JOURNAL OF APPLIED ENGINEERING SCIENCES
ISSN: 2247-3769 / e-ISSN: 2284-7197

sciendo

VOL. 15(28), ISSUE 2/2025
ART.NO. 456 pp. 219-226

ANALYSIS OF POWER-LAW FIN-TYPE PROBLEMS USING PHYSICS INFORMED NEURAL NETWORKS

M. Göçer [a, *], S.B. Coşkun [a], M.T. Atay [b]

[a] Kocaeli University, Faculty of Engineering, Civil Engineering Dept., 41380, Türkiye, E-mails: * mrtgocer@gmail.com, sb.coskun@kocaeli.edu.tr
[b] Abdullah Gül University, Faculty of Engineering, Dept. of Engineering Sciences, 38030, Türkiye, E-mail: ataymt@yahoo.com

Received: 17.06.2025 / Accepted: 02.07.2025 / Revised: 08.10.2025 / Available online: 15.12.2025

DOI: 10.2478/jaes-2025-0026

KEY WORDS: physics-informed neural networks, PINN, fin-type problems, nonlinear heat conduction, heat transfer, power-law.

ABSTRACT:

This study aims to model the temperature distribution in a single fin subjected to steady one-dimensional heat conduction with nonlinear thermal behavior. For the modeling and solution of the problem, the Physics-Informed Neural Networks (PINNs) architecture was used. The temperature-dependent heat conduction problem and the nonlinear boundary conditions of this problem were formulated with a differential equation. With the help of the PINN architecture, the loss function was minimized in order to reduce the difference between the true value and the predicted value. During this minimization process, the PINN architecture was forced to be consistent with the physical laws. The results obtained after training the PINN architecture exhibit successful performance in terms of accuracy and reliability when compared with the results in the literature. These findings highlight the potential of PINNs as a powerful alternative to conventional methods for solving complex nonlinear heat conduction problems.

1. INTRODUCTION

Extensive research has been conducted on heat conduction issues pertaining to fins and fin assemblies in which the heat transfer coefficient varies along the fin length (Kern and Kraus, 1972). The heat transfer coefficient, calculated based on the temperature difference between the surrounding fluid and the fin surface, is generally modeled in the literature using a power-law relationship. The temperature distribution along a fin (Fig.1) with a constant cross-sectional area and the thermal conductance at the fin base are described by a one-dimensional, steady heat-conduction equation written in nondimensional form as (Liaw and Yeh, 1994),

$$\frac{d^2\theta}{dx^2} - N\theta^m = 0 \ , \ 0 \le x \le 1 \qquad (1)$$

subject to the following boundary conditions:

$$\theta'(0) = 0 \ , \ \theta(1) = 1 \qquad (2)$$

In the Eq.(1) x represents the dimensionless coordinate along the fin length, and differentiation is taken with respect to x starting from the adiabatic tip. The dimensionless temperature is denoted by θ, the fin convection–conduction parameter by N, and the exponent m is defined according to the heat transfer mechanism. Typical values of m correspond to practical phenomena such as nucleate boiling, free convection, and thermal radiation. The analytical solution presented in (Sen and Trinh, 1986) includes a Dawson integral as referenced in (Liaw and Yeh, 1994), and incorporates a three-parameter hypergeometric function.

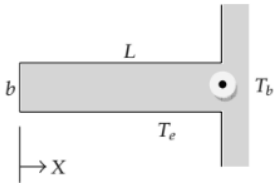

Figure 1. Geometry of a straight fin

Physics-Informed Neural Networks (PINNs) architecture (Raissi *et al.*, 2019a) was developed with the aim that, by integrating the physical laws in nature into deep learning architectures, the final architecture obtained from this integration can overcome the challenges of scientific computing. Compared to other methods, PINNs have produced satisfactory results in solving high-dimensional partial differential equations (PDEs) in terms of both their generalization capabilities and their approximation capabilities (Hu *et al.*, 2019). PINNs fill the gap of scientific computing difficulty in the literature and, by being adapted to various problems without the need for external data, facilitate progress in many fields (Shukla *et al.*, 2021). Applications encompass various domains, including fluid mechanics (Raissi *et al.*, 2019b, *Raissi et al.*, 2020 and *Zhu et al.*, 2023), solid mechanics (Bai *et al.*, 2023, Hao *et al.*, 2023, Ning *et al.*, 2023 and Haghihat *et al.*, 2023), materials science (Chen *et al.*, 2020, Fang and Zhan, 2020) and heat transfer problems (Cai *et al.*, 2021, Russel, 2023 and Madir *et al.*, 2024).

* Corresponding author: Murat Göçer, Graduate Student, Civil Engineering Department, Kocaeli University, e-mail: mrtgocer@gmail.com.

In this study, within the framework of PINNs, a neural network is employed to estimate the solutions of Eq.(1), articulated as

$$N_x[\theta] = 0, \quad x \in \Omega \tag{3}$$

$$\theta(x) = g(x), \quad x \in \partial\Omega \tag{4}$$

where N_x is the nonlinear differential operator; $x \in \mathbb{R}$, Ω and $\partial\Omega$ denote the computational domain and the boundary, $\theta(x)$ is the solution of Eq.(3) with the boundary condition $g(x)$.

