

Physics-Informed Neural Networks for Solving Differential Equations

Mehmet Çevik* 

İzmir Kâtip Çelebi University, Department of Mechanical Engineering, İzmir, Türkiye

**Corresponding author: mehmet.cevik@ikcu.edu.tr*

Abstract

Differential equations (DEs) are fundamental tools for modeling physical phenomena across various scientific and engineering disciplines. Traditional numerical methods for solving these equations often require extensive computational resources, especially when dealing with high-dimensional, nonlinear, or data-scarce problems. In recent years, Physics-Informed Neural Networks (PINNs) have emerged as a powerful alternative, blending the strengths of deep learning with the rigor of physical laws. By embedding DEs, initial conditions, and boundary conditions directly into the loss function of a neural network, PINNs enable the solution of both forward and inverse problems without the need for mesh generation or large datasets. This paper presents an overview of the PINN methodology and applies it to solve a one-dimensional heat conduction problem without relying on empirical data. The results demonstrate that PINNs can accurately approximate the analytical solution, confirming their potential as flexible, mesh-free solvers. The advantages and challenges of PINNs are also discussed, highlighting their role in advancing data-driven scientific computing.

Keywords: Physics-Informed Neural Networks (PINNs), deep learning, heat equation, mesh-free methods

Introduction

Differential equations (DEs) serve as the foundation for modeling physical systems in disciplines ranging from fluid dynamics and quantum mechanics to structural analysis and biomedical engineering. Accurately solving these equations is essential for understanding and predicting real-world phenomena. Ordinary differential equations (ODEs) and partial differential equations (PDEs) can be solved using various approaches, depending on their type and complexity. The main approaches are:

Analytical methods, which aim to find exact, closed-form solutions. Examples are separation of variables (de Oliveira & Maiorino, 2025), integrating factor method (Nie et al., 2008), characteristic equation method (Alcântara et al., 2022), variation of parameters and undetermined coefficients (Aprecio & Roy, 2024), Fourier series (Yıldız et al., 2025), Laplace transforms (Patil et al., 2024, Kurt, 2008), etc. Numerical methods are used when analytic solutions are difficult or impossible. Examples are Euler's method, Runge-Kutta methods (Akinsola, 2023), finite element method (Szabó & Babuška, 2021), finite difference methods (LeVeque, 2007), spectral and collocation methods (Çevik et al., 2014; Savaşaneril, 2018, Çevik et al., 2025; Çevik, 2010), etc. Qualitative and graphical methods (Li et al., 2016) are used to study the behavior of solutions without solving explicitly. Examples are phase plane analysis, Lyapunov's method (Tuan & Trinh, 2018), direction fields. Transform and integral methods are used for complex PDEs or boundary value problems, examples of which are Fourier Transform and Green's function (Duffy, 2015). On the other hand, semi-analytical techniques are hybrid methods that combine the rigor of analytical approaches with the flexibility of numerical methods to solve DEs, particularly when exact solutions are difficult or impossible to obtain. These techniques aim to produce approximate closed-form expressions that retain the interpretability of analytical solutions while extending applicability to complex or nonlinear problems. Common types of semi-analytical methods include the Homotopy Analysis Method (Liao, 2009), Homotopy Perturbation Method (Biazar & Aminikhah, 2009), Adomian Decomposition Method (Li & Pang, 2020; Abbasbandy, 2003), Variational Iteration Method (Bildik & Konuralp, 2006), Differential Transform Method (Ayaz, 2004; Sımr et al., 2018), and the Perturbation-Iteration Algorithm (Pakdemirli, 2016; Bahşi & Çevik, 2015). Each of these methods constructs the solution iteratively, often in the form of a rapidly converging series, and is particularly useful for solving linear and nonlinear ODEs and PDEs with minimal computational effort and high accuracy.

