ISSN (e): 2250 – 3005 || Volume, 14 || Issue, 2|| Mar. - Apr. – 2024 ||
International Journal of Computational Engineering Research (IJCER)

Efficiently Solving High-Order and Nonlinear ODEs with Rational Fraction Polynomial: the Ratio Net

Chenxin Qin[a,#], Ruhao Liu[b,#], Maocai Li[a,*], Shengyuan Li[a], Yi Liu[a], and Chichun Zhou[a,*]

[a] *School of Engineering, Dali University, Dali, Yunnan, China, 671003*
[b] *School of Information Engineering, Nancahng University, Nanchang, Jiangxi, China, 330000*
[#] *These authors contributed equivalently to this work.*
[*] *Corresponding author*

Abstract
Recent advances in solving ordinary differential equations (ODEs) with neural networks have been remarkable. Neural networks excel at serving as trial functions and approximating solutions within functional spaces, aided by gradient backpropagation algorithms. However, challenges remain in solving complex ODEs, including high-order and nonlinear cases, emphasizing the need for improved efficiency and effectiveness. Traditional methods have typically relied on established knowledge integration to improve problem-solving efficiency. In contrast, this study takes a different approach by introducing a new neural network architecture for constructing trial functions, known as ratio net. This architecture draws inspiration from rational fraction polynomial approximation functions, specifically the Pade approximant. Through empirical trials, it demonstrated that the proposed method exhibits higher efficiency compared to existing approaches, including polynomial-based and multilayer perceptron (MLP) neural network-based methods. The ratio net holds promise for advancing the efficiency and effectiveness of solving differential equations.

Keywords
High-order ordinary differential equation, Nonlinear ordinary differential equation, Neural network, Ratio net, rational fraction polynomial

---

Date of Submission: 02-04-2024 Date of acceptance: 13-04-2024

I. Introduction

Ordinary and partial differential equations (ODEs and PDEs) serve as crucial mathematical tools for analyzing and describing a wide range of problems in physics and engineering [1-3].

The methods for solving ordinary differential equations (ODEs) have evolved over time, transitioning from analytical approaches to conventional numerical methods and eventually to neural network techniques. Analytical methods such as the series [4, 5] and the constant transform [4] methods may fail in complicate cases. Unlike traditional rigorous analytical methods, numerical methods primarily employed difference techniques, such as converting ODEs into recursive equations, and can solve ODEs extensively, [6,7] helping us gain insights into properties of solutions. Typical representatives of this approach include Euler's method, the Runge-Kutta method, and finite difference methods [6,7]. However, traditional numerical methods face a challenge in balancing precision and computational efficiency. E.g., improving precision often leads to reduced efficiency [6,7]. Moreover, high-order and nonlinear ODEs introduce stability issues [6,7].

To overcome these challenges, neural network methods have been introduced into the solving of ODEs [8-10,29]. The core idea is to use neural networks with a larger number of parameters as trial functions to approximate solutions, akin to the traditional Ritz method [6,7]. Neural network methods have two key improvements: first, the trial functions employ neural networks with more parameters, enhancing their approximation capabilities [8-10]. Second, they employ backpropagation methods to optimize the parameters within the trial functions, ensuring that they satisfy the constraints of the equations and boundary conditions [8-10].

Compared to traditional numerical methods, neural network-based approaches undeniably provide substantial enhancements in both precision and efficiency. Nevertheless, there remains a new avenue of research dedicated to further augmenting the efficacy and expeditiousness of this method. Effectiveness entails the

development of robust neural network architectures to guarantee the discovery of equation solutions, while efficiency focuses on strategies to expedite the training process.

- Regarding effectiveness, researchers have provided various neural network-based backbones to prove effectiveness [8-14]. Those backbones can be roughly grouped into two categories. 1) multilayer perceptron (MLP) neural network-based [8-10,26-28] and 2) polynomial-based methods [11-13,30-33,35], such as the Chebyshev [31] and Legendre [35] polynomials.
- Regarding efficiency, one of the major advancements is the introduction of incorporating more prior information into neural networks to expedite equation solving efficiency. Notable approaches are the physics–informed neural networks (PINN) [15-17], designed to combine neural networks and physical equations to address scientific and engineering problems.

The Pade approximant, a representative rational fraction polynomial approximation function [18], intriguingly exhibits comparable function fitting capabilities to existing methods like polynomials. Notably, it outperforms these methods, particularly in terms of efficiency [18-20]. Motivated by this, the present paper employs an efficient approach based on rational fraction polynomials, known as the ratio net, for solving ODEs. This method has been previously introduced in [19,20]. We showcase the effectiveness and efficiency of the ratio net in solving ODEs through various illustrative examples, including those from [21–25] and new examples proposed in this study. Furthermore, we conducted a comparative analysis with polynomial-based methods [30-33] and MLP neural network-based methods [26–29], affirming the superior efficiency of the Ratio Net.

To the best of our limited knowledge, the ratio net, which is essentially a kind of rational fraction polynomial approximation functions, is introduced as an alternative method to solving the ODEs for the first time and holds promise for advancing the efficiency and effectiveness of solving differential equations.

The remainder of this paper is organized as follows: Sec. 2 introduces the proposed method and Sec. 3 demonstrates its advantages by applying it to several illustrative examples. Finally, Sec. 4 discusses our findings and concludes this work.

II. Method Description

This section introduces the main methods employed, including the structure of the ratio net, the trial function under boundary conditions, the loss function, and the weight updating algorithm.

2.1 The ratio net: a brief review

The ratio net, introduced in our previous work [19], serves as a neural network backbone. It differentiates itself from traditional neural networks, which rely on nonlinear activation functions and kernel functions, by utilizing rational fraction polynomial approximation functions to address nonlinear difficulties. There were evidence supports its superior efficiency compared to conventional neural networks like MLP and RBF [19]. We will provide a concise overview of the ratio net in the following section.

A neural network can effectively search the mode of the relation between the outputs and the inputs. In the ratio net, the relation between the outputs and the inputs is:

$$y_i^{Ratio}(x) = \frac{\left(\sum_{j=1}^{n} \omega'_{ij} x_j + b'_i\right)\left(\sum_{j=1}^{n} \omega''_{ij} x_j + b''_i\right) \ldots .}{\left(\sum_{j=1}^{n} \omega'''_{ij} x_j + b'''_i\right)\left(\sum_{j=1}^{n} \omega'''_{ij} x_j + b'''_i\right) \ldots .}, \#(2-1)$$

where ω and b are parameters, j ranges from 1 to the dimension of the input x and i from1 to the dimension of the output y. As illustrated in Fig. (1). Here, to solve the ODEs, the input and output dimensions are set to 1.

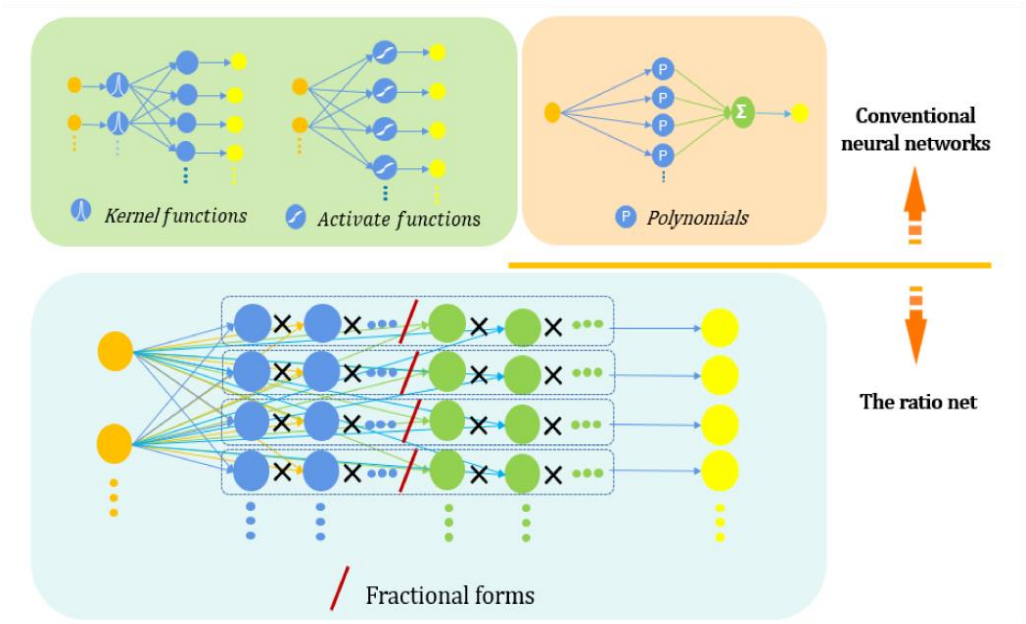


Figure 1. A diagram of Ratio neural networks.

