International Journal of Multidisciplinary Research and Publications
ISSN (Online): 2581-6187

Physics-Informed Neural Networks for Solving Initial Value Problems

Jiaqun Wang
Anhui University of Finance and Economics, Bengbu, 233030, China
Email address: xitiruo_wjq@foxmail.com

Abstract*—Initial Value Problems (IVPs) are fundamental mathematical models describing the dynamic evolution of systems across science and engineering. Their accurate and efficient solution is crucial for both theoretical research and practical applications. Traditional numerical methods, such as the Runge-Kutta and finite difference methods, while mature, face significant challenges when dealing with high-dimensional, nonlinear, and complex problems that benefit from mesh-free solutions. These challenges include the "curse of dimensionality," error accumulation, and high computational costs. In recent years, with the rise of deep learning, Physics-Informed Neural Networks (PINNs) have emerged as a novel scientific machine learning approach that integrates physical laws with neural networks, offering a new paradigm for solving differential equations. This paper aims to systematically explore the application of PINNs in solving Ordinary Differential Equation (ODE) based Initial Value Problems. First, we elaborate on the theoretical foundation of PINNs for solving IVPs, including how a neural network acts as a universal function approximator to parameterize the solution, and how a composite loss function is constructed to incorporate both the dynamic equation's residual and the initial condition constraints. We emphasize the critical role of automatic differentiation in accurately calculating derivatives to form the physics-based residual. Subsequently, this paper presents a case study of a classic two-dimensional dynamical system. By comparing the PINN's predictions with the analytical solution across multiple dimensions—including time-evolution plots, absolute error plots, and phase-space trajectories—we quantitatively and qualitatively validate the method's effectiveness and high precision. The results indicate that PINNs can not only accurately predict the dynamic behavior of systems but also exhibit immense potential in handling complex problems due to their mesh-free nature. Finally, this paper summarizes the advantages of the PINN methodology, discusses its current challenges, and provides an outlook on future research directions.*

***Keywords*—** *Physics-Informed Neural Networks; Initial Value Problem; Deep Learning; Scientific Computing; Automatic Differentiation.*

I. INTRODUCTION

In the vast domains of natural science and engineering, the underlying principles of countless phenomena—from the prediction of celestial orbits to the fluctuation of current in an electrical circuit, from the rate of chemical reactions to the spread of epidemics—can be described by mathematical models that govern the evolution of a system's state with respect to a variable, typically time[1-4]. At the heart of these models lie differential equations. A particularly crucial class of these is the Initial Value Problem (IVP)[5-9]. A typical IVP consists of two components: one or a set of Ordinary Differential Equations (ODEs) that describe the system's dynamic evolution, and the exact state of the system at a specific initial moment (usually t=0). This initial state is akin to the first domino in a line; once it is pushed, the entire future trajectory of the system is uniquely determined by the governing dynamic equations. Consequently, solving an IVP—that is, predicting the system's state at any future time based on its initial conditions and dynamic laws—is fundamental to understanding, predicting, and controlling dynamical systems. It plays an indispensable role in physics, mechanics, biology, economics, and many other disciplines. Whether designing stable control systems, forecasting financial market volatility, or simulating protein folding, the task at its core invariably involves solving a corresponding IVP. Due to its ubiquity and foundational importance, the development of accurate, efficient, and robust methods for solving IVPs has always been a central research topic in computational mathematics and scientific computing.

To meet the immense demand for solutions, computational scientists have developed a series of mature and powerful numerical algorithms over the past century. These traditional numerical methods, such as the Euler method, the improved Euler method, and the more famous and widely used Runge-Kutta family of methods, linear multistep methods (e.g., Adams methods), and the Finite Difference Method (FDM), form the cornerstone of modern scientific computing [3, 4]. The core idea of these methods is "discretization": they divide the continuous time domain into a series of discrete time steps and then, using a recursive formula, march forward step-by-step from the initial point to approximate the system's state at each discrete time point. High-order methods like Runge-Kutta significantly improve single-step accuracy by performing multiple "trial" calculations within each time step. These algorithms perform exceptionally well when solving low-dimensional, well-behaved (non-stiff) IVPs. Their theory is complete, their errors are controllable, and they are indispensable tools in the toolkits of engineers and scientists.

