ISSN: 0067-2904

Solving 2nd Order Volterra Integro-Differential Equations by Using IDRLnet Library and Physics-informed neural networks

Oday Ahmed Jasim[1]*, Abdulghafor M. Al-Rozbayani[2]
[1]*Department of Mathematics, College of Basic Education, University of Mosul, Mosul, Iraq*
[2]*Department of Mathematics, College of Computer science and Mathematics, University of Mosul, Mosul, Iraq*

Received: 10/10/2024 Accepted: 23/ 2/2025 Published: 30/3/2026

Abstract

Physics-informed neural networks (PINNs) are a novel deep learning model that excels at solving both inverse and forward non-linear partial differential equation (PDE) problems. This paper presents a new algorithm implemented using Python and the IDRLnet library. The new algorithm of the open-source deep learning package is designed to solve PDEs using PINNs and is mainly used for physics-informed deep learning. This paper uses PINN to solve the 2nd-order Volterra integro-differential equations (VIDE) for the first time, in which a series of iterative solutions are established based on the Gauss-Legendre quadratic method after converting the 2nd-order VIDE into a PDE. In addition, three different examples are presented and solved using the new algorithm to highlight the accuracy and efficiency of the proposed method. It is found that the L2 relative error is between 0.1e-3 and 0.1e-4, but the absolute error is between 0.1e-3 and 0.1e-5, which is a good result., and the "training" time for all cases is 414.08, 724.2, 1339.31, and 2260.8 seconds for all the cases taken.

Keywords: Volterra Integro-Differential Equations (VIDE, Physics-Informed Neural Networks (PINN), IDRLnet Liberty, Deep Learning (DL), Gauss-Legendre Quadrature (GLQ).

حل معادلات فولتيرا التكاملية التفاضلية من الدرجة الثانية باستخدام مكتبة *IDRLnet* والشبكات العصبية المستندة إلى الفيزياء

عدي احمد جاسم[1]*, عبد الغفور محمد امين[2]
[1]قسم الرياضيات، كلية التربية الاساسية، جامعة الموصل، موصل، العراق
[2]قسم الرياضيات، كلية علوم الحاسوب والرياضيات، جامعة الموصل، موصل، العراق

الخلاصة

الشبكات العصبية المستنيرة بالفيزياء (PINNs) هي نموذج جديد للتعلم العميق يتفوق في حل كل من مشاكل المعادلات التفاضلية الجزئية غير الخطية العكسية والأمامية. تقدم هذه الورقة خوارزمية جديدة تم تنفيذها باستخدام برنامج البايثون ومكتبة IDRLnet. تم تصميم الخوارزمية الجديدة لحزمة التعلم العميق مفتوحة المصدر لحل المعادلات التفاضلية الجزئية باستخدام الشبكات العصبية المستنيرة بالفيزياء، حيث

* Email odayalnoamy@uomosul.edu.iq

تستخدم بشكل أساسي التعلم العميق المستنير بالفيزياء .تستخدم هذه الورقة الشبكات العصبية المستنيرة بالفيزياء لحل معادلات فولتيرا التفاضلية التكاملية من الرتبة الثانية لأول مرة، يتم إنشاء سلسلة من الحلول التكرارية بناء على طريقة كاوس ليجندر التربيعية بعد تحويل معادلات فولتيرا التفاضلية التكاملية من الرتبة الثانية إلى المعادلات التفاضلية الجزئية. يتم تقديم ثلاثة أمثلة مختلفة وحلها باستخدام الخوارزمية الجديدة لتسليط الضوء على دقة وكفاءة الطريقة المقترحة .وجد أن الخطأ النسبي L2 في جميع الحالات يتراوح بين 0.1e−3 و0.1e−4 وهي نتيجة جيدة، ووقت التدريب لجميع ,414.08 ,724.2 ,1339.31 و2260.8 ثانية وفقا للمعلمات المختارة.

1. Introduction

Volterra's population dynamics studies from the start of the 20th century inspired the need for modeling systems with spatiotemporal linkages beyond local modeling, which led to Integro-Differential Equations (IDEs). IDEs have unique properties due to their non-local nature [1-3]. Computational biology, physics, and engineering uses functional differential equations, like IDEs, to represent dynamical systems with non-local properties, as Ordinary Differential Equations (ODEs) and Differential Delay Equations (DDEs) cannot. Similar to ODEs and PDEs, one of the main issues in the theory is the existence and uniqueness problem for IDEs. Several general solutions, such as those obtained by the Tychono fixed point or Schauder theorems, guarantee that the initial value problem for Equation (1) admits solutions provided that the function k above is subject to acceptable regularity constraints. Under other assumptions, the solutions are likewise distinct and continuous with respect to the initial data [4]. Furthermore, a range of strategies for resolving IDEs are presented in [3, 5-9].

Integro-differential equations were used to explain a wide range of physical processes. These formulas are essential to the creation of novel physical phenomenon explanations as well as the mathematical representation of several theories. For instance, one is often interested in one of the following types of phenomena while studying continuous media or fields [5]:

- ❖ Sound wave propagation in turbulent media is a phenomenon that cannot be moved from one location to another.
- ❖ Phenomenon, when biological molecules are transported or diffused via permeable soils [10].

Numerous physical processes, including the glass process [11], nanohydrodynamics [12], drop-wise condensation [13], and wind ripple in the desert [14], result in integro-differential equations. Such problems may be solved using a variety of numerical and analytical techniques, but each approach is limited to a particular class of integro-differential equations.

Volterra studied the genetic implications while he was analyzing a population development model. The research effort resulted in a peculiar topic, where both integral and differential operators happened simultaneously in the same equation. This new kind of equation was dubbed VIDEs, defined in the form:

In general, integro-differential equations are:

$$p^{(n)}(x) = f(x) + \lambda \int_0^x K(x,t)\, p(t)\ dt, \quad p^{(i)}(0) = a_i,\ \ 0 \le i \le k-1, \tag{1}$$

where $p^{(n)}(x) = \frac{d^n p}{dx^n}$.

The VIDE (1) necessitates defining initial conditions such as p(0), p'(0), ..., $p^{n-1}(0)$ to ascertain the particular solution p(x). This requirement arises because the resulting equation in (1) merges the integral operator with the differential operator. A VIDE is defined by the presence of one or more derivatives outside the integral sign, such as $p'(x)$, $p''(x)$, … . When we use the Leibnitz rule to transform an initial value issue into an integral equation, we can get the VIDE, [15].

Numerous vision-related tasks, such as image identification and classification, video analysis, and medical image analysis, have shown the exceptional efficacy of the CNN family of Deep Learning algorithms [16-19].

The term "PINN" was first used by Rassi et al. in 2019, [20]. Physics-informed neural networks are those that perform supervised learning tasks within physical boundaries or somehow capture the partial differential equation (via the loss function, for example).