2.2 The trial function under the boundary condition

The ODEs' boundary condition imposes additional challenges beyond constructing the trial functions. Indeed, typically, researchers consider the boundary conditions as an additional constraint and transform it into an extra term in the loss function [8-10]. This strategy forces the model's parameters to be trained to fit the target function shape and meet the boundary condition. However, the boundary condition may prevent the neural network from finding the target function. An alternative approach introduces a trial function by constructing a network that automatically satisfies the boundary conditions, such as a trial function is in the form [34]

$$y^{trial}(x) = Net(x)B(x) + g(x), \#(2-2)$$

where $B(x)$ and $g(x)$ are designed to satisfy the boundary conditions, and $Net(x)$ are designed to search the target function. In this section, the ratio net-based trial function takes the form as proposed in [34]. However, $B(x)$ and $g(x)$ are chosen as power and power sum series that can be obtained by solving equations determined by the boundary conditions (see code in github). That is, in our approach, the trial function, Eq. (2-2), can be obtained without any manual configuration.

Specifically, considering ODEs in the interval $[a, b]$ with boundary conditions $y^{(n)}(a) = A_n$ and $y^{(m)}(b) = B_m$, here $y^{(n)}(a)$ and $y^{(m)}(b)$ are the n-th and m-th derivative at a and b respectively. A_n and B_m are real numbers representing the boundary values. The trial function suggested is in the form,

$$y_{ratio}^{trial}(x) = y_{ratio}(x)(x-a)^{M_a}(b-x)^{M_b} + g(x), \#(2-3)$$

where M_a and M_b are the maximum order $+1$ of the derivative in the boundary condition at a and b, respectively, $y_{ratio}(x)$ is the ratio net of Eq. (2.1) with the input and output dimensions set to 1. $g(x)$ is a series of highest order L, with L the number of equations in the boundary conditions. $g(x)$ meets the boundary conditions and thus is decided by the boundary conditions automatically. For example, if the boundary conditions read $y^{(0)}(a) = A$ and $y^{(0)}(b) = B$, then, $M_a = 1$, $M_b = 1$, and $g(x) = (xB - xA + bA - aB)/(b-a)$. In the context of problems associated with initial conditions, this fundamental concept remains applicable.

2.3 Other existing methods: a brief review

To demonstrate our method's efficiency in solving the ODEs, we compare it with existing methods. Without loss of generality, we consider two typical conventional neural networks, the Legendre polynomials-based, and the MLP-based methods. This section briefly reviews both these methods.

The polynomial based methods: the Legendre polynomials. The Legendre polynomial is a typical class of orthogonal polynomials, with their linear combination up to order L can be used as an effective function approximant. The Legendre polynomials [35] are defined as $P_0(x) = 1$, $P_1(x) = x$, and for $P_n(x)$ with $n > 1$:

$$P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x), \#(2-4)$$

where $P_n(x)$ is the Legendre polynomial of order n. The trial function based on the Legendre polynomials is:

$$y_{Legendre}^{trial}(x) = y_{Legendre}(x)(x-a)^{Ma}(b-x)^{Mb} + g(x), \#(2-5)$$

where

$$y^{Legendre}(x) = \sum_{j=0}^{L} \omega_j P_j(x), \#(2-6)$$

with ω_j the weight to be learned.

The neural network-based methods: the MLP. The MLP is a classic neural network that comprises an input layer, several hidden layers, and an output layer. To obtain the next layer, a non-linear activation function is applied to the linear combination of the previous layer. For instance, a one-hidden layer MLP with its input and output layer dimensions set to 1 is given by:

$$y^{MLP}(x) = \sum_{j=1}^{n} \omega_{ij}\sigma\left(\sum_{i=1}^{L} \omega_i x + b_1\right) + b_2, \#(2-7)$$

where L is the size of hidden layer and $\sigma(x)$ is the activation function. Typically, the activation function can be $\sigma(x) = (e^x - e^{-x})/(e^x + e^{-x})$, $\sigma(x) = 1/(1+e^{-x})$, and etc.. The trial function of MLP-based methods is:

$$y_{MLP}^{trial}(x) = y_{MLP}(x)(x-a)^{Ma}(b-x)^{Mb} + g(x). \#(2-8)$$

To conclude, the following trial functions are considered in this work:

$$y^{trial} = \begin{cases} y_{ratio}(x)(x-a)^{Ma}(b-x)^{Mb} + g(x) \\ y_{Legendre}(x)(x-a)^{Ma}(b-x)^{Mb} + g(x) \\ y_{MLP}(x)(x-a)^{Ma}(b-x)^{Mb} + g(x) \end{cases} \#(2-9)$$

2.4 An illustration of the advantages of the ratio net

It proved that rational fraction polynomials share the function-fitting capabilities of traditional polynomials while offering several advantages, such as a larger convergence radius [18] and higher efficiency [19,20]. In this section, we demonstrate the superior efficiency of the ratio net by visualizing the changes in the trial function as we adjust the parameters, as shown in Fig. 2. By modifying the weights, the trial function dynamically explores the target function within the functional space. In this demonstration, the weights undergo random changes of up to 10.0% with each step or adjustment. The results illustrate that the ratio net exhibits both sensitivity to parameter variations and a consistent response to these changes, providing insights into its efficiency when navigating the functional space.

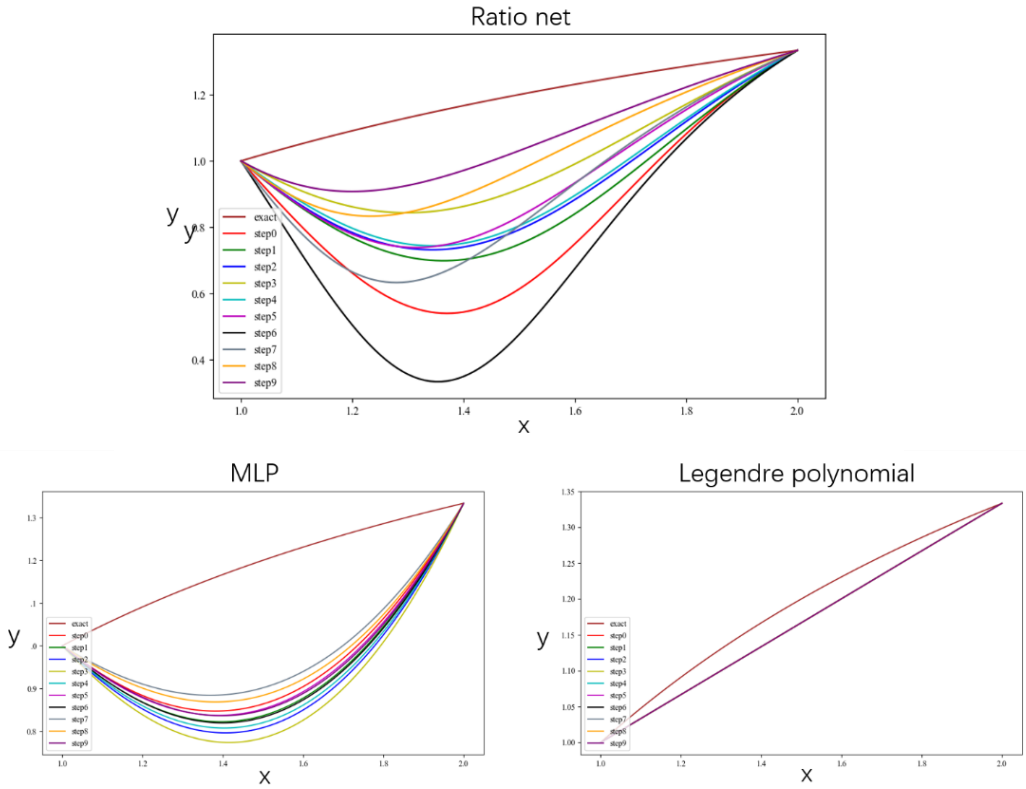


Figure 2. An illustration of the trial functions' efficiency. There are 9 steps plotted with different colors. In each step, the parameters of the trial function randomly changed by 10%. Panel a. The trial functions of the proposed ratio net. Panels b and c. The trial functions of the MLP and Legendre polynomial, respectively. It shows that the ratio net is more sensitive to the change of parameters.