However, despite their tremendous success, the inherent step-by-step and grid-based nature of traditional methods also introduces several intractable limitations. The first is the issue of error accumulation. As each step of the calculation introduces truncation and round-off errors, these minute errors can accumulate and amplify over thousands of steps, potentially causing the result of a long-term integration to deviate significantly from the true solution. Second, for high-dimensional dynamical systems, the computational cost of traditional methods increases dramatically. For instance, in

Jiaqun Wang, "Physics-Informed Neural Networks for Solving Initial Value Problems," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 8, Issue 3, pp. 216-221, 2025.

molecular dynamics simulations or certain control theory problems, the state vector's dimension can reach thousands or even higher. In such cases, the complexity of grid meshing and the volume of computation grow exponentially, a phenomenon known as the "Curse of Dimensionality." Furthermore, traditional methods provide solutions on a predefined discrete grid. Obtaining the solution at an arbitrary non-grid point requires additional interpolation, which not only adds computational complexity but may also introduce new errors. Finally, for problems with complex nonlinearities or stiffness, a very small time step is required to ensure stability and accuracy, which makes the computation extremely inefficient. These limitations have motivated researchers to explore new computational paradigms to overcome the bottlenecks of traditional methods.

The 21st century has witnessed revolutionary breakthroughs in artificial intelligence, led by deep learning. Its powerful function approximation capabilities and potential for handling high-dimensional data have attracted widespread attention in the scientific computing community. Against this backdrop, a novel method aimed at deeply integrating physical knowledge with neural networks—the Physics-Informed Neural Network (PINN)—came into being. The concept of PINNs can be traced back to the 1990s with the work of Dissanayake and Phan-Thien, among others, but it was the series of seminal papers published by Raissi, Perdikaris, and Karniadakis between 2017 and 2019 that truly systematized the field and brought it to prominence [10-15]. They clearly articulated the general framework for PINNs: incorporating the residual of a partial differential equation (PDE) or an ordinary differential equation (ODE) as a regularization term in the neural network's loss function. This "forces" the network's output to not only fit the data points but also to obey the underlying physical laws.

When PINNs are applied to solve IVPs, their unique advantages become apparent. The core difference from traditional methods is that PINNs do not adopt a "step-by-step" strategy but instead transform the solution process into an optimization problem. They treat a neural network as a universal function approximator that directly learns the continuous mapping from time t to the system state u(t), denoted as. This mapping is analytical, continuous, and uniquely determined by the network's parameters θ. Its main advantages are [10-15]:

1. Mesh-free Solution: PINNs learn the functional form of the solution directly within the continuous time domain. After training, one can obtain the solution at any time t simply by feeding it into the network, without any need for interpolation. This is extremely beneficial for applications requiring high-resolution solutions or queries at arbitrary points.
2. Mitigating the Curse of Dimensionality: Neural networks have a natural advantage in handling high-dimensional inputs. For a high-dimensional ODE system, the input to the PINN is still the one-dimensional time t, while the output is the high-dimensional state vector. Although the network's complexity increases with the dimension, this growth is typically linear or polynomial, showing great potential for handling high-dimensional problems compared to the exponential growth of grid-based methods.
3. Leveraging Automatic Differentiation: Modern deep learning frameworks (like PyTorch and TensorFlow) have powerful built-in Automatic Differentiation (AD) engines. PINNs cleverly utilize this tool to accurately and efficiently compute derivatives of the network's output with respect to its input, to any order. This allows for the precise calculation of the ODE residual, avoiding the discretization errors introduced by traditional finite difference methods.
4. Natural Suitability for Inverse Problems: In addition to solving forward problems (i.e., finding the solution given the equation and initial conditions), the PINN framework can be naturally extended to solve inverse problems, such as inferring unknown parameters in an ODE from partial observational data. This is achieved by simply treating these parameters as additional variables to be optimized.