In recent years, the study of deep learning has focused increasingly on PINN for data-driven PDE solution development. The fundamental concept behind PINN is to use a neural network to represent the solutions to PDEs. The parameters of the neural network are trained using gradient descent of the loss function connected with PDEs, together with boundary and initial conditions. The deep learning community specifically uses the automated differentiation approach to handle derivatives[21].

PINN is solved the 2^{nd}-order VIDE, in which a series of iterative solutions are established based on the Gauss-Legendre quadratic method after converting the 2nd-order VIDE into a PDE [22].

Because of its versatility and grid-less design, PINN has been extensively used in scientific computing up to this point, particularly in the forward and inverse problem solutions of nonlinear PDEs [23-25]. Both the governing differential equations and data are used to train PINN simultaneously. PINNs are considered a useful tool for solving differential equations that control physical systems, and they are attractive because they can solve inverse problems with the least amount of code change needed for forward problems [21, 26-29].

Although partial differential equations have been solved using NNs since the 1990s [30], there has been a recent upsurge in interest in PINN in various scientific fields. This is attributable to the ease of access to open-source libraries like PyTorch [31] and TensorFlow [32], which provide practical features like high-performance processing and automated differentiation. Several libraries use Physics-Informed Neural Networks (PINNs) to solve partial differential equations (PDEs) & (VIDE). For instance, see DeepXDE [22, 25], NeuroDiffEq [33], IDRLnet [34].

The following scientific papers deal with the topic of solving integro-differential equations using PINN as follows:

Lu et al. [25] published the Python library DeepXDE for PINN, which is meant to serve both as a teaching tool and a research tool for solving difficulties in computational science and engineering. To improve PINN training efficiency, he also developed a novel residual-based adaptive optimization (RAR) approach. More specifically, inverse issues with extra data and forward problems with initial boundary conditions are both handled by DeepXDE. More broadly, DeepXDE facilitates the quick development of the emerging field of scientific machine learning. Lu et al. used a degree 20 GLQ to approximate the integral and found that the L2 relative error is 2e-3 when solving the 1st-order VIDE using PINN and DeepXDE.

Wei Peng [34] published IDRLnet, a Python framework for methodically modeling and tackling issues using PINN. IDRLnet provides a base for a large variety of PINN algorithms and applications. The numerical integral is approximated by a Gaussian-Legendre square of degree 10.

The auxiliary PINN (A-PINN), invented by Yuana et al. [35], is a unique PINN architecture intended to tackle forward and inverse problems with nonlinear integral differential equations. It is proven that the recommended A-PINN gives a more accurate solution than the usual PINN by specifying the auxiliary output variables to represent the integrals in the governing equation and employing automated differentiation of the auxiliary output to replace the integral operator. The 2D nonlinear Volterra IDE, the Volterra nonlinear IDEs, the Fredholm nonlinear IDE, and the 10D nonlinear Volterra IDE were among the front-end challenges in which A-PINN excelled. Lastly, the inverse issue of nonlinear IDs is tackled

using the A-PINN framework, and the results demonstrate that even with highly noisy data, unknown parameters can be recognized sufficiently.

The objectives of this study are to solve the second-order Volterra integro-differential equations for the first-time using PINN and the IDRLnet library (All previous works did not solve the 2nd -order VIDEs) find and reduce the square of the relative error and absolute error. The strength of the new proposed algorithm is demonstrated through calculated and decreased L2 relative and absolute error, and three separate examples are used to illustrate this.

The following is the structure of the parts that follow in this work: In Section 2, the IDRLnet library's core design for the second-order VIDE solver is introduced, along with a synopsis of the PINN. In Section 3, it demonstrates three examples using PINN and IDRLnet to apply the second-order VIDE and provide the outcomes. Lastly, Section 4 provides the most significant conclusions.

2. Problem statement

A family of non-local functional differential equations in time are called integral-differential equations (IDEs). IDEs naturally describe many dynamical systems, nuclear reactor physics, including population dynamics, and viscoelastic fluids, as studied in depth. Further examples are models of brain dynamics, as well as infectious illness spreading.

Volterra investigated the genetic effects when assessing a population growth model. The research effort produced a unique topic in which the differential operators and integral appeared simultaneously in an equation. In this paper it will discuss a special case of these equations, namely the second-order Volterra integro-differential equations, as follows:

$$y^{(2)}(x) = f(x) + \lambda \int_0^x K(x,t)\, y(t)\, dt. \tag{2}$$

Where $y^{(2)}(x) = \frac{d^2y}{dx^2}$.

The Volterra integro-differential Equation (2) requires the definition of initial conditions $y(0), y'(0)$ in order to get the specific solution y(x). This is because the resultant Equation (2) combines the differential operator with the integral operator.

3. Methodology

IDRLnet is an open-source tool developed in Python based on physics-informed deep learning and in the PyTorch framework. That is, the programming interface of this work is aimed at the scientific and engineering purposes such as variational minimization, parameter identification, surrogate training, and differential equations solving.

The library is named after the Robust Learning and Intelligent Design (IDRL) Laboratory (IDRLnet), a Python framework for systematically modeling and addressing issues using PINN. IDRLnet provides a foundation for many other PINN applications and algorithms. It gives a formal framework to merge data sources, geometric objects, ANN, optimizers, and loss metrics within Python.

3.1. Physics-informed neural networks

The main concept behind using neural networks to solve PDEs is to reframe the issue as one of optimization, whereby the residual of the DEs has to be minimized. The PDE with IC and BC conditions may be described as follows without losing generality:

$$\begin{aligned} \mathcal{L}[u](x) &= q(x), && x\epsilon\, \Omega \text{ X}[0,T] \\ \mathcal{B}[u](x) &= q(x), && x\epsilon\, \partial\Omega \text{ X}[0,T] \\ u(x,0) &= u_0(x), && x\epsilon\Omega, \end{aligned} \tag{3}$$

where $\mathcal{L}$ and $\mathcal{B}$ are the differential operators, u is the solution, and Ω is the spatio-temporal domain [34].

Given that an exact solution u frequently resides in an infinite-dimensional space, parameterizing the solution u as u_θ is one possible numerical technique to get close to the real solution. Within the PINN framework, the model u_θ serves as a replacement artificial neural network, with the neural network's biases and weights represented by the parameter θ.