2.5 The loss function and the update of the weights

The general form of differential equations with boundary conditions can be expressed as

$$F[x, y(x), y^{(1)}(x), \ldots\ldots, y^{(k)}(x)] = 0, x \in [a, b] \#(2-10)$$

and

$$C[x, y(x), y^{(1)}(x), \ldots\ldots, y^{(1)}(x)] = 0, x = a, b, \#(2-11)$$

where k is the order of the ODE and $C(x)$ gives the boundary condition. Since the trial function in Eq. (2-3) meets the boundary conditions automatically, one only need to minimize the following loss function

$$loss = F^2\left[x, y^{\text{trial}}(x), \frac{dy^{\text{trial}}}{dx}, \ldots\ldots, \frac{d^n y^{trial}}{dx^n}\right] \#(2-12)$$

by training the ratio net. I.e., the weights in the ratio net are trained to minimize the loss function Eq. (2-12) through the gradient descent method:

$$\omega' = \omega + k\frac{\partial loss}{\omega} \#(2-13)$$

with ω the parameters in the networks. The updating of the weight in the ratio net is implemented through TensorFlow [36]. Further details on the algorithm's implementation are presented in the illustrative examples and the code is given in github:https://github.com/zhiyiqin/The-Ratio-Net-solve-High-Order-and-the-Non-Linear-Ordinary-Differential-Equations.

III. Illustrative examples

In this section, the ratio net is applied to various illustrative examples from [21–26]. The result is compared with that of current representative methods. It reveals that the ratio net affords a higher efficiency.

3.1 Non-linear ordinary differential equations.

Example 1. Considering the ODE

$$y'(x)=2y(x)-y^2(x)+1 \#(3-1)$$

with boundary conditions $y(0) = 1 + \sqrt{2}\tanh\left[\frac{1}{2}log\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)\right]$ and $y(1) = 1 + \sqrt{2}\tanh\left[\sqrt{2} + \frac{1}{2}log\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)\right]$, which has the exact solution $y(x) = 1 + \sqrt{2}\tanh\left[\sqrt{2} + \frac{1}{2}log\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)\right]$. Fig. 3 illustrates the results of the three neural networks, where the effectiveness is characterized by the fitting diagram and the relative error between the numerical and the analytical solutions. The solutions, along with its first and second derivatives, are both displayed. The decreasing trend of the loss function shows each method's efficiency.

Example 2. Considering the ODE

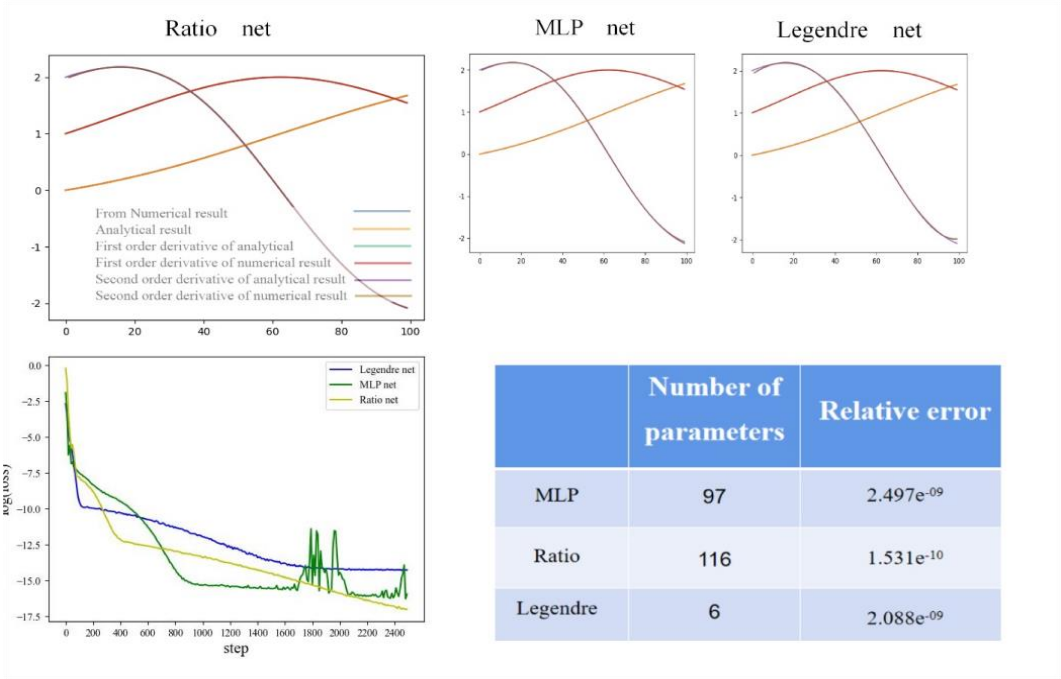
$$y''(x) = \frac{y^3(x) - 2y^2(x)}{2x^2} \#(3-2)$$

with boundary conditions $y(0) = 0$ and $y(2) = \frac{4}{3}$, which has the exact solution $y(x) = 2x/(x+1)$. Fig. 4 depicts the corresponding results.

Example 3. Considering the ODE

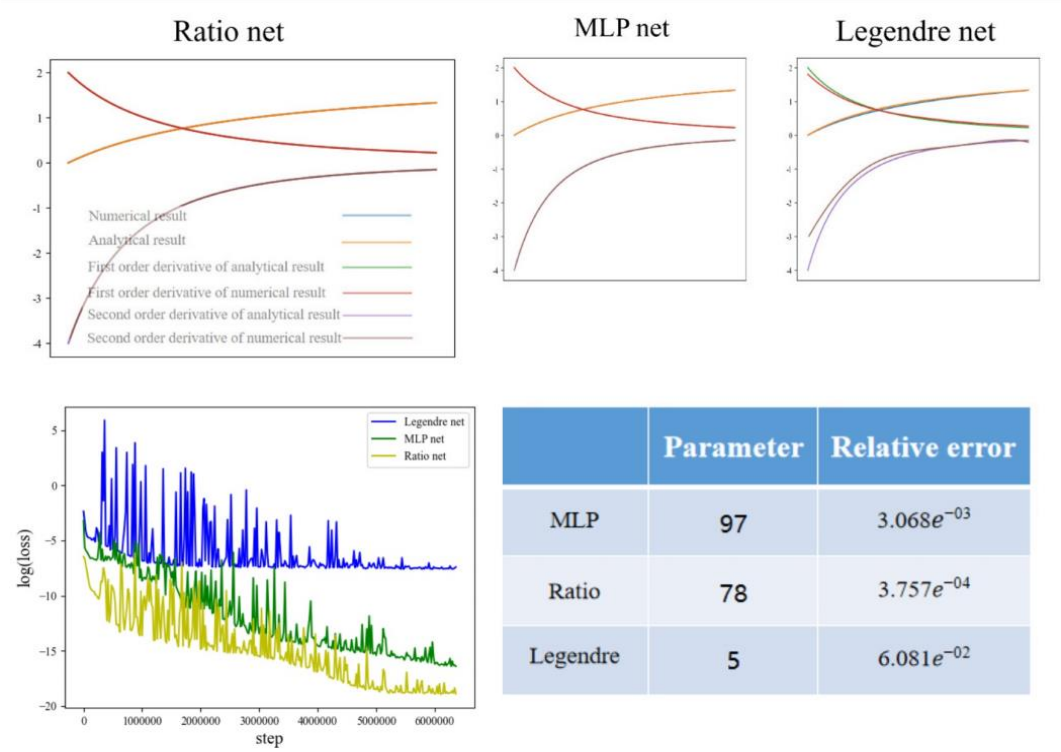
$$y'''(x) = -y^2(x) - \cos(x) + \sin^2(x) \#(3-3)$$

with boundary conditions $y(0)=0$, $y'(0)=1$, and $y(\pi)=0$, which has the exact solution $y(x) = sin(x)$. Fig. 5 depicts the corresponding results.



