Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi
Afyon Kocatepe University – Journal of Science and Engineering
https://dergipark.org.tr/tr/pub/akufemubid

Araştırma Makalesi / Research Article
e-ISSN: 2149-3367
DOI: https://doi.org/10.35414/akufemubid.1558289
AKÜ FEMÜBİD **25** (2025) 031303 (489-496)
AKU J. Sci. Eng. **25** (2025) 031303 (489-496)

Orthogonal Embedding-Based Artificial Neural Network Solutions to Ordinary Differential Equations

Makale Bilgisi / Article Info
Alındı/Received: 30.09.2024
Kabul/Accepted: 04.01.2025
Yayımlandı/Published: 10.06.2025

Adi Diferansiyel Denklemlerin Ortogonal Gömme Tabanlı Yapay Sinir Ağı Çözümleri

Tolga Recep UÇAR[1*], Hasan Halit TALİ[2,3]

[1] *Koç University, Graduate School of Sciences and Engineering, Department of Mathematics, İstanbul, Türkiye*
[2] *Haliç University, Department of Mathematics, İstanbul, Türkiye*
[3] *Galata University, Faculty of Engineering, Department of Computer Engineering, İstanbul, Türkiye*

Abstract

Providing numerical solutions to differential equations in cases where analytical solutions are not available is of great importance. Recently, obtaining more accurate numerical solutions with artificial neural network-based machine learning methods are seen as promising developments for numerical solutions of differential equations. In this paper, a low-cost, orthogonal embedding-based network with fast training by simple gradient descent algorithm is proposed to obtain numerical solutions of differential equations. This architecture is essentially a two-layer neural network that takes orthogonal polynomials as input. The efficiency and accuracy of the method used in this paper are demonstrated in various problems and comparisons are made with other methods. It is observed that the proposed method stands out especially when compared with high-cost solutions.

Keywords: *Non-linear ordinary differential equations; Numerical approximation; Artificial neural networks; Orthogonal polynomials.*

Öz

Analitik çözümlerin mevcut olmadığı durumlarda diferansiyel denklemler için nümerik çözümler elde etmek büyük önem taşımaktadır. Son zamanlarda, yapay sinir ağı tabanlı makine öğrenmesi yöntemleriyle daha tutarlı nümerik çözümlerin elde edilmesi diferansiyel denklemlerin nümerik çözümleri için ümit verici gelişmeler olarak görülmektedir. Bu makalede, diferansiyel denklemlerin nümerik çözümlerini elde etmek için basit gradyan düşüm algoritması ile hızlı eğime sahip düşük maliyetli bir ortogonal gömme tabanlı ağ önerilmektedir. Bu mimari, temelde, ortogonal polinomları girdi olarak alan iki katmanlı bir sinir ağıdır. Bu makalede kullanılan yöntemin verimliliği ve tutarlılığı, çeşitli problemlerde gösterilmiş ve diğer yöntemlerle karşılaştırmalar yapılmıştır. Kullanılan yöntemin, özellikle yüksek maliyetli çözümlerle karşılaştırıldığında öne çıktığı görülmüştür.

Anahtar Kelimeler: *Doğrusal olmayan adi diferansiyel denklemler; Nümerik yaklaşım; Yapay sinir ağları; Ortogonal polinomlar.*

1. Introduction

Many disciplines, especially physics and economics, express their most important problems in the form of differential equations. The differential equation modeling the independent and dependent variables present in the phenomena along with the conditions that we know are true for the given independent variable formulate the initial value problem. The general initial value problem is

$$G\left(x, y(x), y'(x), \dots, y^{(n)}(x)\right) = 0 \qquad (1)$$

subject to the conditions $y(x_0) = y_0$, $y(x_0') = y_0'$,..., $y\left(x_0^{(n-1)}\right) = y_0^{(n-1)}$ where $x \in \mathbb{R}$. Our objective is to compute $y(x)$. Analytic solutions are limited to certain classes of differential equations, and they are mostly not available, so we are obliged to come up with different methodologies. Numerical methods, building the bridge between mathematics and real-life phenomena, has become an effective approach in the absence of an analytic solution. Classical methods such as Euler's and Runge-Kutta are useful, yet they require heavy computation and extensive math when applied to higher order problems. The resulting approximation being only discrete and not continuous is another limitation for these classical methods (Strang, 2007). Lately, neural network based methods are being proposed to overcome the obstacles before solving differential equations (Meade and Fernandez, 1994). The neural network based approximation is a continuous approximation of the unknown function and it is in closed, differentiable form (Lagaris et al., 1998; Hornik et al., 1989). This approximation can also handle differential equations with singularity (Chakraverty and Mall, 2017). Methodology of the neural network based solution is easy to implement and is of low computational demand when an effective architecture design is provided. The robustness of neural networks make them applicable to both ordinary and partial differential equations and system of ordinary or

*Sorumlu Yazar/Corresponding Author: Tolga Recep UÇAR e-posta/e-mail: tucar24@ku.edu.tr

partial differential equations with trivial changes to the theory.