The R^d into R^N is mapped by the surrogate $u_\theta(x)$. With the activation function $\sigma(x) = \text{swish}(x)$ [10], IDRLnet uses a basic Multi-Layer Perceptron (MLP) by default, i.e.,

$$u_\theta := L_m \, o \, \sigma \, o \, L_{m-1} o \, \sigma \, o \, \cdots \, o \, \sigma \, o \, L_1.$$

Where

$$\begin{aligned} L_1(x) &:= W_1 + b_1, & W_1 \in R^{d_1 \times d}, b_1 \in R^{d_1}, \\ L_i(x) &:= W_i + b_i, & W_1 \in R^{d_i \times d_{i-1}}, b_i \in R^{d_i}, \\ L_m(x) &:= W_{im} + b_m, & W_m \in R^{m \times d_{D-1}}, b_D \in R^m. \end{aligned} \qquad \forall i = 2,3,\cdots,m-1,$$

If the differential equation is solved for u_θ, then the residual terms are found.

$$\begin{aligned} R_{pde} &= \mathcal{L}[u_\theta] - q, & x \epsilon \Omega \, \mathrm{X}[0,T] \\ R_b &= \mathcal{B}[u_\theta] - u_b, & x \epsilon \, \partial\Omega \, \mathrm{X}[0,T] \\ R_0 &= u_\theta(.,0) - u_0, & x \epsilon \Omega \end{aligned} \qquad (4)$$

are identically zero [34].

Unfortunately, because u_θ is a finite-dimensional approximation, it is nearly impossible for (4) equals to 0. During a neural network's training phase, one common method is to use the PDE-induced residuals as part of the loss function. Therefore, the so-called PINN is fundamentally PDE-informed, which is the core principle that IDRLnet builds on, but other forms of loss functions are obtained from observable external data.

With a specified loss function $L(\cdot)$, solving the PDE turns into an optimization problem [34]:

$$\min_\theta Loos = \frac{1}{|\mathrm{S_{pde}}|} \sum_{x \epsilon s_{pde}} L\left(R_{pde}(x)\right) + \frac{1}{|\mathrm{S_b}|} \sum_{x \epsilon s_b} L\left(R_b(x)\right) + \frac{1}{|\mathrm{S_0}|} \sum_{x \epsilon s_0} L\left(R_0(x)\right), \qquad (5)$$

where sets of sample points from the initial, inner, and boundary domains are represented by the variables $S_0, S_b,$ and S_{pde}, respectively. An instance of a PINN structure is provided in Figure 1.

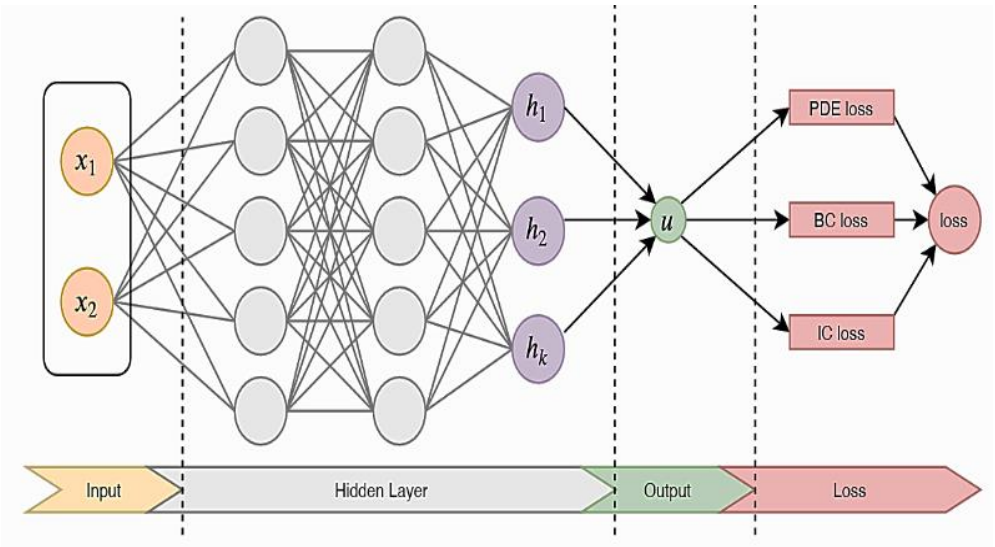

Figure 1: A PINN scheme for resolving generic PDEs, [34].

3.2. Solving 2nd-Order VIDEs

The second-order VIDEs on [a, b], with $u = h(x)$, are the exact solutions as:

$$\frac{d^2u}{dx^2} + g(x) = \int_0^x (x-t)u(t)\,dt. \quad (6)$$

Algorithm 1 for implementing the solution of the second order of differential integral equations is as follows:

Algorithm 1 IDRLnet for solving 2nd order Volterra integro-differential equations.

Step 1	Import the required packages or libraries: ○ Import 'idrlnet.shortcut' package as sc. ○ Import a 'Sympy' package named sp. ○ Import a 'Numpy' package named np. ○ Import the 'Matplotlib.pyplot' package as plt.
Step 2	Definition of symbols and functions: ○ The symbol x is defined. ○ Definition the symbol t as time. ○ Definition the u as function. ○ Definition the derivative of the function u with respect to x.
Step 3	Definition of points and intervals and bounds for Volterra Integro-Differential Equation: ○ Define the 'interior function' to specify interior points and the constraints associated with them ○ Define the 'init function' to specify initial points and their associated constraints. ○ Define the 'infer function' to specify the points that will be used in inference.
Step 4	Neural network definition: ○ Defining a neural network using the code sentence 'sc.get_net_node'
Step 5	Definition of partial differential equations: ○ Define partial differential equations (PDEs) that contain the constraints that need to be solved.
Step 6	Problem solving: ○ Using the code 'solver.infer_step' to solution the problem.
Step 7	Neural network optimization
Step 8	Error calculation and Plot the results: ○ Calculate absolute error and relative error. ○ After solving the problem, the results are plotted using the code 'matplotlib.pyplot.' ○ Time calculation.

4. Results And discussion

3.3. Results

It will solve three cases of the 2nd order VIDE with starting conditions using Algorithm 1 to demonstrate the PINN's precision and effectiveness and support the success of the suggested approach.

The aim is to ascertain the optimal solutions for the second-order Volterra integral-differential equations by calculating the number of points for (x) to calculate (IDRLnet and PINN) and the accuracy of solving the integral portion using the Gauss-Legendre quadratic technique. Along with assessing the deep neural network's training time, for every case, the outcomes were documented in the tables and figures that follow.

Example 4.1.1:

The VIDE of the 2nd order in [0,6] and $u(x) = sinx$ is the exact solution, [5].

$$u''(x) = -x + \int_0^x (x-t)\, u(t)dt, \qquad u(0) = 0, u'(0) = 1. \tag{7}$$

Solution:

When solving VIDE, it uses the method of automatic differentiation (AD) to compute integer-order derivatives, both numerically and analytically. Additionally, it employs the GLQM to estimate integral operators. First, use the GLQM to calculate the integral part:

$$\int_0^x (x-t)\, u(t)dt \approx \sum_{i=1}^{n} w_i\big(x - t_i(x)\big)u\big(t_i(x)\big)\, dt. \tag{8}$$

Now, using PINN, we will solve the specified PDE Equation (9) instead of the original Equation (7).

$$\frac{d^2u}{dx^2} + x \approx \sum_{i=1}^{n} w_i(x - t_i(x))\, u\big(t_i(x)\big)\, dt \tag{9}$$

As shown in Table 1, the solution to Equation (9) yields the value of X, the predicted value of Equation (9), the exact value of Equation (7), and the absolute error. Figure 2 shows how the sine function converges on the interval [0, 6] at different numbers of iterations.