This paper will systematically introduce and demonstrate how to utilize Physics-Informed Neural Networks to solve Ordinary Differential Equation Initial Value Problems. The second section will delve into the theoretical core and mathematical details of the PINN algorithm, explaining how a neural network parameterizes the solution, how a composite loss function incorporating the ODE dynamics residual and initial condition constraints is constructed, and how the optimization process works. The third section will present a concrete case study of a two-dimensional dynamical system with a known analytical solution, visualizing the results from multiple perspectives (including solution accuracy, error distribution, and phase-space trajectory) for intuitive analysis and discussion. The fourth section will conclude the paper, summarizing the advantages and potential challenges of PINNs for solving IVPs and offering an outlook on future research directions.

II. Detailed Introduction to Solving IVPs with the PINN Algorithm

The core idea behind solving Initial Value Problems with Physics-Informed Neural Networks is to reframe the process of solving a differential equation as an optimization problem guided by both data and physical laws. The neural network serves as the "solution carrier," and its training process is a search for the optimal network parameters that enable the function it represents to simultaneously satisfy the initial conditions and the governing dynamical equations over the entire time domain.

2.1 The Neural Network as a Universal Function Approximator

The foundation of PINNs is the powerful function representation capability of neural networks. According to the Universal Approximation Theorem, a feedforward neural network with sufficient width and depth, using a non-linear activation function, can approximate any continuous function on a compact set to an arbitrary degree of accuracy. For an IVP, the solution $\mathbf{u}(t)$ is a continuous function from time t to

Jiaqun Wang, "Physics-Informed Neural Networks for Solving Initial Value Problems," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 8, Issue 3, pp. 216-221, 2025.

the state space R^D. Therefore, we can construct a neural network $\hat{\mathbf{u}}(t;\theta)$ to serve as a surrogate model or a parameterized form of the true solution $u(t)$.

The structure of this neural network is typically as follows:

Input Layer: Receives a single scalar input, time t.

Hidden Layers: Comprise several layers, each with multiple neurons. Neurons are connected through linear transformations involving weights and biases, followed by a non-linear activation function (such as Tanh, SiLU, or Sigmoid). These hidden layers form the core of the network, responsible for learning the complex non-linear mapping between time t and state $u(t)$. The network's depth (number of layers) and width (neurons per layer) determine its expressive power.

Output Layer: Outputs a D-dimensional vector, corresponding to the system's state $[x_1, x_2, \cdots, x_D]^{\mathrm{T}}$ at time t.

The entire neural network can be viewed as a complex composite function $\hat{\mathbf{u}}(t;\theta)$, where θ represents all the trainable parameters of the network (i.e., all weights $\boldsymbol{W}$ and biases b of all layers). Our goal is to find the optimal set of parameters θ^* such that $\hat{\mathbf{u}}(t;\theta^*)$ is as close as possible to the true solution $\mathbf{u}(t)$

2.2 Construction of the Physics-Informed Loss Function

How do we determine how good a set of parameters θ is? This requires defining a Loss Function $L(\theta)$. This loss function must quantify the extent to which the function represented by the neural network, $\hat{\mathbf{u}}(t;\theta)$, "violates" the requirements of the IVP. For an IVP, there are two requirements: satisfying the initial conditions and satisfying the dynamical equations. Therefore, the PINN loss function is composed of these two parts.

Let's consider a first-order system of Ordinary Differential Equations (ODEs) defining an IVP of the form:

$$\frac{d\mathbf{u}}{dt} = \mathrm{F}\ (t, \mathbf{u}), \quad t \in [0, T] \qquad (1)$$

subject to the initial condition: $\mathbf{u}(0) = \mathbf{u}_0$, where $\mathbf{u}(t) \in \mathrm{R}^D$ is the D-dimensional state vector, F is a (possibly nonlinear) function that defines the system's dynamics, and u_0 is the known initial state vector.