Classical numerical methods—such as finite difference, finite element, and Runge-Kutta techniques—have long been the standard tools for solving DEs. However, these methods can be computationally intensive and inflexible, particularly when dealing with high-dimensional, nonlinear, or data-scarce problems. In response to these limitations, recent advances in artificial intelligence (AI) have led to the development of alternative, data-driven approaches for solving DEs. These AI-based methods aim to offer greater adaptability, improved scalability, and, in some cases, enhanced accuracy compared to conventional techniques (Jamaludin et al., 2024; Görüş et al., 2024; Canyakan, 2025).

Among the most notable AI approaches are PINNs, which embed the governing physical laws directly into the training process of neural networks; the Deep Galerkin Method (Sirignano & Spiliopoulos, 2018; Al-Arabi et al., 2022), which generalizes the traditional Galerkin method for high-dimensional PDEs; and Neural Ordinary Differential Equations (Chen et al., 2018), in which neural networks model the continuous dynamics of systems. Other emerging methods include the Deep Ritz approach (Yang et al., 2022), which is grounded in the Ritz variational principle; deep Backward Stochastic Differential Equation (Chessari et al., 2023) solvers for solving high-dimensional PDEs; reinforcement learning-based techniques that treat DEs as sequential decision-making problems; and the use of Generative Adversarial Networks (Gui et al., 2023) for learning solutions to complex PDEs. Additionally, hybrid AI-numerical methods have been proposed to integrate AI with classical solvers, leveraging the strengths of both paradigms to improve efficiency, generalization, and solution quality. These developments mark a significant shift in the field, opening new possibilities for solving DEs in scientific and engineering applications.

With the rise of data-driven modeling and machine learning, researchers began to explore neural networks as approximators of functions and solutions. However, traditional neural networks typically require large datasets to train effectively, and their predictions may not respect the underlying physical laws of the system. PINNs address this shortcoming by embedding physical knowledge directly into the neural network training process. The key innovation is that PINNs do not just learn from data; they learn from the physics itself, enabling them to generalize better and require fewer data points for training.

By bridging deep learning and scientific computing, PINNs enable the solution of ODEs and PDEs equations without requiring a structured grid. Moreover, they offer generalizability across different domains and robustness in handling inverse problems, parameter estimation, and systems with incomplete data. This paper aims to provide an overview of the PINN methodology, demonstrate its application to a representative DE problem, and discuss usage.

Overview of PINNs

PINNs have recently gained significant attention as a mesh-free and data-efficient alternative. Introduced by Raissi et al. (2019), PINNs incorporate the governing DEs, associated boundary, and initial conditions directly into the loss function of a neural network. This innovative approach enables the network to learn solutions that satisfy both data (if available) and the underlying physical laws simultaneously.

The term "PINN" reflects the integration of physical laws into the neural network's architecture. Unlike traditional neural networks that rely solely on data, PINNs are designed to respect the underlying physics of the problem by embedding the governing equations directly into the training process. This approach ensures that the network's predictions are not only data-driven but also physically plausible. When physical data is available, it is incorporated into the PINN by adding a data loss term to the overall loss function. By minimizing this combined loss function, the PINN learns solutions that are consistent with both the physical laws and the observed data.

Numerous researchers have explored the development and applications of PINNs. For instance, Cai et al. (2021) demonstrated the versatility of PINNs by addressing complex heat transfer problems. Deresse and Bekela (2025) employed PINNs to solve nonlinear telegraph equations with varying boundary conditions, while Barbulescu et al. (2025) applied them to model highly nonlinear dynamic systems.

A typical PINN is a deep neural network that takes spatial and/or temporal coordinates as inputs and outputs the predicted values of the solution of interest, such as temperature, velocity, displacement, or pressure. The network is trained to minimize a loss function that includes not only discrepancies from data but also penalties for violating the governing PDEs and boundary/initial conditions. Figure 1 illustrates the general architecture of a PINN, where the neural network is trained to minimize the residuals of the DEs as well as any mismatch with observed or initial data.