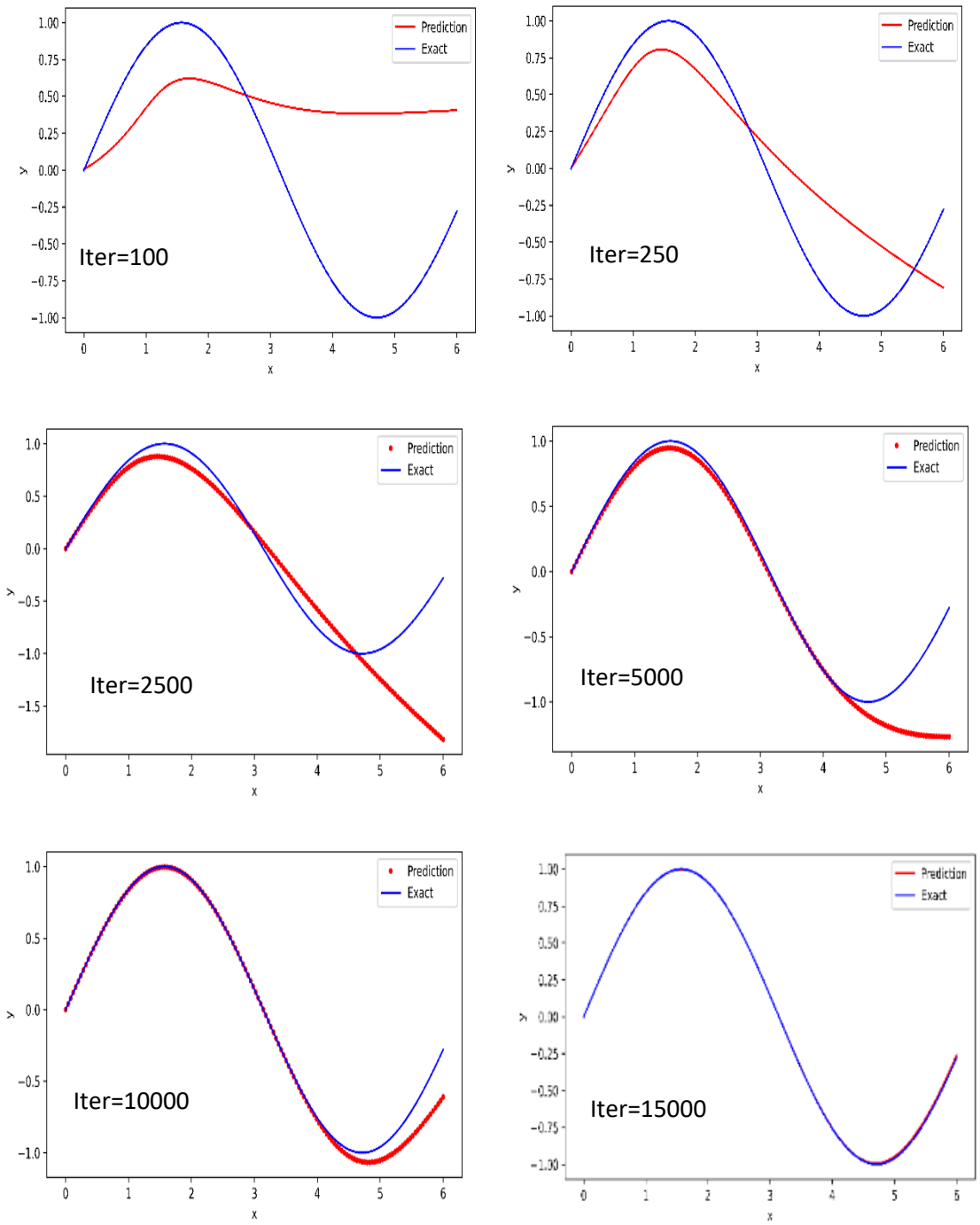

Figure 2: The graph shows how the sine function converges on the interval [0, 6] at different numbers of iterations (Iter), L2 Relative Error = 0.007457045, Gauss-Legendre quadratic = 30, and time 'training' = 2260.8 second.

Table 1: Represents the exact solution, the predicted value, and the absolute error at several test points in the [0, 6] domain. Legendre quadrature points are applied for training by 30 degrees, time 'training' = 2260.8 second, numbers of iteration =15000.

No. of x	Value of x	Pred of Equation (9)	Exact solution of Equation (7)	Absolute error
1	0	-1.27917160E-05	0	1.2791716E-05
2	0.122449	0.12215488	0.1221432	1.1660E-05
3	0.244898	0.24249779	0.2424573	4.0460E-05
4	0.367347	0.35919663	0.3591406	5.6000E-05
5	0.489796	0.47052902	0.4704458	8.322E-05
20	2.326531	0.72840744	0.727768	6.3938E-04
21	2.44898	0.63928604	0.63855	7.3570E-04
22	2.571429	0.54060227	0.53977	8.3202E-04
23	2.693878	0.43383098	0.432907	9.2394E-04
42	5.020408	-0.94698626	-0.95294	5.94974E-03
43	5.142857	-0.90184134	-0.90877	6.92916E-03
44	5.265306	-0.84290081	-0.851	8.09545E-03
45	5.387755	-0.77105981	-0.78048	9.41843E-03
50	6	-0.26154885	-0.27942	1.786665E-02

Example 4.1.2:

Let the 2nd-order VIDE be in the interval [0, 4], with the exact solution $p(x) = e^x$, [5].

$$p''(x) = 1 + x + \int_0^x (x - t)\, p(t)\, dt, \qquad p'(0) = p(0) = 1 \tag{10}$$

Solution:

When solving VIDE, it uses the method of automatic differentiation (AD) to compute integer-order derivatives, both numerically and analytically. Additionally, it employs the GLQM to estimate integral operators. First, use the GLQM to calculate the integral part:

$$\int_0^x (x - t)\, p(t)\, dt \approx \sum_{i=1}^{n} w_i\big(x - t_i(x)\big)p\big(t_i(x)\big)\, dt. \tag{11}$$

Now, using PINN, it will solve the specified PDE Equation (12) instead of the original Equation (10).

$$\frac{d^2p}{dx^2} - 1 - x \approx \sum_{i=1}^{n} w_i\big(x - t_i(x)\big)p\big(t_i(x)\big)\, dt. \tag{12}$$

The solution of Equation (12) contains the value of X, the predicted value of Equation (12), the precise value of Equation (10), and the absolute error. These results are shown in Table 2. Figure 3 illustrates the accuracy between the estimated solutions and the actual value (e^x). The estimated value was obtained by solving the IDE using PINN. Figure 4 illustrates the convergence of the expositional function across the interval [0, 4] at varying iterations.