This is the most straightforward part of the loss function. It measures the discrepancy between the neural network's prediction at the initial time t=0 and the true initial value u_0. We typically use the Mean Squared Error (MSE) to quantify this gap.

By feeding t=0 into the neural network, we obtain the predicted initial state $\hat{\mathbf{u}}(0;\theta)$. The initial condition loss L_{IC} is defined as:

$$L_{IC}(\theta) = \| \hat{\mathbf{u}}(0;\theta) - \mathbf{u}_0 \|_2^2 \qquad (2)$$

Minimizing L_{IC} drives the neural network's output to be precisely "pinned" to the given initial condition at the point t=0.

This is the most innovative part of the PINN methodology, as it encodes the physical law (in the form of the ODE) into the loss function. We define a dynamical residual function $\mathbf{f}(t;\theta)$ that represents the extent to which the neural network's solution fails to satisfy the ODE:

$$\mathbf{f}(t;\theta) = \frac{d\mathbf{u}(t;\theta)}{dt} - \mathrm{F}\ (t, \mathbf{u}(t;\theta)) \qquad (3)$$

If $\hat{\mathbf{u}}(t;\theta)$ were the exact solution to the ODE, then the residual $\mathbf{f}(t;\theta)$ would be identically zero across the entire time domain [0,T]. Our goal, therefore, is to make this residual function as small as possible throughout the domain.

To achieve this in practice, we employ a strategy based on collocation points. We sample a large number of points $\{t_f^i\}_{i=1}^{N_f}$ either randomly or uniformly from within the time domain [0,T]. At these points, we calculate the ODE residual and use its mean squared error as the dynamical equation residual loss L_{ODE}

$$L_{ODE}(\theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} \| \mathbf{f}(t_f^i;\theta) \|_2^2 \qquad (4)$$

The number of collocation points, N_f, is typically large (in the thousands or tens of thousands) to ensure that the neural network is constrained by the physical law across the entire domain.

A critical step in calculating L_{ODE} is computing the derivative of the network's output $\hat{\mathbf{u}}(t)$ with respect to its input t i.e., $d\hat{\mathbf{u}}(t)/dt$. Traditional numerical methods would approximate this derivative using finite differences, such as $[\hat{\mathbf{u}}(t+\Delta t) - \hat{\mathbf{u}}(t)]/\Delta t$, which introduces discretization errors.

The elegance of PINNs lies in their complete avoidance of such approximations. Since a neural network is an analytical function composed of a series of analytical operations (linear transformations and activation functions), its derivative can also be computed analytically. One of the core features of modern deep learning frameworks like PyTorch and TensorFlow is Automatic Differentiation (AD). AD utilizes the chain rule to compute the derivative of any complex function accurately and efficiently, with precision limited only by the machine's floating-point accuracy. Thus, PINNs can obtain the exact analytical value of $d\hat{\mathbf{u}}(t)/dt$, ensuring that the calculation of the ODE residual introduces no numerical approximation error. This is a significant advantage of PINNs over some traditional methods.

Finally, the total loss function $L(\theta)$ is a weighted sum of the initial condition loss and the dynamical equation loss:

$$L(\theta) = w_{ode} L_{ODE}(\theta) + w_{ic} L_{IC}(\theta) \qquad (5)$$

where w_{ode} and w_{ic} are weight coefficients used to balance the importance of the two loss terms. In practice, they are often set to 1, but for some complex or stiff problems, dynamically

Jiaqun Wang, "Physics-Informed Neural Networks for Solving Initial Value Problems," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 8, Issue 3, pp. 216-221, 2025.

adjusting these weights (e.g., using an adaptive weighting scheme) might improve training performance.

