The role of adaptive activation functions in Fractional Physics-Informed Neural Networks

C. Coelho,[1, a)] M. Fernanda P. Costa,[1, b)] and L.L. Ferrás[2, c)]

[1)] *Centre of Mathematics, University of Minho, 4710 - 057 Braga, Portugal*
[2)] *Department of Mechanical Engineering - Section of Mathematics, FEUP - University of Porto, Portugal*

[a)] *Corresponding author: cmartins@cmat.uminho.pt*
[b)] *Electronic mail: mfc@math.uminho.pt*
[c)] *Electronic mail: lferras@fe.up.pt*

Abstract. In this work, we use adaptive activation functions for regression fractional physics-informed neural networks (fPINNs) to approximate nonsmooth solutions. The adaptive activation function has better learning capabilities than the traditional one because it improves the convergence rate and solution accuracy. Our simulation results show that the adaptive parameter contributes less to the improvement of the results as the singularity becomes more strong (α decreases), because the errors incurred from the discretization and optimization of the loss function become dominant.

INTRODUCTION

Fractional Physics-informed neural networks (fPINN) [1, 2, 3] are neural networks trained to solve supervised learning tasks while obeying arbitrary physical laws described by fractional partial differential equations.

In this work we aim to analyse the effect of adaptive activation functions on the convergence rate and solution accuracy of fPINNs. For this purpose, we consider the numerical solution of the fractional Laplacian equation (FLE),

$$(-\Delta)^{\alpha/2}u(x) = f(x), \quad u(0) = u(1) = 0, \quad x \in (0,1),\ \alpha \in (1,2) \tag{1}$$

where $(-\Delta)^{\alpha/2}$ is the fractional Laplacian operator (see [4] for more details).

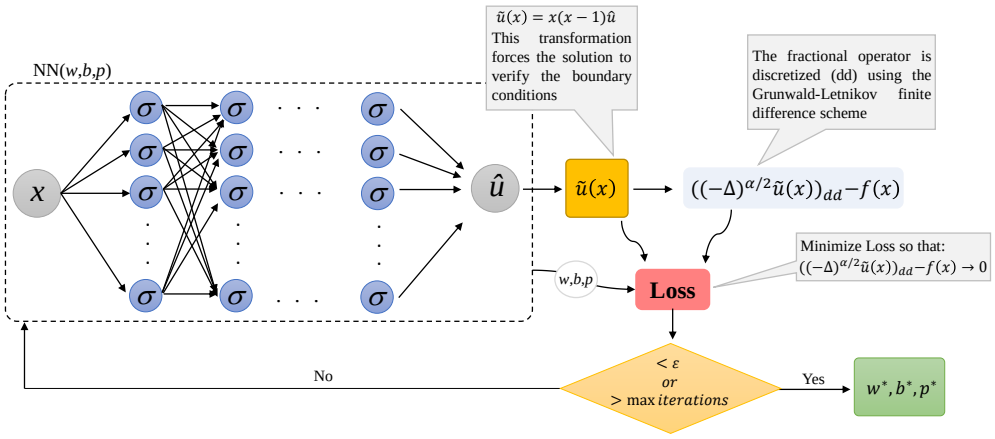

FIGURE 1. Schematic of the methodology used in the fPINN method for the solution of differential equations using fractional (integral) operators. Since the boundary conditions are $u(0) = u(1) = 0$, we simply choose the surrogate model $\tilde{u}(x) = x(x-1)\hat{u}(x)$ to satisfy the boundary condition automatically, where $\hat{u}(x)$ is the output of the NN with weights w, bias b and parameters p.

To solve the 1D FLE using fPINN, we follow the schematic shown in Figure 1. The minimization of the loss function of fPINN [3] is performed over the training dataset $\{x_i : x_i \in (0,1),\ \ i = 1,...,N\}$, where this minimization process is called training. At the end of the training process, we expect to have a set of optimal weights and biases that allow our model to predict a good solution for each point in the test data set $\{x_j : x_j \in (0,1),\ \ j = 1,...,M\}$, where each $x_j \neq x_i$ for $i = 1,\ldots,N$. For the case of fractional differential equations, the classical chain rule is no longer valid and consequently automatic differentiation cannot be used. Therefore, the analytical expressions for the construction

of the loss function are obtained by discretizing the original operators (e.g., a finite integral can be approximated by a finite sum). In this work, we use the Grünwald-Letnikov finite difference scheme. In addition to the training and test data sets, another data set of points (called auxiliary points) is created to help in the computation of the fractional derivatives.

A neural network (NN) learns a mapping from the input to the output of a given training dataset by adapting the weights w^k and bias b^k of each k^{th} layer through a linear transformation, $x^k = w^k x^{k-1} + b^k$, where x^{k-1} is the output of the previous layer.