The PINN architecture proposed by (Raissi *et al*., 2019a) consists of a neural network in which all neurons are fully connected. In this architectural structure $\theta(x)$ is estimated by taking x as the spatial coordinate input and returning $\theta_{NN}(x)$ as the output. This architecture, which consists of many hidden layers, takes as input at each layer ($X = [x_1, x_2, \ldots, x_i]$) and produces as outputs ($\theta = [\theta_1, \theta_2, \ldots, \theta_j]$) corresponding to these inputs. While the inputs are transmitted between neurons, they are computed as follows

$$\theta_j = \sigma\left(w_{i,j}x_i + b_j\right) \tag{5}$$

where $w_{i,j}$ represents the weight to be optimized, while b_j represents the bias to be optimized. The function $\sigma(.)$ in the equation denotes a basic nonlinear activation function. The parameters of the architecture are trained to minimize the sum of the loss functions

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_b \tag{6}$$

where

$$\mathcal{L}_r = \frac{1}{n_r}\sum_{i=1}^{n_r}\left|N_x\left[\theta\left(x^i\right)\right]\right|^2 \tag{7}$$

$$\mathcal{L}_b = \frac{1}{n_b}\sum_{i=1}^{n_b}\left|\theta\left(x^i\right) - g^i\right|^2 \tag{8}$$

In Eqs.(6-8), $\mathcal{L}_r$ and $\mathcal{L}_b$ penalize the residuals associated with governing equations and boundary conditions, respectively, while n_r and n_b represent the quantities of data points for various terms.

The variable $N_x[\theta(x)]$ is taken as input, and after the output is produced, derivatives with respect to the input are required to compute the residual error term, $\mathcal{L}_r$. This requirement for differentiation is handled by an intrinsic feature of the PINN architecture. Known as automatic differentiation, this capability applies the chain rule efficiently and operates reliably even for complex functions. Popular deep learning libraries such as TensorFlow (Abadi *et al*., 2016) and PyTorch (Paszke *et al*., 2017) incorporate automatic differentiation engines that can compute derivatives of arbitrary order. Consequently, the need for manual differentiation is eliminated, and the implementation of PINN architectures is significantly simplified.

Figure 2 presents a schematic representation of the PINN framework, illustrating its application to the governing equation. As shown in the figure, the first component involves approximating the solution $\theta(x)$ using a fully connected feed-forward neural network. The second component involves evaluating the governing equation by computing the required derivative d^2/dx^2 and applying the identity operator I to θ. The network parameters are adjusted by gradient descent optimization algorithm, in which the total loss is backpropagated through the model. Note that the collocation (residual) points used to compute the residual loss $\mathcal{L}_r$ may be sampled randomly over the space–time domain, and both their number and their positions can be chosen by the user.

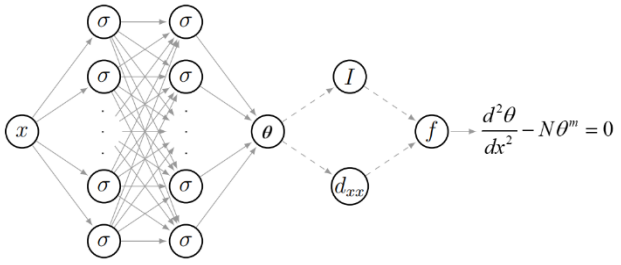

Figure 2. Conceptual PINN for the governing equation

In the present study, power-law fin-type heat-transfer problems are addressed using physics-informed neural networks (PINNs), and the study concentrates on the effects of the convective–conductive parameter N, and the power-law exponent m, which characterizes the mode of heat transfer in the fin. Various neural network architectures are employed, and corresponding loss functions are systematically evaluated to determine the most effective model configuration for accurately solving the problem.

2. THE ANALYSIS USING PINN

A PINN is not a conventional deep learning technique; rather, it integrates physical principles into the neural network framework. Specifically, it represents a specialized neural network architecture and training strategy designed to address supervised learning tasks. In addition, it yields satisfactory results in solving physical laws that can be modeled as differential equations (DEs).

The training of the network is guided not only by observed data but also by the physical laws modeled as differential equations and their boundary values. Thanks to the automatic differentiation feature, the PyTorch library was used and the partial differential equation (PDE) was integrated into the loss function. In this way, the network is trained to obtain solutions that inherently respect the governing physical laws. Such an approach is especially advantageous in scenarios with sparse data or when addressing complex, nonlinear differential equations.

The PINN model employed is a simple feed-forward NN (a `torch.nn.Sequential` model). It takes a single input feature and one or more hidden layers with `nn.Tanh()` activation functions. `Tanh` is a common choice in PINNs as its derivative is easy to compute, which is crucial for the physics-informed loss. The output layer is a single neuron producing a single output value. The NN architecture used in the study is described in the next section.

The core idea is to train the neural network to satisfy both boundary conditions and the governing equation. The total loss is a combination of two parts:

- Physics Loss: mean squared error of this residual of the governing equation.
- Boundary Loss: it evaluates the network at boundary conditions.

An Adam optimizer is used to update the network's weights and biases. 10,000 epochs are conducted with learning rate of 0.01

and a batch of random collocation points is generated within the domain [0, 1]. The total loss (physics loss + boundary loss) is calculated. To minimize the total loss value, the network parameters that need to be updated have their derivatives computed and stored by a method called backpropagation (`loss.backward()`), and are then updated. And finally, the optimizer updates the network parameters based on the calculated gradients (`optimizer.step()`).

3. NN ARCHITECTURES

3.1 Networks with One Hidden Layer

Two different NNs are employed for one hidden layer models. The first network is a 1×4×1 NN shown in Fig.3 and the second one is a 1×8×1 NN shown in Fig.4.

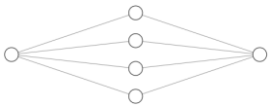

Figure 3. Network #1: 1×4×1

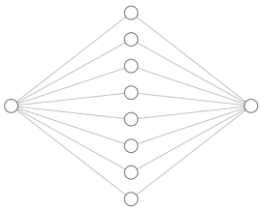

Figure 4. Network#2: 1×8×1

3.2 Networks with Two Hidden Layers

Two separate neural networks with two hidden layers are used. The third network has a 1×8×4×1 architecture, as shown in Fig. 5, and the fourth network follows a 1×8×8×1 architecture, as presented in Fig. 6.