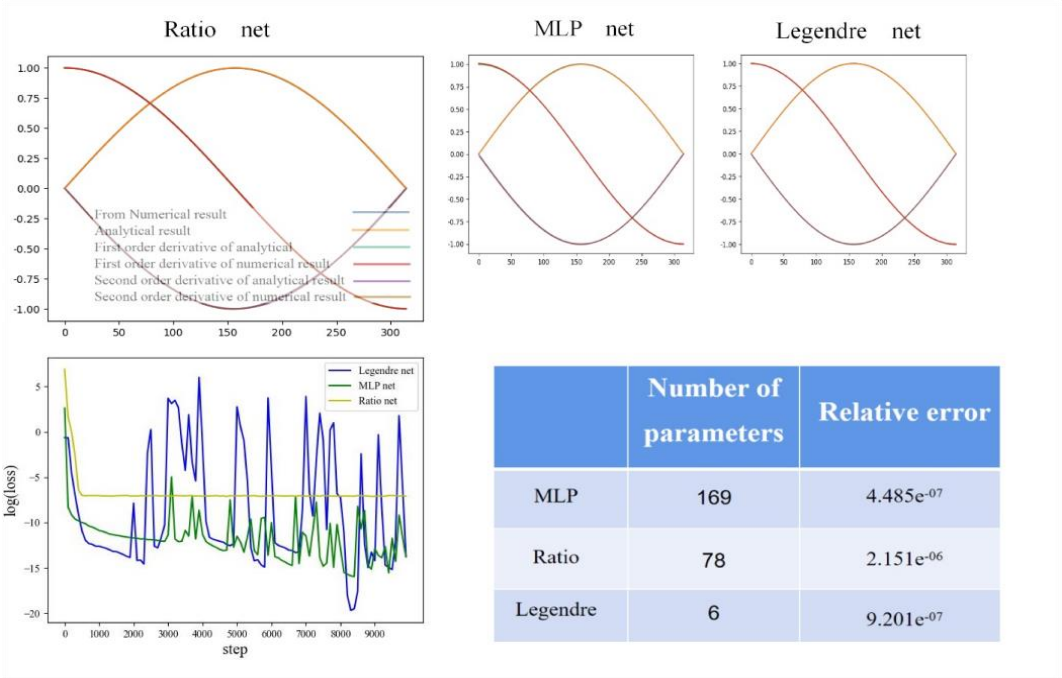
	Number of parameters	Relative error
MLP	97	$2.497e^{-09}$
Ratio	116	$1.531e^{-10}$
Legendre	6	$2.088e^{-09}$

Figure 3. Comparison of the results given by the three neural networks of example 1. The learning rates are all 0.01.



	Parameter	Relative error
MLP	97	$3.068e^{-03}$
Ratio	78	$3.757e^{-04}$
Legendre	5	$6.081e^{-02}$

Figure 4. Comparison of the results given by the three neural networks of example 2. The learning rates are all 0.1.



	Number of parameters	Relative error
MLP	169	4.485e-07
Ratio	78	2.151e-06
Legendre	6	9.201e-07

Figure 5. Comparison of the results given by the three neural networks of example 3. The learning rates are all 0.01.

3.2 High-order ordinary differential equations.

Example 4. Considering the ODE

$$y^{(4)}(x)=120x\#(4-1)$$

with boundary conditions $y(-1)=1$, $y'(-1)=5$, $y(1)=3$, and $y'(1)=5$, which has the exact solution $y(x)=x^5+2$. Fig. 6 depicts the results.

Example 5. Considering the ODE

$$y^{(4)}(x) = -\frac{x^2}{1+y^2(x)} - 72(1-5x+5x^2) + \frac{x^2}{1+(x-x^2)^6} \#(4-2)$$

with boundary conditions $y(0)=0$, $y'(0)=0$, $y(1)=0$, and $y'(1)=0$, which has the exact solution $y(x) = x^3(1-x)^3$. Fig. 7 depicts the results.

Example 6. Considering the ODE

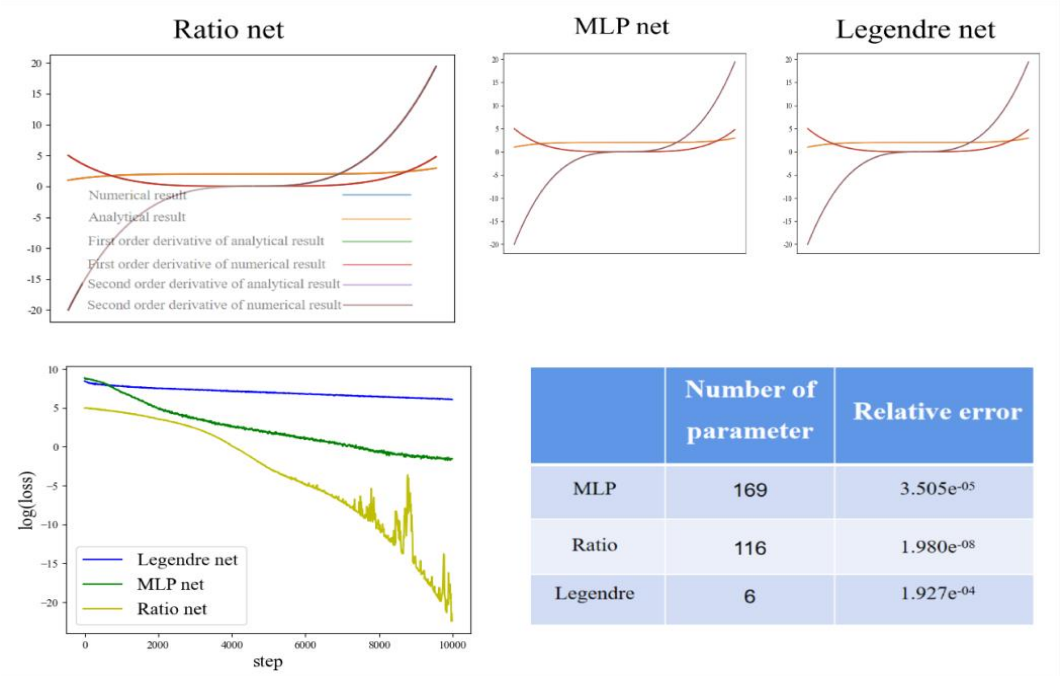
$$y^{(4)}(x)+y(x)=\left[\left(\frac{\pi}{2}\right)^4+1\right] cos\left(\frac{\pi}{2}x\right) \#(4-3)$$

with boundary conditions $y(-1)=0$, $y'(-1)=\frac{\pi}{2}$, $y(1)=0$, and $y'(1)=-\frac{\pi}{2}$, which has exact solution $y(x)=\cos\left(\frac{1}{2}\pi x\right)$. Fig. 8 depicts the results.

Example 7. Considering the ODE

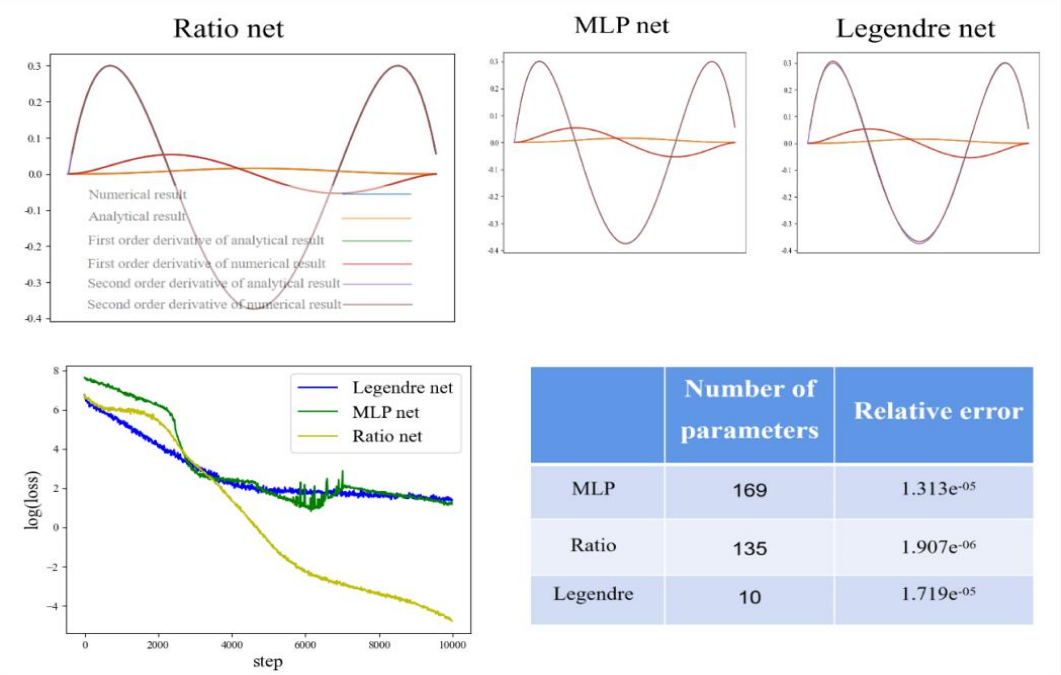
$$y^{(4)}(x) + y'(x) = 4x^3 + 24 \#(4-4)$$

with boundary conditions $y(0)=0$, $y'(0)=0$, $y(1)=1$, and $y'(1)=4$,
which has the exact solution $y(x) = x^4$. Fig. 9 illustrates the results.