To learn more complex transformations, an activation function, σ, is applied to the linear transformation to give the NN nonlinear properties, $x^k = \sigma(w^k x^{k-1} + b^k)$. To add flexibility to the network, one can introduce a single trainable parameter p that is shared by the activation functions of the NN and optimised along with the weights and bias [5], $x^k = \sigma(p(w^k x^{k-1} + b^k))$. As the parameter value p varies, the slope of the activation function changes. If the activation function σ depends on the parameter p, σ is called an adaptive activation function.

However, there are different approaches to implementing adaptive activation functions, namely: p can be a single parameter shared by all NN activation functions [5], it can be one parameter per layer shared by all neurons of the same layer, or one parameter per NN neuron [6].

In [5] and [6], studies on the introduction of adaptive activation functions into the performance of PINNs are reported using a single parameter, one parameter per layer, and one parameter per neuron. Their results showed an increase in the performance of convergence and accuracy of PINNs.

In this work, we analyse the performance of fPINNs when one adaptive parameter per layer is used in solving the 1D Fractional Laplacian Equation with a non-smooth analytical solution. Moreover, we analyse the limitations of fPINNs when solving the FLE with a series of gradually smaller non-integer order α values.

RESULTS AND DISCUSSION

We consider the numerical solution of (1) with $f(x) = \left(2\cos\left(\frac{\pi\alpha}{2}\right)\right)\Gamma(\alpha+2)x$, where Γ is the gamma function . The analytical solution is not smooth, and is given by $u(x) = x(1-x^2)^{\alpha/2}$. As we have seen in an earlier work [4], this case is more extreme and should present greater difficulties in approximating the fractional operator (through the difference scheme) and in solving the equation by fPINN.

We considered a feed-forward NN with 5 hidden layers and a configuration of 128-64-32-16-8 neurons. Several activation functions were used, namely *tanh*, *relu*, *elu*, *selu*, *swish*, *sin*, *silu*, and *sigmoid*. In this paper, only the results for *tanh* and *silu* are presented since they give the best results for the problem at hand and show large differences when the adaptive parameters are introduced.

The adaptive versions of the activation functions are:

$$tanh_{adaptive}(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} \text{ and } silu_{adaptive}(u) = \frac{u}{1+e^{-u}}$$

where $u = p^k(w^k x^{k-1} + b^k)$. In the numerical experiments, the vector p was initialised as a vector of ones and the bias vector b was initialised as a null vector. Since the FLE is a NP -hard problem (non-linear, non-convex, non-smooth, multi-modal), we implemented a multi-start strategy to avoid being trapped in a local optimal solution. In this work, we considered 5 different initial weights generated by 3 different distributions: Glorot Normal with mean and standard deviation given respectively by 0 and $\sqrt{\frac{2}{n^k+n^{k-1}}}$ (n^j is the number of neurons in the layer $j = k-1, k$) and uniform distributions given by $U(-1,1)$ and $U(-2,2)$.

In our previous work [4], we found that the Limited-memory BFGS (L-BFGS) method is the optimization algorithm that achieves the best results with less computation time, when solving FLE with fPINNs. Therefore, we also used L-BFGS in this work with its default settings in all training configurations. During training, we found that L-BFGS stops due to the relative reduction of the loss function to a value less than or equal to $2.22e-09$. To evaluate the models, the relative error L_2 was calculated.

The results obtained are summarised in Table , where for each α value the best relative L_2 error from the 5 runs for each weight initialization (with and without adaptive parameters) is given.

The NNs with the adaptive parameters in the activation functions achieved better performance and faster convergence, even though the number of training variables increased. This behaviour was already seen with the PINNs [5, 6].