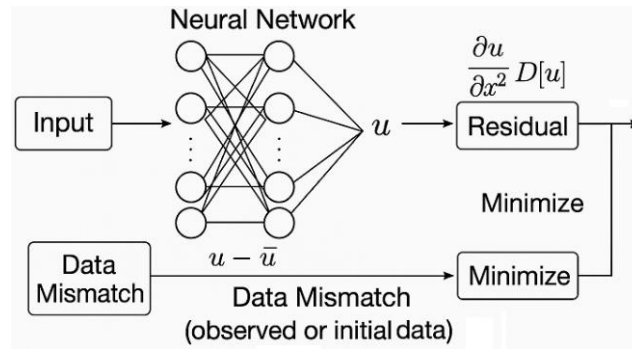


Figure 1. General architecture of a PINN

A total (composite) loss function is a loss function that combines multiple terms to balance different objectives during training, which ensures that the model satisfies both observed data and underlying physical laws, leading to more accurate and meaningful results. The total loss function \mathcal{L}_{total} in a PINN has three main components:

1. PDE residual loss (\mathcal{L}_{PDE}): Ensures that the predicted solution satisfies the governing equations at randomly chosen collocation points.
2. Boundary/initial condition loss (\mathcal{L}_{BIC}): Enforces physical constraints at the domain boundaries or initial time.
3. Data loss (\mathcal{L}_{data}): Measures the error between the network predictions and any available observational or experimental data.

Thus, the total loss is typically expressed as:

$$\mathcal{L}_{total} = \lambda_{PDE} \mathcal{L}_{PDE} + \lambda_{BIC} \mathcal{L}_{BIC} + \lambda_{data} \mathcal{L}_{data} \tag{1}$$

where λ 's are weighting factors that balance the contributions of each term.

PINNs derive their strength from the unique ability to learn solutions to physical problems even in the absence of observational or experimental data, relying solely on the governing equations –such as PDEs– along with the appropriate boundary and initial conditions. When observational data is available, PINNs integrate both the data and the physical laws (e.g., combining sensor measurements with the Navier–Stokes equations), leading to improved accuracy, particularly in cases where the data is sparse or noisy. In contrast, in data-free scenarios, the neural network is trained exclusively to satisfy the physical constraints imposed by the DEs and their associated boundary and initial conditions. This approach is especially advantageous for addressing forward and inverse problems where real-world data is unavailable, difficult, or expensive to collect. Therefore, while physical data can enhance the model's performance, it is not a strict prerequisite; what is fundamentally required is a reliable mathematical formulation of the underlying physics governing the system.

Solution of 1-D Heat Equation by PINN as a Test Problem

We want to solve the 1-D heat equation (without any data) on a rod of length $L = 1$, over a time interval $\in [0,5]$:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{2}$$

where

$u(x, t)$ is the temperature at position x and time t

α is thermal diffusivity (assume, $\alpha = 0.01$)

Boundary conditions:

$u(0, t) = u(1, t) = 0$ (ends of the rod are held at 0°C)

Initial condition:

$u(x, 0) = \sin(\pi x)$

The steps are:

1. We define a neural network $u_\theta(x, t)$ with inputs x and t , and output u (temperature).

2. Automatically compute the partial derivatives $\partial u/\partial t$ and $\partial^2 u/\partial x^2$ using automatic differentiation, a technique commonly available in deep learning libraries like TensorFlow and PyTorch.
3. Construct a residual function $\mathcal{R} = \frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2}$
4. Minimize the mean squared error of the residual at collocation points, ensuring that the neural network output satisfies the PDE.

The residual function represents the error in satisfying the differential equation(s) that encode the physical law. Essentially, you take the candidate solution output by the neural network and plug it into the differential equation.