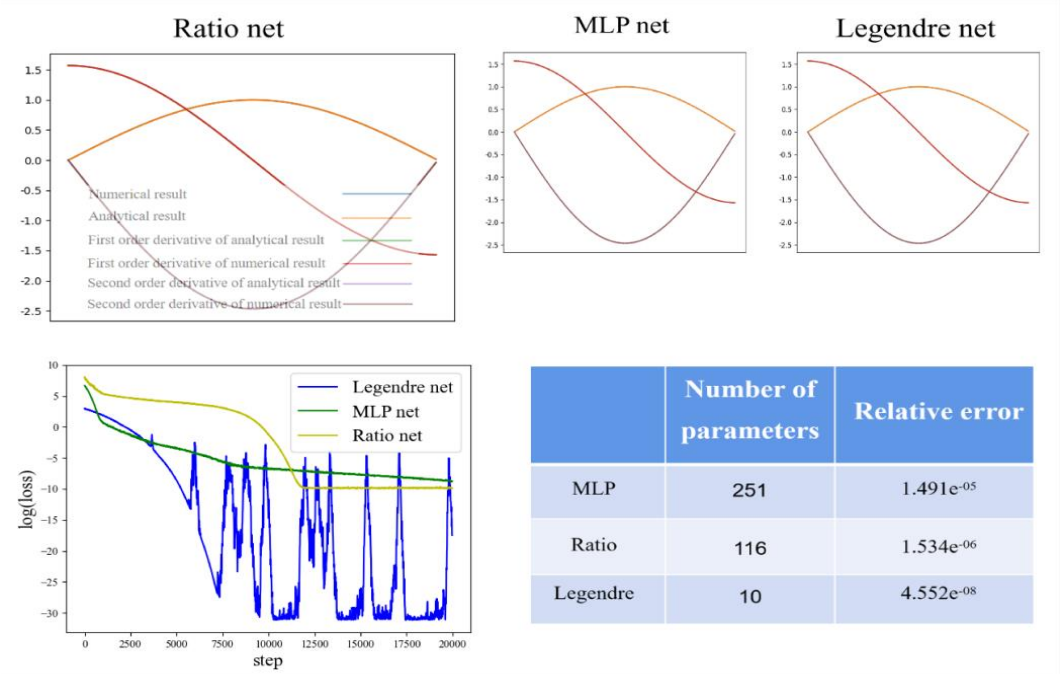
	Number of parameter	Relative error
MLP	169	$3.505e^{-05}$
Ratio	116	$1.980e^{-08}$
Legendre	6	$1.927e^{-04}$

Figure 6. Comparison of the results given by the three neural networks of example 4. The learning rates are all 0.0001.



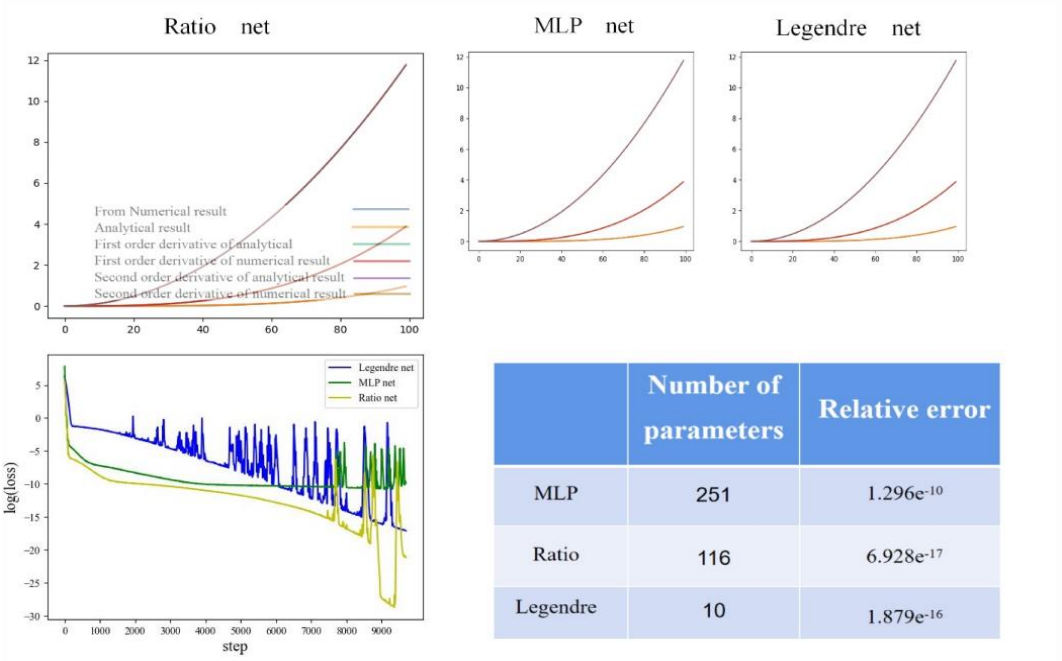
	Number of parameters	Relative error
MLP	169	$1.313e^{-05}$
Ratio	135	$1.907e^{-06}$
Legendre	10	$1.719e^{-05}$

Figure 7. Comparison of the results given by the three neural networks of example 5. The learning rates are all 0.0001.



	Number of parameters	Relative error
MLP	251	$1.491e^{-05}$
Ratio	116	$1.534e^{-06}$
Legendre	10	$4.552e^{-08}$

Figure 8. Comparison of the results given by the three neural networks of example 6. The learning rates are all 0.0001.



	Number of parameters	Relative error
MLP	251	$1.296e^{-10}$
Ratio	116	$6.928e^{-17}$
Legendre	10	$1.879e^{-16}$

Figure 9. Comparison of the results given by the three neural networks of example 7. The learning rates are all 0.0001.

3.3 Non-linear and high-order ordinary differential equations.

Example 8. Considering the ODE

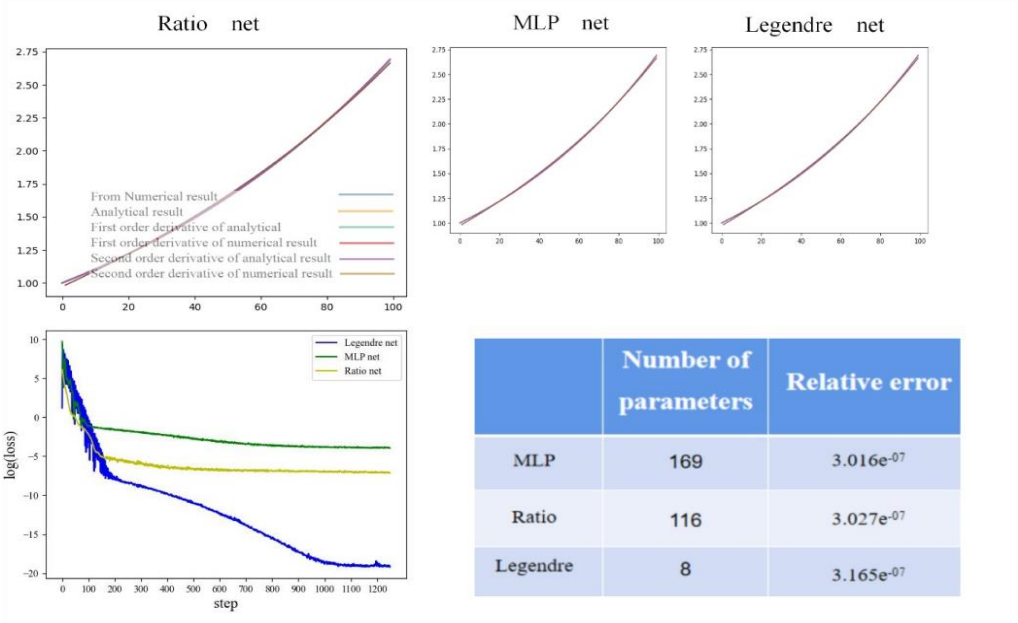
$$y^{(5)}(x) = e^{-x}y^2(x) \#(5-1)$$

with boundary conditions $y(0) = 1$, $y'(0) = 1$, $y''(0) = 1$, $y(1) = \mathrm{e}$, and $y'(1)=\mathrm{e}$, which has the exact solution $y(x) = e^x$. Fig. 10 depicts the results.