There are various machine learning based methods for solving initial value problems. The least squares support vector machine methodology from statistical learning literature is successfully applied on differential equations (Mehrkanoon *et al.,* 2012). The fundamental neural network, i.e. multilayer perceptron with nonlinear activation functions, is considered a universal approximator. In this regard Cybenko (1989) shows that a single hidden layer along with the sigmoid function can approximate any continuous function with support in the unit hypercube, to an arbitrary degree. Later on, Pinkus (1999) reviews the answers of questions in approximation theory of neural networks. Malek and Beidokhti (2006) use a hybrid method mixing neural networks and Nelder-Mead optimization method. Wen et al. (2022) combines neural networks and the Lie group applications on differential equations. In Schiassi et al. (2021), the Theory of Functional Connections (TFC) is used to create an expression that utilizes neural networks, and this reduces the training solely to a simple least-squares, since the only parameters tuned by this methodology are the output weights. An advancement on the classic neural networks are orthogonal networks using the orthogonal polynomials as hard-coded activation functions. Most basic of these architectures is equivalent to the Chebyshev collocation method with optimized coefficients (Mall and Chebyshev, 2014). Legendre polynomials which are also the foundation of our proposal here are previously applied in the work of Mall and Chakraverty (2016). A more complex version include the repeated use of orthogonal layers in deeper neural networks (Parand *et al.*, 2024). In particular, Günel and Gör (2022) examine the swarm intelligence methods employing neural networks for solving Dirichlet boundary problems. A survey of neural network methods along with radial basis function methods for solving differential equations is the work of Kumar and Yadav (2011). Knoke and Wick (2021) present some numerical tests for the sensitivity of the objective function to show that some misleading initial weights can minimize the cost function while not satisfying the equation.

We propose an orthogonal embedding based artificial neural network with simple adjustments providing the following contributions:

1. Robust methodology of our proposal makes the solution applicable to problems from any discipline.

2. The appropriate utilization of Legendre polynomial based orthogonal representations lead to a low demand, fast runtime exceeding the accuracy of highly demanding deep architectures.

3. The efficient backpropagation results in easy convergence by basic gradient descent with constant learning rate.

We start by introducing the mathematical properties and formulate the architecture methodology in Section 2. In Section 3 the method is applied to some initial value problems. The Section 4 discusses the results and concludes the proposal.

2. Methods

In this section we discuss the Legendre polynomials, formulate the architecture and present the optimization procedure. Among the many ways to obtain approximations are series expansions. The most fundamental method in this regard is Taylor's series, which is a useful method since one can easily calculate the expansion coefficients. Orthogonal polynomial expansions are also used to obtain approximations and the theory is well studied to show that they possess useful properties for numerical analysis (Snyder, 1966). We first define the Legendre polynomials, then note that they fundamentally form an orthogonal basis, which will help us obtain useful representations of input scalars.

2.1 Legendre polynomials

Consider the below differential equation, namely Legendre's differential equation.

$$(1-x^2)P_n''(x) - 2xP_n(x) + n(n+1)P_n(x) = 0 \qquad (2)$$

The solutions $P_n(x)$ to the differential equation (2) are called the Legendre polynomials. The polynomials $P_n(x)$ are defined explicitly as

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n. \qquad (3)$$

The first five Legendre polynomials which we will use in order to create our embeddings are as the following:

$$P_0(x) = 1,$$

$$P_1(x) = x,$$

$$P_2(x) = \frac{1}{2}(3x^2-1),$$

$$P_4(x) = \frac{1}{2}(5x^3-3x).$$

The polynomials given above are graphed in Figure 1.

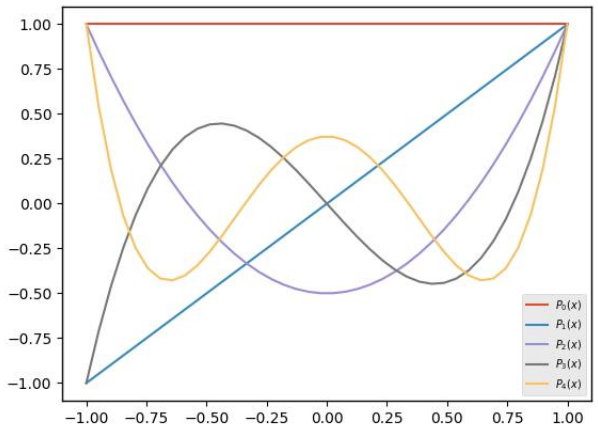

Figure 1. The first five of the Legendre polynomials.