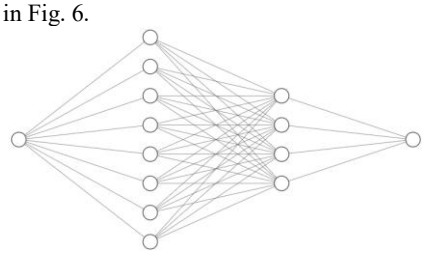

Figure 5. Network#3: 1×8×4×1

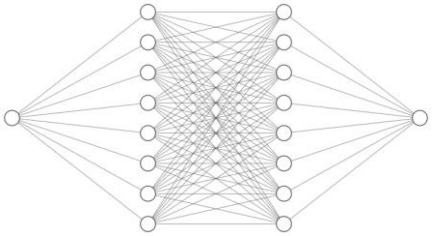

Figure 6. Network#4: 1×8×8×1

3.3 Networks with Three Hidden Layers

For the models with three hidden layers, two different neural networks are utilized. The first one, shown in Fig. 7, follows a 1×8×8×4×1 structure, while the second one, illustrated in Fig. 8, adopts a 1×8×8×8×1 architecture. As the last network, a 1×8×16×8×1 architecture is selected shown in Fig.9.

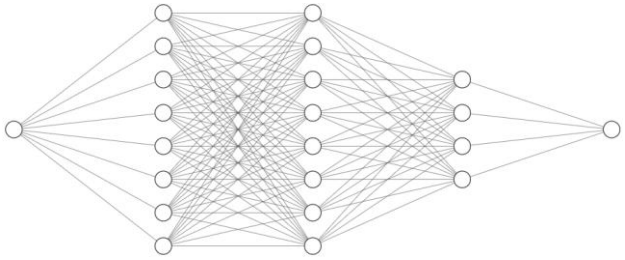

Figure 7. Network#5: 1×8×8×4×1

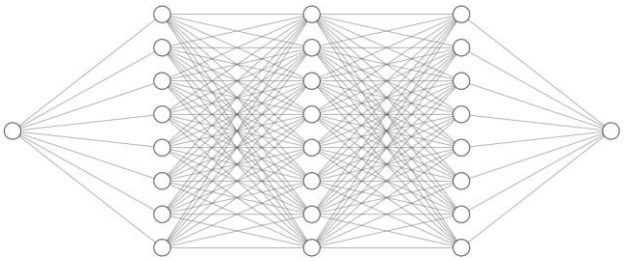

Figure 8. Network#6: 1×8×8×8×1

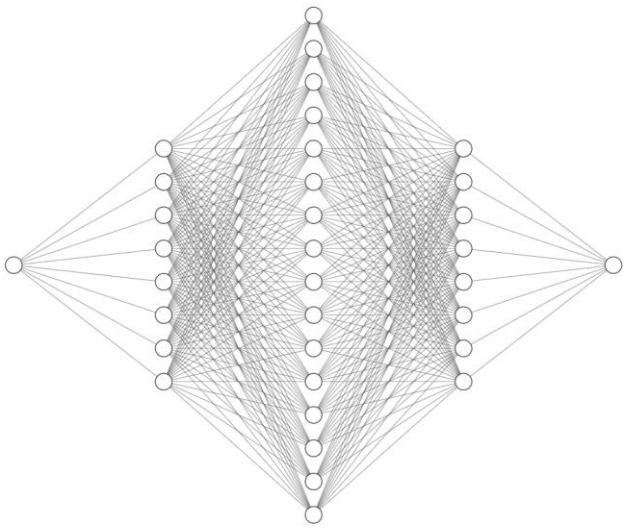

Figure 9. Network#7: 1×8×16×8×1

4. RESULTS

The following section presents the results obtained using seven proposed neural network architectures for various values of *N*. In the figure captions, *lr* denotes the learning rate.

The results for single hidden layer models with *N=1* are presented in Fig. 10. As shown, the models exhibited comparable performance for *m=2* and *m=4*. However, for *m=3*, the 1×8×1 architecture achieved notably better performance in terms of training loss, despite exhibiting more pronounced fluctuations.

The results for two-hidden-layer models are presented in Fig. 11. As shown, the 1×8×8×1 architecture outperforms the 1×8×4×1 model for both *m=2* and *m=4*. Compared to single hidden layer models, the addition of a second hidden layer reduces the training loss by an order of magnitude, from 10^{-5} to 10^{-6}. However, the training loss exhibits larger fluctuations across a wider range of

values. To address this issue, the learning rate was decreased and the total number of training epochs was raised. The results of this adjustment are shown in Fig. 12, where the number of epochs was increased to *50,000,* while the learning rate was adjusted to *0.0001*. As evident from Fig. 12, the variation in training loss became more stable, and its value decreased further to the order of 10^{-7}. As seen in Fig. 12, both models show excellent agreement. However, for *m=2* and *m=3*, additional training epochs are still required to achieve full convergence.

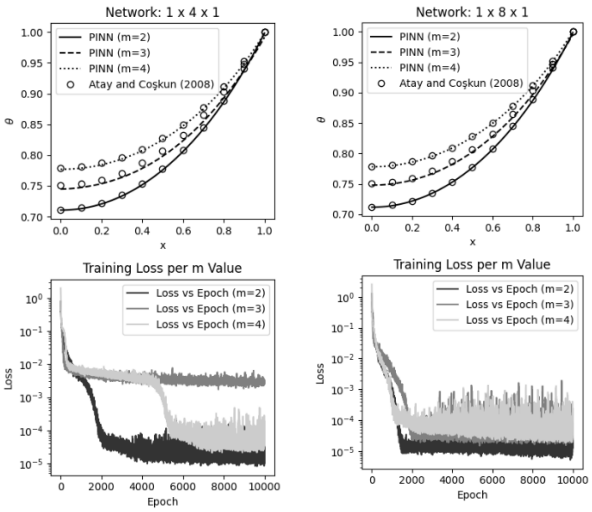

Figure 10. One-hidden-layer network analysis ($N=1$, $lr=10^{-2}$)