Example 9. Considering the ODE

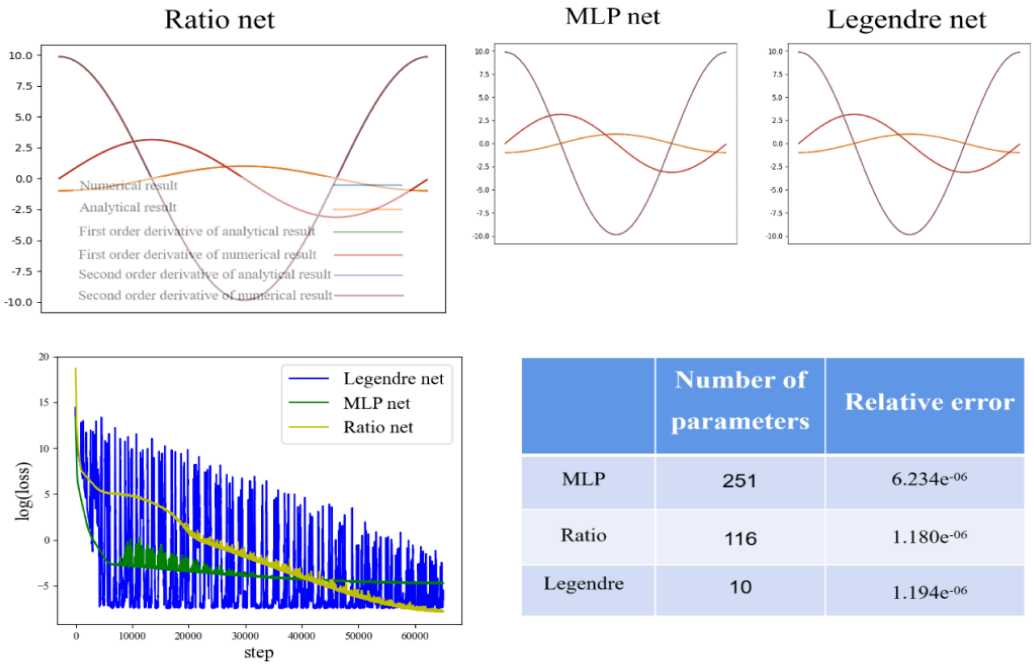
$$y^{(6)}(x) + y(x)y''(x) + y'(x)y^{(5)}(x) - \pi^2 \sin(\pi x)\, y'''(x) + \pi y^2(x) = -\pi^6 \cos(\pi x) \#(5-2)$$

with boundary conditions $y(-1) = \cos(-\pi)$, $y'(-1) = -\pi\sin(-\pi)$, $y''(-1) = -\pi^2\cos(-\pi)$, $y(1) = \cos(\pi)$, $y'(1) = -\pi\sin(\pi)$, and $y''(1) = -\pi^2\cos(\pi)$, which has the exact solution $y(x) = \cos(\pi x)$. Fig. 11 presents the results.



	Number of parameters	Relative error
MLP	169	$3.016e^{-07}$
Ratio	116	$3.027e^{-07}$
Legendre	8	$3.165e^{-07}$

Figure 10. Comparison of the results given by the three neural networks of example 8. The learning rates are all 0.01.



	Number of parameters	Relative error
MLP	251	6.234e-06
Ratio	116	1.180e-06
Legendre	10	1.194e-06

Figure 11. Comparison of the results given by the three neural networks of example 9. The learning rates are all 0.001.

3.4 Linear ordinary differential equations.

This section considers several linear ODEs.

Example 10, $y''(x) = y'(x) = -2\sin(x)$ with boundary conditions $y(0)$=-1 and $y\left(\frac{\pi}{2}\right)$=1, which has the exact solution $y = \sin(x) - \cos(x)$.

Example 11, $y'(x) + \frac{1}{5}y = e^{-x/5}\cos(x)$ with boundary conditions $y(0) = 0$ and $y(1) = \frac{\sin(1)}{e^{1/5}}$, which has the exact solution $y(x) = e^{-x/5}\sin(x)$.

Example 12, $y'(x) + \left(x + \frac{1+3x^2}{1+x+x^3}\right)y(x) = x^3 + 2x + \frac{x^2+3x^4}{1+x+x^3}$ with boundary conditions $y(0) = 1$ and $y(1) = 1 + \frac{1}{3e^{1/2}}$, which has the exact solution $y(x) = e^{-x^2/2}/(1 + x + x^3) + x^2$.

Example 13, $y'(x) - \sin(x)y(x) = 2x - x^2\sin(x)$ with boundary conditions y(-1) = 1 and $y(1) = 1$, which has the exact solution $y(x) = x^2$.

Example 14, $y''(x) + xy'(x) - 4y(x) = 12x^2 - 3x$with boundary conditions $y(0) = 0$ and $y(2) = 18$, which has the exact solution $y(x) = x^4 + x$.

Example 15, $y''(x) - y'(x) = -2\,sin(x)$ with boundary conditions $y(0) = -1$ and $y\left(\frac{\pi}{2}\right) = 1$, which has the exact solution $y(x) = \sin(x) - \cos(x)$.

Example16, $y''(x) + 2y'(x) + y(x) = x^2 + 3x + 1$ with boundary conditions $y(0) = 0$ and $y(1) = -e^{-1} + 1$, which has the exact solution $y(x) = -e^{-x} + x^2 - x + 1$.

Example17, $y''(x) + \frac{1}{5}y'(x) + y(x) = -\frac{1}{5}e^{-x/5}\cos(x)$ with boundary conditions $y(0) = 0$ and $y(2) = \sin(2)\,e^{-\frac{2}{5}}$, which has the exact solution $y(x) = \sin(x)e^{-x/5}$.

Example 18, $y'(x) = y(x) - x^2 + 1$with boundary conditions $y(0) = 0.5$ and $y(1) = 4 - \frac{\pi}{2}$, which has the exact solution $y(x) = (x + 1)^2 - 0.5e^x$.

Fig. 12 illustrates the results of examples 11-18.

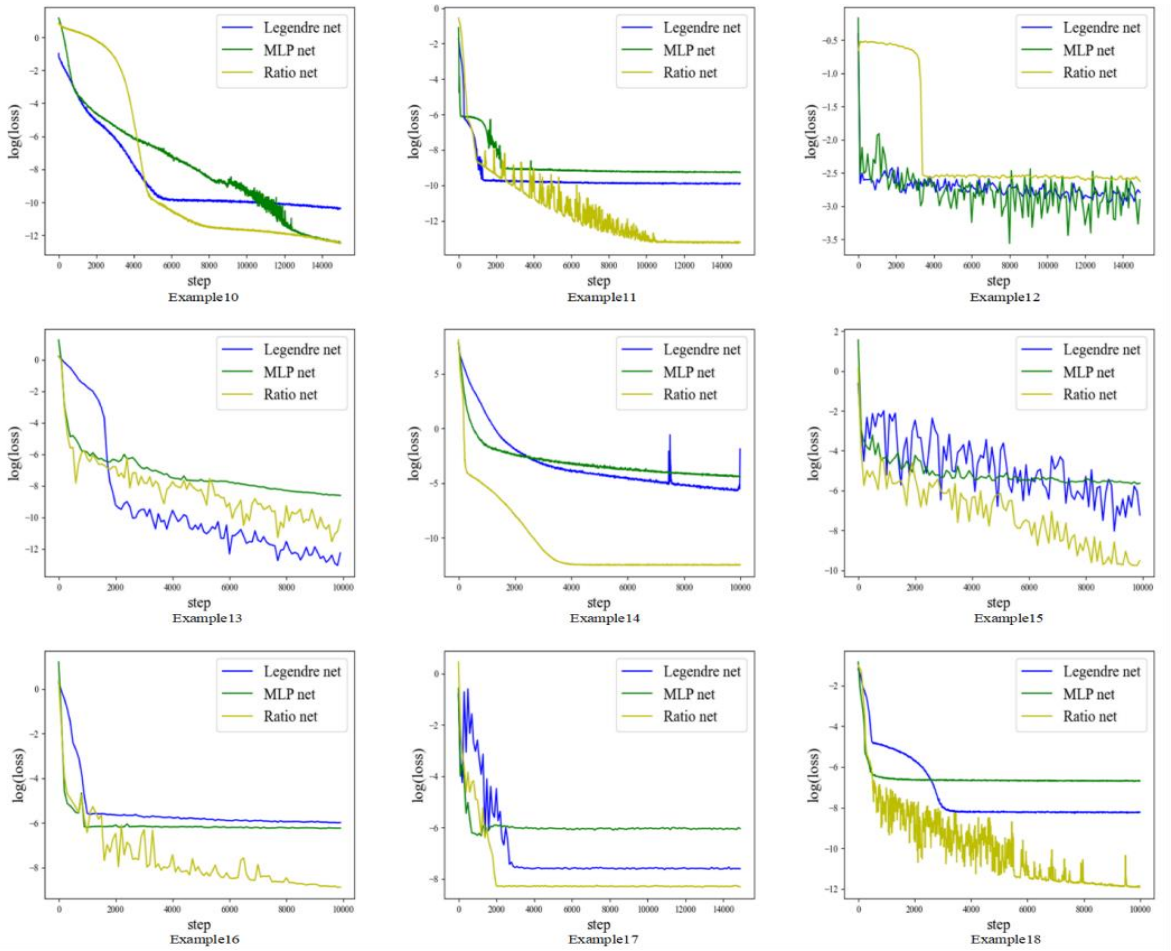


Figure 12. The loss versus steps of example 10 to 18 ,the learning rates of each examples may be different, but for the same examples, the learning rates of different models are same.