2.2 Orthogonality and completeness

Now we state a theorem to show the orthogonality and completeness of Legendre polynomials. The Legendre polynomials are orthogonal with $w(x) = 1$ on the interval $[-1,1]$, i.e.

$$\int_{-1}^{1} P_n(x)\, P_m(x) dx = \frac{2}{2n+1}\delta_{mn} \tag{4}$$

where δ_{mn} is the Kronecker delta. Morevover, the Legendre polynomials are complete. We refer to Lebedev (1965) for orthogonality and Weidmann (1980) for completeness proof.

2.3 Orthogonal embedding-based artificial neural network

Initially, the real continuous space of the problem domain is discretized into D from which the input $x \in D \subset \mathbb{R}$ is received. In the first layer of the architecture, the orthogonal embedding represents the input x by the orthogonal basis the Legendre polynomials exhibit. The evaluated Legendre polynomials are then fed into a two-layer neural network whose hidden layers are the same size as the orthogonal embedding, parameterized by $\mathbf{p}$, with sigmoid activation function. Denoting the operations of the linear layers with matrices $L_{\mathbf{p}}^1$ and $L_{\mathbf{p}}^2$, respectively, we obtain the formulation below for the architecture output.

$$N_{\mathbf{p}}(x) = L_{\mathbf{p}}^2 L_{\mathbf{p}}^1 \begin{bmatrix} P_0(z) \\ P_1(z) \\ P_2(z) \\ P_3(z) \\ P_4(z) \end{bmatrix} \tag{5}$$

where $z = \tanh(x)$.

The computation of the orthogonal embedding is fast and the optimization is only required for the small neural network using these representations. Although small, this network can learn powerful approximations working on the orthogonal representations of inputs.

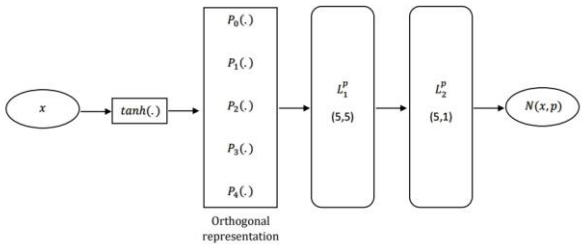

Figure 2. Orthogonal embedding-based artificial neural network. Here (5,5) and (5,1) denote the matrix dimensions of linear transformations $L_{\mathbf{p}}^1$ and $L_{\mathbf{p}}^2$, respectively.

2.4 Approximation

We formulate the approximation of $y(x)$, namely $\hat{y}_{\mathbf{p}}(x)$, using the appropriate trial solution form. For a first-order differential equation, we write

$$\hat{y}_{\mathbf{p}}(x) = y_0 + xN_{\mathbf{p}}(x) \tag{6}$$

and for a second-order differential equation we write

$$\hat{y}_{\mathbf{p}}(x) = y_0 + xy_0' + x^2 N_{\mathbf{p}}(x). \tag{7}$$

Substituting the appropriate trial solution in the equation (1) we obtain the following optimization problem.

$$\frac{1}{2}\sum_{i=1}^{N} G\left(x_i, \hat{y}_{\mathbf{p}}(x_i), \hat{y}_{\mathbf{p}}'(x_i), \dots, \hat{y}_{\mathbf{p}}^{(n)}(x_i)\right)^2 = 0 \tag{8}$$

The optimization objective is accomplished by the gradient descent algorithm. The principle is to update the parameters $\mathbf{p}$ based on the gradient of the function we aim to optimize, namely

$$F_{\mathbf{p}}(x) = \frac{1}{2}\sum_{i=1}^{N} G\left(x_i, \hat{y}_{\mathbf{p}}(x_i), \hat{y}_{\mathbf{p}}'(x_i), \dots, \hat{y}_{\mathbf{p}}^{(n)}(x_i)\right)^2. \tag{9}$$

At each step we compute

$$\nabla_{\mathbf{p}} F_{\mathbf{p}} = \frac{dF}{d\hat{y}} \nabla_{\mathbf{p}} \hat{y}_{\mathbf{p}}, \tag{10}$$

then perform the update

$$\mathbf{p} = \mathbf{p} - \gamma \nabla_{\mathbf{p}} F_{\mathbf{p}} \tag{11}$$

where γ is the learning rate tuning the update magnitude. In our numerical applications, learning rate can remain constant throughout the iterations, thus providing a simple low-cost training procedure.

3. Results and Discussions

In this section we consider some initial value problems to illustrate the efficiency of our method. First three examples are ordinary Lane-Emden type equations. The fourth example is a nonlinear ordinary Riccati type equation. Our method takes 20-equidistant points from

the problem domain and outputs a continuous approximation of the unknown function. We perform the computations on 2.20GHz Intel Xeon CPU with 12 gigabytes of memory.