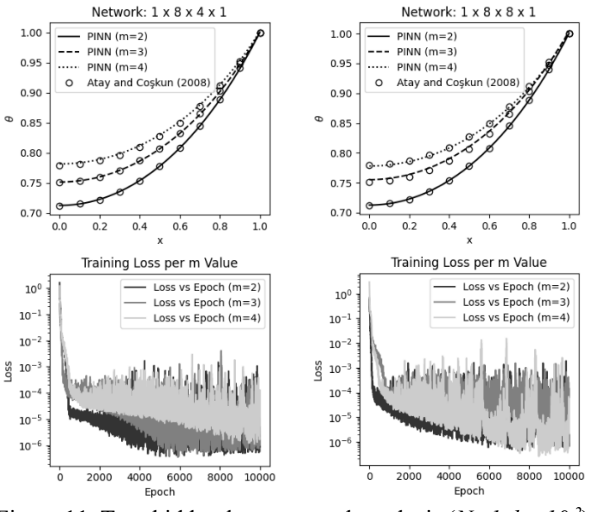

Figure 11. Two-hidden-layer network analysis ($N=1$, $lr=10^{-2}$)

The performance of models with three hidden layers and $N=1$ is illustrated in Figs. 13 and 14, using different learning rates. The 1×8×8×4×1 architecture achieved a reduction in training loss by an order of magnitude, from 10^{-6} to 10^{-7}. However, this model exhibited large fluctuations in the training loss, ranging between 10^{-2} and 10^{-7}. As shown in Fig. 13, the 1×8×8×8×1 model required more training epochs and a lower learning rate—particularly for *m=4,* in order to make the fluctuations more stable. This stability was achieved by reducing the learning rate to *0.0001* and increasing the number of epochs to *80,000*. The corresponding results obtained with these updates are presented in Fig. 14, and show quite accurate performance compared with the reference solutions. Moreover, the training loss exhibits greater stability compared to the results obtained with a learning rate of *0.01*.

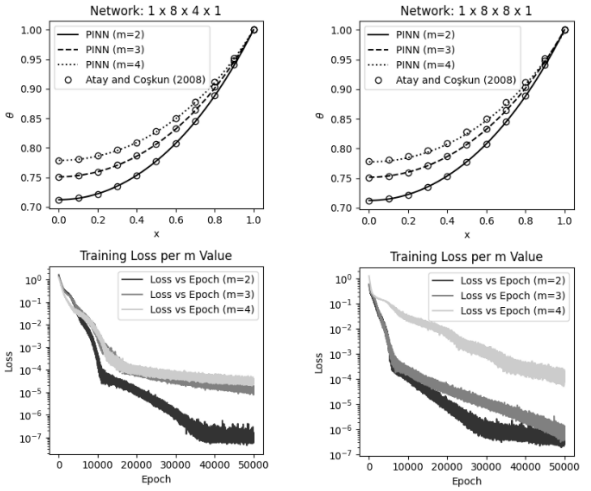

Figure 12. Two-hidden-layer network analysis ($N=1$, $lr=10^{-4}$)

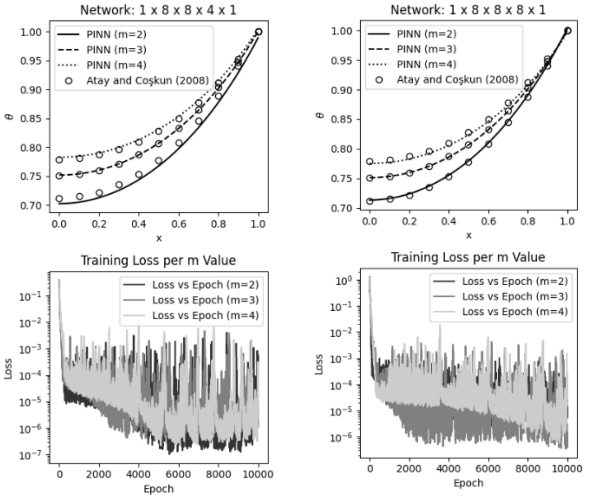

Figure 13. Three-hidden-layer network analysis ($N=1$,$lr=10^{-2}$)

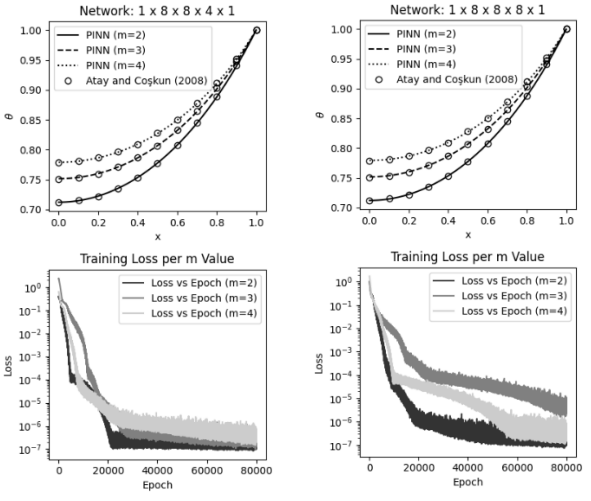

Figure 14. Three-hidden-layer network analysis ($N=1$,$lr=10^{-4}$)

For $N=1$, it is evident that adding more hidden layers led to a reduction in training loss. Nevertheless, even the models with a single hidden layer performed well for the case with $N=1$.

For *N=2*, the 1×4×1 model fails to capture the solution when *m=3*, as the training loss does not decrease as it does for *m=2* and *m=4* as shown in Fig.15. In contrast, the 1×8×1 model successfully captures the solution across all three cases, with the training loss decreasing to the order of 10^{-4} to 10^{-5}.

