Hindawi
International Journal of Intelligent Systems
Volume 2024, Article ID 6641674, 12 pages
https://doi.org/10.1155/2024/6641674

WILEY | Hindawi

Research Article

Physics-Informed Neural Networks for Solving High-Index Differential-Algebraic Equation Systems Based on Radau Methods

Jiasheng Chen,[1] **Juan Tang**,[1,2,3,4] **Ming Yan**,[3,4] **Shuai Lai**,[2] **Kun Liang**,[1] **Jianguang Lu**,[5] **and Wenqiang Yang**[6]

[1]*Institute of Computing Science and Technology, Guangzhou University, Guangzhou, China*
[2]*School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China*
[3]*Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore*
[4]*Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore*
[5]*State Key Laboratory of Public Big Data, Guizhou University, Guiyang, China*
[6]*Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China*

Correspondence should be addressed to Juan Tang; tangjn16@gzhu.edu.cn

Received 24 October 2023; Revised 22 February 2024; Accepted 26 February 2024; Published 29 March 2024

Academic Editor: Subrata Kumar Sarker

As is well known, differential algebraic equations (DAEs), which are able to describe dynamic changes and underlying constraints, have been widely applied in engineering fields such as fluid dynamics, multi-body dynamics, mechanical systems, and control theory. In practical physical modeling within these domains, the systems often generate high-index DAEs. Classical implicit numerical methods typically result in varying order reduction of numerical accuracy when solving high-index systems. Recently, the physics-informed neural networks (PINNs) have gained attention for solving DAE systems. However, it faces challenges like the inability to directly solve high-index systems, lower predictive accuracy, and weaker generalization capabilities. In this paper, we propose a PINN computational framework, combined Radau IIA numerical method with an improved fully connected neural network structure, to directly solve high-index DAEs. Furthermore, we employ a domain decomposition strategy to enhance solution accuracy. We conduct numerical experiments with two classical high-index systems as illustrative examples, investigating how different orders and time-step sizes of the Radau IIA method affect the accuracy of neural network solutions. For different time-step sizes, the experimental results indicate that utilizing a 5th-order Radau IIA method in the PINN achieves a high level of system accuracy and stability. Specifically, the absolute errors for all differential variables remain as low as 10^{-6}, and the absolute errors for algebraic variables are maintained at 10^{-5}. Therefore, our method exhibits excellent computational accuracy and strong generalization capabilities, providing a feasible approach for the high-precision solution of larger-scale DAEs with higher indices or challenging high-dimensional partial differential algebraic equation systems.

1. Introduction

The concept of differential algebraic equations (DAEs) was formally proposed by Gear in the study of network analysis and continuous system simulation problems [1]. Petzold made it explicit through her study of numerical methods that DAEs are not ordinary differential equations (ODEs) [2]. DAE systems are composed of coupled ODE systems and algebraic equation systems with physical significance. These systems encompass both differential and algebraic variables, and their system form is more generalized compared to traditional ODE systems. DAEs have gained significant attention since their inception, as they can accurately describe systems that some ODEs cannot represent. They have found extensive applications in various fields, including fluid dynamics, multi-body dynamics, electronic circuits, mechanical systems, control theory, and chemical engineering.

In different developmental periods and research fields, DAEs are also known as singular systems, general systems, descriptor systems, or constrained systems, among other

names. They often exhibit various structural forms, such as linear DAEs, nonlinear DAEs, semi-explicit DAEs, implicit DAEs, and Hessenberg-type DAEs. Fortunately, in practical physical modeling, most of the system models obtained are either low-index DAEs or high-index (≥ 2) Hessenberg-type DAEs [3]. The index of DAEs measures the "distance" between DAEs and ODEs. Generally, a higher index implies greater difficulty in transforming DAEs into ODEs or in directly solving DAEs using ODE numerical methods. Traditional numerical methods for solving DAE systems include implicit Runge–Kutta methods [4], BDF methods [5], pseudospectral methods [6], Adomian decomposition method [7], exponential integrators [8], generalized-α methods [9], and Lie group methods [10–12]. It is worth noting that these direct numerical methods can solve DAEs with an index of 1. However, for high-index DAE systems, these methods are only applicable to a certain class of DAEs and may result in varying order reduction of numerical accuracy.

With the rapid advancement of neural network technology and hardware resources, neural networks demonstrate increasingly powerful capabilities. Compared to traditional numerical computing methods, neural networks offer several advantages, including strong generalization, fault tolerance, and the ability for parallel computation. In 1998, Lagaris et al. [13] approximated solutions to ODEs or PDEs problems by constructing parameterized trial functions. These trial functions consist of two parts: one part satisfies initial conditions or boundary conditions which do not contain trainable parameters, while the other part is a simple feed forward neural network with trainable parameters. In 2019, Raissi et al. [14] introduced an important technique known as physics-informed neural networks (PINNs) for the numerical approximation of partial differential equation (PDE) problems. The PINN loss function includes not only initial or boundary conditions that reflect physical properties but also a residual term at selected points in the time-space domain where the PDEs hold. It is worth noting that PINN is a data-driven approach that does not require prior knowledge of the analytical form of the solution; instead, it learns the solution from data. Various variants of PINN have been proposed based on different collocation methods, such as variational hp-VPINN [15] and conservative PINN (CPINN) [16]. Specifically, Wu et al. [17] recently introduced an innovative PINN designed to address Hausdorff derivative Poisson equations across irregular domains. This method leverages the Hausdorff fractal derivative to reformulate the numerical resolution of partial differential equations into an optimization challenge, encompassing the principal equation and its boundary conditions. Notably, this technique is characterized by its simplicity, clarity, and programming convenience. Additionally, PINN has been widely applied to solve problems in various fields, including fluid dynamics [18–21], seismic wave prediction [22], and optical problems [23].

In recent years, many researchers have attempted to construct neural network models from different perspectives to solve various types of DAEs systems influenced by these methods. For Hessenberg-DAEs with control variables and an index of 3, Kozlov and Tiumentsev [24] achieved the implementation of BDFs method using a semi-empirical neural network model. Zhao et al. [25] constructed a single-layer feed-forward neural network (FFNN) to solve Hessenberg-type DAEs systems. They augmented the loss function in their special Euler–Lagrange equation system with penalty terms for algebraic equations to avoid drifting in the results. Experimental results in their paper showed that the FFNN method with the Sigmoid activation function provided approximate analytical solutions close to the numerical solutions of corresponding Runge–Kutta methods, but they did not provide further details about the method's accuracy. For linear DAEs systems, Liu et al. [26] selected Jacobi polynomials as activation functions and constructed a single-hidden-layer feed-forward neural network (JNN). They determined the network parameters using the classical ELM algorithm. Through experimental comparisons with other approximation methods such as Padé approximation, ADM method, and Adams methods, they illustrated the feasibility and superiority of the JNN method. It is worth noting that the examples in the paper involve DAEs with an index of 1 or linear DAEs that have been reduced to index 1. For DAEs systems with an index of 1, Moya and Lin [27] proposed a neural network architecture called DAE-PINN based on the PINN method for solving DAEs systems. This neural network model is a discrete-time model based on the implicit Runge–Kutta method, which can directly address most index-1 differential-algebraic equation problems. However, it cannot solve high-index DAEs problems and suffers from low accuracy issues. To address the high-accuracy computation challenges in high-index DAEs systems, we have combined the Radau IIA numerical method with an improved fully connected neural network. We have proposed a PINN computational framework based on the Radau method. Furthermore, we have improved the efficiency and accuracy of the solution by applying a strategy of domain decomposition.