For each of the examples the optimization of model parameters is demonstrated through the graph of the entries in $L_{\mathbf{p}}^2$, denoted in the figures as

$$w_i^1 \coloneqq (L_{\mathbf{p}}^2)_i. \tag{12}$$

3.1 Standard Lane-Emden with $g(y) = 1$

$$y''(x) + \frac{2}{x}y'(x) + 1 = 0 \tag{13}$$

with initial conditions $y(0) = 1,\ y'(0) = 0$. The exact solution of this initial value problem is

$$y(x) = 1 - \frac{x^2}{3!}. \tag{14}$$

In Table 1 we compare our approximations with the approximations presented by ChNN (Mall and Chebyshev, 2014). In Table 2 we compare them with the results of Parand et al. (2024). Both comparisons make it clear that our method can produce highly accurate approximations while remaining low-cost. Table 3 tests the approximation on some randomly generated points to demonstrate the smoothness of the result. In Figure 3 we compare our estimated graph for the unknown function with the exact solution (14). We give a summary in Table 4. In Figure 4 we see that the optimization successfully converges to some specific values from random initialization.

Table 1. Comparison with ChNN for example 3.1.

Input	Exact	ChNN	Proposed method	Absolute error
0	1.0000	1.0000	1.0000	0
0.1	0.9983	0.9993	0.9983	2.38e-06
0.2	0.9933	0.9901	0.9933	2.98e-06
0.3	0.9850	0.9822	0.9850	8.34e-07
0.4	0.9733	0.9766	0.9733	1.78e-07
0.5	0.9583	0.9602	0.9583	6.37e-06
0.6	0.9400	0.9454	0.9400	1.78e-07
0.7	0.9183	0.9139	0.9183	3.69e-06
0.8	0.8933	0.8892	0.8933	3.39e-06
0.9	0.8650	0.8633	0.8650	4.76e-07
1.0	0.8333	0.8322	0.8333	3.45e-06

Table 2. Comparison with Parand et al. for example 3.1.

Input	Exact	Parand et al.	Proposed method	Absolute error
0	1.0000	0.9999	1.0000	0
0.01	0.999983	0.999974	0.999984	2.38e-07
0.02	0.999933	0.999925	0.999934	5.96e-07
0.05	0.999583	0.999575	0.999585	2.02e-06
0.10	0.998333	0.998321	0.998336	2.38e-06
0.50	0.958333	0.958322	0.958340	6.37e-06
1.00	0.833333	0.833326	0.833337	3.45e-06

Table 3. Absolute errors on example 3.1. for randomly generated points.

Input	Exact	Proposed method	Absolute error
0.2840	0.9866	0.9866	8.92e-05
0.7554	0.9049	0.9049	4.67e-05
0.8290	0.8855	0.8854	1.48e-05
0.7711	0.9009	0.9009	3.63e-05
0.6168	0.9366	0.9366	7.56e-06

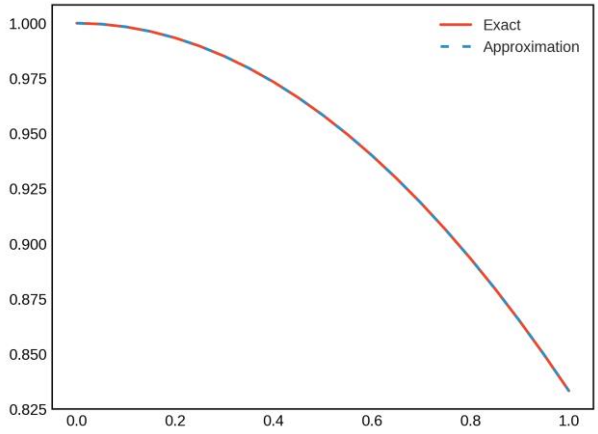

Figure 3. Comparison of the proposed solution with exact solution for example 3.1.

Table 4. Comparison of methods for example 3.1.

Method	Number of parameters	Optimization algorithm	Mean squared error
ChNN	5	Gradient descent	9.36e-06
Parand et al.	550	Adam	1.04e-10
Proposed	30	Gradient descent (constant learning rate)	6.64e-12

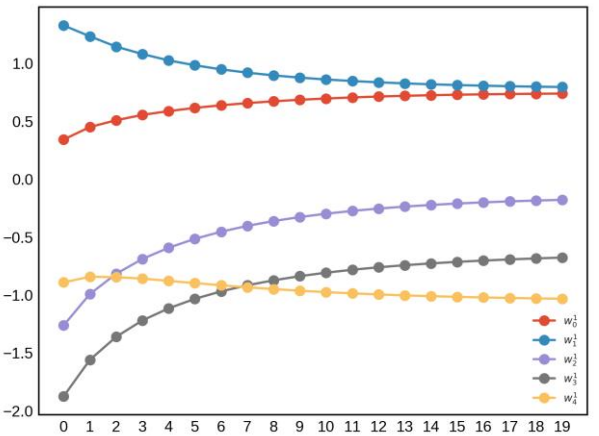

Figure 4. Convergence of the model parameters for example 3.1.