During training, the neural network not only learns from observed data but is also guided by known physical laws, which enhances its robustness and generalization. In this case, no data is provided to the network; instead, we enforce that:

- $u_\theta(x, 0) = \sin(\pi x)$ which is the initial condition
- $u_\theta(0, t) = u_\theta(1, t) = 0$ which are the boundary conditions
- The function $u_\theta(x, t)$ satisfies the heat equation:

$$\left| \frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} \right| \rightarrow 0 \tag{3}$$

The PINN learns a function that approximates the analytical solution:

$$\frac{\partial u}{\partial t} u(x, t) = \sin(\pi x) e^{-\pi^2 \alpha t} \tag{4}$$

The results obtained from the PINN are graphically visualized over the time interval [1,5] and presented in Figure 2.

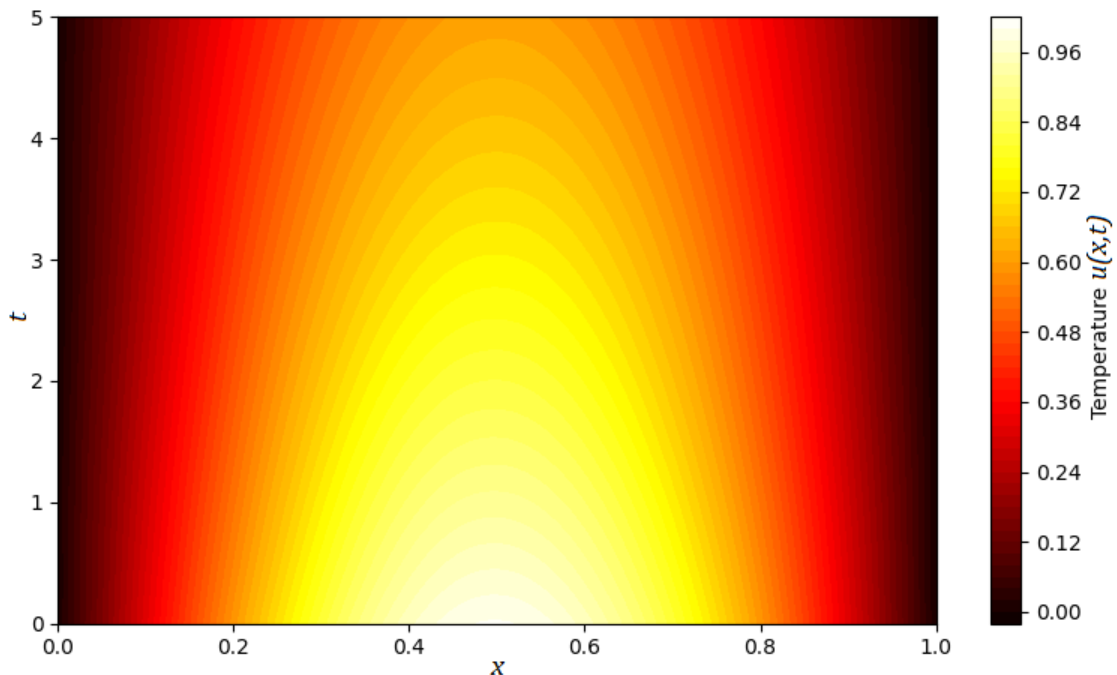


Figure 1. PINN solution of the 1-D heat equation

In this solution, no experimental data is employed; instead, the formulation relies solely on the governing PDE and the specified initial and boundary conditions. This highlights the capability of PINNs to solve complex physical problems in a manner similar to traditional numerical solvers, while offering distinct advantages such as being mesh-free, highly adaptable, and easily extendable to inverse problems or parameter estimation.