		Best L_2 Relative Error			
Initialization	α	Tanh	Tanh$_{adaptive}$	Silu	Silu$_{adaptive}$
Glorot Normal	1.9	0.009703334	0.008445772	0.013962873	0.008224198
U (-1, 1)		0.045120027	0.029161492	0.09528824	0.016481573
U (-2, 2)		1.247241	0.90385956	0.513845	0.025741622
Glorot Normal	1.8	0.017017787	0.019129587	0.020696983	0.0178637
U (-1, 1)		0.026224619	0.021309556	0.06786978	0.07207132
U (-2, 2)		0.20724408	0.19052675	8.201091	0.074855186
Glorot Normal	1.7	0.021984076	0.021637242	0.031672735	0.027998013
U (-1, 1)		0.052932385	0.04272195	0.092276946	0.026273258
U (-2, 2)		0.38647774	0.13118427	6.187943	0.20918174
Glorot Normal	1.6	0.02594118	0.021637242	0.029716883	0.036974136
U (-1, 1)		0.06875729	0.04568334	0.103585996	0.026273258
U (-2, 2)		0.45244825	1.1522498	0.38301727	0.10194409
Glorot Normal	1.5	0.028199576	0.030122932	0.032759454	0.03203833
U (-1, 1)		0.07265402	0.050882217	0.14441098	0.044384737
U (-2, 2)		0.44004643	0.33642536	1.5502751	0.15884954
Glorot Normal	1.4	0.035398327	0.037240285	0.035142127	0.039704762
U (-1, 1)		0.06461266	0.08425152	0.04308845	0.037726227
U (-2, 2)		0.28789306	0.2866603	1.8212881	0.12640741
Glorot Normal	1.3	0.04562477	0.04782937	0.04611545	0.04798889
U (-1, 1)		0.06730618	0.094012655	0.07204201	0.061122436
U (-2, 2)		0.6169999	0.25393987	0.46321905	0.17878965
Glorot Normal	1.2	0.06352478	0.06425263	0.065306604	0.06555646
U (-1, 1)		0.073436975	0.08922008	0.09551617	0.07524242
U (-2, 2)		0.27006406	0.5440407	0.29246098	0.111379206

TABLE I. Performance metrics of the best model obtained by solving the FLE for different α values, initialising the weights and activation functions with and without the layer-wise adaptive parameters.

Table shows that the ability of the NN to approximate the solution of the FLE decreases as α decreases. $\alpha = 1.3$ represents the current acceptable solution limit. This is because an $L_2 \geq 0.05$ leads to a very poor approximation to the true solution. The numerical solution deteriorates significantly when $L_2 \geq 0.05$. Thus, we can conclude that the introduction of the adaptive parameters leads to an improvement of the relative error L_2, but the obtained solution does not agree with the analytical solution at low α values. When the solution becomes more regular, for example, when $\alpha = 1.9$, the use of an adaptive parameter leads to an improvement of the L_2 error.

Initialising the weights of the NN with *larger* distributions did not improve the convergence of the NN. However, a major difference in the L_2 relative error is observed when using $U(-2,2)$ for the NN without and with adaptive parameters (especially when considering the results provided by *silu*). This shows that the adaptive parameters help to regulate poor initialization of the weights, which would otherwise lead to models with extremely poor performance.

CONCLUSIONS

The influence of introducing adaptive parameters (per layer) in the activation functions of fractional physics-informed neural networks to approximate non-smooth solutions of FLE was investigated. Different values of α were used to see the limitations of fPINNs in the presence of stronger singularities. For each α, different combinations of hyperparameters were used for training the NN, namely 3 strategies for initialising the weights and 2 activation functions *tanh* and *silu* and their adaptive versions.

The adaptive parameters improved the performance of fPINNs for both activation functions and allowed the network to converge to an optimal point faster.

As expected, the performance of the NN solution decreases with decreasing α. The errors incurred from the discretization and optimization of the loss function become dominant.

The use of adaptive activation functions in the presence of strong singularities is meaningless, since fPINNs cannot provide a reasonable solution. This is due to the strongly non-convex loss function, which causes the optimization algorithm to get trapped in local minima. Recall that in this case we cannot use automatic differentiation to compute the loss function (the classical chain rule does not apply here). Also, the discretization error introduced by the

approximation of the non-integer derivatives contributes to the poor performance of the NN solutions. Instead, a discretization should be used that takes into account the strength of the singularity.

Attempts to circumvent bad local minima by using different strategies to initialise the weights with larger ranges of values were unsuccessful. Although, a poor choice of weights initialization can be attenuated with adaptive activation functions, as the results show large differences in relative error L_2 when using $U(-2,2)$.

A Jupyter notebook with modified code to replicate the experiments is available at:
https://github.com/CeciliaCoelho/icnaam2022.git

ACKNOWLEDGMENTS

The authors acknowledge the funding by Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) through CMAT projects UIDB/00013/2020 and UIDP/00013/2020.
C. Coelho would like to thank FCT for the funding through the scholarship with reference 2021.05201.BD.

REFERENCES

1. L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Review **63**, 208–228 (2021).
2. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics **378**, 686–707 (2019).
3. G. Pang, L. Lu, and G. E. Karniadakis, "fpinns: Fractional physics-informed neural networks," SIAM Journal on Scientific Computing **41**, A2603–A2626 (2019), https://doi.org/10.1137/18M1229845.
4. C. Coelho, M. F. P. Costa, and L. Ferrás, "The influence of the optimization algorithm in the solution of the fractional laplacian equation by neural networks," in *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (ICNAAM-2021)* (AIP Conference Proceedings, accepted).
5. A. D. Jagtap and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," Journal of Computational Physics **404**, 109136 (2020).
6. A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **476**, 20200334 (2020).