With the total loss function defined, solving the IVP is transformed into a standard deep learning optimization problem:

$$\theta^* = \arg\min_{\theta} L(\theta) \qquad (6)$$

We use a gradient-based optimization algorithm to find the optimal parameters θ^*. The most common algorithm is Adam (Adaptive Moment Estimation), which combines the advantages of momentum and RMSProp to search efficiently on complex loss surfaces. The training process is as follows:

1. Initialization: Randomly initialize the neural network's parameters θ,
2. Forward Pass:
   - Feed the initial point t=0 and a batch of collocation points $\{t_f^i\}$ into the network to get predictions $\hat{\mathbf{u}}(0;\theta)$ and $\hat{\mathbf{u}}(t_f^i;\theta)$
   - Use automatic differentiation to compute the derivatives $d\hat{\mathbf{u}}(t_f^i;\theta)/dt$
   - Calculate L_{IC} and L_{ODE} according to their formulas, and then the total loss $L(\theta)$.
3. Backward Pass (Backpropagation): Use automatic differentiation to compute the gradient of the total loss $L(\theta)$ with respect to every network parameter θ_j
4. Parameter Update: The optimizer (e.g., Adam) updates the network parameters θ based on the computed gradients, moving them in the direction that decreases the loss: $\theta_{new} = \theta_{old} - \eta \nabla_\theta L(\theta)$ (where η is the learning rate).
5. Iteration: Repeat steps 2-4 until the loss function converges to a sufficiently small value or a preset number of training epochs is reached.

Once training is complete, we obtain an optimized neural network $\hat{\mathbf{u}}(t;\theta^*)$. This network itself is a continuous, analytical approximate solution to the IVP. We can input any time point $t \in [0,T]$ of interest into the network to get an accurate prediction of the system's state at that moment.

III. Case Study

To concretely demonstrate the process and effectiveness of PINNs for solving IVPs, we choose a classic two-dimensional dynamical system with a known analytical solution: the Simple Harmonic Oscillator. The advantage of selecting a problem with an analytical solution is that we can directly compare the PINN's predictions with the true solution, allowing for a precise evaluation of its performance.

We consider an undamped simple harmonic motion described by the second-order ODE:

$$\frac{d^2x}{dt^2} + x = 0 \qquad (7)$$

To convert this into a first-order ODE system, we introduce the state variable $y = -dx/dt$. The system can then be rewritten as:

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = -x \end{cases} \qquad (8)$$

This is a two-dimensional linear dynamical system with the state vector $\mathbf{u}(t)=[x(t),y(t)]^T$. We set the following initial conditions: $x(0)=0, y(0)=1$.

Physically, this corresponds to an oscillator starting from its equilibrium position with a unit positive initial velocity. The analytical solution to this IVP is straightforward:

$$\begin{aligned} x(t) &= sin(t) \\ y(t) &= cos(t) \end{aligned} \qquad (9)$$

In the phase space (x–y plane), the system's trajectory is a unit circle centered at the origin, traversed in a clockwise direction. Our goal is to train a PINN to "discover" this solution using only the dynamical equations and the initial conditions.

After 10,000 epochs of training (which typically takes only 71s on a consumer-grade CPU), we obtain a trained PINN model. Next, we evaluate its performance from multiple perspectives.

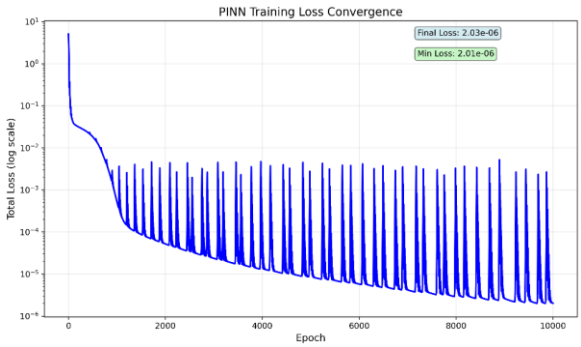

Figure 1. Loss Function Convergence Curve

First, we plot the total loss function as it changes with the number of training epochs. Figure 1 shows that the total loss decreases rapidly in the early stages of training and then stabilizes after about 2,000 epochs, eventually converging to a very low value (on the order of 10^{-4}). This indicates that the optimization process was successful and the neural network parameters have been adjusted to a state that satisfies both the ODE and the initial conditions well. The logarithmic scale shows that the loss consistently and steadily decreases throughout the training, without violent oscillations, indicating a stable training process.