Table 1. The relative errors of examples 10–18

Relative error	Ratio	MLP	Legendre
Example 10	**3.119e−06**	3.496e−04	2.061e−05
Example 11	**1.592e−07**	1.247e−06	1.202e−06
Example 12	**1.799e−02**	2.715e−02	1.832e−02
Example 13	5.591e−06	2.236e−05	**3.136e−07**
Example 14	1.066e−05	**4.515e−06**	9.550e−05
Example 15	**5.041e−07**	9.846e−04	9.964e−05
Example 16	4.967e−06	6.803e−06	**2.391e−06**
Example 17	**3.863e−06**	8.451e−03	7.467e−06
Example 18	**8.191e−07**	2.684e−05	1.055e−05

Example 19, $y''(x)+y(x)=2$ with boundary conditions $y(0)=1$ and $y(1)=0$, which has the exact solution$y(x)=[(\cos(1)-2)/\sin(1)]\sin(x)-\cos(x)+2$.

Example 20, $y''(x)+\frac{2}{x}y'(x)+2y(x)=0$ with boundary conditions $y(0.001)=1$ and $y(1)=\sin(\sqrt{2})/\sqrt{2}$, which has the exact solution $y(x)=\sin(\sqrt{2}x)/\sqrt{2}x$.

Example 21, $y'(x)+\frac{\cos(x)}{\sin(x)}y(x)=\frac{1}{\sin(x)}$with boundary conditions $y(1)=\frac{3}{\sin(1)}$ and $y(2)=\frac{4}{\sin(2)}$,which has the exact solution $y(x)=(x+2)/\sin(x)$.

Example 22, $y''(x) - \frac{1}{1+e^x}y'(x) - \frac{15e^{2x}}{(1+e^x)^2}y(x) = \frac{e^{2x}}{(1+e^x)^6}$ with boundary conditions $y(-1) = \frac{1}{(1+e^{-1})^4}$, $y(0) = \frac{1}{2^4}$, which has the exact solution $y(x) = (1+e^x)^{-4}$.

Example 23, $y'(x) = 4x^3 - 3x^2 + 2$ with boundary conditions $y(0) = 0$ and $y(1) = 2$, which has the exact solution $y(x) = x^4 - x^3 + 2x$.

Example 24, $y'(x) = y(x)$ with boundary conditions $y(0) = 1$ and $y(1) = e$, which has the exact solution $y(x) = e^x$.

Example 25, $y'(x) = 3\cos(x)\sin^2(x) + 6\sin(3x)$ with boundary conditions $y(0) = -3$ and $y(1) = -2\cos(3) + \sin^3(1) - 1$, which has the exact solution $y(x) = -2\cos(3x) + \sin^3(x) - 1$

Example 26, $y''(x) = \frac{3}{x}[1 - y'(x)] - \frac{27}{x^3}$ with the boundary conditions $y(1) = 2$ and $y(2) = \frac{27}{4}$, which has the exact solution $y(x) = x - 3 + 27/x - 23/x^2$.

Fig. 13 presents the results of examples 19-26.

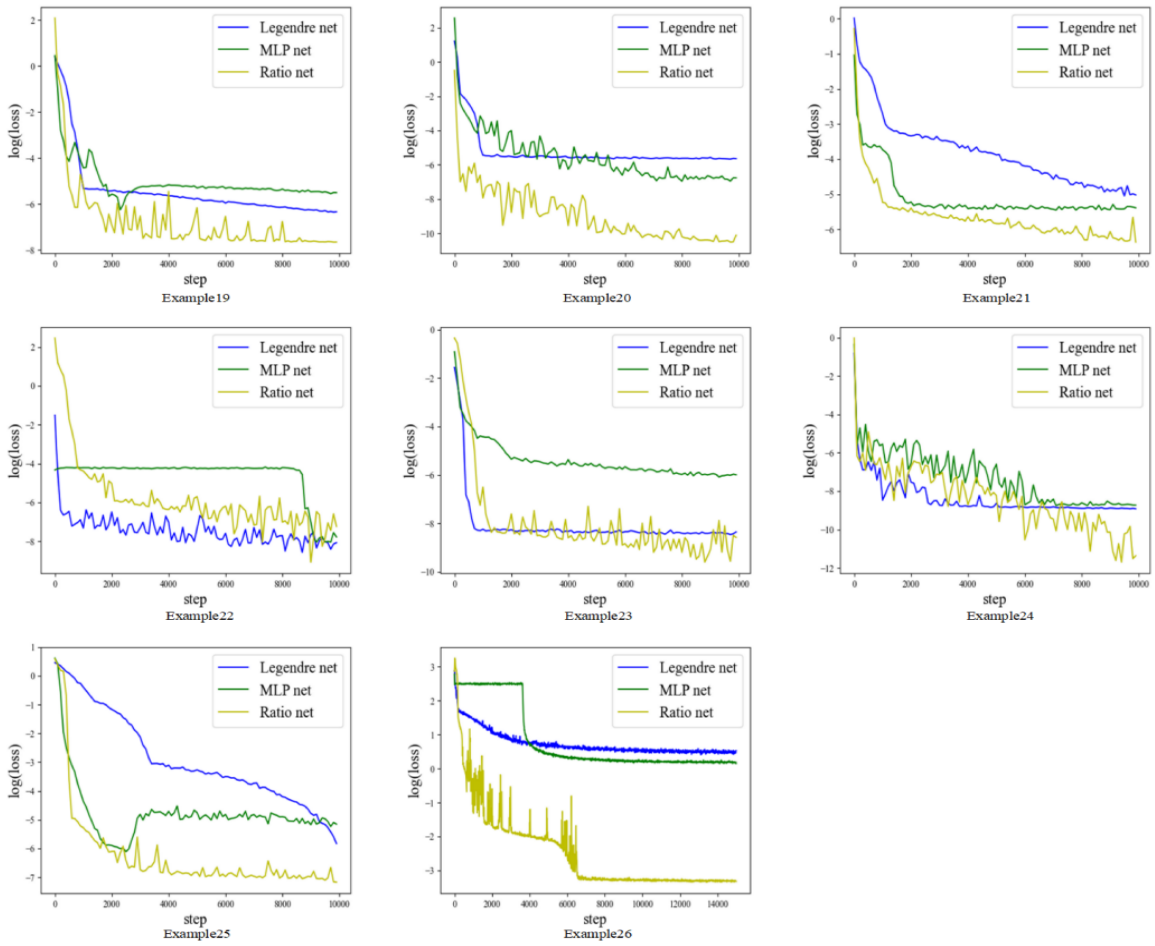


Figure 13. The loss versus steps of example 19 to 26, the learning rates of each examples may be different, but for the same examples, the learning rates of different models are same.

Table 2. The relative errors of examples 19–26

Relative error	Ratio	MLP	Legendre
Example 19	2.625e−05	4.880e−05	**2.376e−06**
Example 20	**2.092e−07**	5.598e−05	6.479e−06
Example 21	**4.484e−05**	1.578e−04	3.319e−04
Example 22	4.916e−05	**1.053e−05**	1.497e−05
Example 23	2.534e−05	5.515e−05	**9.775e−06**
Example 24	1.008e−05	**1.264e−07**	6.382e−06
Example 25	**5.541e−05**	1.110e−04	2.172e−04

Example 26	**2.730e−04**	1.473e−02	1.742e−02

IV. Conclusions and discussions

This paper proposes an effective backbone, the ratio net, that can effectively and efficiently solve the ODEs, especially the high-order and non-linear ODEs. Extensive comparisons against existing solutions such as the polynomials-based and multilayer perceptron neural network-based methods prove that the ratio net has higher efficiency.

