L^2 error that hovers around 2%-3.5%. These models converged to the same final shallow structure, formed by 40 neurons in total. Rather uniquely, they all present the same number of residuals (2560 in total) as well. Although we have obtained satisfactory results for this trial, we can still observe better performances from the networks produced during the counterpart test executed for the basic PINN sensitivity analysis, in which we even employed a lower number of training spots. Figure 12 shows a visual illustration of a six time-frame evolution of the best network's graph next to the sought sinusoidal solution target, which presents an amplitude that decays exponentially over time. As we can also see from these plots, the represented curves basically overlap for small time values (hence close to the initial condition) while they progressively detach from one another as time increases, especially in correspondence to the second peak of the target solution.



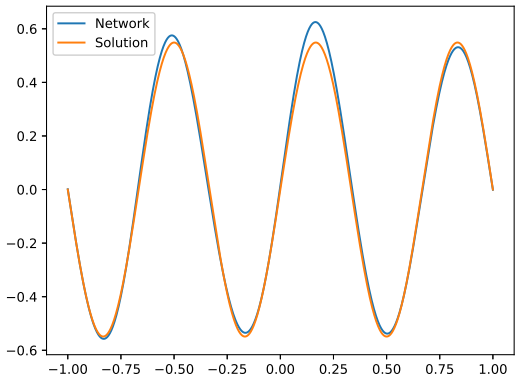
(a) Target solution and network approximation, $t = 0$. (b) Target solution and network approximation, $t = 0.2$.



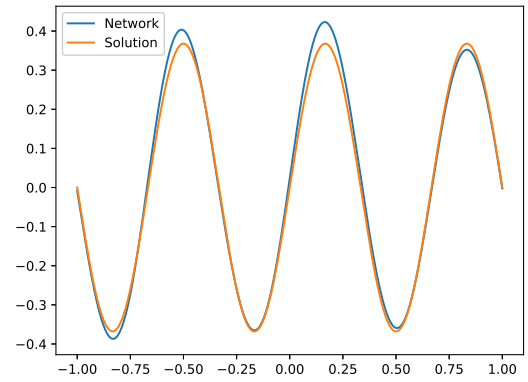
(c) Target solution and network approximation, $t = 0.4$.



(d) Target solution and network approximation, $t = 0.6$.



(e) Target solution and network approximation, $t = 0.8$.



(f) Target solution and network approximation, $t = 1$.

Figure 12: Best approximation plots, at six uniformly-spaced time steps, of the target solution to (5) by the first network trained with the hyperbolic tangent activation function.

Concerning the other (not reported here) adaptive tests performed, we can draw a few important comments. The bell-like profile target, already encountered in one of the trials studied for the basic PINN sensitivity analysis, has been imposed as the solution of the one-dimensional Poisson equation with homogeneous boundary conditions (1) also in this framework. For its approximation we obtained good overall performances, that however do not reach the same level of accuracy that we managed to achieve in the previous case (with our basic PINNs). Another experiment that has been carried out also in this context is the one involving the so-called Burger's regularized differential problem, for which we imposed the same solution that we encountered in the previous basic PINN study. In this test we noticed the appearance of similar difficulties with respect to the ones that we have just observed for the models trained to learn the Heat equation's solution.
Before drawing the conclusions of our work, we highlight another important potential use for our adaptive algorithm. The latter, indeed, might be extremely useful even when it does not provide satisfactory levels of accuracy for the solution of the problem at hand: this might be actually exploited for the individuation of a convenient architecture to be employed for the numerical resolution of a certain differential problem (including the number and location of the residual points inside the computational domain), even when the network does not manage to grasp the essential features of the solution (presenting a high loss function value). In such a way, we can at least discover a valid architectural configuration that we can feed to a basic (with a fixed structure) PINN, which should then be accordingly initialized and trained to learn the solution in a more reliable manner. The results exposed in this framework are to be intended as simple trial outcomes of our newborn algorithm, and all the consequent considerations must be accordingly taken with a grain of salt. It is important to contextualize their significance in terms of mere initial experimental attempts in a new and possibly promising direction for the development of PINNs, that nonetheless still remains unpretentious in its current form.