The convergence time for the solution of example 3.1. is 5 seconds.

3.2 Standard Lane-Emden with $g(y) = y^5(x)$

$$y''(x) + \frac{2}{x}y'(x) + y^5(x) = 0 \quad (15)$$

with initial conditions $y(0) = 1,\ y'(0) = 0$. The exact solution of this initial value problem is

$$y(x) = (1 + \frac{x^2}{3})^{-\frac{1}{2}}. \quad (16)$$

As above we make comparison with ChNN (Mall and Chebyshev, 2014) and Parand et al. (2024) in Table 5 and 6, respectively. In Table 7, we give some randomly generated examples to prove continuity. Figure 5 contains the graph of our estimation and the graph of the exact solution (16). The efficient usage of neural network parameters can be observed in Table 8. The convergence performance of the simple gradient descent with constant learning rate is visible in Figure 6.

Table 5. Comparison with ChNN for example 3.2.

Input	Exact	ChNN	Proposed method	Absolute error
0	1.0000	1.0000	1.0000	0
0.10	0.9983	0.9981	0.9983	4.82e-06
0.20	0.9934	0.9935	0.9934	5.36e-07
0.30	0.9853	0.9899	0.9853	5.96e-08
0.40	0.9744	0.9712	0.9744	9.59e-06
0.50	0.9607	0.9684	0.9608	1.18e-05
0.60	0.9449	0.9411	0.9449	1.84e-06
0.70	0.9271	0.9303	0.9272	7.27e-06
0.80	0.9078	0.9080	0.9078	3.45e-06
0.90	0.8874	0.8830	0.8873	7.86e-06
1.00	0.8660	0.8651	0.8660	5.00e-06

Table 6. Comparison with Parand et al. for example 3.2.

Input	Exact	Parand et al.	Proposed method	Absolute error
0	1.0000	1.0000	1.0000	0
0.01	0.999983	0.999976	0.999984	3.57e-07
0.02	0.999933	0.999927	0.999934	1.07e-06
0.05	0.999584	0.999579	0.999588	3.93e-06
0.10	0.998338	0.998332	0.998342	4.82e-06
0.50	0.960769	0.960762	0.960781	1.18e-05
1.00	0.866025	0.866022	0.866030	5.00e-06

Table 7. Absolute errors on example 3.2. for randomly generated points.

Input	Exact	Proposed method	Absolute error
0.6596	0.9345	0.9345	7.27e-06
0.4847	0.9630	0.9630	2.61e-05
0.8548	0.8967	0.8967	2.43e-05
0.2336	0.9910	0.9910	6.97e-06
0.8425	0.8993	0.8992	2.06e-05

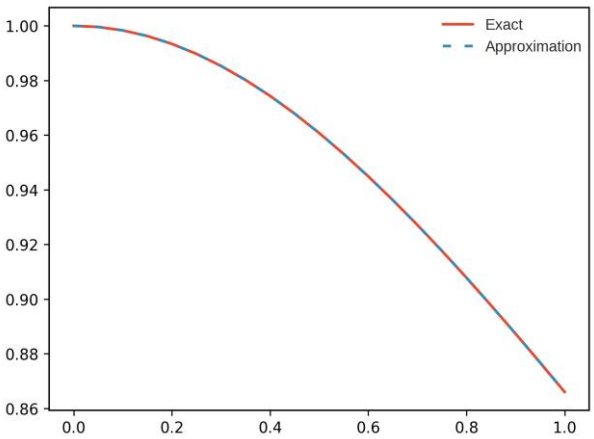

Figure 5. Comparison of the proposed solution with exact solution for example 3.2.

Table 8. Comparison of methods for example 3.2.

Method	Number of parameters	Optimization algorithm	Mean squared error
ChNN	5	Gradient descent	1.21e-05
Parand et al.	550	Adam	5.87e-11
Proposed	30	Gradient descent (constant learning rate)	3.74e-11

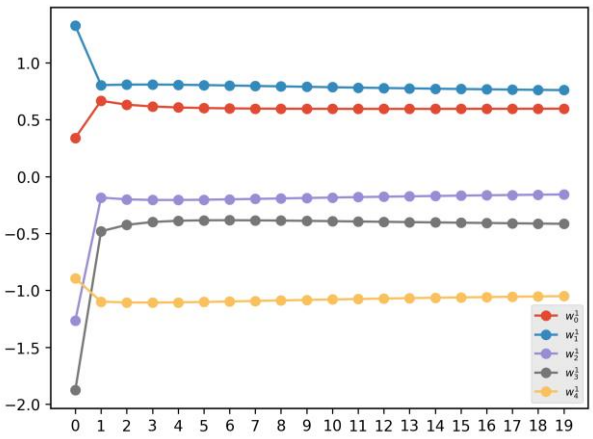

Figure 6. Convergence of the model parameters for example 3.2.

