Mathematics and Computers in Simulation 218 (2024) 311–333

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

A new modern scheme for solving fractal–fractional differential equations based on deep feedforward neural network with multiple hidden layer

Mohd Rashid Admon [a], Norazak Senu [a,*], Ali Ahmadian [b,c,d,*], Zanariah Abdul Majid [a], Soheil Salahshour [c,e,f]

[a] Institute for Mathematical Research, Universiti Putra Malaysia, Selangor, Malaysia
[b] Decisions Lab, Mediterranea University of Reggio Calabria, Reggio Calabria, Italy
[c] Faculty of Engineering and Natural Sciences, Istanbul Okan University, Istanbul, Turkey
[d] Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon
[e] Faculty of Engineering and Natural Sciences, Bahcesehir University, Istanbul, Turkey
[f] Faculty of Science and Letters, Piri Reis University, Tuzla, Istanbul, Turkey

ARTICLE INFO

Keywords:
Fractal–fractional differential equation
Artificial Neural Network
Deep feedforward neural network
Vectorized algorithm
Adam optimization

ABSTRACT

The recent development of knowledge in fractional calculus introduced an advanced superior operator known as fractal–fractional derivative (FFD). This operator combines memory effect and self-similar property that give better accurate representation of real world problems through fractal–fractional differential equations (FFDEs). However, the existence of fresh and modern numerical technique on solving FFDEs is still scarce. Originally invented for machine learning technique, artificial neural network (ANN) is cutting-edge scheme that have shown promising result in solving the fractional differential equations (FDEs). Thus, this research aims to extend the application of ANN to solve FFDE with power law kernel in Caputo sense (FFDEPC) by develop a vectorized algorithm based on deep feedforward neural network that consists of multiple hidden layer (DFNN-2H) with Adam optimization. During the initial stage of the method development, the basic framework on solving FFDEs is designed. To minimize the burden of computational time, the vectorized algorithm is constructed at the next stage for method to be performed efficiently. Several example have been tested to demonstrate the applicability and efficiency of the method. Comparison on exact solutions and some previous published method indicate that the proposed scheme have give good accuracy and low computational time.

1. Introduction

Over a last few decades, the emerging concept of fractional derivatives and integrals in a new topic known as Fractional Calculus (FC) has give remarkable history in various field of studies. Some of real world applications have appeared during its early peak of development such as modelling in viscoelastic materials, feedback amplifier design, electrical circuits involving resistors and capacitors, electroanalytical chemistry, electrical conductance of membranes of cells in biological organisms, diffusion that can be found in [1,2]. As time goes through year by year, abundance of new applications of FC have been discovered in science and engineering which highlighted through comprehensive review paper prepared by [3]. The ability to described the dependency of

* Corresponding authors.
E-mail addresses: norazaksenu@upm.edu.my (N. Senu), ahmadian.hosseini@unirc.it (A. Ahmadian).

https://doi.org/10.1016/j.matcom.2023.11.002
Received 22 June 2023; Received in revised form 18 October 2023; Accepted 6 November 2023
Available online 24 November 2023

any present event or process is based on the entire history in the past starting from the initial time or can be term as memory effect, is one of the main reason that this concept is spanning over in modelling various higher order dynamics and complex nonlinear event compared to standard Leibniz's derivative [4]. Ones can benefit this feature by manipulating the order of fractional operator from integer to non-integer which clearly provide a better degree of freedom when analysing certain mathematical models of interest. There are three most influential fractional operators in FC, often categorized through its kernel representation such as power law kernel with singular type that associate with Riemann–Liouville and Caputo fractional operator, exponential decay law kernel with non-singular type that associate with Caputo–Fabrizio fractional operator and generalized Mittag-Leffler law kernel with non-singular type that associate with Atangana–Baleanu fractional operator.

The implementation of these fractional operators involving fractional differentiation in fractional differential equations (FDEs) have been proven on giving accurate representation in mathematical modelling of real world problem [5–9]. In spite of this innovative idea, there exists another concept that extend the notion of Leibniz regarding on standard local derivative that scaled with integer dimension to a new metric time–space that scaled with non-integer dimension, known as fractal derivative [10,11]. This differential operator have been used in anomalous diffusion which frequently occurs mostly in soft matters or porous (fractal media) such as glass, oil, various of porous media including aquifer and turbulence, in which the macromolecules are grouped together in non-lattice and porous mesostructures. The necessity of fractal derivative is due to the existence of interactions and motions between these macromolecules could give a huge impact in a variety of physical behaviours such as inertial effects and external forces. As the current physical law such as Fick's Law, Darcy's Law and Fourier's Law are designed in classical derivative through rate of change in concept of classical derivative, the process of anomalous diffusion are not well-measured or described through this concept. Thanks to the author in [10], an excellent theory of fractal derivative found out to be effective on capturing those complex process. Other applications of fractal derivative have been studied in several works [12,13].

Differ from fractional derivative, fractal derivative is a local operator while the former is a global operator. Both operator have plays significant role in applications such as modelling of diffusion using differential equation where fractal differential equation manifest the stretched Gaussian process while fractional differential equation portray Lèvy process [11]. Realizing the capability of these two different concept in application, a new concept of differentiation was further introduced just recently by Abdon Atangana known as fractal–fractional derivative (FFD) [14]. This derivative is combination of two earlier concept which are fractal derivative and fractional derivative that consists of two types of order which are fractional-order and fractal-order. The author proposed the FFD based on two version which are Riemann–Liouville sense and Caputo sense with power law kernel (FFDPRL/FFDPC), exponential decay law kernel (FFDERL/FFDEC) and Mittag-Leffler law kernel (FFDMRL/FFDMC). This novel derivative seems to be a simple modification in theoretically, but, it give a huge success in delivering accurate description process in modelling of disease [15–17], chaotic system [18,19] and economy [20] through fractal–fractional differential equations (FFDEs).

Much efforts have been devoted by various researchers to find the solution of FFDEs in order to understand and predict the nature's complexities. In fact, most of the methods derive for solving an ordinary FDEs are extends to solve FFDEs to suit with the theoretical features lies behind the FFD operator. In spite of introduced the new concept called FFD, Atangana [14] provided the numerical approximation for it by first changing the existence of fractal derivative in the operator into an equivalent form that consists of classical derivative. The classical derivative then can be approximated using any numerical differentiation such as first order approximation. The authors also introduced fractal–fractional integral (FFI) based on power law kernel, exponential decay law kernel and Mittag-Leffler law kernel. Two different methods was presented to approximate FFI which are 3/8 Simpson rule and Boole's rule. The method presented have been applied to simple FFDEs and applications at the Darcy scale describing flow in a dual medium. Atangana and Qureshi [21] presents three numerical schemes for FFDEs with different kernels in Riemann–Liouville sense to illustrate various type of chaotic dynamical systems such as Chen attractor, Lu Chen attractor, Modified Lu Chen attractor and Modified Chua chaotic attractor. They first convert the system into Volterra integral equation and replace the RL derivative into Caputo derivative to make use the integer initial condition. The numerical scheme then successfully develop by make use the two-point Lagrange piecewise interpolation technique for all FFDEs with different kernels.

The previous techniques has been expanded for various work afterwards in investigating the effect of varying the order of fractional derivative and fractal derivative in applications such as competition system involving banks in Indonesia [22], modelling of memristor [23], modelling the COVID-19 disease in Pakistan [24], modelling of vaccine model of COVID-19 [16], and modelling of Brusselator [25]. Abro and Atangana [18] extend the classical Adam–Bashforth–Moulton method to investigate the hyperchaos, abrupt chaos and coexisting attractors in meminductor and memcapacitor through FFDEs with Caputo–Fabrizio operator. Through different type of polynomial, Mekkaoui et al. extend the predictor–corrector method to solve FFDEs using the Newton polynomial [26]. However, the work proposed by the author are quite cumbersome compared with the usual approach that using Lagrange polynomial. Rayal and Verma [27] extend the use of wavelet methods by make use fractional-order Legendre wavelets and collocation methods for solving pantograph FFDEs. A widely used operational matrix method in solving FDEs are also have been found to be extend in some works that aims to solve FFDEs. An operational matrix of FFD using shifted Chebyshev polynomials on solving nonlinear Ginzburg–Landau equation in Riemann–Liouville sense has been developed by Heydari et al. with the aid of collocation scheme [28]. Recently, Shloof et al. [29] develop a novel operational matrix of solving FFDEs with generalized Caputo fractional derivative using shifted Legendre polynomials. The main contribution of this work lies on the inclusion of fractal derivative in generalized Caputo derivative, resulting three different orders on the operator.

In delivering the methods for solving differential equations with the involvement of new fractal–fractional operator, the same style of extending the classical numerical scheme for integer-order differential equation to FDEs as have been reviewed recently is a common approach for this situation. Looking into different angles, this research intends to widen the scope of producing numerical techniques from the classical ones to the modern approach. In 1999, Lagaris et al. [30] has introduced a new numerical techniques

known as artificial neural network (ANN) to solve various type of integer-order differential equations. This technique is originally developed in machine learning and has ability to approximate nonlinear behaviour of functions [31]. Due to this characteristic, the previous author design ANN with one hidden layer to approximate the solution of integer-order differential equations using optimization techniques to train the network, whose weights and biases are changed to minimize the appropriate error function. The technique gains numerous enticing aspects through the use of this novel approach such as the governing solution that can be referred as approximate solution (AS) is differentiable and closed analytic form that ease us to proceed for any further subsequent calculations, the AS have good generalization properties and the method is general which can be applied to single equation or system of equation of differential equation [30,32,33]. Another huge advantage of this technique is lies on its ability to skip the repeating the process on finding AS within the discretized point. In standard numerical technique, we only get the AS at discretized point based on the step size that we fix before run the computation. If we need to find the AS at other different points, we need to fix the step size and run the process again. It is clearly impractical and not efficient, but not for ANN which is the other way around.

The journey of expanding ANN to be a part of numerical technique in finding solution of differential equations has shown major success through various improvement in terms of network architecture and optimization procedure. Raja et al. [34] design unsupervised fractional neural network with one hidden layer using second-order optimization technique known as interior point algorithm for solving Bagley–Torvik equation in Caputo sense. The proposed method give AS to be accurate not only compared for ANN with stochastic optimizer but also analytical technique such as variational iteration method. Pakdaman et al. [33] extend the framework from Lagaris to solve linear and nonlinear FDEs in Caputo sense using ANN with one hidden layer by make use BFGS optimization method to train the network. Panghal and Kumar [35] present deep feedforward neural network (DFNN) to solve delay and system of delay integer-order differential equation. In their work, they try to compare the result of absolute error of AS with three different hidden layer in the network. The results indicate that DFNN with two hidden layer give AS the best accuracy to the solution of the differential equation compared to network with one and three hidden layer. T.T. Dufera [36] developed vectorized algorithm to solve system of integer-order differential equation. The framework is similar to Lagaris, but they concern on elevating the performance of the DFNN to the network by compute the forward propagation in one time and implement automatic differentiation for fast computation of first-order derivative. Shloof et al. [37] then attempts to use ANN with one hidden layer to solve FDEs with newly fractal–fractional operator in generalized Caputo sense. Babu et al. [38] provide a new approach based on quaternion-valued neural networks on solving FFDEs with time-delays. Admon et al. [39] extends the work from T.T Dufera by implement vectorized algorithm to solve linear and nonlinear FDEs in Caputo sense. The authors investigate the roles of hidden layer, number of neurons and learning rate in first-order optimization technique towards the accuracy of the FDEs. They found out deep ANN with two hidden layers with reasonable choice of number of neurons and learning rate can increase the accuracy of the FDEs.

The motivation of our works lies on the roles of hidden layer to the accuracy of FDEs. In aspect of machine learning application, the addition of hidden layer have proved to give dramatic improvement in computer vision, image processing, pattern recognition and cybersecurity [40–42]. In terms of mathematical research, it also has been shown that through the work done in [35,36,39], the addition of hidden layer are capable to improve the accuracy of FDEs in Caputo sense. Thus, this work intends to implement DFNN with two hidden layers (DFNN-2HL) to solve FFDE with power law kernel in Caputo sense (FFDEPC) using Adam optimization technique. To the best of our knowledge, there is no work related to DFNN to solve this kind of FFD operator. We focuses on improving the performance of the computation by implementing vectorization algorithm to reduce the burden of computation in looping for evaluating forward propagation and evaluation of nonlocal FFD operator for the first time. The Adam optimization technique is chosen compared from others optimization technique because it has simple implementation and low memory consumption. It also make use the adaptive learning mechanism compared from other class of first-order optimization technique that using adaptive learning rate parameter which can be adjusted in response during the training process of the network parameters. Another new innovation that considered in this work is the adjustment of the network parameters during the training process are selective. This approach is totally new not only in the research of ANN in mathematics, but also in traditional implementation of ANN. The motivation of this approach is based on extreme learning machine algorithm that are adjusted at only weights that coming from hidden layer to the output layer [43].