In Section 2, we briefly introduce the fundamental concepts of DAEs systems, the Radau IIA numerical method, the improved fully connected neural network, and the discrete-time model of PINN. Building upon this foundation, we provide a detailed construction of the PINN computing framework based on the Radau IIA method. Additionally, we employ a time domain decomposition strategy for neural network. Section 3 uses the neural network designed in this paper to solve two high-index DAEs systems, and we analyze the solving accuracy of this neural network. Finally, we discuss and summarize the advantages, challenges, and potential avenues for improvement in the Radau-PINN architecture.

2. Scientific Machine Learning Methods

This section first sequentially introduces the basic concepts of DAEs and the classical Radau IIA numerical method. Then, we introduce an improved fully connected neural network architecture and the discrete-time model of PINN. Building upon this, we construct a PINN based on the Radau IIA method. Finally, we enhance the

efficiency and accuracy of neural network solutions for DAEs systems by utilizing the concept of domain decomposition.

2.1. Radau IIA Method for DAE Systems. This article first provides a brief introduction to DAEs with an index of 2, with the specific form as follows:

$$\begin{cases} y'(t) = f(t, y(t), z(t)), \\ 0 = g(t, y(t)), \end{cases} \tag{1}$$

where $y(t) \in \mathrm{R}^n$ is the differential function variable, $z(t) \in \mathrm{R}^m$ is the algebraic function variable, $t \in [t_0, T]$, t_0 is the initial time point, and $y_0 = y(t_0)$ is the initial value. Both $f(t, y, z) \in \mathrm{R}^n$ and $g(t, y) \in \mathrm{R}^m$ are sufficiently smooth, and the Jacobian matrix $g_y f_z$ is non-singular.

The Radau IIA method is a class of implicit Runge–Kutta methods, typically defined in the following general form:

$$\begin{aligned} \xi_i &= y_n + h \sum_{j=1}^{v} a_{i,j} f\left(\xi_j, \zeta_j\right), \\ g\left(\xi_i\right) &= 0, \\ y_{n+1} &= y_n + h \sum_{j=1}^{v} b_j f\left(\xi_j, \zeta_j\right), \\ g\left(y_{n+1}\right) &= 0, \end{aligned} \tag{2}$$

where $\xi_i = y(t_n + c_i h)$, $\zeta_i = z(t_n + c_i h)$, h is the step size, n is the current step number, $\left\{a_{\mathrm{ij}}, b_j, c_i\right\}$ are parameters, and $c_i = \sum_{j=1}^{v} a_{\mathrm{ij}}$, $i, j = 1, \ldots, v$.

In Table 1, different sets of parameters lead to different implicit Runge–Kutta methods, such as commonly used Gauss method, Radau method, and Lobatto method. These parameters are determined using Gauss polynomials, Radau polynomials, and Lobatto polynomials, respectively. Among them, the Radau IIA method is a high-precision numerical method with excellent numerical stability. Therefore, in this paper, the Radau IIA method is chosen, and the parameters need to satisfy the following conditions:

$$\begin{aligned} B(2v-1)&: \sum_{i=1}^{v} b_i c_i^{k-1} = \frac{1}{k}, \quad k = 1, \ldots, 2v-1, \\ C(v)&: \sum_{j=1}^{v} a_{\mathrm{ij}} c_j^{k-1} = \frac{c_i^k}{k}, \quad k = 1, \ldots, v, \\ D(v-1)&: \sum_{i=1}^{v} b_i c_i^{k-1} a_{\mathrm{ij}} = \frac{b_j}{k}\left(1 - c_j^k\right), \quad k = 1, \ldots, v-1, \end{aligned} \tag{3}$$

and $c_v = 1$, $b_j = a_{\mathrm{vj}}$, $i, j = 1, 2, \ldots, v$.

2.2. Improved Fully Connected Neural Network Structure. Building upon the DAE-PINN structure, we employ adaptive activation functions (4) to train an improved fully connected neural network structure. The specifics are as follows:

Table 1: The parameter table of the v-stage implicit Runge-Kutta methods.

c_1	a_{11}	a_{12}	a_{13}	$\cdots$	a_{1v}
c_2	a_{21}	a_{22}	a_{23}	$\cdots$	a_{2v}
c_3	a_{31}	a_{32}	a_{33}	$\cdots$	a_{3v}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_v	a_{v1}	a_{v2}	a_{v3}	$\cdots$	a_{vv}
	b_1	b_2	b_3	$\cdots$	b_v

The architecture of the improved fully connected neural network model is primarily constructed using two transformer networks, denoted as U and R, to build two stacked-layer networks, as illustrated in Figure 1. Both neural networks map the input variable X (differential function variable y) to a high-dimensional feature space. Subsequently, each hidden layer forms new residual connections using element-wise multiplication operations, as expressed below:

$$\begin{aligned} U &= \varphi\left(\mathrm{XW}^1 + b^1\right), \\ R &= \varphi\left(\mathrm{XW}^2 + b^2\right), \\ H^{(1)} &= \varphi\left(\eta \cdot l \cdot \mathrm{XW}^{o,1} + b^{o,1}\right), \\ M^{(k)} &= \varphi\left(H^{\mathrm{k}} W^{o,k} + b^{o,k}\right), \\ H^{(k+1)} &= \left(1 - M^{(k)}\right) \odot U + M^{(k)} \odot R, \\ P_\theta(X) &= H^{\mathrm{d}+1} W + b, \end{aligned} \tag{4}$$

where X represents the input vector of the neural network, $W^{o,k}$ is the collection of weights for the o-th neuron in the k-th layer, $b^{o,k}$ denotes the set of biases for the o-th neuron in the k-th layer, φ is the activation function, $\odot$ represents element-wise multiplication, d indicates the number of hidden layers (the depth of the neural network), $P_\theta(X)$ is the final output vector of the neural network, η is a predetermined hyper-parameter that ensures the slope is greater than 1, and l is a parameter that can modify the slope of the activation function.

2.3. Discrete-Time Model of PINN. In this section, we will provide a detailed introduction to the PINN for discrete-time model as outlined in [14].

The main idea of the discrete-time model in PINN is to integrate neural networks with traditional Runge–Kutta methods. Considering the general form of PDE, it can be expressed as follows:

$$u_t + \mathcal{N}[u] = 0, x \in \Omega, t \in [0, T], \tag{5}$$

where $u(t, x)$ denotes the solution of PDE, $\mathcal{N}[\cdot]$ is a nonlinear differential operator, and Ω is a subset of $\mathbb{R}^D$.

Substituting the general form of the v-stage Runge-Kutta method into the above PDE (5), we can obtain as follows:

$$\begin{aligned} u^{n+c_i} &= u^n - h \sum_{j=1}^{v} a_{\mathrm{ij}} \mathcal{N}\left[u^{n+c_j}\right], \; i = 1, \ldots v, \\ u^{n+1} &= u^n - h \sum_{j=1}^{v} b_j \mathcal{N}\left[u^{n+c_j}\right]. \end{aligned} \tag{6}$$

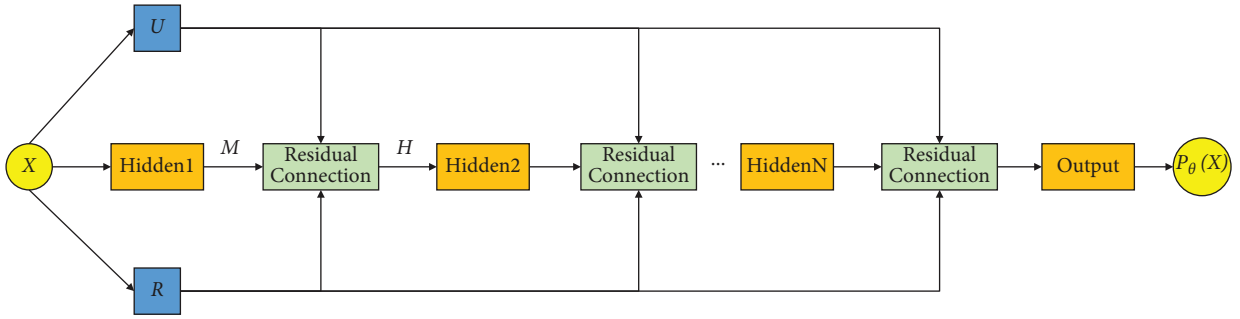

Figure 1: Improved fully connected neural network structure.