Advantages and Challenges of the PINNs

PINNs offer a transformative approach for solving DEs by incorporating physical laws into the learning process. Compared to traditional numerical methods, PINNs present several advantages:

- **Mesh-free computational framework:** Unlike conventional numerical solvers such as finite element or finite difference methods, PINNs do not require spatial discretization or mesh generation. This characteristic enhances their applicability to complex geometries and irregular domains, making them well-suited for problems with evolving boundary conditions.
- **Integration of prior physical knowledge:** PINNs leverage PDEs and governing physical laws as constraints within their loss function. This integration reduces reliance on large labeled datasets, facilitating generalization and improving accuracy when experimental or observational data are sparse.
- **Unified representation of forward and inverse problems:** A key advantage of PINNs is their ability to solve both forward and inverse problems within a single neural network framework. In addition to approximating solutions to PDEs, they can be utilized for parameter estimation, and data assimilation, thereby offering a flexible approach to real-world applications.
- **Improved generalization and extrapolation capabilities:** Unlike traditional machine learning models, which predominantly rely on empirical data, PINNs incorporate physics-based constraints that enhance their ability to generalize beyond training samples. This property is particularly beneficial for extrapolating solutions in regions where data is unavailable.
- **Multiphysics and multiscale integration:** PINNs provide a natural framework for integrating multiple physical domains within a unified neural network. This makes them effective for modeling multiphysics phenomena, such as coupled fluid-structure interactions or thermo-mechanical systems, where different governing equations must be satisfied simultaneously.

Despite their advantages, PINNs also face several fundamental challenges that influence their practical implementation and scalability:

- **Computational complexity and training cost:** The optimization of PINNs requires minimizing a total loss function that balances data-driven accuracy and physics-informed constraints. This process is computationally demanding, especially in high-dimensional problems where automatic differentiation incurs significant memory and processing overhead.
- **Hyperparameter sensitivity and optimization challenges:** The effectiveness of PINNs is highly dependent on the appropriate selection of network architecture, activation functions, and weighting factors for different loss components. Improper tuning of these hyperparameters can lead to poor convergence, numerical instability, or failure to satisfy PDE constraints adequately.
- **Difficulty in handling stiff equations and sharp gradients:** PINNs struggle with solving stiff PDEs or problems involving discontinuities and steep gradients. Traditional numerical methods, such as spectral methods or adaptive mesh refinement techniques, often outperform PINNs in these cases.
- **Challenges in scaling to large-scale and high-dimensional systems:** Applying PINNs to complex systems with multiple interacting PDEs and extensive spatial-temporal domains presents scalability issues. Computational techniques such as physics-informed Gaussian processes, distributed neural network architectures, and physics-aware transfer learning are being explored to mitigate these constraints.
- **Interpretability and robustness in real-world applications:** While PINNs exhibit strong theoretical advantages, their adoption in practical engineering and scientific applications requires rigorous validation. Uncertainty quantification and sensitivity analysis remain crucial areas of research to ensure their robustness in real-world scenarios.

Conclusion

PINNs represent a significant advancement in the intersection of scientific computing and machine learning. By embedding the underlying physical laws into the architecture and training of neural networks, PINNs offer a mesh-free, data-efficient, and flexible framework for solving both forward and inverse problems governed by DEs. This paper has demonstrated the effectiveness of PINNs through the solution of a one-dimensional heat equation, showing that even in the absence of experimental data, accurate and physically consistent results can be obtained. Despite their promising capabilities, PINNs face challenges related to computational cost, optimization complexity, and scalability, particularly in high-dimensional or stiff systems. As the field matures, PINNs are poised to become a core tool for tackling complex, real-world problems across physics, engineering, and applied sciences.