The convergence time for the solution of example 3.2. is 30 seconds.

3.3 Lane-Emden with $g(y) = -2(2x^2 + 3)$

$$y''(x) + \frac{2}{x}y'(x) - 2(2x^2 + 3)y(x) = 0 \quad (17)$$

with initial conditions $y(0) = 1,\ y'(0) = 0$. The exact solution of this initial value problem is

$$y(x) = e^{x^2}. \quad (18)$$

Table 9 and 10 compare the obtained results with ChNN (Mall and Chebyshev, 2014) and Parand et al. (2024), respectively. In Table 11 we present examples for points not present among the training points. Table 12 compares the methods. Figure 7 shows the graph of the exact solution (18) and the estimated solution obtained by our method. The succcessful convergence is observed in Figure 8.

Table 9. Comparison with ChNN for example 3.3.

Input	Exact	ChNN	Proposed method	Absolute error
0	1.0000	1.0000	1.0000	0
0.10	1.0101	1.0094	1.0101	1.97e-05
0.20	1.0408	1.0421	1.0408	2.01e-05
0.30	1.0942	1.0945	1.0942	1.70e-05
0.40	1.1735	1.1598	1.1735	2.58e-05
0.50	1.2840	1.2866	1.2840	2.02e-05
0.60	1.4333	1.4312	1.4334	2.98e-05
0.70	1.6323	1.6238	1.6323	3.01e-05
0.80	1.8965	1.8924	1.8965	2.95e-05
0.90	2.2479	2.2392	2.2480	5.12e-05
1.00	2.7183	2.7148	2.7183	5.17e-05

Table 10. Comparison with Parand et al. for example 3.3.

Input	Exact	Parand et al.	Proposed method	Absolute error
0.00	1.0000	1.0001	1.0000	0
0.01	1.0001	1.0002	1.0001	1.54e-06
0.02	1.0004	1.0005	1.0004	5.00e-06
0.05	1.0025	1.0026	1.0025	1.63e-05
0.06	1.0036	1.0037	1.0036	1.87e-05
0.07	1.0049	1.0050	1.0049	2.00e-05
0.10	1.0101	1.0101	1.0101	1.97e-05
0.20	1.0408	1.0409	1.0408	2.01e-05
0.30	1.0942	1.0942	1.0942	1.70e-05
0.40	1.1735	1.1736	1.1735	2.58e-05
0.50	1.2840	1.2841	1.2840	2.02e-05
1.00	2.7183	2.7185	2.7183	5.17e-05

Table 11. Absolute errors on example 3.3. for randomly generated points.

Input	Exact	Proposed method	Absolute error
0.5694	1.1423	1.1423	2.14e-05
0.0545	1.0016	1.0016	1.21e-05
0.9987	1.2980	1.2981	2.01e-05
0.5940	1.0691	1.0691	2.55e-05
0.6833	1.1653	1.1653	2.24e-05

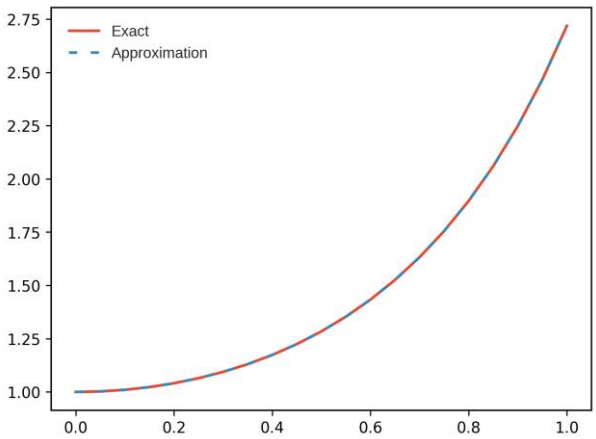

Figure 7. Comparison of the proposed solution with exact solution for example 3.3.

Table 12. Comparison of methods for example 3.3.