Table 2: The table showed the (x) values, the true values, and the approximate values that resulted from the new algorithm using PINN, numbers of iteration (Iter) = 10000, L2 Relative Error = 0.00018295646, Absolute Error= Gauss-Legendre quadratic = 15, time 'training' = 724.2 seconds, and number of interior points = 100, Number of prediction points=50.

No. of x	Value of x	Exact solution of Equation (12)	Pred of Equation (10)	Absolute Error	L2 Relative Error
1	0.00000000000000E+00	1.00000000000000E+00	1.00018095970153E+00	1.80959701538085E-04	1.80959701538085E-04
2	8.16326513886451E-02	1.08505713939666E+00	1.08475327491760E+00	3.03864479064941E-04	2.80044681858271E-04
3	1.63265302777290E-01	1.17734909057617E+00	1.17710435390472E+00	2.44736671447753E-04	2.07870951271615E-04
4	2.44897961616516E-01	1.27749097347259E+00	1.27815485000610E+00	6.63876533508300E-04	5.19672175869345E-04
5	3.26530605554580E-01	1.38615059852600E+00	1.38622117042541E+00	7.05718994140625E-05	5.09121455252170E-05
12	8.97959172725677E-01	2.45458841323852E+00	2.45251917839050E+00	2.06923484802246E-03	8.43006826471537E-04
13	9.79591846466064E-01	2.66336894035339E+00	2.66286706924438E+00	5.01871109008789E-04	1.88434700248762E-04
14	1.06122446060180E+00	2.88990712165832E+00	2.89081168174743E+00	9.04560089111328E-04	3.13006632495671E-04
15	1.14285719394683E+00	3.13571500778198E+00	3.13704133033752E+00	1.32632255554199E-03	4.22972923843190E-04
16	1.22448980808258E+00	3.40242958068847E+00	3.40321540832519E+00	7.85827636718750E-04	2.30960737098939E-04
22	1.71428573131561E+00	5.55270767211914E+00	5.55237865447998E+00	3.29017639160156E-04	5.92535507166758E-05
23	1.79591834545135E+00	6.02500486373901E+00	6.02547121047973E+00	4.66346740722656E-04	7.74018844822421E-05
29	2.28571438789367E+00	9.83270835876464E+00	9.83237743377685E+00	3.30924987792968E-04	3.36555276589933E-05
30	2.36734700202941E+00	1.06690492630004E+01	1.06680450439453E+01	1.00421905517578E-03	9.41245089052245E-05
40	3.18367338180541E+00	2.41352462768554E+01	2.41322154998779E+01	3.03077697753906E-03	1.25574733829125E-04
41	3.26530623435974E+00	2.61881294250488E+01	2.61867027282714E+01	1.42669677734375E-03	5.44787581020500E-05
42	3.34693884849548E+00	2.84156150817871E+01	2.84143447875976E+01	1.27029418945312E-03	4.47040874860249E-05
43	3.42857146263122E+00	3.08325634002685E+01	3.08299369812011E+01	2.62641906738281E-03	8.51832883199676E-05
47	3.75510215759277E+00	4.27385864257812E+01	4.27435531616210E+01	4.96673583984375E-03	1.16211980639491E-04
48	3.83673477172851E+00	4.63738021850585E+01	4.63831672668457E+01	9.36508178710937E-03	2.01947681489400E-04
49	3.91836738586425E+00	5.03182220458984E+01	5.03301277160644E+01	1.19056701660156E-02	2.36607535043731E-04
50	4.00000000000000E+00	5.45981483459472E+01	5.46160507202148E+01	1.79023742675781E-02	3.27893445501103E-04

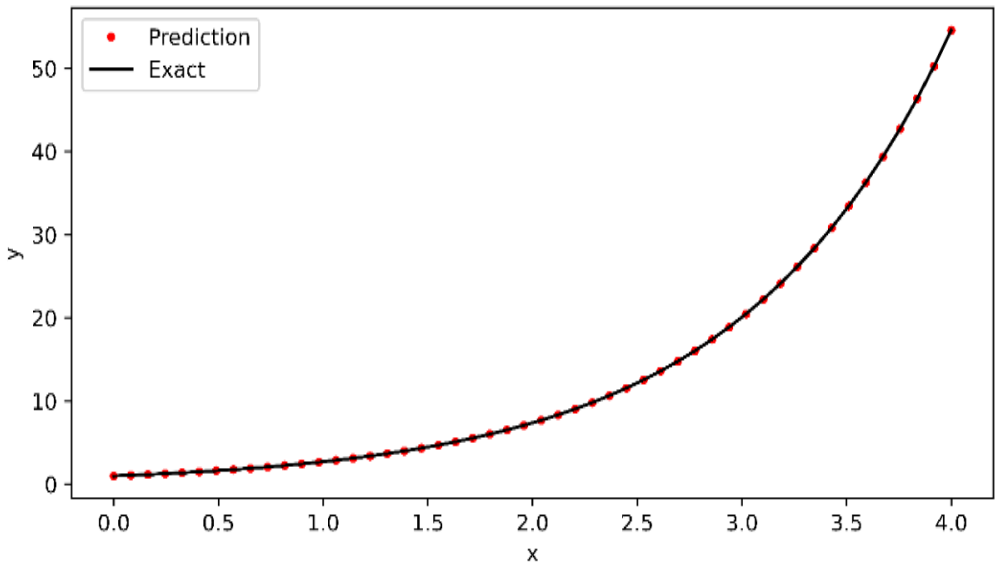

Figure 3: The graph shows the exact (exponential function) solutions with the approximate solutions (which emerged from solving the 2nd order VIDE using PINN) on the interval [0, 4], numbers of iteration (Iter) = 10000, L2 Relative Error = 0. 00018295646, Gauss-Legendre quadratic = 15, time 'training' = 724.2 seconds, and number of interior points = 100.