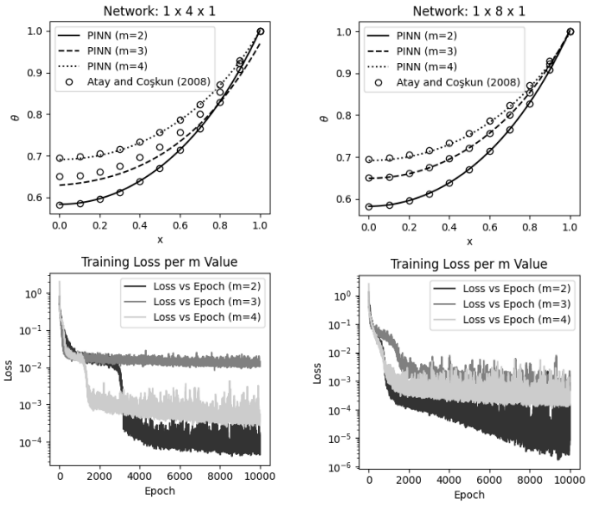

Figure 15. One-hidden-layer network analysis ($N=2$, $lr=10^{-2}$)

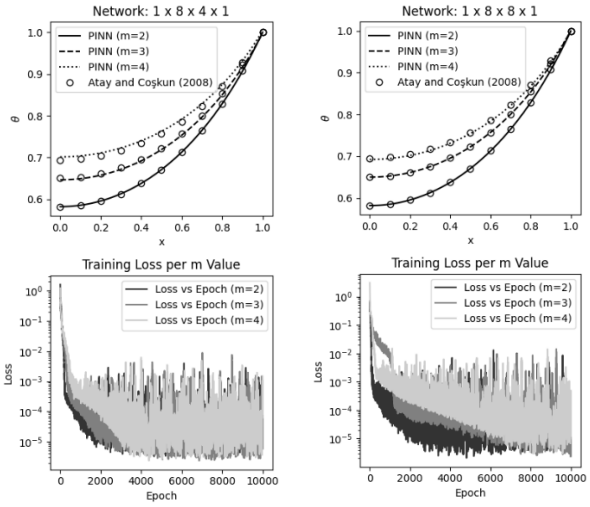

Figure 16. Two-hidden-layer network analysis ($N=2$, $lr=10^{-2}$)

For *N=2*, introducing a second hidden layer leads to a reduction in training loss by an order of magnitude from 10^{-5} to 10^{-6} compared to single hidden layer models. However, similar to the case with *N=1*, the training loss exhibits considerable fluctuations. In order to reduce these fluctuations, the learning rate was reduced to 10^{-4}, and the number of epochs was increased to *60,000*. The results obtained after this update are presented in Fig. 17, and when compared with the results in Fig. 16, a more stable training loss curve is observed, with values decreasing down to the order of 10^{-6}. As also illustrated in Fig. 12, both models exhibit excellent agreement with the reference solution.

In Figs. 18 and 19, the performance of PINN architectures with three hidden layers and *N=2* is shown under different learning rates. Although the models perform well with a learning rate of *0.01* and *10,000* epochs, the training loss still exhibits significant fluctuations. To improve training stability, the analysis was repeated using a reduced learning rate of *0.0001* and an increased number of training epochs set to *100,000*.

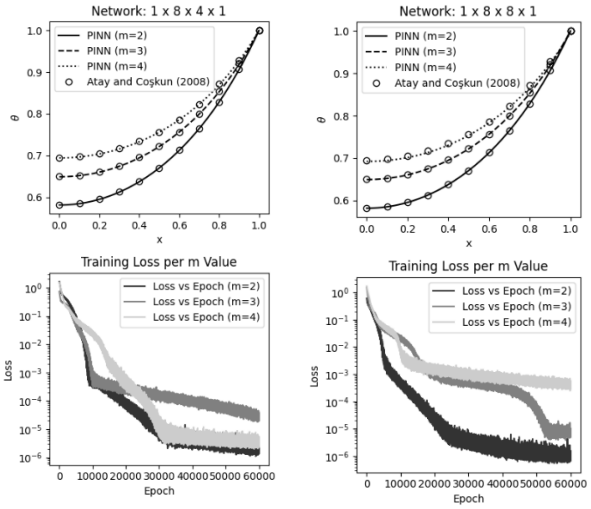

Figure 17. Two-hidden-layer network analysis ($N=2$, $lr=10^{-4}$)

The results for the three-hidden-layer models, trained with a learning rate of 10^{-4} and *100,000* epochs, are presented in Fig. 19. These results demonstrate excellent agreement with the reference solutions. Furthermore, the training loss shows significantly improved stability compared to that observed with a learning rate of *0.01*.

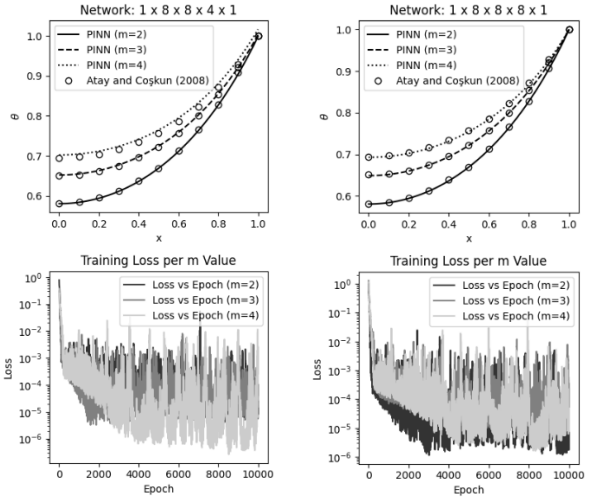

Figure 18. Three-hidden-layer network analysis ($N=2$,$lr=10^{-2}$)

Training stability is improved, and convergence is significantly accelerated by using a reduced learning rate of *0.0001* and increasing the number of training epochs to *100,000*.