Here, $u^{n+c_j}(x) = u(t^n + c_j h, x)$ for $j = 1, \ldots, v$, h is the step size. For ease of writing and comprehension, equation (6) above can be equivalently represented as follows:

$$\begin{aligned} u^n &= u_i^n, \quad i = 1, \ldots, v, \\ u^n &= u_{v+1}^n, \end{aligned} \tag{7}$$

where

$$\begin{aligned} u_i^n &:= u^{n+c_i} + h \sum_{j=1}^{v} a_{ij} \mathcal{N}\left[u^{n+c_j}\right], \quad i = 1, \ldots v, \\ u_{q+1}^n &:= u^{n+1} + h \sum_{j=1}^{v} b_j \mathcal{N}\left[u^{n+c_j}\right]. \end{aligned} \tag{8}$$

The output layer of the PINN for discrete-time models has a number of neurons equal to the stage of the Runge–Kutta method plus one $(v+1)$. The output layer is described as follows:

$$\left[u^{n+c_1}(x), \ldots, u^{n+c_q}(x), u^{n+1}(x)\right]. \tag{9}$$

By integrating the Runge–Kutta formulas (8) and the output values of the PINN for discrete-time model (9), we can derive a neural network architecture with x as input and

$$\left[u_1^n(x), \ldots, u_v^n(x), u_{v+1}^n(x)\right], \tag{10}$$

as output.

In the discrete-time model of a PINN, the time axis is typically divided into several time steps. Besides spatial coordinates, the neural network's input also includes information about the current time step. This enables the neural network to learn temporal dynamics and predict the solution at each time step. Based on the PINN's discrete-time model, we have constructed a PINN architecture based on the Radau IIA method for solving high-index DAEs, which will be described in the following section.

2.4. PINN Based on Radau IIA Method. In this section, we use the discrete-time model of PINN as the foundation, incorporating an improved fully connected neural network. We have constructed a PINN architecture based on the Radau IIA method, as illustrated in Figure 2.

Firstly, construct a neural network with multiple inputs and multiple outputs, where the inputs consist of the collection of differential variables y_n, and outputs

$$\begin{aligned} &\xi_1^\theta, \xi_2^\theta, , \ldots\ldots, \xi_v^\theta, y_{n+1}^\theta;, \\ &\zeta_1^\theta, \zeta_2^\theta, , \ldots\ldots, \zeta_v^\theta, z_{n+1}^\theta. \end{aligned} \tag{11}$$

The first v values of ξ_i^θ represent intermediate differential variables, and the first v values of ζ_i^θ represent intermediate algebraic variables, where $i = 1, 2, \ldots\ldots, v$.

Secondly, based on the structure of the DAEs system with an index of 2 and the characteristics of the Radau IIA method, we further design an improved fully connected neural network for both the differential variable part and the algebraic variable part of the system. There are two specific design approaches: one assigns a single neural network to all differential variables and two neural networks to the algebraic variables, and the other assigns a separate neural network to each differential variable while keeping the algebraic variable part unchanged. In the case of the algebraic variable part, one of the neural networks is used to predict the first v values, while the other neural network is used to predict the $v+1$-th value. Theoretically, the second method, illustrated in Figure 2, involves creating a separate neural network for each differential or algebraic variable. This approach differs from the first by effectively increasing the size of the network. Additionally, it offers the benefit of avoiding the negative impact on other variables that could result from unsuccessful parameter optimization of a single variable during training. As a result, this enhances both the precision and the ability of the model to generalize. Through further testing, the second approach's training results are more precise than those of the first approach, consistent with the expected results. As a result, all subsequent experiments in this paper are implemented based on the second approach.

Thirdly, based on the designed network structure, this paper constructs the loss function as follows:

$$\mathcal{L}(\theta; \mathcal{T}) = W_f \mathcal{L}_f(\theta; \mathcal{T}) + W_g \mathcal{L}_g(\theta; \mathcal{T}) + W_s \mathcal{L}_s(\theta; \mathcal{T}). \tag{12}$$

(i) $\mathcal{L}_f(\theta; \mathcal{T})$ is the loss related to the differential network and is expressed as follows:

$$\begin{aligned} &\frac{1}{N_{\mathcal{T}}(v+1)} \sum_{k=1}^{N_{\mathcal{T}}} \sum_{i=1}^{v+1} \left\| y_{n,k} - y_{n,k}^i(\theta) \right\|_2^2, \\ &y_{n,k}^i(\theta) = \xi_{i,k}^\theta - h \sum_{j=1}^{v} a_{i,j} f\left(\xi_{j,k}^\theta, \zeta_{j,k}^\theta\right), \quad i = 1, \ldots, v, \\ &y_{n,k}^{v+1}(\theta) = y_{n+1,k}^\theta - h \sum_{i=1}^{v} b_i f\left(\xi_{i,k}^\theta, \zeta_{i,k}^\theta\right);. \end{aligned} \tag{13}$$

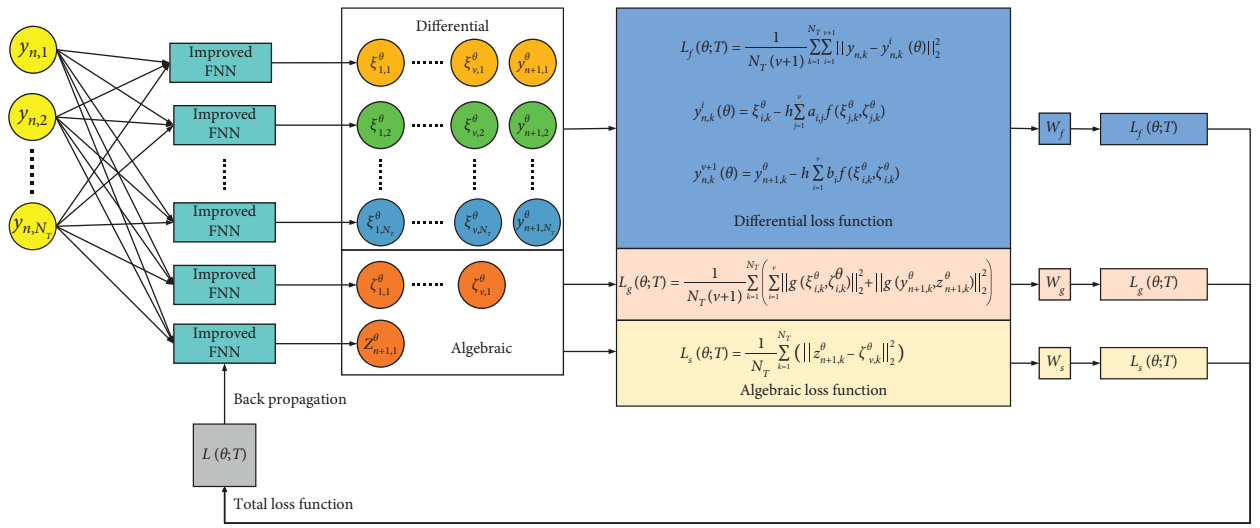

FIGURE 2: The schematic diagram of PINN based on Radau IIA method.