To quantitatively assess the accuracy, we plot the absolute error between the predicted and true solutions over time. Figure 2 shows that for both x(t) and y(t), the absolute error remains at a very low level throughout the time domain, with the maximum error not exceeding 10^{-3}. This indicates that the PINN's predictions have high and consistent accuracy.

Jiaqun Wang, "Physics-Informed Neural Networks for Solving Initial Value Problems," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 8, Issue 3, pp. 216-221, 2025.

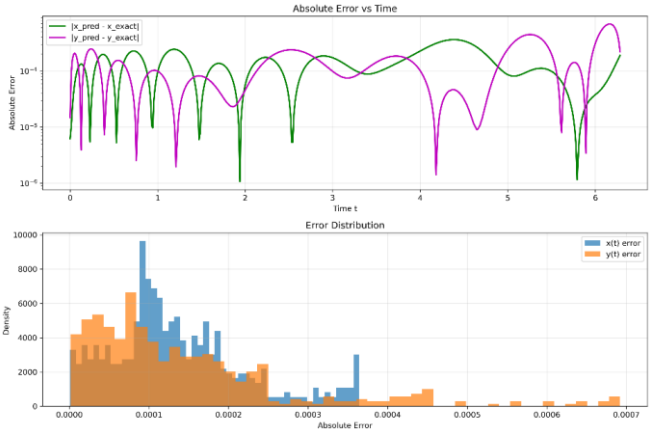

Figure 2 Absolute Error

Finally, we examine whether the model has learned the overall dynamical structure of the system, which can be visualized by comparing the trajectories in the phase space. As shown in Figure 3, the phase-space trajectory predicted by the PINN (red dashed line) perfectly overlaps with the analytical unit circle trajectory (blue solid line). This indicates that the PINN has not only fitted the function values at isolated time points but, more importantly, has successfully captured the intrinsic relationship between $x(t)$ and $y(t)$ dictated by the dynamical equations. This is crucial for understanding and predicting the long-term behavior of the system.

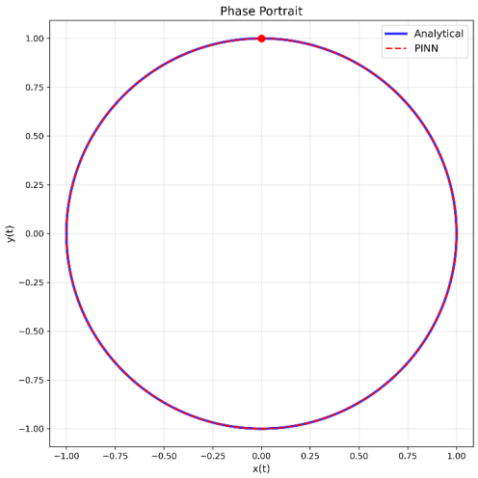

Figure 3: Phase Portrait Comparison

IV. Conclusion

This paper has systematically investigated the application of Physics-Informed Neural Networks (PINNs) for solving Ordinary Differential Equation Initial Value Problems (IVPs). Starting from the importance of IVPs in science and engineering, we reviewed the achievements and limitations of traditional numerical methods, which motivated the introduction of PINNs as an emerging paradigm in scientific computing.

On a theoretical level, we detailed the core idea of PINNs: transforming the problem of solving a differential equation into an optimization problem. By using a neural network as a universal function approximator to parameterize the system's solution and constructing a composite loss function that includes both the dynamical equation (ODE) residual and the initial condition (IC) constraints, PINNs directly encode physical laws into the deep learning training process. We specifically highlighted the pivotal role of automatic differentiation, which enables the precise calculation of physical residuals, thereby avoiding the discretization errors inherent in traditional numerical differencing schemes.