For *N=5*, and particularly when *m=4*, the nonlinearity of the problem becomes significantly more pronounced. The performance of single hidden layer models is presented in Fig. 20, using a learning rate *of 0.01* and *10,000* training epochs. Although both the 1×4×1 and 1×8×1 architectures show good agreement with the reference solutions for *m=2* and *m=4*, they fail to capture the solution accurately for *m=3*. This indicates that a single hidden layer is insufficient to model the increased nonlinearity, necessitating the use of an additional hidden layer. The results for two-hidden-layer models are presented in Figs. 21 and 22. In Fig. 21, a learning rate of *0.01* and *10,000* epochs were used. Similar to previous cases, the training loss exhibits considerable fluctuations; however, the inclusion of a second

hidden layer leads to improved performance in capturing the nonlinear characteristics of the problem.

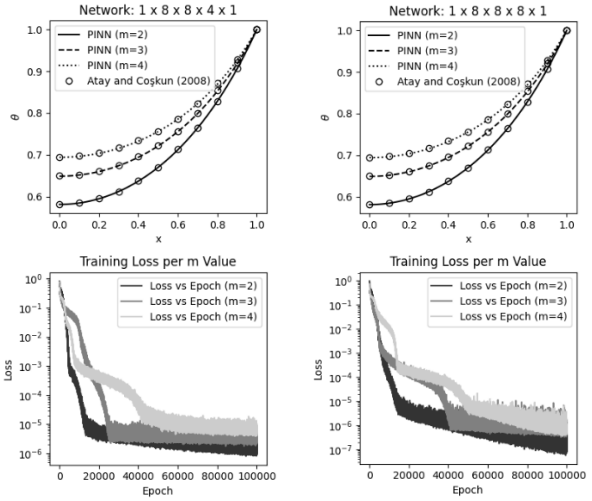

Figure 19. Three-hidden-layer network analysis ($N=2, lr=10^{-4}$)

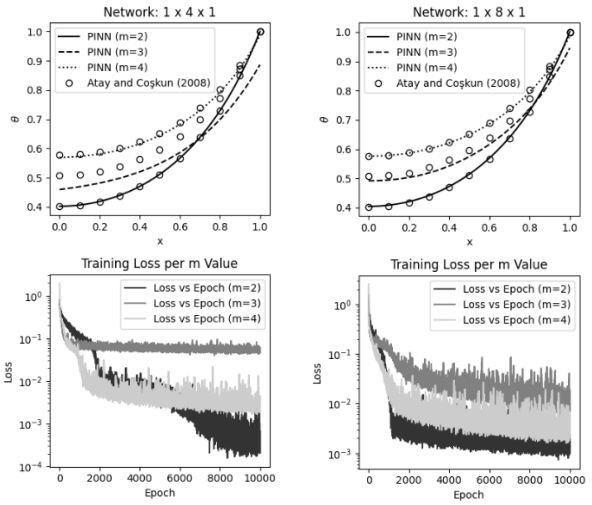

Figure 20. One-hidden-layer network analysis ($N=5$, $lr=10^{-2}$)

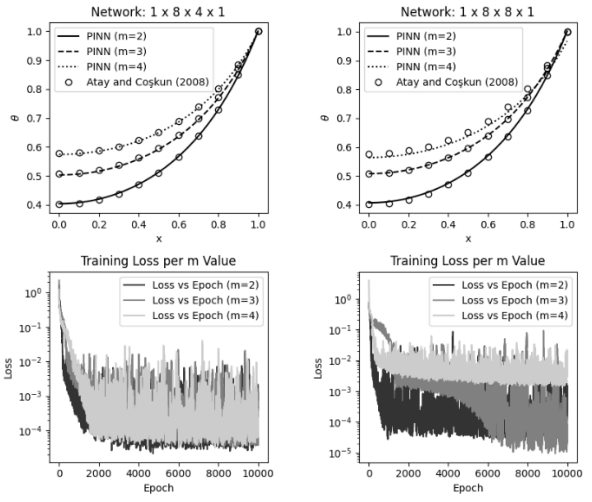

Figure 21. Two-hidden-layer network analysis ($N=5$, $lr=10^{-2}$)

To further stabilize the training loss, the learning rate was reduced to 10^{-4} and the number of training epochs was extended to *60,000*. The results, as presented in Fig. 22, show that the fluctuations in the training loss have become more stable. However, it is observed that increasing the number of epochs is still necessary to achieve closer agreement with the reference solutions. Nevertheless, the model's predictions exhibit excellent agreement with the reference solutions.

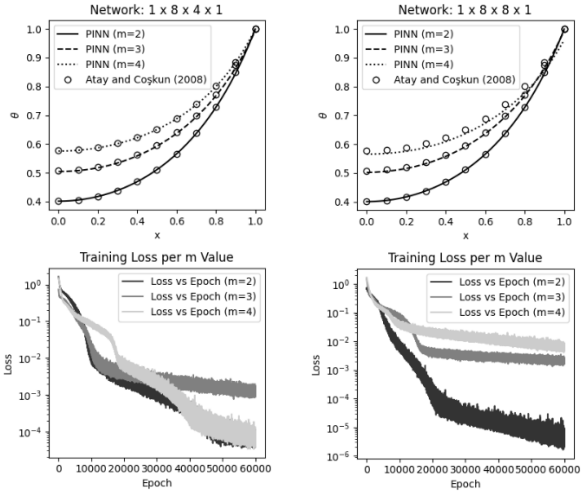

Figure 22. Two-hidden-layer network analysis ($N=5, lr=10^{-4}$)

The performance of three-hidden-layer models with $N=5$ is presented in Figs. 23 and 24, evaluated under different learning rates. In Fig. 23, there is a large difference between the 1×8×8×4×1 model and the reference solution for $m=3$. When m is updated to $m=2$ and $m=4$, and the model is trained with a learning rate of *0.01* and *10,000* epochs an increase in accuracy is observed. Although the learning rate of *0.01* is consistent with previous observations, the training loss exhibits large fluctuations in the range of 10^{-1} to 10^{-5}. To make these fluctuations more stable, the model was run again by reducing the learning rate to *0.0001* and increasing the number of epochs to *100,000*. With this final update, acceptable improvements have been observed in terms of both accuracy and stability. The outputs of the final model show very good agreement with the reference solutions. However, as illustrated in Fig. 24, additional epochs are still required, particularly for $m=2$, to further reduce the training loss and achieve full convergence.