(ii) $\mathscr{L}_g(\theta;\mathscr{T})$ is the loss associated with the algebraic network and can be expressed as follows:

$$\frac{1}{N_{\mathscr{T}}(\nu+1)}\sum_{k=1}^{N_{\mathscr{T}}}\left(\sum_{i=1}^{\nu}\left\|g\left(\xi_{i,k}^{\theta},\zeta_{i,k}^{\theta}\right)\right\|_2^2+\left\|g\left(y_{n+1,k}^{\theta},z_{n+1,k}^{\theta}\right)\right\|_2^2\right);. \tag{14}$$

(iii) $\mathscr{L}_s(\theta;\mathscr{T})$ is the loss related to the last value of the controlled algebraic variable and can be expressed as follows:

$$\frac{1}{N_{\mathscr{T}}}\sum_{k=1}^{N_{\mathscr{T}}}\left(\left\|z_{n+1,k}^{\theta}-\zeta_{\nu,k}^{\theta}\right\|_2^2\right), \tag{15}$$

where W_f represents the loss weight for the differential neural network, W_g is the weight for the algebraic neural network, and W_s signifies the weight for the control of algebraic variable prediction neural network. The parameters $a_{i,j}$ and b_i are specific to the Radau IIA method. $\mathscr{T}$ is the total number of samples, $N_{\mathscr{T}}$ is the number of training samples in the current batch, and θ denotes the neural network parameters. Here, f represents the differential network, and g represents the algebraic network. $y_{n,k}$ corresponds to the sample data of the model, $\xi_{i,k}^{\theta}$ stands for the values of intermediate differential variables, and $\zeta_{i,k}^{\theta}$ signifies the values of intermediate algebraic variables. Furthermore, $y_{n,k}^{i}(\theta)$ represents the output values of the differential neural network. The notation $\|\cdot\|_2^2$ refers to the square of the L2 norm, $z_{n+1,k}^{\theta}$ represents the final output of the algebraic neural network, and $\zeta_{n,k}^{\theta}$ denotes the penultimate output of the algebraic neural network.

Finally, we use gradient descent to solve for the weights, biases, and other parameters of the PINN,

$$\theta^{*}=\arg\min_{\theta}\mathscr{L}(\theta;\mathscr{T}). \tag{16}$$

2.5. Time-Domain Decomposition of Neural Networks. In this section, based on an analysis of the existing limitations of the PINN architecture, we adopt a time-domain decomposition strategy using neural networks.

One limitation of the PINN model is that it exhibits relatively low accuracy in predicting solutions. This is because the inherent inaccuracies involved in solving high-dimensional non-convex optimization problems can lead to local minima, making it challenging to achieve absolute errors below 10^{-5}. Another evident limitation is the high training cost [16]. Similarly, our proposed PINN model based on the Radau IIA method may encounter similar issues. Furthermore, the iterative format of the Radau IIA method does not fully exploit its high-precision advantages during training.

To address these issues, we propose a time-domain decomposition strategy for neural networks, as illustrated in Figure 3. With this approach, we partition the original problem into segments, enhancing solution accuracy while leveraging iterative training. In other words, the predicted values from the previous time segment can serve as input values for the subsequent segment. This means that knowing the data values at the initial point t_0 for the first segment is sufficient to iteratively compute the solutions over the entire time domain. This approach significantly reduces the amount of required data. Specifically, only the data at the initial point t_0 for a set of differential variables, denoted as y_0, are needed. Using the time-domain decomposition structure, we can iteratively determine the desired values within the range $[t_0, T]$. If we can obtain the initial values for each network at every time segment, parallel training of each neural network becomes possible, significantly reducing the model training time.

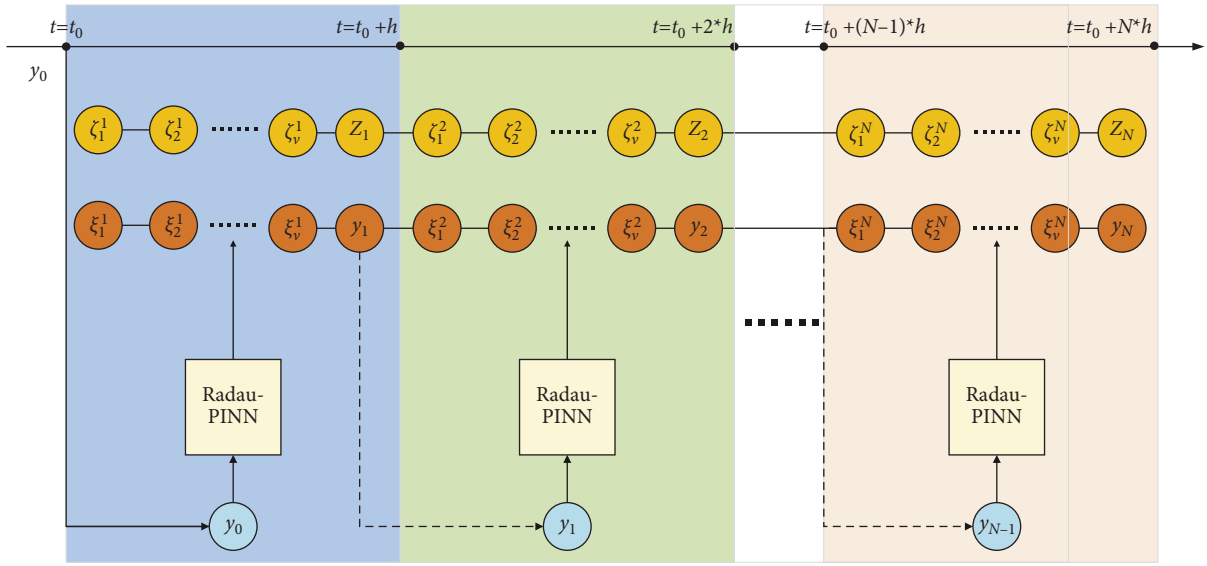

Figure 3: The time domain decomposition of neural networks.

3. Numerical Experiments

In this section, we apply PINN based on the Radau IIA method to solve two high-index DAEs systems separately and further investigate the influence of the order of the Radau IIA method on the solution results. The experiments were conducted on a Windows 10 operating system with an Intel(R) Core(TM) i7-10875H CPU @2.30 GHz processor. We used Python 3.9 software and coded the neural network architecture using PyTorch 1.12.1, the GPU version. Additionally, this paper involves two formulas to measure the accuracy of the experiments. One is the commonly used absolute error (AE) formula, defined as $\mathrm{AE} = |y_{\mathrm{true}} - y_{\mathrm{pred}}|$, which reflects the magnitude of the deviation between the neural network's predicted solution and the true solution. The other metric is the mean absolute error (MAE) formula, defined as $\mathrm{MAE} = (1/n)\sum_{i=1}^{n}|y^i_{\mathrm{true}} - y^i_{\mathrm{pred}}|$, used to assess the differences in accuracy among different orders of the Radau IIA method.

3.1. Hessenberg-Type DAEs System. In this section, we explore classical Hessenberg-type DAE systems with an index of 2 that possess exact analytical solutions [12], as follows:

$$\begin{cases} y_1'(t) = \left(y_3(t)y_4(t) + y_1(t)y_2(t)\right)y_5(t), \\ y_2'(t) = -y_3(t)y_4(t)^2y_2(t)^2y_5(t), \\ y_3'(t) = 2y_3(t)y_4(t)y_1(t)y_2(t), \\ y_4'(t) = -y_3(t)y_4(t)y_2(t)^2, \\ 0 = y_1(t)y_4(t) - y_2(t)y_3(t), \end{cases} \tag{17}$$

where $t \in [0, 1]$, and the initial values $y_0 = (1, 1, 1, 1, 1)$. The functions $y_1(t), y_2(t), y_3(t), y_4(t)$ represent differential variables, while $y_5(t)$ is an algebraic variable. The system's exact solution expressions are $y_1(t) = e^{2t}$, $y_2(t) = e^{-t}$, $y_3(t) = e^{2t}$, $y_4(t) = e^{-t}$, and $y_5(t) = e^t$.