In summary, PINNs exhibit several key advantages for solving IVPs:

- High Accuracy: By leveraging automatic differentiation, they can impose physical constraints precisely, leading to highly accurate solutions.
- Flexibility and Extensibility: The PINN framework is easily extensible, capable of handling not only nonlinear and high-dimensional systems but also being naturally suited for inverse problems like parameter identification.

However, the PINN methodology still faces some challenges. For instance, errors may accumulate in long-term integration problems. For stiff problems, the optimization of the loss function can become very difficult, leading to training failure or slow convergence. Furthermore, the choice of neural network architecture, hyperparameter tuning, and the weighting strategy for different terms in the loss function still rely heavily on heuristics and experimentation, lacking mature theoretical guidance.

In conclusion, as a bridge connecting physics and artificial intelligence, Physics-Informed Neural Networks offer a powerful and promising new tool for solving Initial Value Problems and, more broadly, a wide range of scientific computing challenges. Although there are still hurdles to overcome, with continuous improvements in theory and practice, PINNs are poised to play an increasingly important role in future scientific discoveries and technological innovations.

This work was supported by the Project for Scientific Research of Universities in Anhui Province (Grant No.: 2024AH050014） and Quality Engineering Project of Anhui University of Finance and Economics (Grant No. acszjyyb2024008).

References

[1] Liu G R. Meshfree methods: Moving beyond the finite element method, second edition [M]. 2009.
[2] Seydel R. Practical bifurcation and stability analysis [M]. New York, NY: Springer, 2010.
[3] Burden A M, Burden R L, Faires J D. Numerical analysis, 10th ed. [M]. 2016.
[4] Butcher J C. Numerical methods for ordinary differential equations [M]. Third ed. Chichester: John Wiley & Sons, Ltd, 2016.
[5] Newmark N M. A method of computation for structural dynamics [J]. Journal of the Engineering Mechanics Division, 1959, 85(3): 67-94.
[6] Wilson E L, Farhoomand I, Bathe K-J. Nonlinear dynamic analysis of complex structures [J]. Earthquake Engineering & Structural Dynamics, 1972, 1(3): 241-52.
[7] Bathe K-J. Finite element procedures (second) [M]. Watertown: Prentice Hall, 2014.
[8] Wen W B, Deng S Y, Liu T H, et al. An improved quartic b-spline based explicit time integration algorithm for structural dynamics [J]. European Journal of Mechanics a-Solids, 2022, 91: 23.

[9] Zhang H M, Zhang R S, Zanoni A, et al. A novel explicit three-sub-step time integration method for wave propagation problems [J]. Archive of Applied Mechanics, 2022, 92(3): 821-52.
[10] Raissi M. Deep hidden physics models: Deep learning of nonlinear partial differential equations [J]. J Mach Learn Res, 2018, 19: 24.
[11] Pang G F, Lu L, Karniadakis G E M. Fpinns: Fractional physics-informed neural networks [J]. Siam Journal on Scientific Computing, 2019, 41(4): A2603-A26.
[12] Raissi M, Perdikaris P, Karniadakis G E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations [J]. Journal of Computational Physics, 2019, 378: 686-707.
[13] Chew A W Z, He R F, Zhang L M. Physics informed machine learning (piml) for design, management and resilience-development of urban infrastructures: Concept, state-of-the-art, challenges and opportunities [J]. Archives of Computational Methods in Engineering, 2024: 40.
[14] Herrmann L, Kollmannsberger S. Deep learning in computational mechanics: A review [J]. Computational Mechanics, 2024: 51.
[15] Wang F J, Zhai Z, Zhao Z B, et al. Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis [J]. Nat Commun, 2024, 15(1): 12.

Jiaqun Wang, “Physics-Informed Neural Networks for Solving Initial Value Problems,” *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 8, Issue 3, pp. 216-221, 2025.