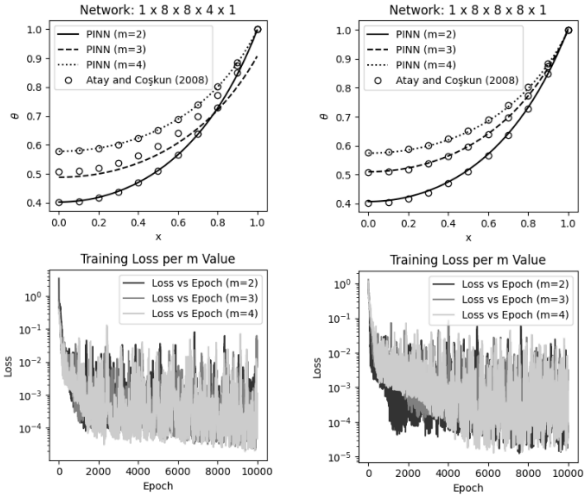

Figure 23. Three-hidden-layer network analysis ($N=5, lr=10^{-2}$)

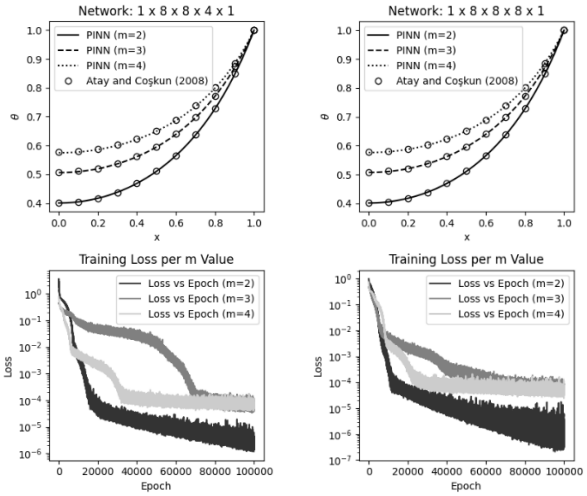

Figure 24. Three-hidden-layer network analysis ($N=5, lr=10^{-4}$)

Until this point, the 1×8×16×8×1 network architecture had not been utilized. The models evaluated thus far indicate that increasing the number of hidden layers enhances the model's capacity to capture nonlinear behavior. However, these deeper architectures require a reduced learning rate, typically on the order of 10^{-4}, and a minimum of *100,000* training epochs to achieve stable and accurate results. Accordingly, the final model was selected as the 1×8×16×8×1 architecture, and the corresponding results are presented in Fig. 25

For the 1×8×16×8×1 model, a learning rate of *0.00005* was employed with *100,000* training epochs. As shown in Fig. 25, the obtained results match the reference solution very closely. In addition, the evolution of the training loss is much smoother than in the earlier runs that employed larger learning rates.

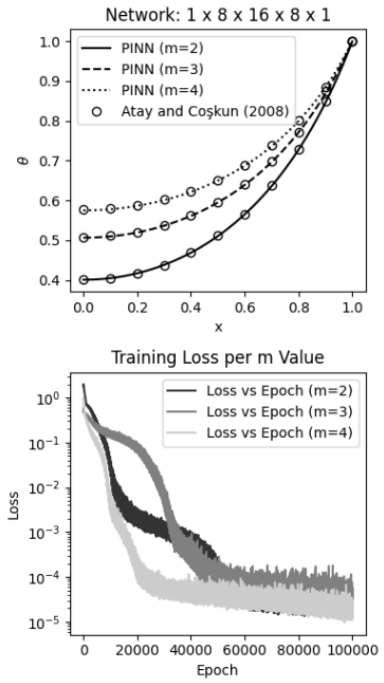

Figure 25. Results for 1×8×16×8×1 model ($N=5$)

5. DISCUSSION

The results of this study underscore the effectiveness of PINNs in solving nonlinear, steady-state, one-dimensional heat conduction problems in fin-type structures governed by power-law behaviour. A key challenge observed throughout the numerical experiments is the sensitivity of PINN training to hyperparameter choices, particularly the learning rate, network depth, and number of training epochs. For PINN architectures to achieve stable convergence without fluctuations and high accuracy for difficult-to-solve nonlinear problems, the learning rates must be kept as low as possible and the training durations must be increased. These requirements lengthen computation times and increase resource consumption. For this, more powerful computer hardware is needed. These constraints limit the use of PINN architectures in real-time, instantaneous problem solutions.

In this study, different architectural structures were determined through manual experimentation. Manual experimentation introduces a generalization issue for solving different problems. In addition, although automatic differentiation offers a significant advantage in computing higher-order derivatives, in the absence of regularization strategies, stability and accuracy can easily deteriorate and, especially in models that require deeper networks, may produce results that are far from the true solution.

The PINN method is regarded as a flexible method and, thanks to this flexibility, it can also be applied to more advanced problems. This study can first be extended so that it can be applied to the transient heat conduction problem in fin-type structures. Then, instead of one dimension, it can be applied to two-dimensional or axisymmetric geometries that better represent real fin systems. In addition, by applying PINN to complex physical problems such as combined convective–radiative heat transfer or thermo-mechanical coupling, solutions can be developed.