References

- Abbasbandy, S. (2003). Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method. *Applied Mathematics and Computation*, 145(2-3), 887-893.
- Akinsola, V. (2023). Numerical methods: Euler and Runge-Kutta. In *Qualitative and Computational Aspects of Dynamical Systems*. IntechOpen.
- Alcântara, S. C. S., Lima, A. A. S., Ochoa, A. A. V., Leite, G. de N. P., da Costa, J. Â. P., dos Santos, C. A. C., Cavalcanti, E. J. C., & Michima, P. S. A. (2022). Implementation of the characteristic equation method in quasi-dynamic simulation of absorption chillers: Modeling, validation and first results. *Energy Conversion and Management: X*, 13, 100165. <https://doi.org/10.1016/j.ecmx.2021.100165>
- Aprecio, N. V. C., & Roy, F. A. Jr. (2024). Method of undetermined coefficients in solving third-order linear differential equations: A comprehensive analysis. *Journal of Engineering and Technological Advances*, 9(1), 105-118.
- Al-Aradi, A., Correia, A., Jardim, G., Naiff, D. de F., & Saporito, Y. (2022). Extensions of the deep Galerkin method. *Applied Mathematics and Computation*, 430, 127287. <https://doi.org/10.1016/j.amc.2022.127287>
- Ayaz, F. (2004). Solutions of the system of differential equations by differential transform method. *Applied Mathematics and Computation*, 147(2), 547–567. [https://doi.org/10.1016/S0096-3003\(02\)00794-4](https://doi.org/10.1016/S0096-3003(02)00794-4)
- Bahşi, M. M., & Çevik, M. (2015). Numerical solution of pantograph-type delay differential equations using perturbation-iteration algorithms. *Journal of Applied Mathematics*, 2015(1), 139821.
- Barbulescu, R., Ciuprina, G., Duca, A., & et al. (2025). Physics-informed neural networks for a highly nonlinear dynamic system. *Journal of Mathematics in Industry*, 15, 7. <https://doi.org/10.1186/s13362-025-00172-1>
- Bildik, N., & Konuralp, A. (2006). Two-dimensional differential transform method, Adomian's decomposition method, and variational iteration method for partial differential equations. *International Journal of Computer Mathematics*, 83(12), 973-987.
- Biazar, J., & Aminikhah, H. (2009). Study of convergence of homotopy perturbation method for systems of partial differential equations. *Computers & Mathematics with Applications*, 58(11–12), 2221–2230. <https://doi.org/10.1016/j.camwa.2009.03.030>
- Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6), 060801.
- Canyakan, S. (2025). Machine learning in audio mastering: A comparative study. *Journal for the Interdisciplinary Art and Education*, 6(1), 47–65. <https://doi.org/10.5281/zenodo.15074948>
- Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. Part of *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.
- Chessari, J., Kawai, R., Shinozaki, Y., & Yamada, T. (2023). Numerical methods for backward stochastic differential equations: A survey. *Probability Surveys*, 20, 486-567.
- Çevik, M. (2010). Application of Taylor matrix method to the solution of longitudinal vibration of rods. *Mathematical and Computational Applications*, 15(3), 334-343. <https://doi.org/10.3390/mca15030334>
- Çevik, M., Bahşi, M. M., & Sezer, M. (2014). Solution of the delayed single degree of freedom system equation by exponential matrix method. *Applied Mathematics and Computation*, 242, 444–453. <https://doi.org/10.1016/j.amc.2014.05.111>
- Çevik, M., Savaşaneril, N. B., & Sezer, M. (2025). A review of polynomial matrix collocation methods in engineering and scientific applications. *Archives of Computational Methods in Engineering*. <https://doi.org/10.1007/s11831-025-10235-6>