Method	Number of parameters	Optimization algorithm	Mean squared error
ChNN	5	Gradient descent	3.44e-05
Parand et al.	2291	Adam	1.22e-08
Proposed	30	Gradient descent (constant learning rate)	9.22e-10

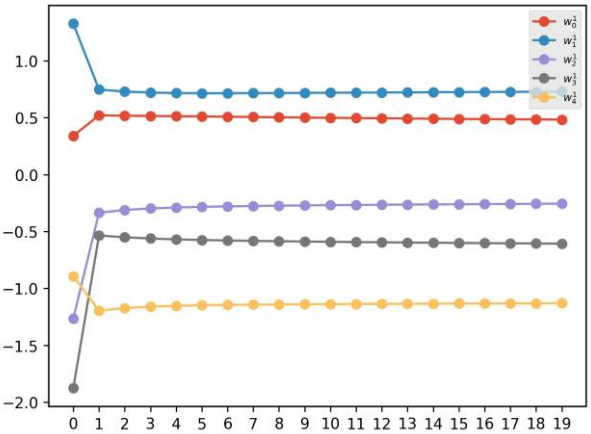

Figure 8. Convergence of model parameters for example 3.3.

The convergence time for the solution of example 3.3. is 24 seconds.

3.4 Riccati equation

Consider the Riccati equation

$$y'^{(x)} = y(x) - 2y^2(x) \tag{19}$$

with initial condition $y(0) = 1$. The exact solution of this initial value problem is

$$y(x) = \frac{1}{2-e^{-x}}. \tag{20}$$

The problem is previously solved by Gülsü and Sezer (2006) using the Taylor matrix method. We compare the errors of methods in Table 13. Our proposal achieves to exceed previous accuracies without heavy mathematics of the Taylor matrix method. Table 14 test the estimation on some new points to prove continuity. In Figure 9 we present the estimated graph and the exact solution (20). Figure 10 presents the convergence of architecture parameters.

Table 13. Comparison of methods for example 3.4.

Input	Exact	Euler's Method (Absolute Error)	Taylor Matrix (Absolute Error)	Proposed Method (Absolute Error)
0.10	0.9131	5.85e-03	2.94e-07	2.68e-04
0.20	0.8465	8.47e-03	1.48e-04	1.50e-04
0.30	0.7942	1.55e-02	1.97e-03	2.20e-04
0.40	0.7521	9.73e-03	1.11e-02	1.44e-04
0.50	0.7176	9.54e-03	4.14e-02	5.02e-05

Table 14. Absolute errors on example 3.4. for randomly generated points.

Input	Exact	Proposed method	Absolute error
0.5694	0.6973	0.6975	4.92e-04
0.0545	0.9497	0.9507	9.60e-04
0.9987	0.6129	0.6130	1.97e-04
0.5940	0.6907	0.6909	5.17e-04
0.6833	0.6689	0.6691	4.15e-04

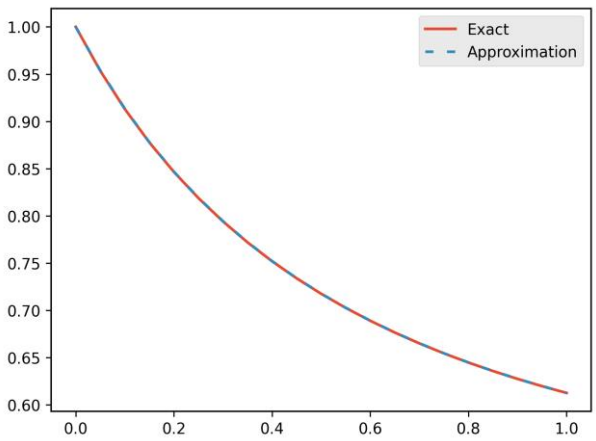

Figure 9. Comparison of the proposed solution with exact solution for example 3.4.

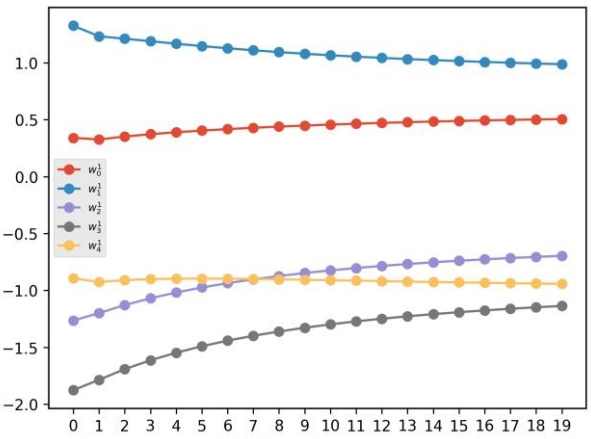

Figure 10. Convergence of model parameters solving example 3.4.

The convergence time for the solution of example 3.4. is 29 seconds.