6. CONCLUSION

In this study, the use of Physics-Informed Neural Networks (PINNs) with different architectures was investigated for modeling the temperature distribution in a fin subjected to nonlinear, one-dimensional, steady-state heat conduction. Temperature-dependent properties and boundary conditions, which give rise to nonlinearity, lead to computational challenges in classical methods. In contrast to these conventional approaches, PINNs work directly with physical laws and incorporate them as part of the training process. The differential equation to be solved is transformed into a residual-based optimization problem, and this transformation allows common numerical difficulties to be overcome effectively.

The study systematically assessed the impact of varying the convective-conductive parameter N and the power-law exponent m, which governs the thermal behavior within the fin. Different PINN architectures were tested, and it was shown that as the networks become deeper, that is, as the number of layers increases, they become more successful in solving nonlinear problems. However, to achieve this performance and stability, learning rates on the order of 10^{-4} or lower and a large number of training iterations exceeding *100,000* epochs were required.

As a result of the comparative analyses, it has been observed that the use of the PINN architecture achieves an excellent agreement with the reference solutions. In particular, for complex problems involving nonlinear heat conduction, PINNs have been shown to offer a practical and powerful alternative to classical methods. This approach can be said to hold high potential for future applications in more complex heat transfer problems and broader engineering systems.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Kudlur, M., 2016, Tensorflow: A System for Large-Scale Machine Learning. *OSDI'16: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, 16, pp. 265–283. *doi: 10.48550/arXiv.1605.08695.*

Atay, M.T., Coşkun, S.B., 2008. Comparative Analysis of Power-Law Fin-Type Problems Using Variational Iteration Method and Finite Element Method. *Math. Probl. Eng*., 2008, 635231. *doi:10.1155/2008/635231.*

Bai, J., Rabczuk, T., Gupta, A., Alzubaidi, L., Gu, Y., 2023. A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics. *Comput. Mech*., 71, pp. 543-562. *doi: 10.1007/s00466-022-02252-0.*

Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E., 2021. Physics-Informed Neural Networks for Heat Transfer Problems. *J. Heat Transf*., 143, 060801. *doi: 10.1115/1.4050542.*

Chen, Y., Lu, L., Karniadakis, G.E., Dal Negro, L., 2020. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express*, 28, 11618. *doi: 10.1364/OE.384875.*

Fang, Z., Zhan, J., 2020. Deep physical informed neural networks for metamaterial design. *IEEE Access*, 8, pp. 24506–24513. *doi: 10.1109/ACCESS.2019.2963375.*

Haghighat, E., Raissi, M., Moure, A., Gomez, H., Juanes, R., 2021. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Engrg*., 379, 113741. *doi: 10.1016/j.cma.2021.113741.*

Hao, W.Q., Tan, L., Yang, X.G., Shi, D.Q., Wang, M.L., Miao, G.L., Fan, Y.S., 2023. A physics-informed machine learning approach for notch fatigue evaluation of alloys used in aerospace. *Int. J. Fatigue*, 170, 107536. *doi: 10.1016/j.ijfatigue. 2023.107536.*

Hu, Z., Jagtap, A.D., Karniadakis, G.E., Kawaguchi, K., 2019. When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization? *arXiv*, arXiv:2109.09444. *doi: 10.1137/21M1447039.*

Kern, D.Q., Kraus, A.D., 1972. *Extended Surface Heat Transfer*. McGraw-Hill, New York, NY, USA. ISBN: 978-0070341951.

Liaw, S.P., Yeh, R.H., 1994. Fins with temperature dependent surface heat flux—I: single heat transfer mode. *Int. J. Heat Mass Transf.*, 37(10), pp. 1509–1515. *doi: 10.1016/0017-9310(94)90152-X.*

Madir, B.E., Luddens, F., Lothode, C., Danaila, I., 2024. Physics Informed Neural Networks for heat conduction with phase change. *arXiv*, arXiv:2410.14216. *doi: 10.48550/arXiv.2410. 14216.*

Ning, L., Cai, Z., Dong, H., Liu, Y., Wang, W., 2023. A peridynamic-informed neural network for continuum elastic displacement characterization. *Comput. Methods Appl. Mech. Engrg*., 407, 115909. *doi: 10.1016/j.cma.2023.115909.*

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A., 2017. Automatic Differentiation in Pytorch. *31st Conference on Neural Information Processing Systems (NIPS 2017), CA, USA. https://openreview.net/forum?id=BJJsrmfCZ.*

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019a. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378, pp. 686–707. *doi: 10.1016/j.jcp.2018.10.045.*

Raissi, M., Wang, Z., Triantafyllou, M.S., Karniadakis, G.E., 2019b. Deep learning of vortex-induced vibrations. *J. Fluid Mech*., 861, pp. 119–137. *doi: 10.48550/arXiv.1808.08952.*

Raissi, M., Yazdani, A., Karniadakis, G.E., 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367, pp. 1026–1030. *doi: 10.1126/science.aaw4741.*

Russel, C., 2023. Applying Physics-Informed Neural Networks to the Solution of Parametric, Nonlinear Heat Conduction Problems. M.Sc. Thesis, University of Michigan-Flint, University of Michigan Library, Michigan, USA. doi: *10.7302/21927.*

Sen, A.K., Trinh, S., 1986. An exact solution for the rate of heat transfer from a rectangular fin governed by a power law-type temperature dependence. *J. Heat Transfer*, 108(2), pp. 457-459. *doi: 10.1115/1.3246946.*

Shukla, K., Jagtap, A.D., Karniadakis, G.E., 2021. Parallel physics-informed neural networks via domain decomposition. *J. Comput. Phys.*, 447, 110683. *doi: 10.48550/arXiv.2104.10013.*

Zhu, Q., Zhao, Z., Yan, J., 2023. Physics-informed machine learning for surrogate modeling of wind pressure and optimization of pressure sensor placement. *Comput. Mech*., 71, pp. 481-491. *doi: 10.1007/s00466-022-02251-1.*