The article organized as follows: in Section 2, we review some important definitions on fractal–fractional derivative with the power law kernel in Caputo sense and briefly discussed the architecture of ANN used in this research. In Section 3, we present how to solve FFDEPC using the new proposed scheme in two stage which are basic framework and the vectorized algorithm. To validate our newly proposed scheme, we use several linear and nonlinear examples of FFDEPC by observing its absolute error and comparison with other method available in literature in Section 4. Finally in Section 5, the main conclusions of the study are highlighted.

2. Preliminaries and notation

In this section, a definition of fractal–fractional operator and fundamental concept of ANN that used in this work are presented.

2.1. Fractal–fractional definition

Definition 2.1 (*[14] Fractal-Fractional Derivative With The Power Law Kernel In Caputo Sense (FFDPC)*)**.**

Let $f(t)$ is a differentiable function with order η on interval (t_0, t_f). Then, fractal–fractional derivative of order ν with power law kernel in Caputo sense is defined as

$$\left({}^{FFDPC}_{t_0}D_t^{\nu,\eta}\right) f(t) = \frac{1}{\Gamma(m-\nu)} \int_{t_0}^{t} (t-s)^{m-\nu-1} \frac{df}{ds^{\eta}} ds, \tag{1}$$

where $\frac{df}{ds^{\eta}} = \lim_{t\to s} \frac{f(t)-f(s)}{t^{\eta}-s^{\eta}}$, $m-1 < \nu \leq m$, and $0 < m-1 < \eta \leq m$, $m \in \mathbb{N}$.

If f is differentiable over (t_0, t_f), then

$$\begin{aligned} \frac{df}{ds^{\eta}} &= \lim_{t\to s} \frac{f(t)-f(s)}{t^{\eta}-s^{\eta}} \\ &= \frac{f'(s)}{\eta s^{\eta-1}} \\ &= f'(s)\frac{s^{1-\eta}}{\eta}. \end{aligned}$$

2.2. Fundamental concept of ANN

The history emergence of ANN can be dated back in 1943, where the invention of simplified neurons by McCulloch and Pitts that mimics the biological neurons of the human brain [44]. In their article, they define the idea of a neuron as a single cell residing in a network of cells that receive and process inputs to generate outputs. The simplest definition of ANN can defined as computing system that consists of a number of simple highly interconnected processing element called artificial neuron that acquire knowledge through its environment through learning process and store knowledge through its connection [45,46]. From this definition, it can be seen that ANN revolve two major basic things which are processing elements or neurons and learning process. Processing elements in ANN refer to basic elements in ANN that inspired from the event in biological neuron which are weights that mimic incoming stimulus or synapses, summing junction that mimic the dendrites which accumulated the incoming weighted stimulus, activation function that mimic cell body where the conversion of new stimulus take place and bias that mimic the threshold value that increase or decrease the stimulus. While the learning process is the process of modifying the weights and biases that mimic the process of capturing signals occurred in neuron. The aim of this process is minimize the error of the output which usually involve optimization techniques such as gradient-based method (first-order and second-order) and non-gradient based method (heuristic).

ANN in fact is one of the well-known of research fields of artificial intelligence (AI) and models in machine learning. It is an effective and reliable data modelling tools in several complicated problems such as system security [47], pattern recognition [48,49], and image encryption [50]. ANN can be designed to form different architecture that can be simple or complex ones. Early neural network pioneers used very basic architectures known as single layer neural networks, which just featured input and output layers. A single layer neural network becomes a multilayer neural network when hidden layers are added. Thus, the input layer, hidden layer(s), and output layer make up the multilayer neural network. A shallow neural network, sometimes known as a vanilla neural network, is a neural network with only one hidden layer. While multilayer ANN with two or more hidden layers known as deep ANN. This type of ANN contributed a lot of progress in modern neural to tackle various real world problems due to ability on capture more variances in data thus provide more accurate desired output [51].

In this research, we focuses on deep feedforward ANN with two hidden layers (DFNN-2HL) to approximate the solution of FFDEPC. The term feedforward here means that the input that goes in into the network are strictly forward way through input layer, first hidden layer, second hidden layer and output layer. In other words, the output of each neuron in the network does not go backwards or affect the same layer. In order to solve FFDEs, a brief framework of neural network architecture should be proposed. Consider a DFNN-2HL model as shown in Fig. 1. It consists of one single neuron denoted by $p_1^{[0]}$ in the input layer, k number of neurons in first and second hidden layer denoted by $p_j^{[1]}$ and $p_i^{[2]}$ respectively where $j = 1, 2, 3, \ldots, k$ and $i = 1, 2, 3, \ldots, k$ and one single neuron in the output layer denoted by $p_1^{[3]}$. We assume the number of neurons for the first hidden layer and second hidden layer are the same. The weight from the input layer to first hidden layer is denoted by $w_{j1}^{[1]}$, weight from the first hidden layer to second hidden layer is denoted by $w_{ij}^{[2]}$ and weight from the second hidden layer to third hidden layer is denoted by $w_{1i}^{[3]}$. The subscript in the weights such as w_{ij} indicate weights that move from jth neuron in the previous layer to ith neuron in the next layer. While for the biases, $b^{[1]}$ and $b^{[2]}$ represent biases in the first hidden layer and second hidden layer respectively. The superscript in all the network parameters and the neurons refer to the layer. Finally, the output of DFNN-2HL was indicated by $\hat{y}(t, \mathbf{\Phi})$ where $\mathbf{\Phi}$ represent all the weights and bias of the DFNN-2HL.

The first neuron $p_1^{[0]}$ in the input layer will collects the input signal, x defined by the user. Then the input signal will multiplied by the weight $w_{j1}^{[1]}$ to give weighted input. The product then added by bias $b^{[1]}$ and being applied to an activation function ϕ to generate new output for each neurons in the first hidden layer. The activation function used in this work is sigmoid function as it preserve balance between linear and nonlinear behaviour [32]. The same process occur for each neurons in the second hidden layer. Next, the output of neuron $p_1^{[3]}$ in the output layer is then computed by multiplying the output of neuron j in the second hidden layer with the weight $w_{1i}^{[3]}$ and add the bias $b^{[2]}$. The net output is then obtained by summed up all the product and being expressed as $\hat{y}(t, \mathbf{\Phi})$. All this process are known as forward propagation. The following input–output relation related this phase are summarized as follows:

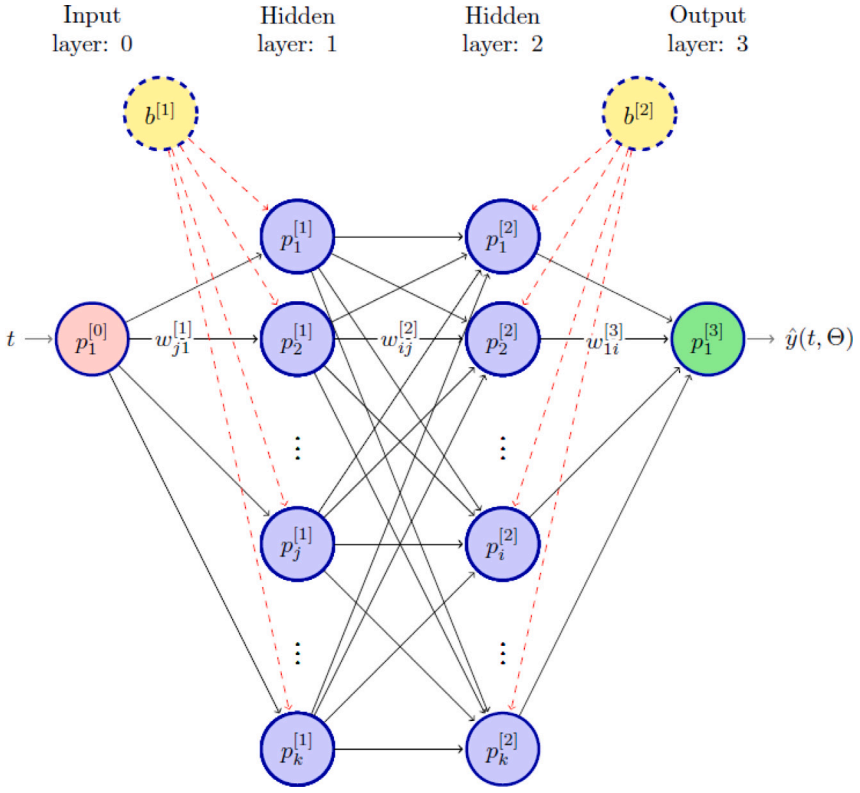


Fig. 1. Schematic diagram of DFNN-2HL.

- In the input layer:

$$p_1^{[0]} = t. \tag{2}$$

- In the first hidden layer:

$$s_j^{[1]} = w_{j1}^{[1]} p_1^{[0]} + b_j^{[1]}, \tag{3}$$

$$p_j^{[1]} = \phi(s_j^{[1]}). \tag{4}$$

- In the second hidden layer:

$$s_i^{[2]} = \sum_{j=1}^{k} w_{ij}^{[2]} p_j^{[1]} + b_i^{[2]}, \tag{5}$$

$$p_i^{[2]} = \phi(s_i^{[2]}). \tag{6}$$

- In the output layer:

$$s_1^{[3]} = \sum_{j=1}^{k} w_{1j}^{[3]} p_i^{[2]}, \tag{7}$$

$$p_1^{[3]} = s_1^{[3]}. \tag{8}$$

3. Methodology description

In this section, the proposed model of ANN which is DFNN-2H will be implemented for solving initial value problem of fractal–fractional differential equation in Caputo sense. Firstly, we highlight the basic framework needed on solving FFDEPC using DFNN-2H. Then, a vectorized algorithm based on this framework is presented.

Table 1
Numerical comparison of AS with ABM of Example 1 at different values of ν and η.

t	$\nu = 0.7, \eta = 0.75$		$\nu = 0.8, \eta = 0.85$		$\nu = 0.9, \eta = 0.95$	
	ABM	AS	ABM	AS	ABM	AS
0	0	0	0	0	0	0
0.1	0.0013601220	0.0024566098	0.0008456593	0.0010454915	0.0005141706	0.0007728137
0.2	0.0074327574	0.0080996959	0.0053086525	0.0054141843	0.0037077799	0.0038492990
0.3	0.0200716240	0.0203021505	0.0155466607	0.0155119552	0.0117757137	0.0116610653
0.4	0.0406149151	0.0413206732	0.0333218123	0.0334144089	0.0267342041	0.0266802666
0.5	0.0701643537	0.0708973450	0.0601924366	0.0602152110	0.0504966929	0.0501631866
0.6	0.1096759405	0.1109246502	0.0975827809	0.0974397255	0.0849044883	0.0842123537
0.7	0.1600042339	0.1618226916	0.1468191697	0.1467810908	0.1317436523	0.1307966986
0.8	0.2219279756	0.2244398872	0.2091518932	0.2088281150	0.1927556877	0.1927563423
0.9	0.2961665058	0.2995427346	0.2857697419	0.2855093107	0.2696448946	0.2675434173
1.0	0.3833910262	0.3883526527	0.3778102855	0.3774693993	0.3640837304	0.3613516935
CPU Time (s)		101.56		103.25		105.34
Cost function		4.83×10^{-4}		3.08×10^{-5}		1.36×10^{-4}