4. Conclusion

We have introduced a low-cost method for solving nonlinear ordinary differential equations which exceeds the efficiency of previous approaches. This fast computation of complex unknown functions is achieved through the fundamental two-layer neural network with improved representations. Our method evaluates the corresponding Legendre polynomials in order to compute an orthogonal representation that work as improved representations to learn neural network parameters. The illustrations on various examples show that this method can achieve better results while also reducing computational cost. Alongside these improvements, the proposed method is applicable to a wide range of problems and the solution is continuous while the input is discrete. We observe that the illustration of orthogonal representations as empowering embeddings for simple neural networks to work on implies further important research.

Declaration of Ethical Standards

The authors declare that they comply with all ethical standards.

Credit Authorship Contribution Statement

Author 1: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Visualization.
Author 2: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – review and editing, Supervision.

Declaration of Competing Interest

The authors have no conflicts of interest to declare regarding the content of this article.

Data Availability

All data generated or analyzed during this study are included in this published article.

5. References

Chakraverty, S. and Mall, S., 2017. Artificial Neural Networks for Engineers and Scientists. CRC Press. https://doi.org/10.1201/9781315155265

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems*, **2**, 303-314. https://doi.org/10.1007/BF02551274

Gülsü, M. and Sezer, M., 2006. On the solution of the Riccati equation by the Taylor matrix method. *Applied Mathematics and Computation*, **176**, 414-421. https://doi.org/10.1016/j.amc.2005.09.030

Günel, K. and Gör, I., 2022. Solving Dirichlet boundary problems for ODEs via swarm intelligence. *Mathematical Sciences*, **16**, 325-341. https://doi.org/10.1007/s40096-021-00424-2

Hornik, K., Maxwell, S. and Halbert, W., 1989. Multilayer feedforward networks are universal approximators. *Neural Network,* **2**, 359-366. https://doi.org/10.1016/0893-6080(89)90020-8

Knoke, T. and Wick, T., 2021. Solving differential equations via artificial neural networks: Findings and failures in a model problem. *Examples and Counterexamples,* **1**. https:/doi.org/10.1016/j.exco.2021.100035

Kumar, M. and Yadav, N., 2011. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey. *Computers & Mathematics with Applications*, **10**, 3796-3811. https://doi.org/10.1016/j.camwa.2011.09.028

Lagaris, I.E., Likas, A. and Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. on Neural Netw.,* **9**, 987-1000. https://doi.org/10.1109/72.712178

Lebedev, N.N., 1965. Special Functions and Their Applications. Prentice-Hall.

Malek, A. and Beidokhti, R.S., 2006. Numerical solution for high order differential equations using a hybrid neural network—Optimization method. *Applied Mathematics and Computation,* **1**, 260-271**.** https://doi.org/10.1016/j.amc.2006.05.068

Mall, S. and Chakraverty, S., 2014. Chebyshev Neural Network based model for solving Lane–Emden type equations. *Applied Mathematics and Computation*, **247**, 100-114. https://doi.org/10.1016/j.amc.2014.08.085

Mall, S. and Chakraverty, S., 2016. Application of Legendre Neural Network for solving ordinary differential equations. *Applied Soft Computing*, **43**, 347-356. https://doi.org/10.1016/j.asoc.2015.10.069

Meade, A.J. and Fernandez, A.A., 1994. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling,* **19**, 1-25**.** https://doi.org/10.1016/0895-7177(94)90095-7

Mehrkanoon, S., Falck, T. and Suykens, J.A.K., 2012. Approximate Solutions to Ordinary Differential Equations Using Least Squares Support Vector Machines. *IEEE Trans. Neural Netw. Learn. Syst.*, **23**, 1356-1367. https://doi.org/10.1109/TNNLS.2012.2202126

Parand, K., Aghaei, A.A., Kiani, S., Ilkhas Zadeh, T. and Khosravi, Z., 2024. A neural network approach for solving nonlinear differential equations of Lane–Emden type. *Engineering with Computers*, **40**, 953-969. https://doi.org/10.1007/s00366-023-01836-5

Pinkus, A., 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, **8**, 143-195. https://doi.org/10.1017/S0962492900002919

Schiassi, E., Furfaro, R., Leake, C., De Florio, M., Johnston, H., Mortari, D., 2021. Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing*, **457**, 334-356. https://doi.org/10.1016/j.neucom.2021.06.015

Snyder, M.A., 1966. Chebyshev Methods in Numerical Approximation. Prentice-Hall.

Strang, G., 2007. Computational Science and Engineering. Wellesley-Cambridge Press. https://doi.org/10.1137/1.9780961408817

Weidmann, J., 1980. Linear Operators in Hilbert Spaces. Springer-Verlag.

Wen, Y., Chaolu, T. and Wang, X., 1980. Solving the initial value problem of ordinary differential equations by Lie group based neural network method. *PLOS ONE*, **17(4)**. https://doi.org/10.1371/journal.pone.0265992