Firstly, we consider the impact of different orders of Radau IIA methods, including 3rd, 5th, 9th, and 13th orders (corresponding to $v = 2, 3, 5, 7$), on the precision of neural network solutions. Secondly, we explore the influence of activation functions on PINN. Common activation functions for hidden layers include Sigmoid, TanH, Sin, and ReLu, among others. When solving smoothly continuous systems, ReLu is generally not chosen; instead, Sigmoid, TanH, or Sin activation functions are preferred. In the experiments, *Sigmoid* resulted in better approximate solutions. Within this neural network framework, the initial values of the differential variables, namely, $y_1(t)$, $y_2(t)$, $y_3(t)$, and $y_4(t)$ for each time segment, are used as a dataset for training. The step size h is 0.05, which means that each time interval has a length of 0.05. Each network model in every time segment comprises 5 hidden layers, with each hidden layer containing 100 neurons. Sigmoid is used as the activation function, and the Adam optimizer is applied for 100,000 iterations. The experimental results within the time interval of 0 to 1 are presented in Figure 4.

From Figure 4, it is evident that the accuracy of the mean absolute errors for the 3rd and 13th-order Radau IIA methods corresponds to the blue Y-axis, while the accuracy of the average absolute errors for the 5th and 9th-order Radau methods corresponds to the red Y-axis. For all the differential function variables, the 3rd and 13th-order Radau IIA methods exhibit significantly higher average absolute errors compared to the 5th and 9th-order methods. For the algebraic variable y_5, the 13th-order Radau IIA method has notably higher average absolute errors than the 3rd, 5th, and 9th-order methods.

Additionally, we further observe that for all differential function variables from red Y-axis, the 9th-order Radau IIA method's overall trend in average absolute errors is significantly higher than the 5th-order method. For the algebraic variable y_5, the 9th-order Radau IIA method exhibits

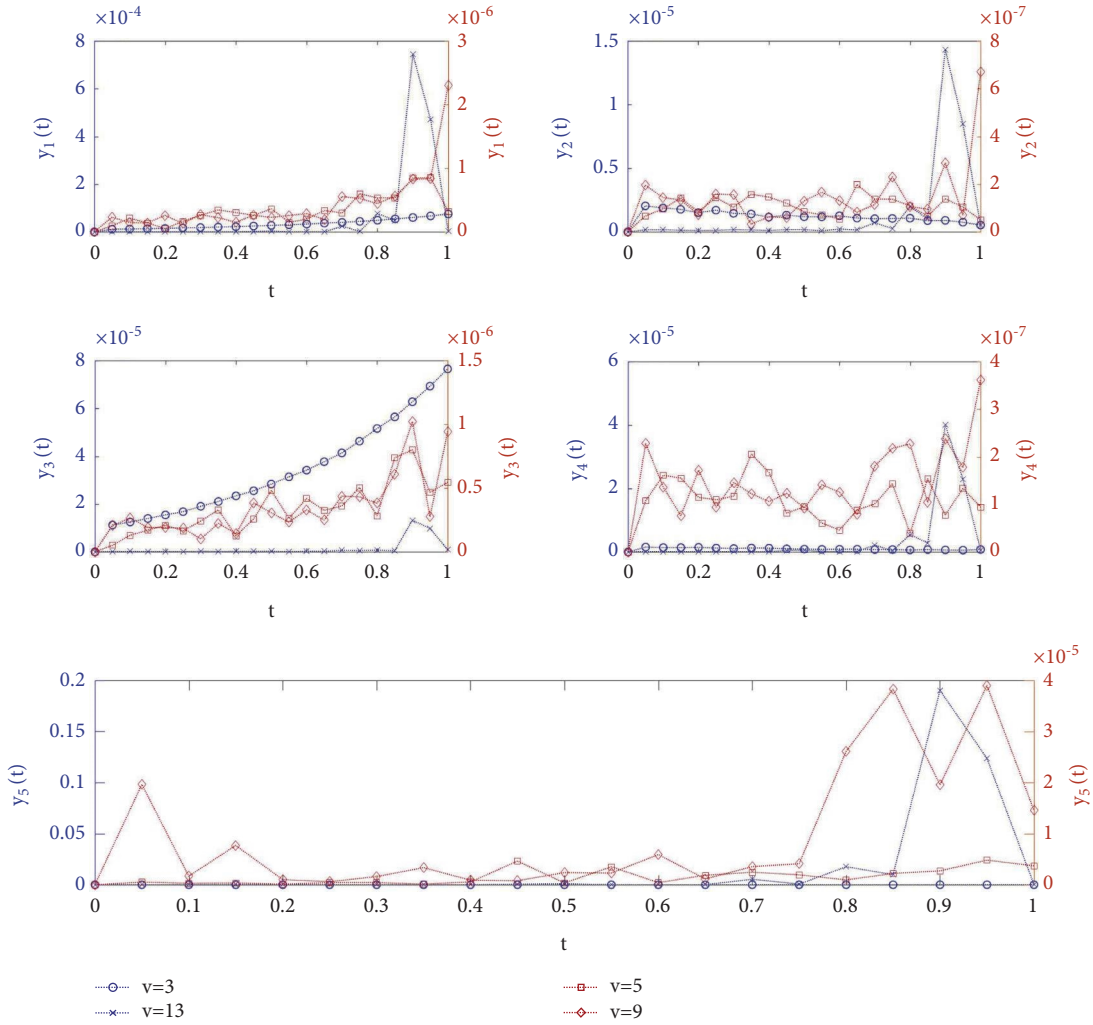

FIGURE 4: The mean absolute errors for solving Hessenberg-DAEs systems using PINN based on Radau IIA of order v = 3, 5, 9, 13. The blue curves represent the mean absolute errors v = 3, 13 on the left Y-axis, while the red curves correspond to v = 5, 9 on the right Y-axis.

notably higher average absolute errors than the 5th-order method. In other words, the 5th-order Radau IIA-based PINN achieves the highest precision in terms of average absolute errors.

The absolute error results obtained using the 5th-order method are shown in Figure 5. The accuracy of the absolute errors for $y_1(t)$, $y_3(t)$, and $y_5(t)$ corresponds to the blue Y-axis, while the accuracy of the absolute errors for $y_2(t)$ and $y_4(t)$ corresponds to the red Y-axis. From the figure, it is evident that the neural network's predicted values for all four differential variables have their lowest precision of absolute errors maintained at the order of 10^{-6}, while the lowest precision of absolute errors for the algebraic variable is kept at 10^{-6}. The experimental results suggest that the neural network's predicted solutions have reached a high level of accuracy.

For the neural network structure designed in this paper, the predicted values of the differential variables can be used as the initial values for the next time step's network input dataset. The precision of the differential variables can affect the results of the next time step's network. In this context, the precision of the differential variables y_1 and y_3 is already at the order of 10^{-6}, and the precision of the differential variables y_2 and y_4 is at the order of 10^{-7}, which will not significantly affect the precision of the next time step.

The key parameters controlling the performance of our algorithm are the total number of Radau IIA stages v and the time-step size h. We consider different orders (3, 5, 9, 13, corresponding to $v = 2, 3, 5, 7$) of the Radau IIA method at various time steps ($h = 0.025, 0.05, 0.1$) to investigate their impact on the precision of neural network solutions. Throughout the experimental process, we maintain a fixed network architecture with 5 hidden layers, each containing 100 neurons, while varying the stages v and time-step size h of the Radau IIA method.