3.1. Method on solving FFDEPC

Consider the following FFDEPC

$$\left({}_0^{FFDPC}D_t^{\nu,\eta} y\right)(t) = F\Big(t, y(t)\Big), \quad t \in [0, t_f], \tag{9}$$

with initial condition $y(0) = a$ and $0 < \nu \leq 1$, $0 < \eta \leq 1$. Let the AS of the problem (9) is denoted by $y_T(t, \mathbf{\Phi})$. The designation of AS contain two important term, the first term must satisfy the initial condition problem (9) and the other one contain the output of the DFNN-2HL denoted as $\mathbf{\Phi}$, that correspond to the all the weights and biases of the network. Thus, the designation of AS can be formulated as follows:

$$y_T(t, \mathbf{\Phi}) = a + \lceil \eta \rceil t^{\lceil \nu \rceil + \lceil \eta \rceil - 1} \hat{y}(t, \mathbf{\Phi}), \tag{10}$$

where $\lceil \nu \rceil$ and $\lceil \eta \rceil$ are the upper integer value of fractional-order and fractal-order respectively and $\hat{y}(t, \mathbf{\Phi})$ is the output of DFNN-2H. It is clear that the Eq. (10) satisfies the behaviour of the FFDEPC's initial condition. Since the nearest integer value for ν and η are both equal to one, thus the AS will reduce to

$$y_T(t, \mathbf{\Phi}) = a + t\hat{y}(t, \mathbf{\Phi}), \tag{11}$$

which independent of fractal-order η. Substituting the Eq. (11) in the Eq. (9) will yield

$$\left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t, \mathbf{\Phi}) = F\Big(t, y_T(t, \mathbf{\Phi})\Big). \tag{12}$$

Then, the resulting Eq. (12) will lead to the following optimization problem:

$$\min_{\mathbf{\Phi}} \{q(\mathbf{\Phi})\} = \sum_{d=1}^{N_{tp}} \left[\left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t, \mathbf{\Phi}) - F\Big(t, y_T(t, \mathbf{\Phi})\Big) \right]^2. \tag{13}$$

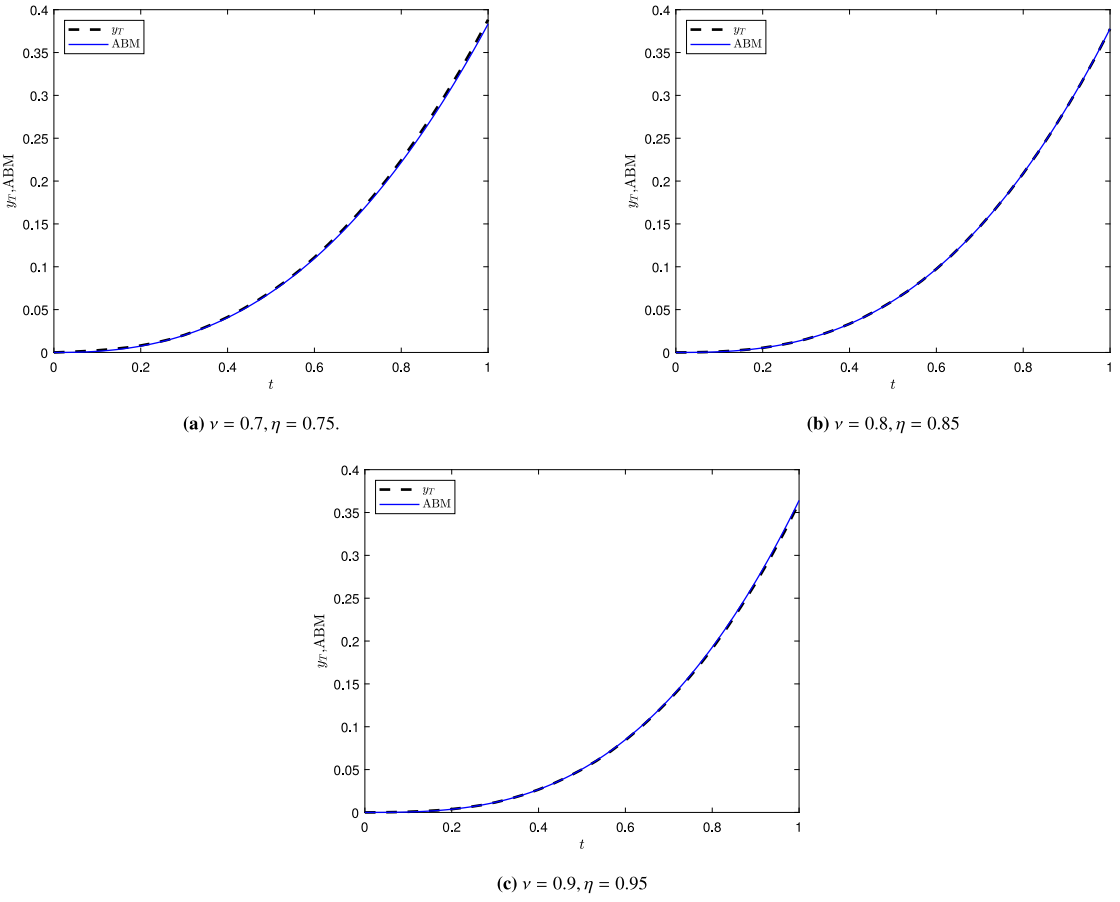
where N_{tp} is the number of equidistant training point that discretize from domain t and $q(\mathbf{\Phi})$ is the error or cost function that need to be minimized through Adam optimization method to obtain the optimal network parameters, $\mathbf{\Phi}^*$. In this work, the network parameters need to be modified are selective, which are only the network parameters at the outermost hidden layer. The motivation of this idea is based on the extreme learning machine technique that freeze the adjustment of the network parameters between outermost hidden layers [9,35,43,52]. Thus, $\mathbf{\Phi}^*$ contain two types of network parameters which are updated network parameters, $\mathbf{\Phi}_1$ and non-updated network parameters, $\mathbf{\Phi}_2$.

To evaluate the cost function in (13), we need to approximate the FFD operator. By the definition FFD with power law kernel in Caputo sense in (1), we obtain

$$\begin{aligned} \left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t, \mathbf{\Phi}) &= \frac{1}{\Gamma(1-\nu)} \int_0^t (t-s)^{-\nu} \frac{dy_T}{ds^\eta} ds, \\ &= \frac{1}{\Gamma(1-\nu)} \int_0^t (t-s)^{-\nu} y_T'(s, \mathbf{\Phi}) \frac{s^{1-\eta}}{\eta} ds. \end{aligned} \tag{14}$$

We discretize the interval $[0, t_f]$ into n subinterval by $h = t_f/n$ to produce $n+1$ training points. At $t^{(n+1)}$,

$$\begin{aligned} \left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(n+1)}, \mathbf{\Phi}) &= \frac{1}{\Gamma(1-\nu)} \int_0^{t^{(n+1)}} (t^{(n+1)} - s)^{-\nu} y_T'(s, \mathbf{\Phi}) \frac{s^{1-\eta}}{\eta} ds \\ &= \frac{1}{\eta\Gamma(1-\nu)} \sum_{r=0}^{n} \int_{t^{(r)}}^{t^{(r+1)}} \frac{y_T(t^{(r+1)}, \mathbf{\Phi}) - y_T(t^{(r)}, \mathbf{\Phi})}{h} (t^{(r)})^{1-\eta} (t^{(n+1)} - s)^{-\nu} ds, \end{aligned}$$



(a) $\nu = 0.7, \eta = 0.75$.

(b) $\nu = 0.8, \eta = 0.85$

(c) $\nu = 0.9, \eta = 0.95$

Fig. 2. Graphical comparison of AS with ABM of Example 1 at different values of ν and η.

$$= \frac{1}{\eta\Gamma(1-\nu)} \sum_{r=0}^{n} \frac{y_T(t^{(r+1)}, \Phi) - y_T(t^{(r)}, \Phi)}{h} (t^{(r)})^{1-\eta} \int_{t^{(r)}}^{t^{(r+1)}} (t^{(n+1)} - s)^{-\nu} ds,$$

$$= \frac{h^{-\nu}}{\eta\Gamma(2-\nu)} \sum_{r=0}^{n} (y_T(t^{(r+1)}, \Phi) - y_T(t^{(r)}, \Phi))(t^{(r)})^{1-\eta}[(n-r+1)^{1-\nu} - (n-r)^{1-\nu}].$$

For all discretized training points in the interval $[0, t_f]$, we defined $t^{(1)} = 0 < t^{(2)} = h < t^{(3)} = 2h < \cdots < t^{(N_{tp}-1)} = nh < t^{(N_{tp})} = (n+1)h$. Then, we further discretize the interval $[0, t^{(d)}]$, where $d = 1, 2, \ldots, N_{tp}$ into θ subinterval by $h_d = t^{(d)}/\theta$ to produce $\theta + 1$ training points such that $t^{(1)} = 0 < t_d^{(2)} = h_d < t_d^{(3)} = 2h_d < \cdots < t_d^{(d-1)} = \theta h_d < t^{(d)} = (\theta+1)h_d$. Then, the FFDPC at each of training point $t^{(d)}$ can be approximated by

$$\left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(d)}, \Phi) = \frac{h_d^{-\nu}}{\eta\Gamma(2-\nu)} \left\{ \begin{array}{l} (y_T(t_d^{(2)}, \Phi) - y_T(t^{(1)}, \Phi))[(\theta+1)^{1-\nu} - \theta^{1-\nu}] \\ + \sum_{r=2}^{d-1} y_T(t_d^{(r+1)}, \Phi) - (y_T(t_d^{(r)}, \Phi))(t_d^{(r)})^{1-\eta}[(\theta-r+2)^{1-\nu} - (\theta-r+1)^{1-\nu}] \\ + y_T(t^{(d)}, \Phi) - (y_T(t_d^{(d-1)}, \Phi))(t_d^{(d-1)})^{1-\eta} \end{array} \right\} \tag{15}$$

With the help of approximation developed in (15), we can finally compute the following optimization problem:

$$\min_{\Phi}\{q(\Phi)\} = \sum_{d=1}^{N_{tp}} \left[\left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(d)}, \Phi) - F\left(t^{(d)}, y_T(t^{(d)}, \Phi)\right) \right]^2. \tag{16}$$

To solve the optimization problem above, we can use any optimization method in the literature such as gradient-based method and heuristic method [53]. Here, we intend to use first-order optimization technique known as Adaptive Moment estimation method (Adam) to due to its simplicity and low memory usage. Let p representing either selected individual weight or biases, we can updated the network parameters as follows

$$v^l = \beta_1 v^{l-1} + (1-\beta_1)\frac{\partial q}{\partial p^l}, \tag{17}$$

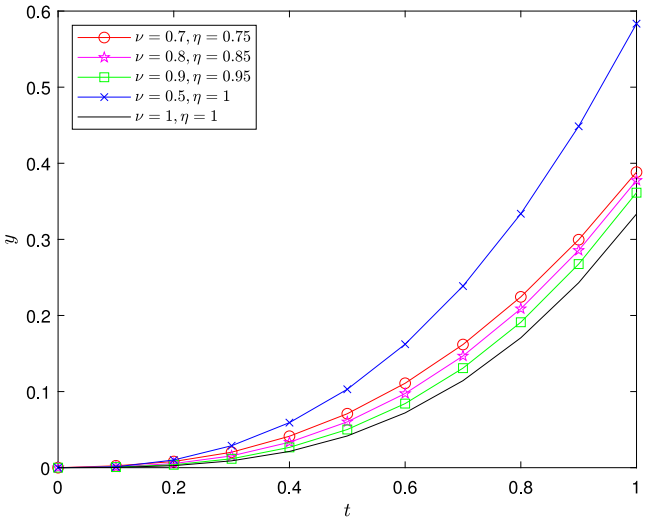


Fig. 3. Graphical comparison of AS of Example 1 at different values of ν and η with exact solution.

$$s^l = \beta_2 s^{l-1} + (1-\beta_2)\Big(\frac{\partial q}{\partial p^l}\Big)^2, \tag{18}$$

$$\hat{v}^l = \frac{v^l}{1-\beta_1^l}, \tag{19}$$

$$\hat{s}^l = \frac{s^l}{1-\beta_2^l}, \tag{20}$$

$$p^{l+1} = p^l - \frac{\nu}{\sqrt{\hat{s}^l + \epsilon}}\hat{v}^l. \tag{21}$$

where v and s are the first moment and second moment of the gradients respectively. These value of parameters are initialized at 0. While β_1 and β_2 are decay rates are in the range of value from 0 to 1, often selected close to 1 as suggested in [39,53,54]. In this research, we will set β_1 and β_2 with 0.95 and 0.995 respectively [39]. The learning rate, ν is the most sensitive parameter which as have been extensively discussed in [39]. Based on their findings, we decided to take the value of this parameter either 0.1 and 0.001 depending on the FFDEPC later. Lastly, the parameter ϵ refer to the very small number to avoid division by zero, which often set as 1×10^{-8} [54]. From the formula above, (17) refer to biased decaying momentum, (18) refer to biased decaying squared gradient, (19) refer to corrected biased decaying momentum, (20) refer to corrected biased decaying squared gradients and (21) is the updated formula for the network parameters at iteration $l+1$.