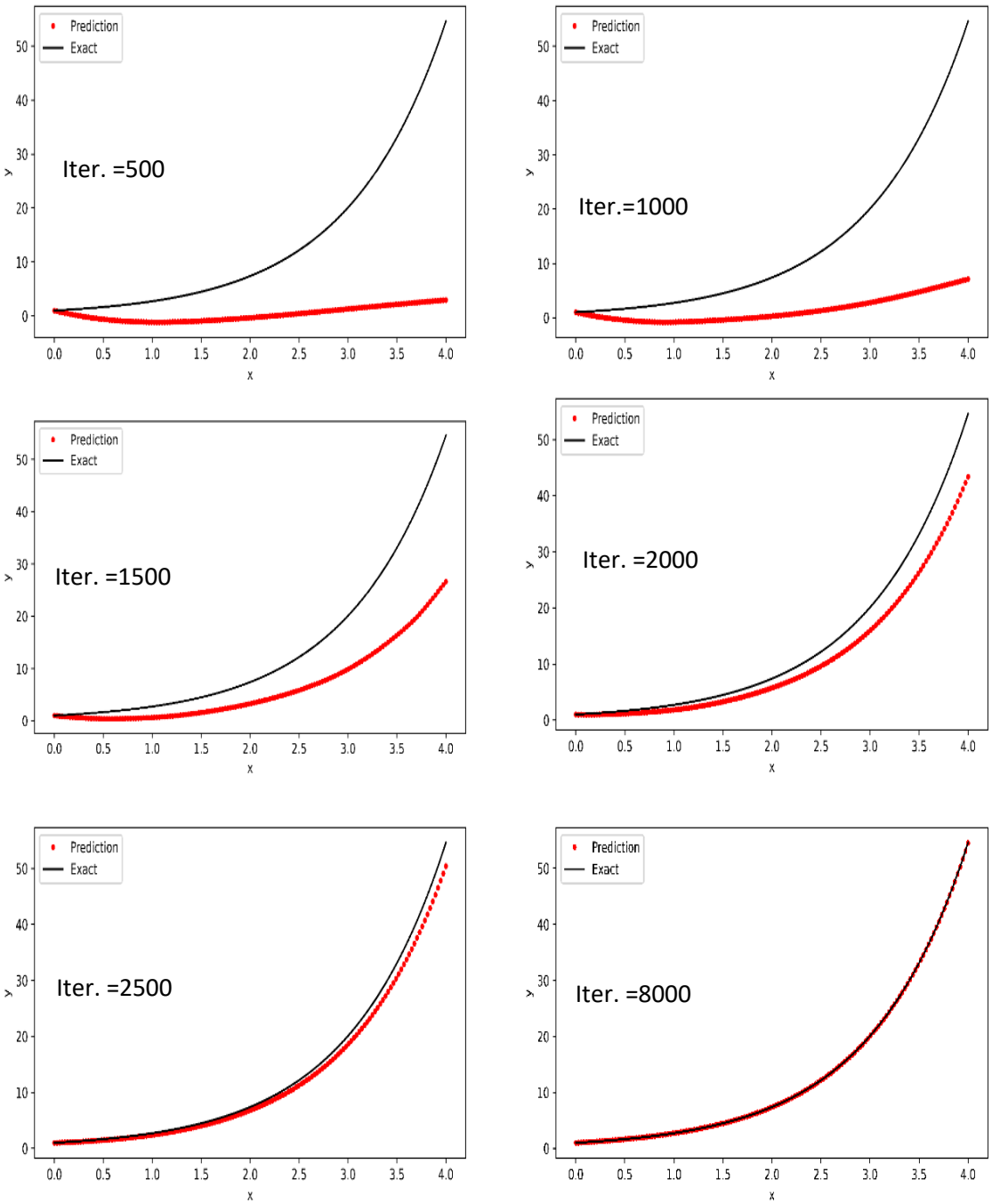

Figure 4: The graphs show the convergence of the exact (exponential function) value of the approximate solution (which originated from solving the 2nd order VIDE using PINN) on the interval [0, 4] at different numbers of iterations (Iter). The following results of Iter = 8000 are: L2 Relative Error = 0.0007446787, Gauss-Legendre quadratic = 5, time 'training' = 414.08 seconds, and number of interior points = 50.

Example 4.1.3:
Solving the VIDE in [0,5], where $u(x) = sinx + cosx$, is the exact solution, [5].

$$u''(x) = -1 - x + \int_0^x (x-t)\, u(t)\, dt, \qquad u(0) = 1, u'(0) = 1. \tag{15}$$

Solution:
To estimate the integral, we first use the Gauss-Legendre quadratic approach to a specific degree.

$$\int_0^x (x-t)\, u(t)\, dt \approx \sum_{i=1}^{n} w_i\big(x - t_i(x)\big)u\big(t_i(x)\big)\, dt\,. \tag{16}$$

Now, using PINN, we will solve the specified PDE Equation (15) instead of the original Equation (17).

$$\frac{\mathrm{d}^2 u}{\mathrm{dx}^2} + 1 + x \approx \sum_{i=1}^{n} w_i \left(x - t_i(x)\right) u\big(t_i(x)\big)\, dt. \tag{17}$$

As seen in Table 3, the solution to Equation (15) provides the value of X, the expected value of Equation (17), the exact value of Equation (15), and the absolute error.

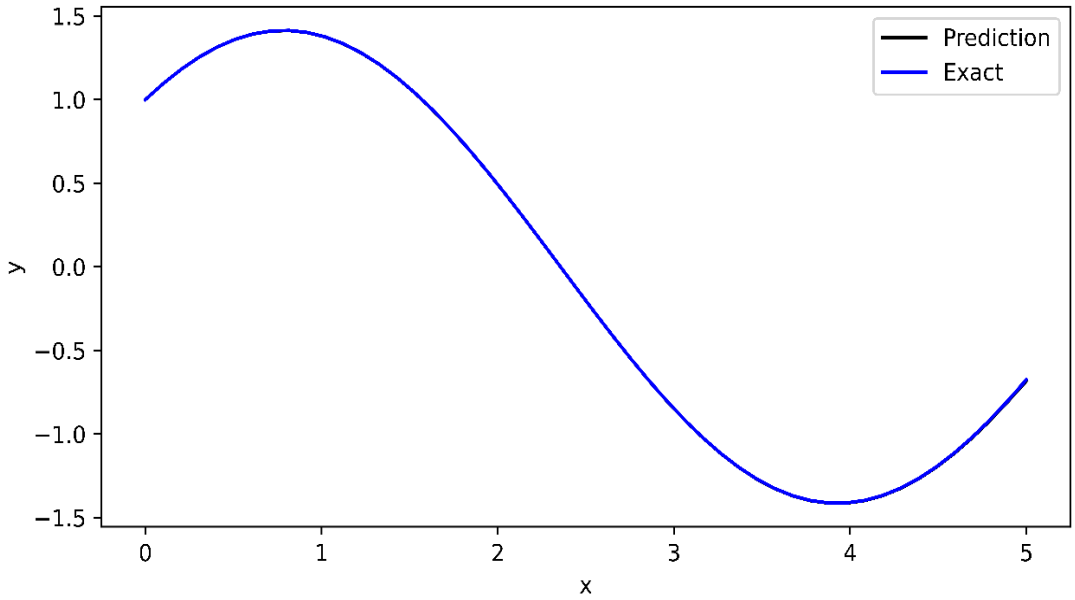

Figure 5: The graph shows the exact (sin(x)+cos(x)) value with the approximate solution (which emerged from solving the 2nd order VIDE using PINN) on the interval [0, 5], numbers of iteration (Iter) = 15000, L2 Relative Error = 0.001478439, Gauss-Legendre quadratic = 5, time 'training' = 1339.31 seconds, and number of interior points = 150.

Table 3: The table showed the (x) values, the true values, and the approximate values that resulted from the new algorithm using PINN, numbers of iteration (Iter) = 15000, L2 Relative Error = 0.001478439, Gauss-Legendre quadratic = 5, time 'training' = 1339.31 seconds, and number of interior points = 150.