From Tables 2 and 3, the bold values in the table represent the optimal accuracy achieved at each time step. we summarize the average absolute error precision of solving the differential and algebraic function variables of the Hessenberg-type DAEs system using Radau IIA method with different stages and time-step size. Specifically, we can clearly observe that when the time-step is set to 0.025, 0.05, or 0.1, the Radau IIA method with a stage v of 3 consistently produces accurate results. Furthermore, for the Radau IIA method with a stage v of 3 and a time-step size h of 0.05, the neural network

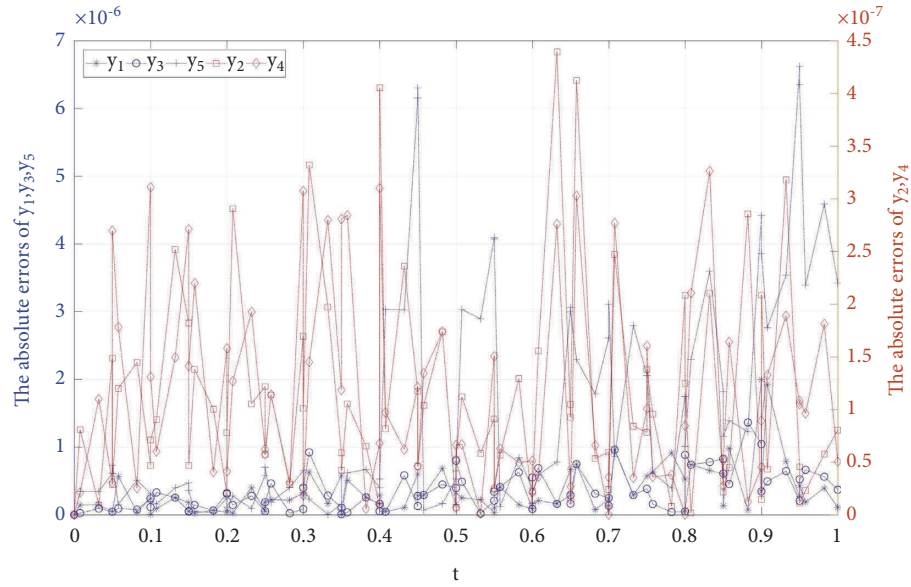

Figure 5: The absolute errors of Hessenberg-DAEs system solved by PINN based on 5th-order Radau IIA. The blue curves represent the absolute errors of y_1, y_3 and y_5 on the left Y-axis, while the red curves correspond to y_2 and y_4 on the right Y-axis.

Table 2: Hessenberg-DAEs systems: Measure the average absolute error of differential function variables for different numbers of Radau IIA stages v and time-step sizes h.

v	h		
	0.025	0.05	0.1
2	$1.9e-06$	$1.7e-05$	$1.8e-04$
3	**$2.2e-07$**	**$2.2e-07$**	$1.4e-06$
5	$9.8e-07$	$2.7e-07$	**$1.0e-06$**
7	$2.1e-05$	$1.8e-05$	$6.0e-06$

Bold values indicate the optimal accuracy achieved at each time step and show the Radau IIA method stages v used for optimal accuracy.

Table 3: Hessenberg-DAEs systems: Measure the average absolute error of algebraic function variable for different numbers of Radau IIA stages v and time-step sizes h.

v	h		
	0.025	0.05	0.1
2	$4.7e-05$	$1.1e-05$	$9.0e-05$
3	**$9.9e-06$**	**$1.6e-06$**	**$1.6e-06$**
5	$6.2e-05$	$9.7e-06$	$2.0e-04$
7	$2.8e-02$	$1.7e-02$	$2.3e-03$

Bold values indicate the optimal accuracy achieved at each time step and show the Radau IIA method stages v used for optimal accuracy.

solution achieves optimal accuracy. Additionally, the overall trend of the average absolute error for the 9th order ($v = 5$) Radau IIA method is similar to the 5th order ($v = 3$) method.

For the differential function variables, the Radau IIA methods of four different orders demonstrate consistently stable accuracy in average absolute error across various time steps. However, for algebraic variables, compared to the 5th and 9th-order methods, the 3rd and 13th-order Radau IIA methods exhibit significantly higher average absolute errors across three different time steps. When using higher-order Radau IIA methods with smaller time steps, there is no doubt that it will enhance the convergence speed of the neural network. However, it also increases the demand for a larger neural network scale. These adjustments require tuning in experiments. Nevertheless, overall, the 5th-order ($v = 3$) Radau IIA method proves to be the most stable. This characteristic is evident in the Hessenberg-DAEs system discussed below, where its numerical stability remains uncompromised. This makes the 5th-order ($v = 3$) Radau IIA method an ideal choice for solving stiff problems.

3.2. DAE System of the Pendulum Model. In this section, we study the classical pendulum DAEs system with an index of 2, as follows:

$$\begin{cases} y_1'(t) = y_3(t), \\ y_2'(t) = y_4(t), \\ y_3'(t) = -y_1(t)y_5(t), \\ \mathrm{m}y_4'(t) = -y_2(t)y_5(t) - \lambda, \\ 0 = y_1(t)y_3(t) + y_2(t)y_4(t), \end{cases} \tag{18}$$

where $t \in [0, 1]$, and the parameters m and λ are variable parameters, both set to 1 in the experiments of this section. The initial values are $y_0 = (1, 0, 0, 1, 1)$. In this context, $y_1(t)$, $y_2(t)$, $y_3(t)$, and $y_4(t)$ are differential function variables, while $y_5(t)$ is an algebraic function variable. This DAEs system does not have an exact analytical expression. In this paper, we directly solve the reduced inner ODEs of this system using high-precision ODE solvers from the Python scientific computing library *Scipy* and compare the obtained approximate solution with the predicted values from the neural network. Similarly, we consider the impact of different orders (3, 5, 9, 13, corresponding to $v = 2, 3, 5, 7$) in the Radau IIA methods on the accuracy of the neural network's solutions. Secondly, we explore the effect of activation functions on PINN. In this experiment, the Sin