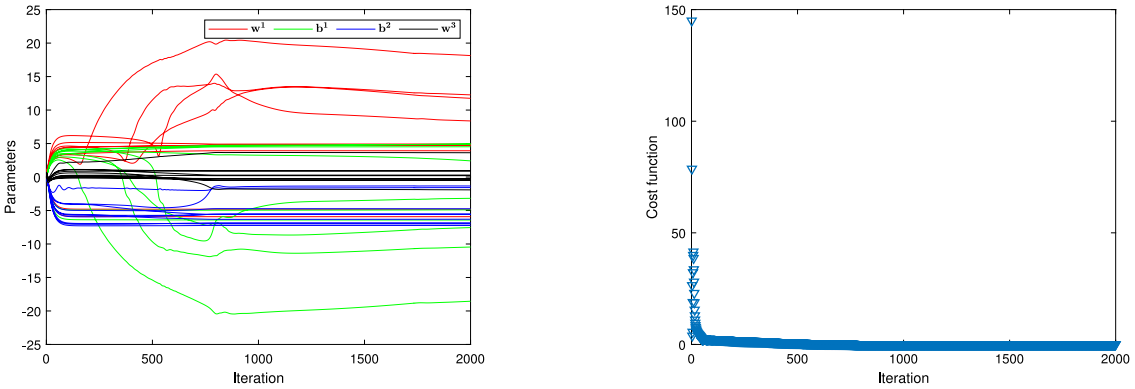
Notice that from the Adam formula in (17)–(18), it is require the evaluation of the first-order derivative of error function with respect to the network parameters. This can be obtained as:

$$\frac{\partial q}{\partial p} = \frac{\partial}{\partial p}\left(\sum_{d=1}^{N_{tp}}\left[\left({}_0^{FFDPC}D_t^{\nu,\eta}y_T\right)(t^{(d)},\mathbf{\Phi}) - F\left(t^{(d)}, y_T(t^{(d)},\mathbf{\Phi})\right)\right]^2\right). \tag{22}$$

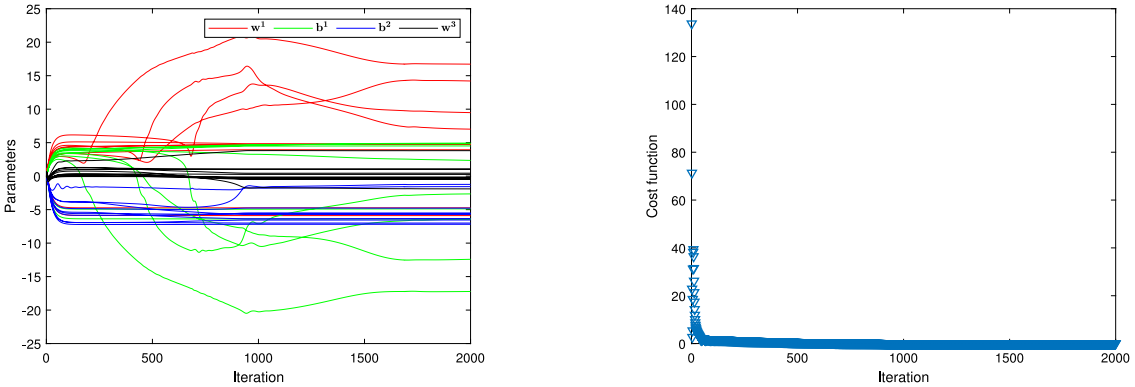
This gradient is actually can be computed through standard method in training of neural network called backpropagation algorithm. It compute the gradient of an cost function as a series of local or intermediate gradients. A common approach to code this algorithm in mathematical software is by using symbolic differentiation, however this technique is computationally expensive due to the results of complex and "swollen" expression [55]. Thus, one way to handle this situation is to perform automatic differentiation (AD), that compute derivatives through accumulation of values during code execution to produce numeric values of the derivatives rather than derivative expression [56]. AD is consider new in machine learning, thus a solver package related on AD may not be available on a certain mathematical software. Thus, a proper sketch of computational graph of the error function is needed before implement the AD which ones can refer in many references such as in [53,56,57], so after that we are ready to code the algorithm efficiently. The training of the algorithm could be ended at a certain stopping criteria such as maximum iteration, to obtain the newly adjusted network parameters, $\mathbf{\Phi}^*$. Finally we can obtain the final AS for a given problem (9).

3.2. Vectorized algorithm

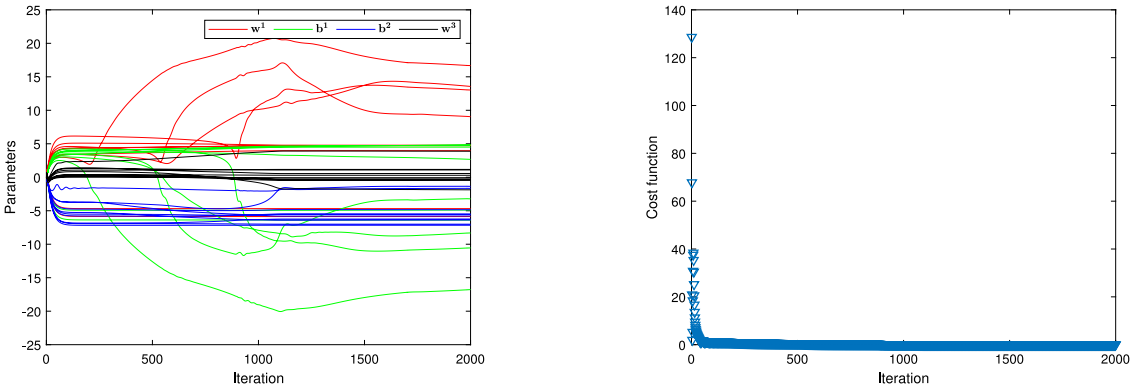
Since we use deep network to solve FFDEPC, the issue of high cost in computational time become a major of interest in this work. This is due to the implementation of looping statement such as in forward propagation that being coded in mathematical software are very slow, especially if the hidden layer of the network are added. This will give rise an alternative approach known as



(a) $\nu = 0.9$ and $\eta = 0.95$



(b) $\nu = 0.8$ and $\eta = 0.85$



(c) $\nu = 0.7$ and $\eta = 0.75$

Fig. 4. Convergence of the parameters and cost function with respective ν and η for Example 1.

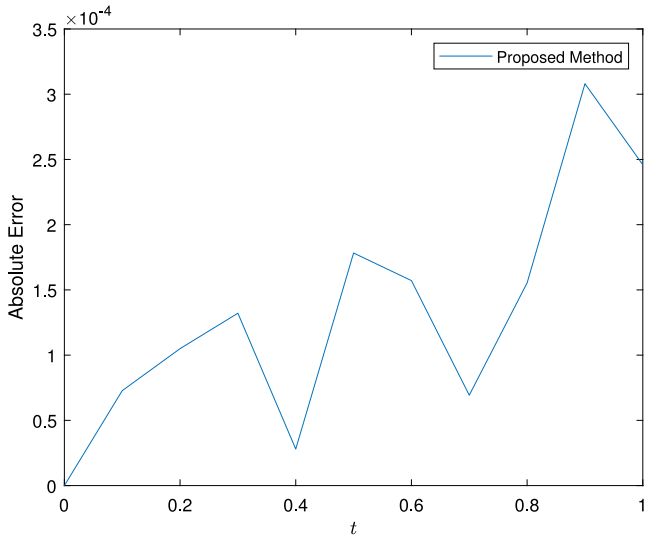
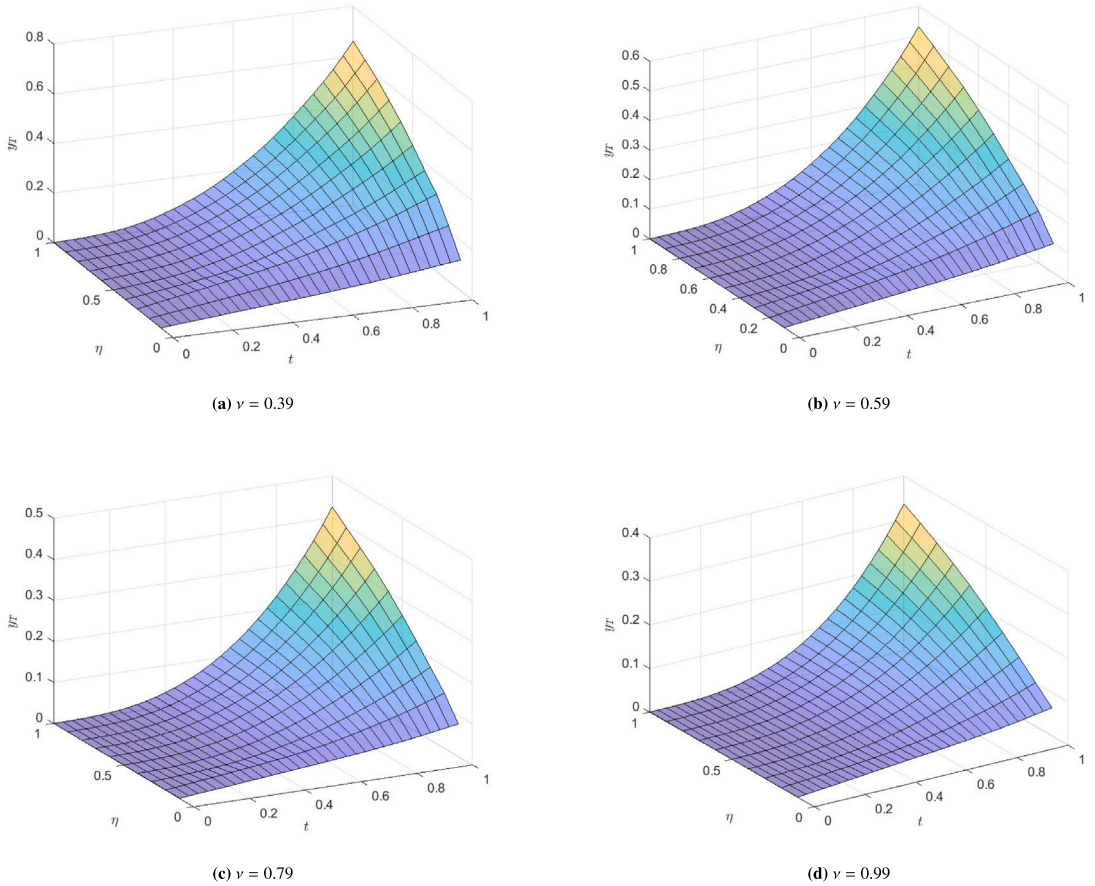


Fig. 5. Absolute error of proposed method when $\nu = 1$ and $\eta = 1$ for Example 1.