No. of x	Value of X	Pred of Equation (17)	Exact solution of Equation (15)	Absolute Error
1	0.000000000E+00	1.000008941E+00	1.000000000E+00	8.940696716E-06
2	1.020408198E-01	1.096621275E+00	1.096662164E+00	4.088878632E-05
3	2.040816396E-01	1.181811333E+00	1.181915402E+00	1.040697098E-04
4	3.061224520E-01	1.254712820E+00	1.254872918E+00	1.600980759E-04
5	4.081632793E-01	1.314560175E+00	1.314775586E+00	2.154111862E-04
16	1.530612230E+00	1.038726330E+00	1.039366007E+00	6.396770477E-04
17	1.632653117E+00	9.356601238E-01	9.362701178E-01	6.099939346E-04
18	1.734693885E+00	8.228754997E-01	8.234341145E-01	5.586147308E-04
19	1.836734653E+00	7.015142441E-01	7.020316124E-01	5.173683167E-04
20	1.938775539E+00	5.728280544E-01	5.733255148E-01	4.974603653E-04
30	2.959183693E+00	-8.023562431E-01	-8.020104766E-01	3.457665443E-04
31	3.061224461E+00	-9.169182777E-01	-9.164905548E-01	4.277229309E-04
32	3.163265228E+00	-1.021954060E+00	-1.021435976E+00	5.180835724E-04
33	3.265306234E+00	-1.116364956E+00	-1.115755558E+00	6.093978882E-04
34	3.367347002E+00	-1.199159145E+00	-1.198467135E+00	6.920099258E-04
45	4.489795685E+00	-1.198153496E+00	-1.196087837E+00	2.065658569E-03
46	4.591836929E+00	-1.115653038E+00	-1.113002658E+00	2.650380135E-03
47	4.693877697E+00	-1.021739006E+00	-1.018338919E+00	3.400087357E-03
48	4.795918465E+00	-9.173874855E-01	-9.130810499E-01	4.306435585E-03
49	4.897959232E+00	-8.036475182E-01	-7.983241677E-01	5.323350430E-03
50	5.000000000E+00	-6.815962791E-01	-6.752620935E-01	6.334185600E-03

3.4. Discussion

❖ By comparing the true solution with the solution of the PINN model, it is visible that the approximate solution is very close to the real one throughout the entire range, and therefore, the main trends of the behaviour of the function are described sufficiently well.

❖ The improved accuracy of the approximation raises the model's reliability when it comes to providing accurate solutions to complex problems.

❖ Quantitative Analysis: The discrepancies between the approximate and exact answers are negligible, demonstrating the model's capacity to manage noisy data and provide precise outcomes.

❖ The chosen interval inside the solution domain determines the degree of the solution's accuracy. In other words, the further the interval, the lower the accuracy, and vice versa.

❖ The importance of the L2 relative error:

1. Accuracy assessment: Provides an accurate estimate of how near the calculated answer is to the genuine solution, assisting in assessing the model's efficacy.
2. Appropriate for large-scale issues: Can be used to examine the correctness of solutions in high-dimensional problems, making it appropriate for assessing PINNs solutions.

3. Ease of interpretation: Plain to interpret which means that smaller numbers represent correct solutions while bigger numbers represent less accurate solutions.

❖ The reason for choosing the relative error in this paper is:

1. complete evaluation: Provides a complete assessment of the correctness of solutions across all points, making it suited for assessing PINN solutions that deal with complicated partial differential equations.
2. Handling high dimensions: Can be utilized to evaluate solutions in high-dimensional situations, making it suited for assessing PINNs solutions that deal with large-scale optimization challenges.
3. Ease of interpretation: Easy to interpret it as a way of understanding the correctness of solutions and to make alternatives to improve models.

❖ Absolute error has many advantages when used to assess the accuracy of solutions in PINNs or in any other methodology. The reason for choosing the Absolute error in this paper is:

1. Ease of Understanding: Absolute error is a simple and easy-to-understand metric, as it directly represents the difference between the true value and the estimated value.
2. Comprehensive Evaluation: Absolute error can be used to assess the accuracy of solutions across all points, providing a comprehensive picture of the model's performance.
3. Handling Small Values: Absolute error is particularly useful when dealing with small values, as it can be more accurate than relative metrics in these cases.
4. Not Affected by Large Values: Unlike some other metrics, absolute error is not disproportionately affected by large values, making it a balanced measure for evaluating accuracy.
5. Ease of Calculation: Calculating absolute error is straightforward and does not require complex computations, making it suitable for use in many applications.

❖ In the above examples, the "time training" ranged from 2750 to 400 seconds in all circumstances.

❖ The GLQ degrees used in this study are 5, 15, and 30, and the greater the value, the more precisely the Volterra equation's integral may be calculated.

❖ The correctness of the solution is significantly impacted by the number of iterations used to train the network; the more iterations, the higher the accuracy. This is made very evident in Figures 1 and 3, as well as in example 1. Please keep in mind that we needed many iterations to adjust for this.

❖ The good discovery is that in all examined cases, the "L2 relative error" is in the range between 0.1e-3 and 0.1e-4.

❖ A good result, the "absolute error" is located between 0.1e-3 and 0.1e-5 in this study.

❖ Finally, we should note that we used Python (9.3) with several libraries such as: Numpy (2.2.0), Deepxde (1.12.2), IDRLnet (2.0.0), PyYAML (6.0.2), paddlepaddle (2.6.2), matplotlib (3.10.0), Torch (2.5.1), mpmath (1.4.0), pandas (2.2.4), packaging (24.2), and many more.

5. Conclusions

This study provides a novel approach that uses the IDRLnet Library and a physics-informed neural network (PINN) to solve the second-order Volterra integro-differential equations. The integration operator of the second-order Volterra integro-differential equations was approximated using the Gauss-Legendre quadratic numerical approach. We then used automated distinction and a neural network with knowledge of physics to solve the resultant problem. It used automated differentiation of extra outputs and differential factors in integro-differential equations (IDEs), selecting a multi-output, multi-input deep neural network to

represent the cooperative and basic integrals in the governing equations concurrently. Three examples for VIDE were resolved to confirm the stability and soundness of the suggested network. The results demonstrated the excellent accuracy and effectiveness of this network. It is found that the L2 relative error is between 0.1e-3 and 0.1e-4, but the absolute error is between 0.1e-3 and 0.1e-5, which is excellent. A "training" period is 414.08, 724.2, 1339.31, 2260.8, seconds for all the cases taken. Finally, it has been found that the solution's accuracy is inversely proportional to the function domain in the solution area and vice versa. The number of network training iterations also influences solution correctness. The Volterra integro- differential equation solution is more accurate as the Gauss-Legendre technique degree increases.