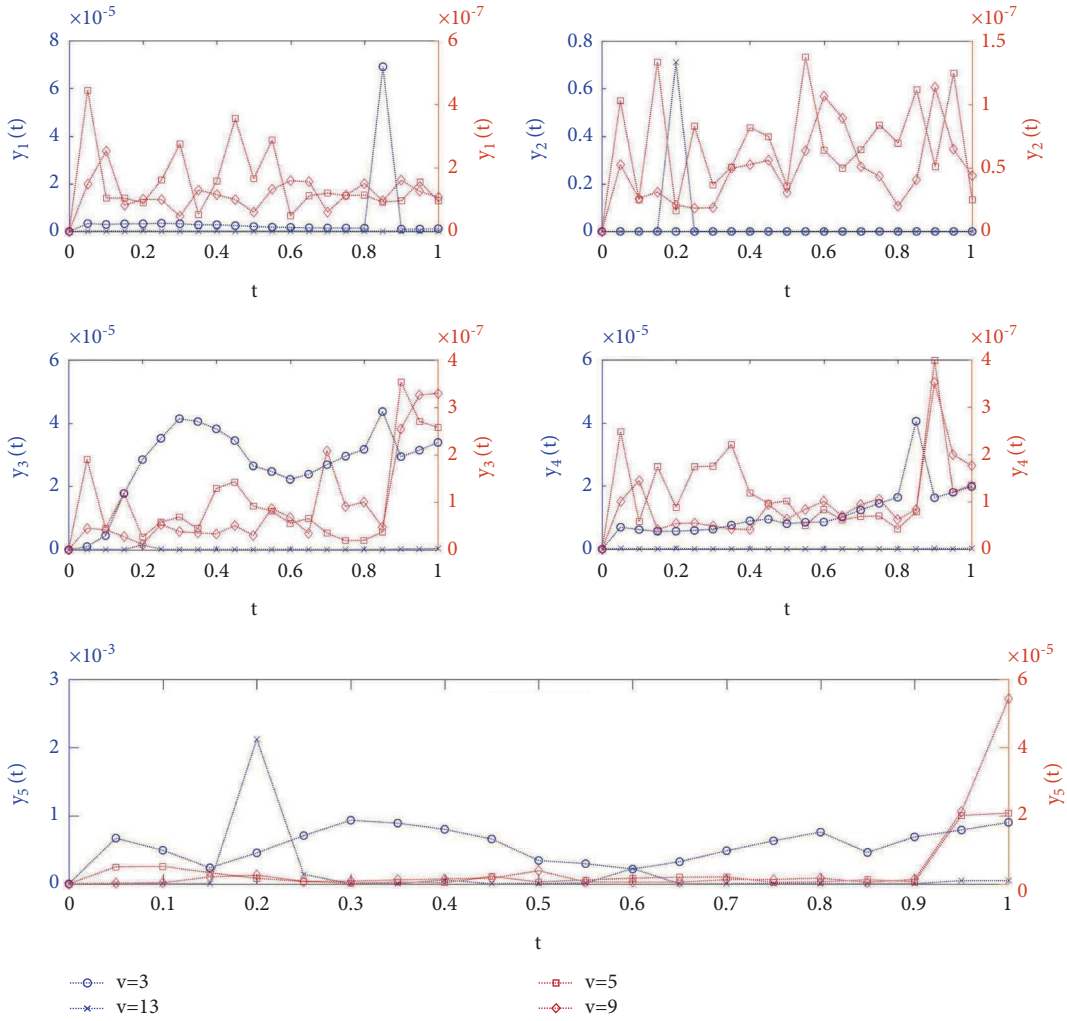

FIGURE 6: The mean absolute errors for solving single Pendulum DAEs systems using PINN based on Radau IIA of order $v = 3, 5, 9, 13$. The blue curves represent the mean absolute errors $v = 3, 13$ on the left Y-axis, while the red curves correspond to $v = 5, 9$ on the right Y-axis.

activation function provides a better approximation. To maintain consistency in the numerical experiments, other network structural information is consistent with the experiments in the previous section. The results obtained are shown in Figure 6.

From Figure 6, we can observe that the accuracy of the mean absolute errors for the 3rd and 13th-order Radau IIA methods corresponds to the blue Y-axis, while the accuracy of the average absolute errors for the 5th and 9th-order Radau methods corresponds to the red Y-axis. For the differential function variables $y_1(t)$, $y_3(t)$, and $y_4(t)$, the 3rd-order Radau IIA method has significantly higher average absolute errors than the 5th, 9th, and 13th-order methods. For the differential function variable $y_2(t)$, the 13th-order Radau IIA method exhibits significantly higher average absolute errors than the 3rd, 5th, and 9th-order methods. For the algebraic variable $y_5(t)$, the 3rd and 13th-order Radau IIA methods have significantly higher average absolute errors compared to the 5th and 9th-order methods.

Additionally, we further observe that for all differential function variables from red Y-axis, the 9th-order Radau IIA method's average absolute error overall trends similarly to the 5th-order method. For the algebraic variable $y_5(t)$, the 9th-order Radau IIA method exhibits significantly higher average absolute errors in the later time regions compared to the 5th-order method. In other words, a PINN based on the 5th-order Radau IIA method achieves the highest precision in terms of average absolute errors.

The absolute error results obtained using the 5th-order method are shown in Figure 7. The accuracy of the absolute errors for $y_1(t)$, $y_2(t)$, $y_3(t)$, and $y_4(t)$ corresponds to the blue Y-axis, while the accuracy of the absolute errors for $y_5(t)$ corresponds to the red Y-axis. From the figure, we can see that the lowest precision of absolute errors for all four differential variables is maintained at 10^{-7}, while the lowest precision of absolute errors for the algebraic variable is kept at 10^{-5}. The experimental results suggest that the neural network's predicted solutions for the pendulum's DAEs system can also achieve high precision.

Similarly, we also consider the impact of different orders (3, 5, 9, 13, corresponding to $v = 2, 3, 5, 7$) in the Radau IIA method at various time steps ($h = 0.025, 0.05, 0.1$) on the precision of the neural network solutions for the pendulum DAEs system. From Tables 4 and 5, the bold

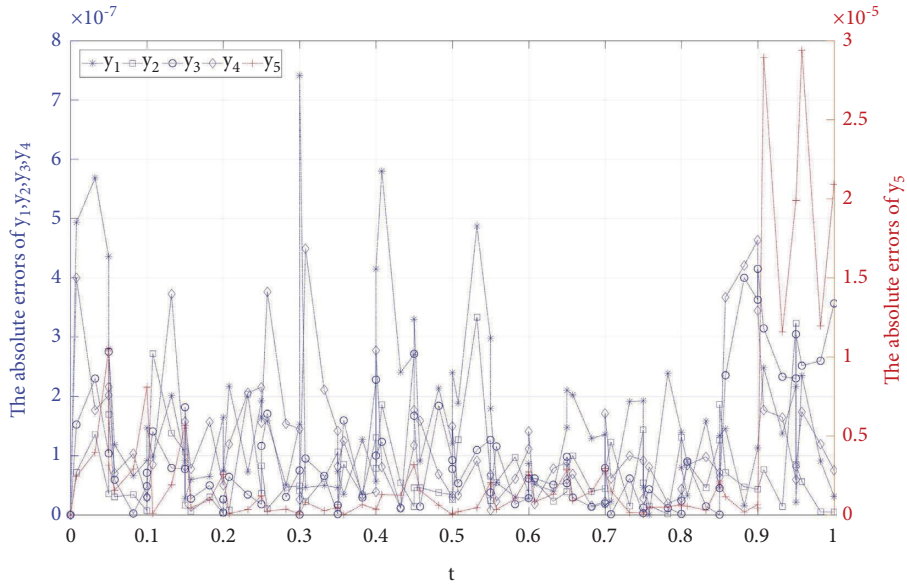

Figure 7: The absolute errors of single Pendulum DAEs system solved by PINN based on 5th-order Radau IIA. The blue curves represent the absolute errors of y_1, y_2, y_3 and y_4 on the left Y-axis, while the red curve corresponds to y_5 on the right Y-axis.

Table 4: Pendulum DAEs systems: Measure the average absolute error of differential function variables for different numbers of Radau IIA stages v and time-step sizes h.

v	h		
	0.025	0.05	0.1
2	**2.3e − 07**	1.3e − 05	3.8e − 05
3	2.9e − 07	1.1e − 07	2.3e − 07
5	7.1e − 07	**9.1e − 08**	**1.1e − 07**
7	2.4e − 06	1.4e − 07	1.2e − 07

Bold values indicate the optimal accuracy achieved at each time step and show the Radau IIA method stages v used for optimal accuracy.

Table 5: Pendulum DAEs systems: Measure the average absolute error of algebraic function variable for different numbers of Radau IIA stages v and time-step sizes h.

v	h		
	0.025	0.05	0.1
2	1.6e − 04	5.9e − 04	2.4e − 03
3	**1.2e − 05**	**3.5e − 06**	1.4e − 05
5	4.7e − 05	4.9e − 06	**7.0e − 06**
7	1.6e − 02	1.3e − 04	2.1e − 05

Bold values indicate the optimal accuracy achieved at each time step and show the Radau IIA method stages v used for optimal accuracy.

values in the table represent the optimal accuracy achieved at each time step. Clearly, the overall trends in the mean absolute errors for both the differential and algebraic function variables of the 5th-order ($v = 3$) and 9th-order ($v = 5$) Radau IIA methods consistently remain within the same order of magnitude. Additionally, the solving accuracy of these two orders of Radau IIA methods remains highly stable regardless of the chosen time step value.