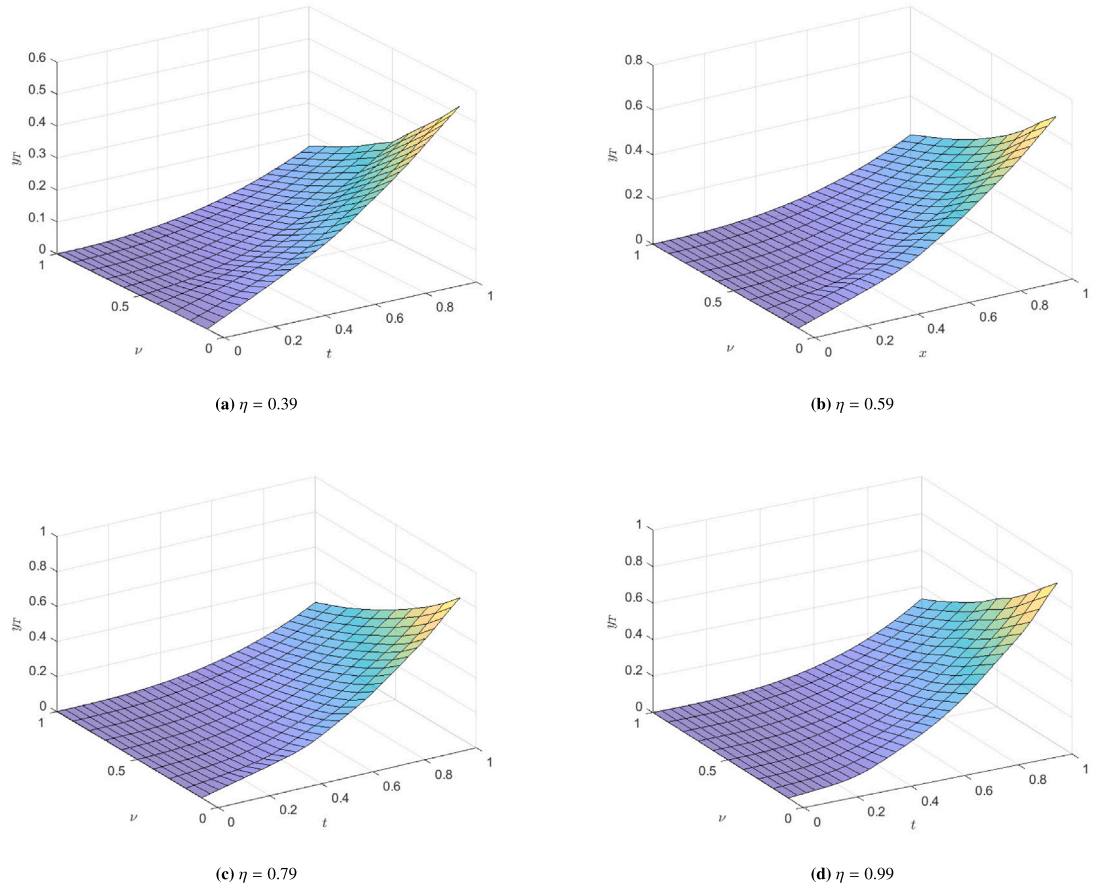


(a) $\nu = 0.39$ **(b)** $\nu = 0.59$

(c) $\nu = 0.79$ **(d)** $\nu = 0.99$

Fig. 6. AS of various η and t with fixed ν for Example 1.

vectorization, that use vector and matrix computation during performing forward propagation and training of the DFNN-2H. In other words, these process can be done at one time for all the training points. In solving FFDEPC, the vectorization cannot be implemented



(a) $\eta = 0.39$ **(b)** $\eta = 0.59$

(c) $\eta = 0.79$ **(d)** $\eta = 0.99$

Fig. 7. AS of various ν and t with fixed η for Example 1.

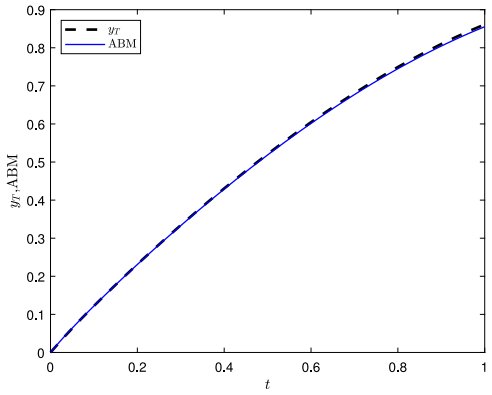
Table 2
Numerical comparison of AS with ABM of Example 2 at different values of ν and η.

t	$\nu = 0.93, \eta = 0.99$		$\nu = 0.95, \eta = 0.95$		$\nu = 0.97, \eta = 0.97$	
	ABM	AS	ABM	AS	ABM	AS
0	0	0	0	0	0	0
0.1	0.1220887806	0.1221157628	0.1259488899	0.1259335857	0.1144328766	0.1150455468
0.2	0.2309759389	0.2315082380	0.2357149697	0.2363123728	0.2198818954	0.2207909383
0.3	0.3330265459	0.3341968396	0.3375781495	0.3387552320	0.3199036559	0.3213123340
0.4	0.4289426406	0.4306268266	0.4327662749	0.4345237658	0.4147663641	0.4165765690
0.5	0.5185630512	0.5208987649	0.5213882159	0.5238012307	0.5041315601	0.5063140366
0.6	0.6014567879	0.6044861773	0.6031933193	0.6062228467	0.5874761748	0.5900746054
0.7	0.6771016127	0.6807794739	0.6777964803	0.6814180235	0.6642220392	0.6672822817
0.8	0.7449596924	0.7494112039	0.7447715248	0.7491321660	0.7337921305	0.7373614753
0.9	0.8045169460	0.8098155213	0.8036984671	0.8087976015	0.7956407452	0.7996576908
1.0	0.8553065827	0.8613360202	0.8541908927	0.8598991742	0.8492721827	0.8536996661
CPU Time (s)		103.38		107.17		104.66
Cost function		4.89×10^{-5}		2.99×10^{-4}		4.39×10^{-4}

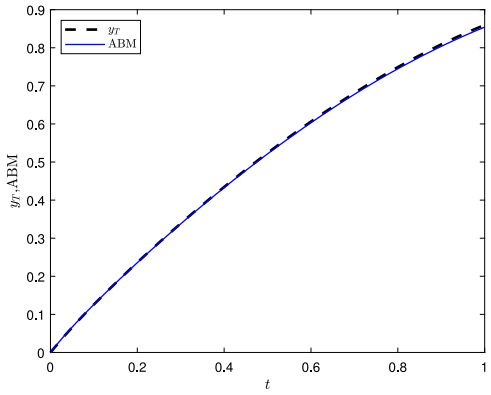
during the computation of FFDPC approximation, which require the evaluation of the approximation at one training point at one time due to the nonlocal properties of the operator. Thus, it is important to minimized the burden of DFNN-2H computation on solving FFDEPC in other aspect such as forward propagation, at least the burden on handling the computational time will be reduced. The designation of the algorithm can be described as follows:

1. *Set up input data:* Take N_{tp} training points from the discretized domain $t \in [0, t_f]$ to form a vector $\tilde{t}$ of size $1 \times N_{tp}$ such that

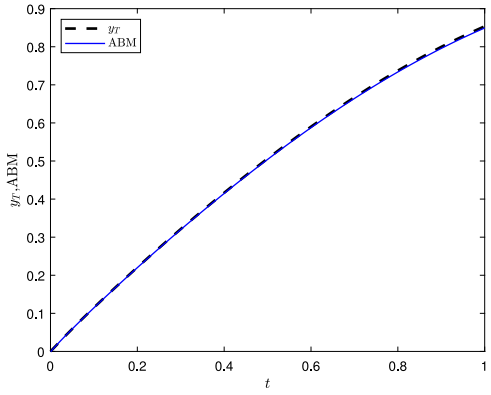
$$\tilde{t} = \left[t^{(1)}, t^{(2)}, \ldots, t^{(N_{tp})}\right].$$



(a) $\nu = 0.93, \eta = 0.99$.



(b) $\nu = 0.95, \eta = 0.95$



(c) $\nu = 0.9, \eta = 0.99$

Fig. 8. Graphical comparison of AS with ABM of Example 2 at different values of ν and η.

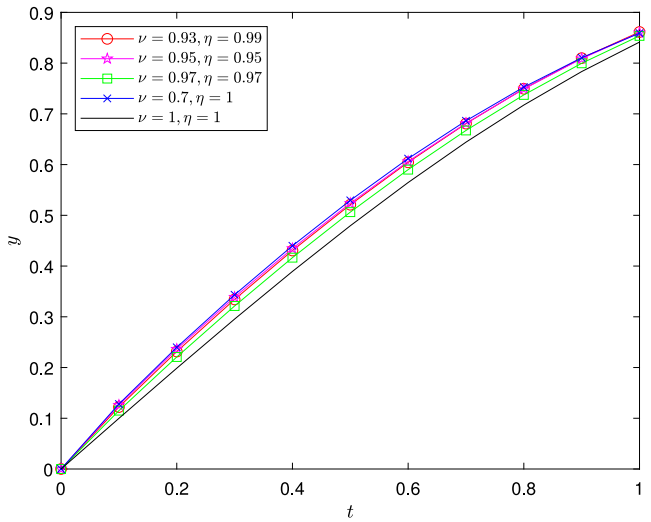
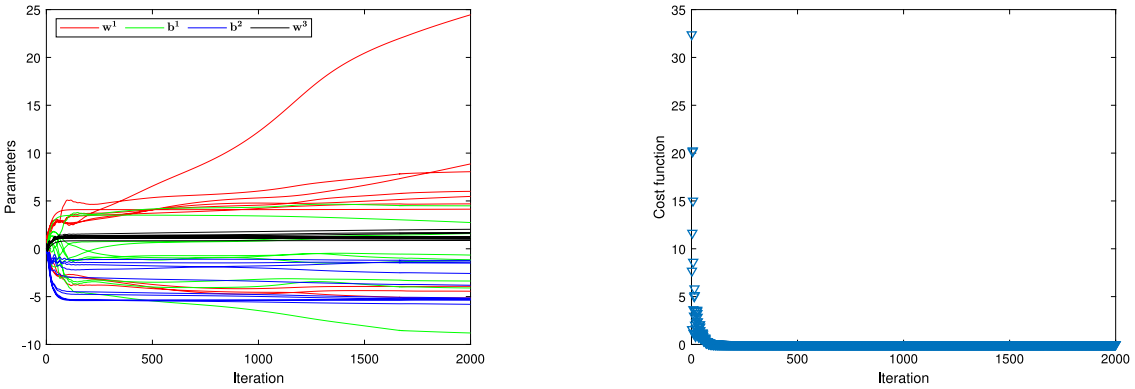
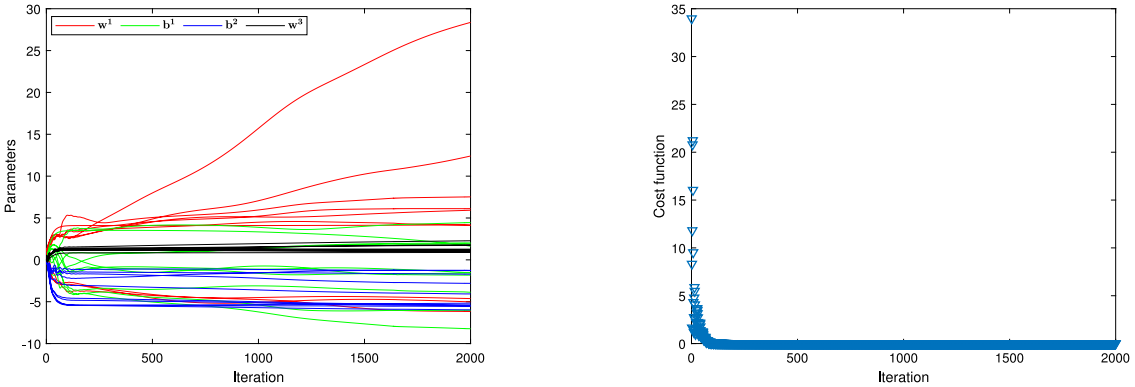


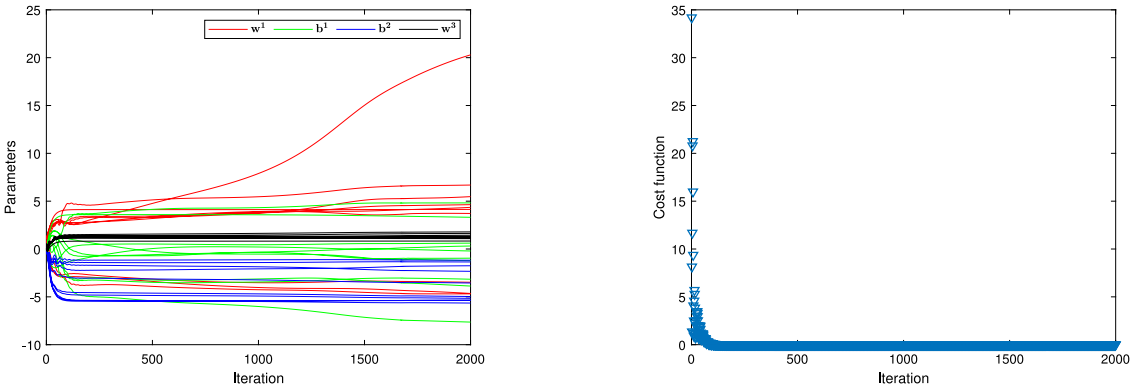
Fig. 9. Graphical comparison of AS of Example 2 at different values of ν and η with exact solution.



(a) $\nu = 0.93$ and $\eta = 0.99$



(b) $\nu = 0.95$ and $\eta = 0.95$



(c) $\nu = 0.97$ and $\eta = 0.97$

Fig. 10. Convergence of the parameters and cost function with respective ν and η for Example 2.