Acknowledgement

This work was supported by University of Mosul/College of Basic Education and College of Computer science and Mathematics. Thanks for the help provided to successfully complete this work.

References

[1] V. Volterra, Lectures on integral equations and integro-differential equations. Gauthier-Villar, 1913.

[2] V. Volterra, Theory of Functionals and of Integral and Integro-differential Equations, 1930.

[3] S. V. Meleshko, Y. N. Grigoriev, N. Kh. Ibragimov, and V. F. Kovalev, Symmetries of Integro-Differential Equations: With Applications in Mechanics and Plasma Physics. Springer Science & Business Media, 2010.

[4] N. J. Linden, D. R. Tabuena, N. A. Steinmetz, W. J. Moody, S. L. Brunton, and B. W. Brunton, "Go with the FLOW: visualizing spatiotemporal dynamics in optical widefield calcium imaging," *Journal of the Royal Society Interface*, vol. 18, no. 181, pp. 20210523, Aug. 2021,

[5] A. M. Wazwaz, Linear and Nonlinear Integral Equations. Springer Science & Business Media, 2011.

[6] O. A. Jasim, "Application of the Operational Matrices for Solving Nonlinear Volterra Integral Equations System of the Second Kind", BERJ, vol. 12, no. 1, pp. 773–785, Mar. 2013.

[7] E. H. H. Doha, M. a. A. Abdelkawy, A. Z. M. Z. M. Amin, and D. Baleanu, "Shifted Jacobi spectral collocation method with convergence analysis for solving integro-differential equations and system of integro-differential equations," *Nonlinear Analysis Modelling and Control*, vol. 24, no. 3, pp. 332–352, Apr. 2019.

[8] R. M. Ganji, H. Jafari, and S. Nemati, "A new approach for solving integro-differential equations of variable order," *Journal of Computational and Applied Mathematics*, vol. 379, pp. 112946, Apr. 2020.

[9] K. Issa, J. Biazar, T. O. Agboola, and T. Aliu, "Perturbed Galerkin Method for Solving Integro-Differential Equations," *Journal of Applied Mathematics*, vol. 2022, pp. 1–8, Apr. 2022.

[10] L. A. Girifalco, Atomic migration and molecular diffusion in porous soils. New York, Harvard University Press; 1999.

[11] H. Wang, H. M. Fu, H. F. Zhang, and Z. Q. Hu, "A Practical Thermodynamic Method to Calculate the Best Glass-forming Composition for Bulk Metallic Glasses," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 8, no. 2, Jan. 2007.

[12] L. Xu, J.-H. He, and Y. Liu, "Electrospun Nanoporous Spheres with Chinese Drug," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 8, no. 2, pp. 99-202 2007.

[13] F. Z. Sun *et al.*, "The Fractal Dimension of the Fractal Model of Dropwise Condensation and Its Experimental Study," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 8, no. 2, Jan. 2007.

[14] T.-L. Bo, L. Xie, and X. J. Zheng, "Numerical Approach to Wind Ripple in Desert," *International*

Journal of Nonlinear Sciences and Numerical Simulation, vol. 8, no. 2, Jan. 2007.

[16] S. S. Mohammed, Z. W. Salman, A. A. Mahdi, "An Effective Approach to SARS-CoV-2 Diagnosis by Developing the CNN Algorithm," *Iraqi Journal of Science*, vol. 65, no. 9, pp. 5270-5280, 2024.

[17] M. Tharmakulasingam, N. S. Chaudhry, M. Branavan, W. Balachandran, A. C. Poirier, M. A. Rohaim, and A. Fernando, "An artificial intelligence-assisted portable low-cost device for the rapid detection of sars-cov-2," *Electronics*, vol. 10, no. 17, pp. 2065, 2021.

[18] K. Shaheed, P. Szczuko, Q. Abbas, A. Hussain, and M. Albathan, "Computer-Aided Diagnosis of COVID-19 from Chest X-ray Images Using Hybrid-Features and Random Forest Classifier," *Healthcare*, vol. 11, no. 6, pp. 837, Mar. 2023.

[19] [2] A. A. Nafea, M. AL-Mahdawi, K. M. Alheeti, M. S. I. Alsumaidaie, and M. M. AL-Ani, "A Hybrid Method of 1D-CNN and Machine Learning Algorithms for Breast Cancer Detection," Baghdad Science Journal, vol. 21, no. 10, Article 19, 2024. doi: 10.21123/bsj.2024.9443.

[20] 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, Nov. 2018.

[21] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, Jan. 2017.

[22] O. A. Jasim and A. M. Al-Rozbayani, "Using Physics-Informed Neural Networks for Solving 2nd-Order Volterra Integro-Differential Equation by DeepXDE Library," Baghdad Science Journal, vol. 22, no. 12, Article 22, 2025. doi: 10.21123/2411-7986.5173

[23] L. Wang and Z. Yan, "Data-driven rogue waves and parameter discovery in the defocusing nonlinear Schrödinger equation with a potential using the PINN deep learning," *Physics Letters A*, vol. 404, p. 127408, May 2021.

[24] J. Pu, J. Li, and Y. Chen, "Solving localized wave solutions of the derivative nonlinear Schrödinger equation using an improved PINN method," *Nonlinear Dynamics*, vol. 105, no. 2, pp. 1723–1739, Jul. 2021.

[25] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A Deep Learning Library for Solving Differential Equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, Jan. 2021.

[26] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, Jan. 2020.

[27] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, "Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems," *Water Resources Research*, vol. 56, no. 5, Apr. 2020.

[28] Q. He, D. Barajas-Solano, G. Tartakovsky, and A. M. Tartakovsky, "Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport," *Advances in Water Resources*, vol. 141, pp. 103610, May 2020.

[29] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *Optics Express*, vol. 28, no. 8, pp. 11618, Mar. 2020.

[30] M. W. M. G. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Communications in Numerical Methods in Engineering*, vol. 10, no. 3, pp. 195–201, Mar. 1994.

[31] A. Paszke, S. Gross, F. Massa, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *In Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[32] M. Abadi, P. Barham, J. Chen, et al., "TensorFlow: a system for large-scale machine learning," *Operating Systems Design and Implementation*, pp. 265–283, Nov. 2016.

[33] F. Chen, D. Sondak, P. Protopapas, et al., "NeuroDiffEq: A Python package for solving

differential equations with neural networks," *Journal of Open-Source Software*, vol. 5, no. 46, pp. 1931, 2020.

[34] W. Peng, J. Zhang, W. Zhou, X, Zhao, W. Yao X. Chen, "IDRLnet: A Physics-Informed Neural Network Library," *The7th International Workshop on Advanced Computational Intelligence and Intelligent Informatics (IWACIII2021) Beijing, China,* Oct.31-Nov.3, 2021.

[35] L. Yuan, Y. Q. Ni, X.Y. Deng, and S. Hao, "A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations," *Journal of Computational Physics*, vol. 462, pp. 111260, Aug. 2022.