6. Conclusions

In this work we have explored the performances of PINNs over a set of predefined test cases with the aim of assessing their sensitivity (measured through the relative L^2 error) with respect to, on one hand, the complexity of the target solutions underlying the studied problems and, on the other, the choices concerning the architectural properties of our networks. First of all, we saw that it is not necessarily true that deeper models perform better than shallow networks. Secondly, we highlight that some of the most common trends (which, nevertheless, are not valid in general presumably due to effects linked to the so-called generalization error) consist in the improvement of performances when we increase the number of neurons per layer or the amount of residual points inside the computational domain. We remark that, in some cases, no clear patterns emerge from the gathered outcomes of our experiments. Along the way we also discovered that indefinitely regular activation functions prove to be much more reliable than irregular profiles such as ReLU, conjecturing and verifying a possible explanation for the failure associated to the latter. Moreover, increasing the maximum order of derivative appearing in the PDE seemingly leads to stiffer and more unstable learning procedures. Overall we can acknowledge success for the presented tool, because for all the trials mentioned up to now we have always been able to find successful instances that provided satisfactory levels of accuracy.
The second part of this project has been dedicated to the introduction and testing of an adaptive version of the PINN, whose aim, other than seeking for a reliable approximation of the solution of the differential problem at hand, consists in searching for the "optimal" architecture for the interpretation of the mentioned target function. In order to do so we employed a revisited version of the Residual Adaptive Refinement technique (introduced in [4]) alongside the Growing Method, all applied to a simple but innovative modification of the usual Feed Forward Neural Network structure. We finally performed preliminary experiments of this adaptive algorithm over a pool of test cases encountered in the first part of the thesis. Some of them turned out to be completely successful, while other still show clear signs of improvements to be made. With that being said we must ascertain that, after all, thanks to the smaller computational cost attached to our new scheme, we are left with promising margins for potential improvements. In order to avoid possible misunderstandings, we feel free to recognize that even in this prime version of the algorithm, whenever we did not achieve completely satisfactory performances, our scheme has nonetheless shown to be partially useful in understanding a convenient structure that could be embedded in a PINN to achieve appreciable results.
Still on this subject, the path drawn by our basic PINN analyses should motivate the need of an alternative road for the resolution of differential problems, if we intend to pursue the usage for this category of techniques. Indeed, the latter (basic) framework generally requires a potentially enormous amount of simulations to be able to reach, if this is even possible, a good model for the description of the phenomenon under study. Indeed, every time a new architectural parameter is considered, the number of trials that must be performed to explore all the possible settings, by varying singularly each parameter in its own range, grows exponentially fast. As an immediate consequence, this framework cannot be considered scalable at all.

6.1. Further Developments

Possible extensions of the present work are:

- Exploit the so-called Pruning techniques (see [3]), for which the main utilities are already included in our library, for the adaptive PINN strategy. This proposal should follow the dual heuristical reasoning that has been pursued in the present work, starting from a very large architecture (that should exhibit overfitting) and progressively cutting the redundant connections of the network (according to a proper criterion that must be chosen) until an optimal performance is reached.
- Modify or enhance our adaptive algorithm with new features such as the adaptive activation functions or the introduction of adaptive weight multiplying the loss function's single terms.
- Exploit PINNs for the resolution of inverse problems or to solve integro-differential equations with the features that have been exposed here.

References

[1] Marcus Frean. The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks. *Neural Computation - NECO*, 1990.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

[3] Yann Lecun, John Denker, and Sara Solla. Optimal Brain Damage. 1989.

[4] Lu Lu, Xuhui Meng, Zhiping Mao, and George Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 2021.

[5] Maryam Najafabadi, Taghi Khoshgoftaar, Flavio Villanustre, and John Holt. Large-scale distributed L-BFGS. *Journal of Big Data*, 2017.

[6] Christian Petersen. Neural Network Theory. 2020.

[7] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks, 2020.