- Deresse, A. T., & Bekela, A. S. (2025). A deep learning approach: Physics-informed neural networks for solving a nonlinear telegraph equation with different boundary conditions. *BMC Research Notes*, 18, 77. <https://doi.org/10.1186/s13104-025-07142-1>
- Duffy, D. G. (2015). *Green's Functions with Applications* (2nd ed.). Chapman and Hall/CRC.
- Görüş, V., Bahşı, M. M., & Çevik, M. (2024). Machine learning for the prediction of problems in steel tube bending process. *Engineering Applications of Artificial Intelligence*, 133(Part F), 108584. <https://doi.org/10.1016/j.engappai.2024.108584>
- Gui, J., Sun, Z., Wen, Y., Tao, D., & Ye, J. (2023). A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 35(4), 3313–3332. <https://doi.org/10.1109/TKDE.2021.3130191>
- Jamaludin, H., Achlison, U., & Rokhman, N. (2024). Enhancing AI model accuracy and scalability through big data and cloud computing. *JTIE: Journal of Technology Informatics and Engineering*, 3(3), Article 203. <https://doi.org/10.51903/jtie.v3i3.203>
- Kurt, N. (2008). Solution of the two-dimensional heat equation for a square in terms of elliptic functions. *Journal of the Franklin Institute*, 345(3), 303–317.
- LeVeque, R. J. (2007). *Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems*. SIAM.
- Li, T., Bohner, M., Candan, T., Rogovchenko, Y. V., & Wang, Q.-R. (2016). Qualitative theory of differential equations, difference equations, and dynamic equations on time scales. *The Scientific World Journal*, 2016, Article ID 9094613. <https://doi.org/10.1155/2016/9094613>
- Li, W., & Pang, Y. (2020). Application of Adomian decomposition method to nonlinear systems. *Advances in Differential Equations*, 2020, Article 67. <https://doi.org/10.1186/s13662-020-2529-y>
- Liao, S. (2009). Notes on the homotopy analysis method: Some definitions and theorems. *Communications in Nonlinear Science and Numerical Simulation*, 14(4), 983–997. <https://doi.org/10.1016/j.cnsns.2008.04.013>
- Nie, Q., Wan, F. Y. M., Zhang, Y.-T., & Liu, X.-F. (2008). Compact integration factor methods in high spatial dimensions. *Journal of Computational Physics*, 227(10), 5238–5255. <https://doi.org/10.1016/j.jcp.2008.01.050>
- de Oliveira, E. C., & Maiorino, J. E. (2024). The method of separation of variables. In *Analytical Methods in Applied Mathematics* (pp. 223–249). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-74794-6_9
- Pakdemirli, M. (2016). Application of the perturbation iteration method to boundary layer type problems. *SpringerPlus*, 5, Article 208. <https://doi.org/10.1186/s40064-016-1859-4>
- Patil, A. S., Chougale, R. K., & Shinde, S. S. (2024). A literature review on applications of Laplace transform in engineering. *The Indian Journal of Technical Education*, 47(1), 375–392.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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. <https://doi.org/10.1016/j.jcp.2018.10.045>
- Savaşaneril, N. B. (2018). Laguerre series solutions of the delayed single degree-of-freedom oscillator excited by an external excitation and controlled by a control force. *Journal of Computational and Theoretical Nanoscience*, 15(2), 606–610. <https://doi.org/10.1166/jctn.2018.7132>
- Sınır, S., Çevik, M., & Sınır, B. G. (2018). Nonlinear free and forced vibration analyses of axially functionally graded Euler-Bernoulli beams with non-uniform cross-section. *Composites Part B: Engineering*, 148, 123–131. <https://doi.org/10.1016/j.compositesb.2018.04.061>
- Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 1339–1364. <https://doi.org/10.1016/j.jcp.2018.08.029>

- Szabó, B., & Babuška, I. (2021). *Finite element analysis: Method, verification, and validation*. Wiley.
- Tuan, H. T., & Trinh, H. (2018). Stability of fractional-order nonlinear systems by Lyapunov direct method. *IET Control Theory & Applications*, 12(17), 2417-2422.
- Yang, J., Zuo, J., Tian, Y., & Lei, M. (2022). Deep Ritz method for solving high-dimensional fractional differential equations. *Engineered Science*, 21, 789. <https://dx.doi.org/10.30919/es8d789>
- Yıldız, B., Sınır, S., & Sınır, B. G. (2025). A general solution procedure for nonlinear single degree of freedom systems including fractional derivatives. *International Journal of Non-Linear Mechanics*, 169, 104966. <https://doi.org/10.1016/j.ijnonlinmec.2024.104966>