2. *Set up DFNN-2H:* Determine the number of layer and number of neurons. Here, we have 1 input layer, 2 hidden layer and 1 output layer. The number of input layer representing the independent variable t and the number of neuron in output layer represent the dependent variable in the FFDEPC. The number of neurons for the first hidden layer and the second hidden layer are set to be equal.

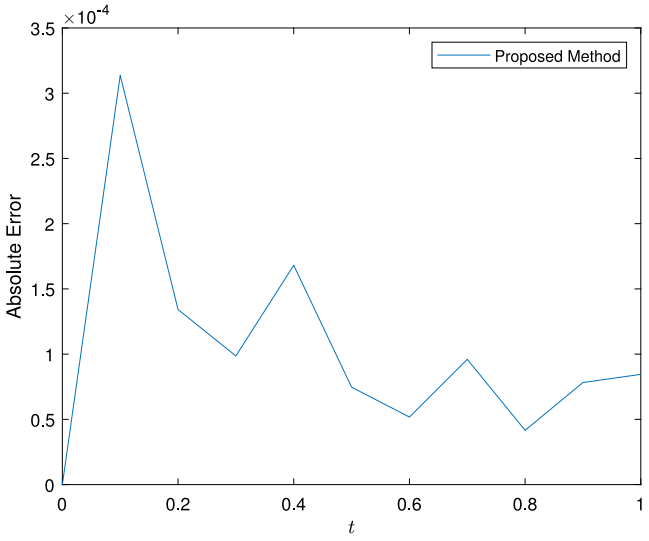


Fig. 11. Absolute error of proposed method when $\nu = 1$ and $\beta = 1$ for Example 2.

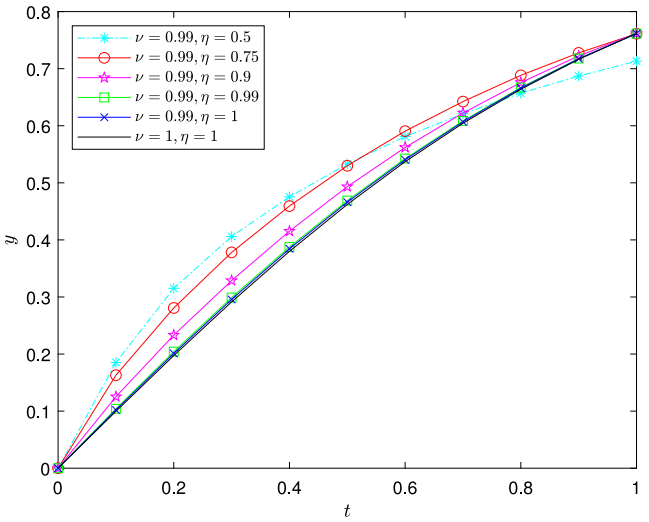


Fig. 12. The comparison of exact solution and AS of proposed method with varying η and $\nu = 0.99$ for Example 3.

3. *Initialize Parameters:* All network parameters are initially set to be random except for the weights from the first hidden layer to second hidden layer where for these weights we use Glorot initialization as suggested by [58]. Through their discussion, the selection of values during initial step is important to avoid the failure during network training.
4. *Forward propagation in vectorized form:* For each training points $t^{(d)}$ for $d = 1, 2, \ldots, N_{tp}$, we rewrite the forward propagation (3)–(8) in condensed form. In the first hidden layer:

$$\mathbf{s}^{[1](d)} = \mathbf{w}^{[1]} p^{[0](d)} + \mathbf{b}^{[1]}, \tag{23}$$

$$\mathbf{p}^{[1](d)} = \phi(\mathbf{s}^{[1](d)}). \tag{24}$$

In the second hidden layer:

$$\mathbf{s}^{[2](d)} = \mathbf{W}^{[2]} \mathbf{p}^{[1](d)} + \mathbf{b}^{[2]}, \tag{25}$$

$$\mathbf{a}^{[2](d)} = \phi(\mathbf{s}^{[2](d)}). \tag{26}$$

and in the output layer:

$$\mathbf{s}^{[2](d)} = \mathbf{W}^{[2]} \mathbf{p}^{[1](d)} + \mathbf{b}^{[2]}, \tag{27}$$

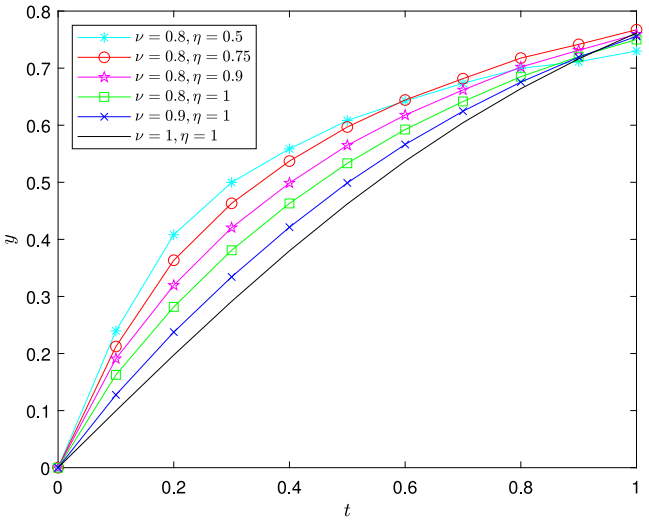


Fig. 13. The comparison of exact solution and AS of proposed method with varying η when $\eta = 0.8, 0.9$ for Example 3.

$$\mathbf{a}^{[2](d)} = \phi(\mathbf{s}^{[2](d)}). \tag{28}$$

By stacking each gridpoints, concatenating all the vectors $\mathbf{s}^{[1](d)}$'s and $\mathbf{p}^{[1](d)}$'s the process can be now put in vectorized form that yields:

$$S^{[1]} = \mathbf{w}^{[1]} P^{[0]} + \mathbf{b}^{[1]}, \tag{29}$$

$$P^{[1]} = \phi(S^{[1]}). \tag{30}$$

where $P^{[0]} = \tilde{\mathrm{t}} \in \mathbb{R}^{1\times N_{tp}}, S^{(1)} \in \mathbb{R}^{k\times N_{tp}}, \mathbf{w}^{[1]} \in \mathbb{R}^{k\times 1}$ and $\mathbf{b}^{[1]} \in \mathbb{R}^{k\times 1}$. Following the same process for the second hidden layer:

$$S^{[2]} = \mathbf{W}^{[2]} P^{[1]} + \mathbf{b}^{[2]}, \tag{31}$$

$$P^{(2)} = \phi(S^{[2]}) \in \mathbb{R}^{k\times N_{tp}}. \tag{32}$$

where $P^{[1]} \in \mathbb{R}^{k\times N_{tp}}, S^{(2)} \in \mathbb{R}^{k\times N_{tp}}, \mathbf{W}^{[2]} \in \mathbb{R}^{k\times k}$ and $\mathbf{b}^{[2]} \in \mathbb{R}^{k\times 1}$. And finally for the output layer:

$$S^{[3]} = \mathbf{w}^{[3]} P^{[2]}, \tag{33}$$

$$P^{[3]} = S^{[3]} \in \mathbb{R}^{1\times N_{tp}}. \tag{34}$$

where $P^{[2]} \in \mathbb{R}^{k\times N_{tp}}, S^{(3)} \in \mathbb{R}^{1\times N_{tp}}$ and $\mathbf{W}^{[2]} \in \mathbb{R}^{1\times k}$. During the development of this phase, broadcasting is implemented in (29) and (31) to get rid the necessity of introducing new variable [59]. Thus, the output of DFNN-2H in vectorized form is $\hat{y}(\tilde{t}, \mathbf{\Phi}) = P^{[3]} \in \mathbb{R}^{1\times N_{tp}}$ where $\mathbf{\Phi} \in \mathbb{R}^{(k^2+4k)\times 1}$.

5. *AS and cost function in vectorized form:* It is straightforward to show that vectorized form of AS and cost function are given as

$$\underbrace{y_T(\tilde{t}, \mathbf{\Phi})}_{\mathbb{R}^{1\times N_{tp}}} = a + \underbrace{\tilde{t} \odot \hat{y}(\tilde{t}, \mathbf{\Phi})}_{\mathbb{R}^{1\times N_{tp}}}, \tag{35}$$

and

$$q = (\hat{\mathbf{G}} - \hat{\mathbf{F}})(\hat{\mathbf{G}} - \hat{\mathbf{F}})', \tag{36}$$

where $\odot$ is Hadamard product or elementwise multiplication and broadcasting is implemented to Eq. (35). While $\hat{\mathbf{G}} = \left[\left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(1)}, \mathbf{\Phi}), \left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(2)}, \mathbf{\Phi}), \ldots, \left({}_0^{FFDPC}D_t^{\nu,\eta} y_T\right)(t^{(N_{tp})}, \mathbf{\Phi})\right] \in \mathbb{R}^{1\times N_{tp}}$ and $\hat{\mathbf{F}} = \left[F\left(t^{(1)}, y_T(t^{(1)}, \mathbf{\Phi})\right), F\left(t^{(2)}, y_T(t^{(2)}, \mathbf{\Phi})\right), \ldots, F\left(t^{(d)}, y_T(t^{(d)}, \mathbf{\Phi})\right)\right]$.

6. *Gradient computation in vectorized form:*

   • Weights from input layer to the first hidden layer:

$$\frac{\partial q}{\partial \mathbf{w}^{[1]}} = 2 \cdot \left(\left[\underbrace{(\hat{\mathbf{G}} - \hat{\mathbf{F}})}_{1\times N_{tp}} \odot \underbrace{\left(\frac{\partial \hat{\mathbf{G}}}{\partial \mathbf{w}^{[1]}} - \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{w}^{[1]}} \right)}_{k\times N_{tp}} \right] \cdot \underbrace{\mathbf{1}}_{N_{tp}\times 1} \right) \in \mathbb{R}^{k\times 1}. \tag{37}$$

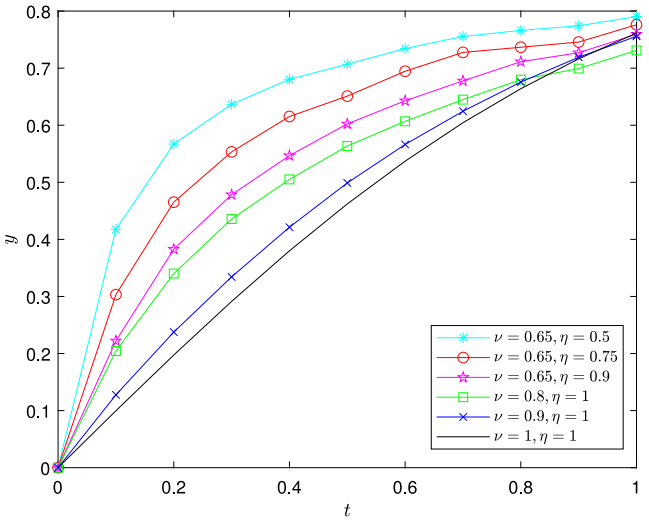


Fig. 14. The comparison of exact solution and AS of proposed method with varying η when $\nu = 0.65, 0.8, 0.9$ for Example 3.

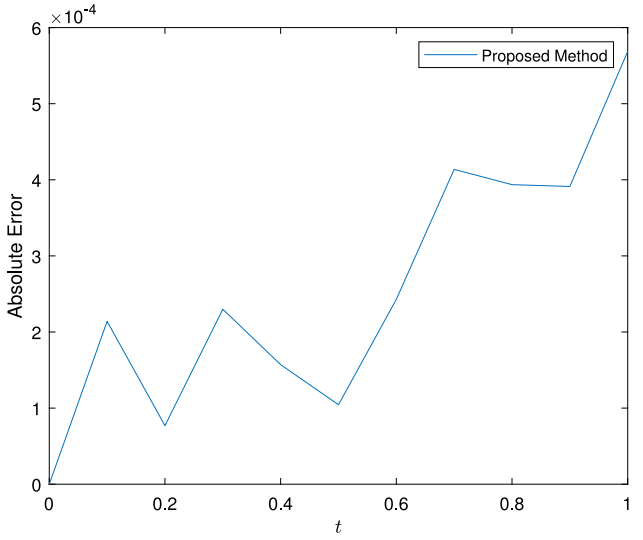


Fig. 15. Absolute error of proposed method when $\nu = 1$ and $\eta = 1$ for Example 3.