The essence of numerical methods lies in harnessing the computational power of computers to sample and search for optimal solutions within potential spaces. Despite computers surpassing human capabilities in search, the search space corresponding to real-world problems grows exponentially, rendering brute force methods impractical. Therefore, enhancing the efficiency of computer algorithms holds paramount significance.

To improve algorithm efficiency, a common approach involves acquiring prior information about the problem and integrating it into the algorithm. This method proves effective as it enables computers to concentrate their efforts on areas where potential optimal solutions may reside. However, another equally essential approach is to enhance the efficiency of the search algorithm itself, a strategy that should not be overlooked.

This study highlights that, compared to other trial functions, ratio net exhibits greater sensitivity to parameter variations, enabling it to expedite the search for the target function within the function space. The value of this research lies in presenting a novel case, demonstrating the efficiency superiority of rational fraction approximation over traditional methods. The ratio net holds promise for advancing the efficiency and effectiveness of solving differential equations.

V. Declarations

5.1 Ethical Approval

Not Applicable

5.2 Availability of supporting data

Source code for the algorism is available at GitHub: https://github.com/zhiyiqin/The-Ratio-Net-solve-High-Order-and-the-Non-Linear-Ordinary-Differential-Equations.

5.3 Competing interests

The authors declare that they have no competing interests regarding the publication of this manuscript.

5.4 Funding

This work is supported by the National Natural Science Funds of China (Grant No. 62106033), Yunnan Youth Fundamental Research Projects (202001AU070020) and Doctoral Programs of Dali University (KYBS201910).

5.5 Authors' contributions

Zhou, C.C.: Conceptualized and implemented the algorithm, analyzed the results, reviewed, and revised the manuscript. Qin, C.X. and Liu, R.H.: Implemented and developed the algorithm, carried out the experiments. Qin, C.X., Li, M.C., and Liu, R.H.: Wrote the original draft. Li, S.Y. and Zhou, C.C.: Analyzed the results. Zhou, C.C., Li, M.C., Li, S.Y. , and Liu, Y.: Reviewed and revised the manuscript.

5.6 Acknowledgments

We are very indebted to Prof. Wu-Sheng Dai for his enlightenment and encouragement. We are very indebted to Profs. Guan-Wen Fang and Yong Xie for their encouragement.

References

[1]. Ellahi, R., Fetecau, C., & Sheikholeslami, M. (2018). Recent advances in the application of differential equations in mechanical engineering problems. Mathematical Problems in Engineering, 2018.
[2]. Antontsev, S. N., Díaz, J. I., Shmarev, S., & Kassab, A. J. (2002). Energy Methods for Free Boundary Problems: Applications to Nonlinear PDEs and Fluid Mechanics. Progress in Nonlinear Differential Equations and Their Applications, Vol 48. Appl. Mech. Rev., 55(4), B74-B75.
[3]. Betounes, D. (2010). Differential Equations: theory and applications (pp. 201-207). New York: Springer.
[4]. Zaitsev, V. F., & Polyanin, A. D. (2002). Handbook of exact solutions for ordinary differential equations. CRC press.
[5]. Corliss, G., & Chang, Y. F. (1982). Solving ordinary differential equations using Taylor series. ACM Transactions on Mathematical Software (TOMS), 8(2), 114-144.
[6]. Butcher, J. C. (2016). Numerical methods for ordinary differential equations. John Wiley & Sons.
[7]. Quarteroni, A., & Valli, A. (2008). Numerical approximation of partial differential equations (Vol. 23). Springer Science & Business Media.
[8]. Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9(5), 987-1000.
[9]. Schneidereit, T., & Breuß, M. (2020). Solving ordinary differential equations using artificial neural networks-a study on the solution variance. In Proceedings of the conference algoritmy (pp. 21-30).

[10]. Qiu, C., Bendickson, A., Kalyanapu, J., & Yan, J. (2023). Accuracy and Architecture Studies of Residual Neural Network Method for Ordinary Differential Equations. Journal of Scientific Computing, 95(2), 50.

[11]. Vinodbhai, C. D., & Dubey, S. (2023). Numerical solution of neutral delay differential equations using orthogonal neural network. Scientific Reports, 13(1), 3164.

[12]. Omidi, M., Arab, B., Rasanan, A. H., Rad, J. A., & Parand, K. (2021). Learning nonlinear dynamics with behavior ordinary/partial/system of the differential equations: looking through the lens of orthogonal neural networks. Engineering with Computers, 1-20.

[13]. Mall, S., & Chakraverty, S. (2016). Application of Legendre neural network for solving ordinary differential equations. Applied Soft Computing, 43, 347-356.

[14]. Qiu, C., Bendickson, A., Kalyanapu, J., & Yan, J. (2023). Accuracy and Architecture Studies of Residual Neural Network Method for Ordinary Differential Equations. Journal of Scientific Computing, 95(2), 50.

[15]. Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. Journal of Scientific Computing, 92(3), 88.

[16]. Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica, 37(12), 1727-1738.

[17]. 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.

[18]. Liu, C., Li, W. D., & Dai, W. S. (2023). Perturbation-based Non-perturbative Method. arXiv preprint arXiv:2308.10996.

[19]. Zhou, C.C., Tu, H.L., Liu, Y., and Hu, J.(2022). Activation functions are not needed: the ratio net. Deep Learning, 1(1), 8-16. arXiv:2005.06678.

[20]. Zhou, C.C., and Liu, Y.. (2022). The pade approximant based network for variational problems. Deep Learning, 1(1), 1-7. arXiv:2004.00711.

[21]. Yang, Y., Hou, M., & Luo, J. (2018). A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. Advances in Difference Equations, 2018(1), 1-24.

[22]. Parapari, H. F., & Menhaj, M. B. (2016, January). Solving nonlinear ordinary differential equations using neural networks. In 2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA) (pp. 351-355). IEEE.

[23]. Ezadi, S., & Parandin, N. (2013). An application of neural networks to solve ordinary differential equations. International Journal of Mathematical Modelling & Computations Vol. 03, (03), 245- 252

[24]. Ramadan, M. A., Raslan, K. R., El Danaf, T. S., & Abd El Salam, M. A. (2017). An exponential Chebyshev second kind approximation for solving high-order ordinary differential equations in unbounded domains, with application to Dawson's integral. Journal of the Egyptian Mathematical Society, 25(2), 197-205.

[25]. Akyüz-Daşcıoğlu, A., & Çerdi, H. (2011). The solution of high-order nonlinear ordinary differential equations by Chebyshev series. Applied Mathematics and Computation, 217(12), 5658-5666.

[26]. Kumar, M., & Yadav, N. (2011). Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. Computers & Mathematics with Applications, 62(10), 3796-3811.

[27]. Alli, H., Uçar, A., & Demir, Y. (2003). The solutions of vibration control problems using artificial neural networks. Journal of the Franklin Institute, 340(5), 307-325.

[28]. Shirvany, Y., Hayati, M., & Moradian, R. (2009). Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. Applied Soft Computing, 9(1), 20-29.

[29]. Gupta, J., Jayaprakash, B., Eagon, M., Selvam, H. P., Molnar, C., Northrop, W., & Shekhar, S. (2023). A Survey on Solving and Discovering Differential Equations Using Deep Neural Networks. arXiv preprint arXiv:2304.13807.

[30]. Mall, S., & Chakraverty, S. (2017). Single layer Chebyshev neural network model for solving elliptic partial differential equations. Neural Processing Letters, 45, 825-840.

[31]. Mall, S., & Chakraverty, S. (2014). Chebyshev neural network based model for solving Lane–Emden type equations. Applied Mathematics and Computation, 247, 100-114.

[32]. Chakraverty, S., & Mall, S. (2020). Single layer Chebyshev neural network model with regression-based weights for solving nonlinear ordinary differential equations. Evolutionary Intelligence, 13, 687-694.

[33]. Patra, J. C., Juhola, M., & Meher, P. K. (2008). Intelligent sensors using computationally efficient Chebyshev neural networks. IET Science, Measurement & Technology, 2(2), 68-75.

[34]. Malek, A., & Beidokhti, R. S. (2006). Numerical solution for high order differential equations using a hybrid neural network—optimization method. Applied Mathematics and Computation, 183(1), 260-271.

[35]. Dattoli, G., Ricci, P. E., & Cesarano, C. (2001). A note on Legendre polynomials. International Journal of Nonlinear Sciences and Numerical Simulation, 2(4), 365-370.

[36]. Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep learning with tensorflow: A review. Journal of Educational and Behavioral Statistics, 45(2), 227-248.