In general, the results of solving the pendulum DAEs system using Radau IIA methods with different orders and time steps are consistent with the results obtained in the solution of the Hessenberg-type DAEs system mentioned above.

4. Summary and Conclusions

DAE systems are widely employed in various domains, including fluid dynamics, multi-body dynamics, and control theory. In practical physical modeling, most DAE models are either low-index DAEs or high-index Hessenberg-type DAEs. Classical implicit numerical methods are suitable for a certain class of high-index DAEs, but they often lead to varying order reduction of numerical accuracy. Recently, a novel neural network method, DAE-PINN, has been developed for solving low-index DAEs. However, it cannot directly handle high-index systems. Therefore, this paper proposes a PINN-based approach using the Radau method to solve high-index DAEs systems. This method combines the strengths of the Radau IIA method with an improved fully connected neural network structure and employs a time-domain decomposition strategy to enhance both efficiency and accuracy in solving these systems.

In this paper, two high-index systems, namely, Hessenberg-type DAEs and pendulum model DAEs, are studied as examples. The research takes into account the influence of different orders and time-step sizes in the Radau IIA methods and the activation functions on the accuracy of neural network solutions. Generally, employing higher-order Radau IIA methods enhances the neural network's generalization capability. However, through comparative experiments with two examples, it is found that employing a 5th-order Radau IIA method in the PINN yields a high degree of system accuracy and stability for varying time-step

sizes. This conclusion is consistent with the notion that Radau-5 is a high-precision numerical method [3]. Further experimental results indicate that in high-index systems, the absolute errors for all differential variables maintain a minimum precision of 10^{-6}, while the absolute errors for algebraic variables maintain a minimum precision of 10^{-5}. This method's numerical accuracy surpasses the corresponding results in the literature [25] and, to some extent, surpasses the accuracy achieved by the DAE-PINN method [27]. This demonstrates that our method can directly and accurately solve high-index DAEs systems, showcasing strong generalization capabilities and offering a viable approach for high-precision solutions to even higher-index DAEs or challenging systems of partial differential algebraic equations. Furthermore, we have maintained the depth and width of the neural networks as in DAE-PINN [27] and have not delved into a detailed study of their impact on the accuracy of our method, which we will need to investigate in our future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A preprint has previously been published [28]. This manuscript has been presented as a paper presentation of International Journal of Intelligent Systems.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The study was supported by the National Natural Science Foundation of China (Grant No. 12201144), the Guangdong Basic and Applied Basic Research Foundation of China (Grant No. 2020A1515110554), the Science and Technology Foundation of Guizhou Province (Grant No. QKHJC-ZK [2021]YB015) of China, and Chongqing Talents Plan Youth Top-Notch Project of China (Grant No. 2021000263).

References

[1] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, 1971.

[2] L. R. Petzold, "Differential/algebraic equations are not ode's," *SIAM Journal on Scientific and Statistical Computing*, vol. 3, no. 3, pp. 367–384, 1982.

[3] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, vol. 61, SIAM, New Delhi, India, 1998.

[4] U. M. Ascher and L. R. Petzold, "Projected implicit Runge–Kutta methods for differential-algebraic equations," *SIAM Journal on Numerical Analysis*, vol. 28, no. 4, pp. 1097–1120, 1991.

[5] J. R. Cash, "Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in odes and daes," *Journal of Computational and Applied Mathematics*, vol. 125, no. 1-2, pp. 117–130, 2000.

[6] M. Saravi, E. Babolian, R. England, and M. Bromilow, "System of linear ordinary differential and differential-algebraic equations and pseudo-spectral method," *Computers and Mathematics with Applications*, vol. 59, no. 4, pp. 1524–1531, 2010.

[7] M. M. Hosseini, "Adomian decomposition method for solution of differential-algebraic equations," *Journal of Computational and Applied Mathematics*, vol. 197, no. 2, pp. 495–501, 2006.

[8] C. K. Newman, "Exponential integrators for the incompressible Navier-Stokes equations," Virginia Tech, Blacksburg, VA, USA, Doctor of Philosophy, 2003.

[9] J. Ding and Z. Pan, "Generalized-α projection method for differential-algebraic equations of multi-body dynamics," *Engineering Mechanics*, vol. 4, pp. 380–384, 2013.

[10] J. Lu, J. Tang, X. Qin, and Y. Feng, "Modified group preserving methods and applications in chaotic systems," *Acta Physica Sinica*, vol. 65, no. 6, Article ID 060503, 2016.

[11] C.-S. Liu, W. Chen, and L.-W. Liu, "Solving mechanical systems with nonholonomic constraints by a lie-group differential algebraic equations method," *Journal of Engineering Mechanics*, vol. 143, no. 9, 2017.

[12] J. Tang and J. Lu, "Modified extended lie-group method for hessenberg differential algebraic equations with index-3," *Mathematics*, vol. 11, no. 10, p. 2360, 2023.

[13] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[15] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "hp-vpinns: variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, 2021.

[16] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, 2020.

[17] G. Wu, F. Wang, and L. Qiu, "Physics-informed neural network for solving hausdorff derivative Poisson equations," *Fractals*, vol. 31, no. 06, Article ID 2340103, 2023.

[18] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, 2020.

[19] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 426, 2021.

[20] B. Zhang, G. Wu, Y. Gu, X. Wang, and F. Wang, "Multi-domain physics-informed neural network for solving forward and inverse problems of steady-state heat conduction in multilayer media," *Physics of Fluids*, vol. 34, p. 11, 2022.

[21] B. Zhang, F. Wang, and L. Qiu, "Multi-domain physics-informed neural networks for solving transient heat

conduction problems in multilayer materials," *Journal of Applied Physics*, vol. 133, no. 24, 06 2023.

[22] P. Borate, J. Rivière, C. Marone, A. Mali, D. Kifer, and P. Shokouhi, "Using a physics-informed neural network and fault zone acoustic monitoring to predict lab earthquakes," *Nature Communications*, vol. 14, no. 1, p. 3693, 2023.

[23] Y. Chen and L. Dal Negro, "Physics-informed neural networks for imaging and parameter retrieval of photonic nanostructures from near-field data," *APL Photonics*, vol. 7, no. 1, 2022.

[24] D. S. Kozlov and Y. V. Tiumentsev, "Neural network based semi-empirical models for dynamical systems described by differential-algebraic equations," *Optical Memory and Neural Networks*, vol. 24, no. 4, pp. 279–287, 2015.

[25] Y. Zhao, J. Lan, and Y. Wu, "On solutions to several classes of differential-algebraic equations based on artificial neural networks," *Applied Mathematics and Mechanics*, vol. 40, no. 2, 2019.

[26] H. Liu, H. Liu, J. Xu, L. Li, and J. Song, "Jacobi neural network method for solving linear differential-algebraic equations with variable coefficients," *Neural Processing Letters*, vol. 53, no. 5, pp. 3357–3374, 2021.

[27] C. Moya and G. Lin, "Dae-pinn: a physics-informed neural network model for simulating differential algebraic equations with application to power networks," *Neural Computing and Applications*, vol. 35, no. 5, pp. 3789–3804, 2023.

[28] J. Chen, J. Tang, M. Yan et al., "Physical information neural networks for solving high-index differential-algebraic equation systems based on radau methods," 2023, https://arxiv.org/abs/2310.12846.