- Weights from the second hidden layer to the output layer:

$$\frac{\partial q}{\partial \mathbf{w}^{[3]}} = 2 \cdot \left(\underbrace{\mathbf{1}^T}_{1\times m_{tp}} \cdot \left[\underbrace{(\hat{\mathbf{G}} - \hat{\mathbf{F}})}_{1\times N_{tp}} \odot \underbrace{\left(\frac{\partial \hat{\mathbf{G}}}{\partial \mathbf{w}^{[3]}} - \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{w}^{[3]}} \right)}_{k\times N_{tp}} \right]^T \right) \in \mathbb{R}^{1\times k}. \tag{38}$$

- Biases at first hidden layer:

$$\frac{\partial q}{\partial \mathbf{b}^{[1]}} = 2 \cdot \left(\left[\underbrace{(\hat{\mathbf{G}} - \hat{\mathbf{F}})}_{1\times N_{tp}} \odot \underbrace{\left(\frac{\partial \hat{\mathbf{G}}}{\partial \mathbf{b}^{[1]}} - \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{b}^{[1]}} \right)}_{k\times N_{tp}} \right] \cdot \underbrace{\mathbf{1}}_{N_{tp}\times 1} \right) \in \mathbb{R}^{k\times 1}. \tag{39}$$

Table 3
Comparison of the present method and method in [37] and [60] for Example 3 at $\nu = 1$ and $\eta = 1$.

t	y	AS	Method in [37]	Method in [60]
0.1	0.0997	0.0997	0.0997	0.0995
0.3	0.2913	0.2913	0.2914	0.2909
0.5	0.4621	0.4621	0.4622	0.4622
0.7	0.6044	0.6044	0.6044	0.6044
0.9	0.7163	0.7163	0.7164	0.6640
CPU Time (s)	–	0.7163	–	–
Cost	–	1.5713×10^{-5}	–	–

- Biases at second hidden layer:

$$\frac{\partial q}{\partial \mathbf{b}^{[2]}} = 2 \cdot \left(\left[\underbrace{(\hat{\mathbf{G}} - \hat{\mathbf{F}})}_{1 \times N_{tp}} \odot \underbrace{\left(\frac{\partial \hat{\mathbf{G}}}{\partial \mathbf{b}^{[2]}} - \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{b}^{[2]}} \right)}_{k \times N_{tp}} \right] \cdot \underbrace{\mathbf{1}}_{N_{tp} \times 1} \right) \in \mathbb{R}^{k \times 1} \tag{40}$$

where broadcasting is applied to $\hat{\mathbf{G}} - \hat{\mathbf{F}}$ before doing Hadamard product [59]. $\mathbf{1}$ is the column vector containing number 1 for all its element.

7. *Update network parameters:* The parameter $\mathbf{\Phi}_1 = [\mathbf{w}^{[1]}, \mathbf{w}^{[3]}, \mathbf{b}^{[1]}, \mathbf{b}^{[2]}]$ is concatenated into column vector of size $4k \times 1$ need to be adjusted through Adam:

$$\mathbf{v}^l = \beta_1 \mathbf{v}^{l-1} + (1 - \beta_1) \nabla q(\mathbf{\Phi}_1^l), \tag{41}$$

$$\mathbf{s}^l = \beta_2 \mathbf{s}^{l-1} + (1 - \beta_2) \nabla \left(\nabla q(\mathbf{\Phi}_1^l) \right)^2, \tag{42}$$

$$\hat{\mathbf{v}}^l = \frac{\mathbf{v}^l}{1 - \beta_1^l}, \tag{43}$$

$$\hat{\mathbf{s}}^l = \frac{\hat{\mathbf{s}}^l}{1 - \beta_2^l}, \tag{44}$$

$$\mathbf{\Phi}_1^{l+1} = \mathbf{\Phi}_1^l - \frac{\nu}{\sqrt{\hat{\mathbf{s}}^l + \epsilon}} \hat{\mathbf{v}}^l. \tag{45}$$

where $\mathbf{v}$, $\mathbf{s}$, $\hat{\mathbf{v}}$ and $\hat{\mathbf{s}}$ are all zeros vector with size $4k \times 1$. Note that the evaluation of gradient computation, $\nabla q(\mathbf{\Phi_1})$ are computed with the help of automatic differentiation to yield vector of size $4k \times 1$.

4. Numerical results

In this section, we implement the algorithm on solving linear and nonlinear FFDEPC. For all cases, the exact solution for $\nu < 1$ and $\eta < 1$ is not available, thus we compare our results with the Adam–Bashforth method (ABM) proposed in [61]. The following parameters were used for Examples 1–3:

- $\nu = 0.1$, $\beta_1 = 0.95$, $\beta_2 = 0.995$ and $\epsilon = 1 \times 10^{-8}$.
- $N_{tp} = 31$ and number of iterations= 2000.

with the same set of weights and biases. For the last example, we consider a special application known as Lotka–Volterra system. To maximize the accuracy of the output, we use different parameters that will mentioned later in the discussion of the example.

Example 1 ([62]). Consider the following FFDEPC

$$\left({}_0^{FFDPC} D_t^{\nu,\eta} y \right)(t) = t^2, \quad t \in [0, 1] \tag{46}$$

with the initial condition $y(0) = 0$. The exact solution when $\nu = 1$ and $\eta = 1$ for this example is

$$y(t) = \frac{t^3}{3}. \tag{47}$$

Based on Eq. (11), the designated AS is

$$y_T(t, \mathbf{\Phi}) = t\hat{y}(t, \mathbf{\Phi}). \tag{48}$$

We compare our proposed method with ABM at different values of fractional-order and fractal-order, where $\nu = 0.7, \eta = 0.75$, $\nu = 0.8, \eta = 0.85$ and $\nu = 0.9, \eta = 0.95$ as presented in Table 1 respectively. The graphical comparison of AS for each result is shown in Fig. 2. We can see that our proposed method very close to ABM. In Fig. 3, we can see that as the value fractal-order increases, the solutions profile go towards the exact for $\nu = 1$ and $\eta = 1$. These solution profile are below than AS when $\nu = 0.5$, $\eta = 1$ because

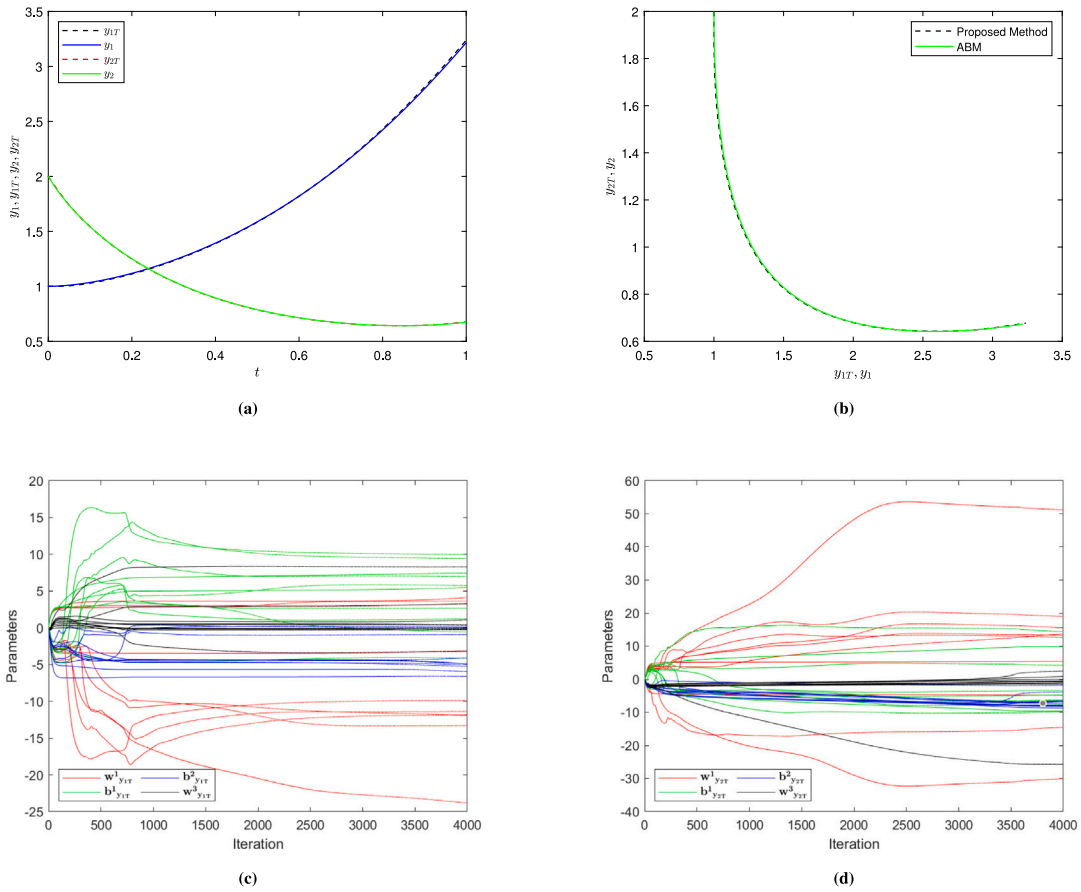


Fig. 16. Numerical results for Example 5 using $\nu = 0.9$ and $\eta = 0.99$ with comparison of ABM [21] in 16(a) and 16(b). While the convergence of network parameters for each AS is portrayed in 16(c) and 16(d) respectively. The CPU time for this case is 398.13 s.

their values of ν is larger than $\nu = 0.5$. The respective plot of convergence for these cases are displayed in Fig. 4. Through this figure, the network parameters attained to an optimal ones with respective cost functions that leads to zero as iteration increases. To demonstrate the accuracy of the proposed scheme with the exact solution, we plot the absolute error in Fig. 5. We finally perform numerical simulations by fixing the fractional-order, $\nu = 0.39, 0.59, 0.79, 0.99$ and vary fractal-order, η and fixing the fractal-order, $\eta = 0.39, 0.59, 0.79, 0.99$ and vary fractional-order, ν as in Figs. 6 and 7 respectively.

Example 2 (*[26]*). Consider the following FFDEPC

$$\left({}_{0}^{FFDPC}D_{t}^{\nu,\eta}y\right)(t) = \cos(t), \quad t \in [0,1], \tag{49}$$

with the initial condition $y(0) = 0$. The exact solution when $\nu = 1$ and $\eta = 1$ for this example is

$$y(t) = sin(t). \tag{50}$$

From (11), the AS can be written as

$$y_T(t, \boldsymbol{\Theta}) = t\hat{y}(t, \boldsymbol{\Theta}). \tag{51}$$

By applying our method, Table 2 shows the numerical comparison for $\nu = 0.93, \eta = 0.99$ and $\nu = 0.95, \eta = 0.95$ $\nu = 0.97, \eta = 0.97$. For these value of fractional-order and fractal-order, Fig. 8 shows that our approximate solutions are close with ABM solutions. As the value of fractal-order increases, we can see that the solutions profile goes towards exact at $\nu = 1$ and $\eta = 1$ as shown in Fig. 9. These solution profile are below than approximate solution when $\nu = 0.5$, $\eta = 1$ because the value of ν is less than others. These solutions behaviour is the same like previous example. For all cases, the network parameters are tends to a constant value over time indicating the optimal value has achieved as shown in Fig. 10. However, only several value of weights are still not consistent to same value over time until iteration ended. Finally, the absolute error of the proposed method when $\nu = 1$ and $\beta = 1$ is plotted in Fig. 11.

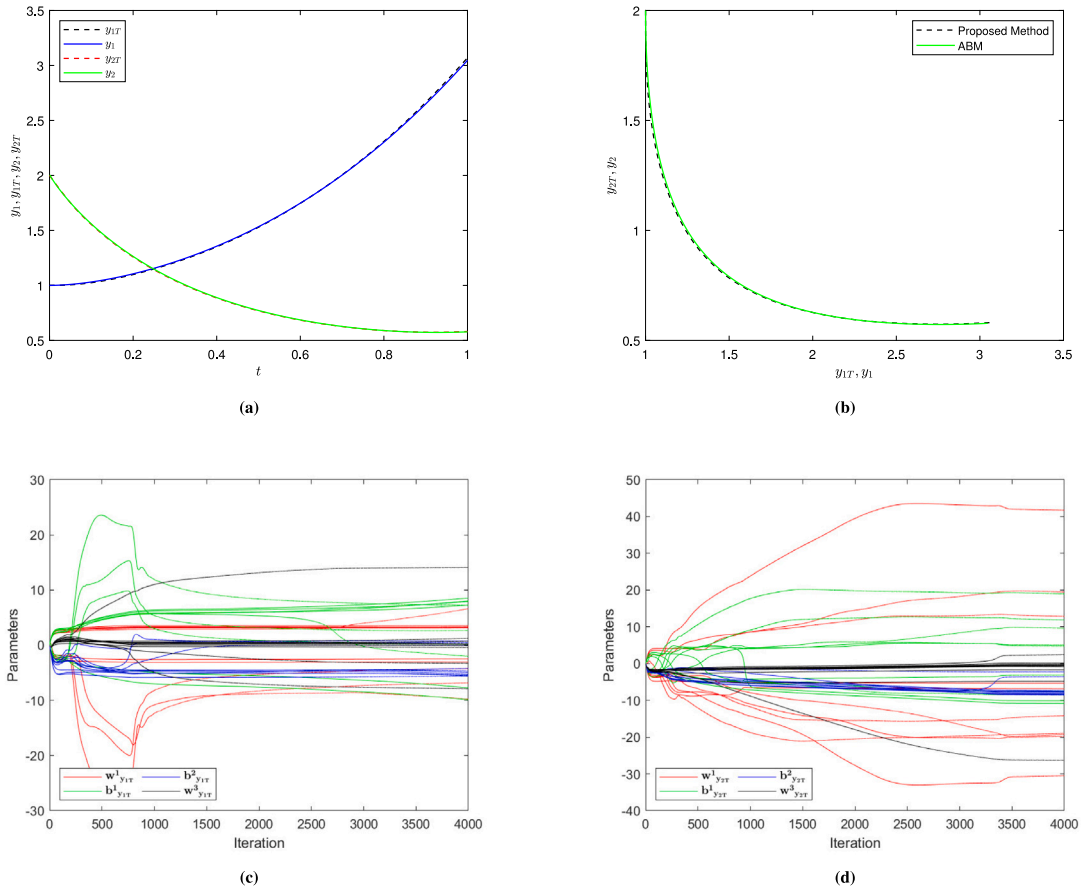


Fig. 17. Numerical results for Example 5 using $\nu = 0.95$ and $\eta = 0.95$ with comparison of ABM [21] in 17(a) and 17(b). While the convergence of network parameters for each AS is portrayed in 17(c) and 17(d) respectively. The CPU time taken for this case is 377.65 s.

Example 3 (*[37]*)**.** Consider the following Riccati FFDEPC

$$\left({}_{0}^{FFDPC}D_{t}^{\nu,\eta}y\right)(t) = 1 - y^2(t), \quad t \in [0,1], \tag{52}$$

with the initial condition $y(0) = 0$. The exact solution for this example when $\nu = \eta = 1$ is

$$y(t) = \frac{e^{2t-1}}{e^{2t+1}}. \tag{53}$$

The AS is

$$y_T(t, \boldsymbol{\Theta}) = t\hat{y}(t, \boldsymbol{\Theta}). \tag{54}$$

In this example, we focuses on comparing various value of fractional-order, ν and fractal-order, η to the exact solution when $\nu = \eta = 1$. Initially, we fix the value of fractional-order and let the fractal-order increases up to 1. Then, we increase the fractional-order until it becomes equal to 1. We tested initially on fractional-order, $\nu = 0.99$ in Fig. 12, $\nu = 0.8, 0.9$ in Fig. 13 and $\nu = 0.65, 0.8, 0.9$ in Fig. 14. From all the results, we can see that as the fractal-order and fractional-order increases, the AS will move towards the exact solution when $\nu = \eta = 1$. We also can observe that for fixed value of ν, the AS that have higher η will located nearer to the exact solution when $\nu = \eta = 1$. As $\eta = 1$, the AS that have higher value of ν will located near the exact solution when $\nu = \eta = 1$. We also show in Table 3 that our proposed method are more accurate than [60] and competitive with [37]. Since our proposed scheme is competitive with method in [37], however, there is unavailability of the time taken from the method to be compared to show which method perform the best. The only indicator to show our proposed scheme is the best is by observe the solution obtained by proposed scheme are all similar with the exact solution, compared from the method by [29] are less accurate at $x = 0.3, 0.5$ and 0.9. The CPU time for this case is 0.7163s, however for other results are ranging from 100.71 up to 107.18. The time is faster for $\nu = 1$ and $\eta = 1$ as there is no approximation of FFDPC, thus no nonlocality involved during the computation. As the final output, we plot the solution for the case of $\nu = 1$ and $\beta = 1$ with the exact solution as in Fig. 15.

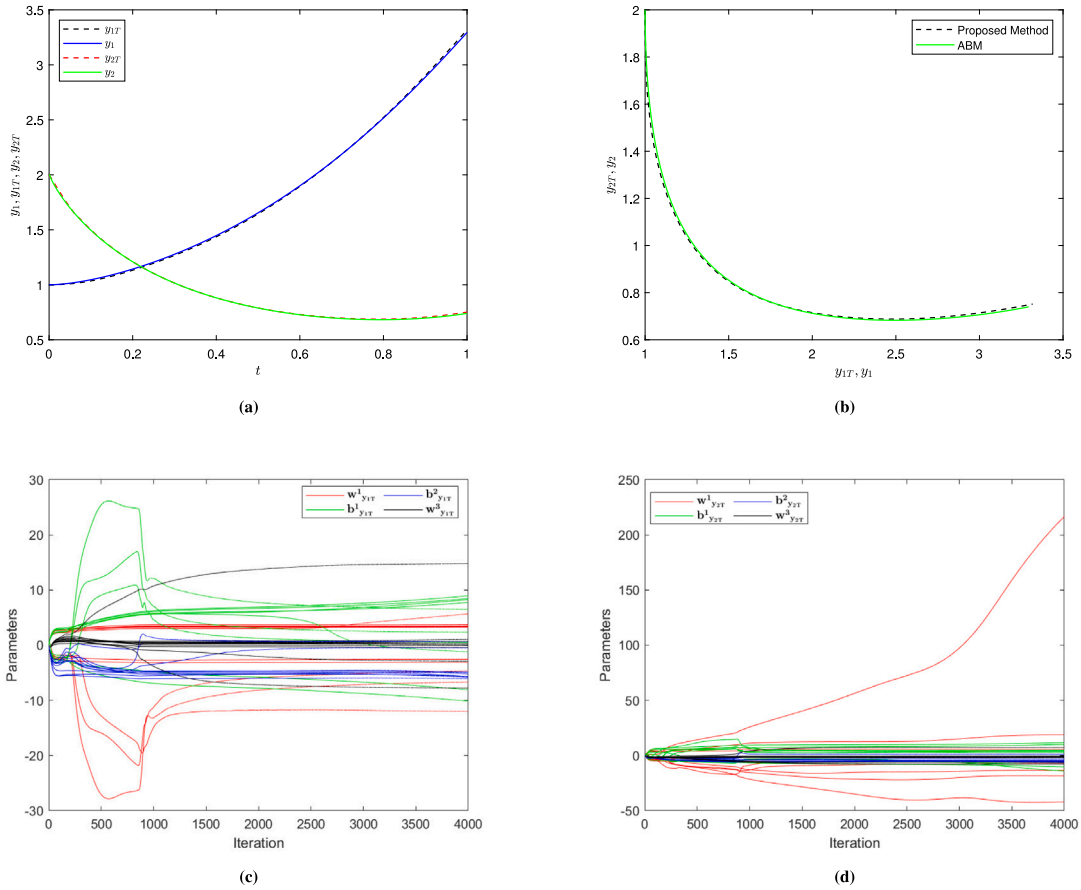


Fig. 18. Numerical results for Example 5 using $\nu = 0.87$ and $\eta = 0.97$ with comparison of ABM [21] in 18(a) and 18(b). While the convergence of network parameters for each AS is portrayed in 18(c) and 18(d) respectively. The CPU time taken for this case is 381.17 s.

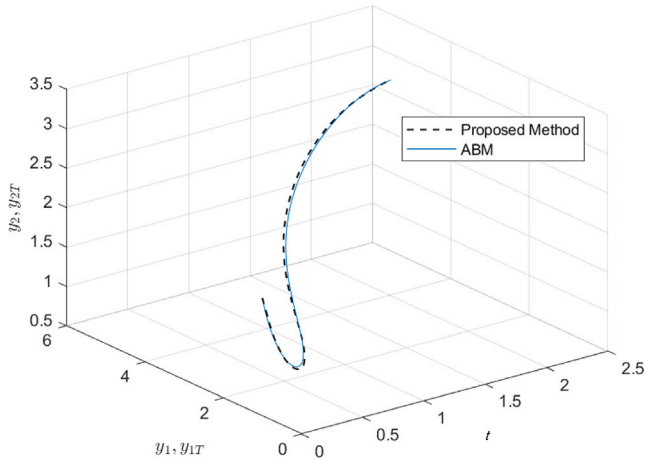


Fig. 19. Numerical results for $\nu = 0.9$ and $\eta = 0.9$ with ABM [21] for Example 5.

Example 4 ([63]). Consider the following FFDEPC of Lotka–Volterra system

$$({}_0^{FFDPC}D_t^{\nu,\eta} y_1)(t) = a_1 y_1(t) - b_1 y_1(t) y_2(t), \tag{55}$$

$$({}_0^{FFDPC}D_t^{\nu,\eta} y_2)(t) = a_2 y_1(t) y_2(t) - b_2 y_2(t). \tag{56}$$

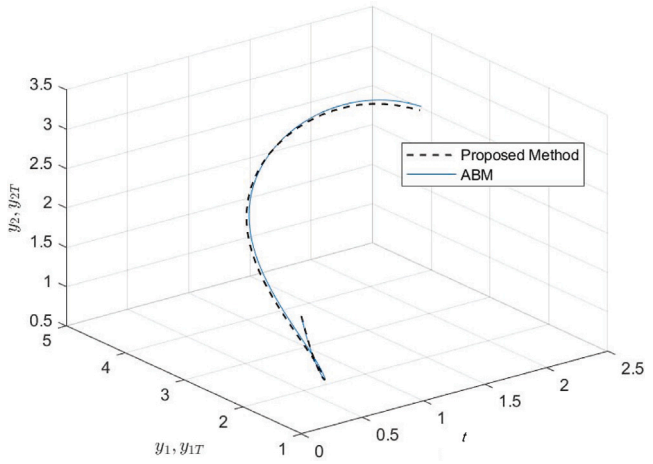


Fig. 20. Numerical results for $\nu = 0.87$ and $\eta = 0.97$ with ABM [21] for Example 5.

where $a_1 = 2, b_1 = 1, a_2 = 1$ and $b_2 = 3$ with the initial conditions $y_1(0) = 1$ and $y_2(0) = 2$. The designated AS is

$$y_{1T}(t, \boldsymbol{\Theta}) = 1 + t\hat{y_1}(t, \boldsymbol{\Theta}), \tag{57}$$

$$y_{2T}(t, \boldsymbol{\Theta}) = 2 + t\hat{y_2}(t, \boldsymbol{\Theta}). \tag{58}$$

Notice that this is a system of FFDEPC that have two unknown variable which is y_1 and y_2. Same architecture of DFNN-2H as in Fig. 1 is used for the both y_1 and y_2 to yield an output $\hat{y_1}(t, \boldsymbol{\Theta})$ and $\hat{y_2}(t, \boldsymbol{\Theta})$. For this application problem, there is no exact solution available for all values of fractional-order, ν and fractal-order, η. Here, we use the ABM proposed by [21] with step size 0.0001 as our reference comparison. The numerical simulations for $\nu = 0.9, \eta = 0.99$, $\nu = 0.95, \eta = 0.95$ and $\nu = 0.87, \eta = 0.97$ are portrayed in Fig. 16, Fig. 17 and Fig. 18 respectively. We finally plot 3D results in Fig. 19 and Fig. 20 for $\nu = 0.9, \eta = 0.9$ and $\nu = 0.87, \eta = 0.97$ in domain $[0, 2]$ respectively. We can see the numerical results for all cases are nearly identical to the ABM which our reference for exact solutions. It is important to mention that for this example we use different sets of parameters:

Results in Fig. 16–Fig. 18

- $\nu = 0.1$, $\beta_1 = 0.95$, $\beta_2 = 0.995$ and $\epsilon = 1 \times 10^{-8}$.
- $N_{tp} = 51$ and number of iterations= 4000.

Results in Fig. 19 and Fig. 20

- $\nu = 0.001$, $\beta_1 = 0.95$, $\beta_2 = 0.995$ and $\epsilon = 1 \times 10^{-8}$.
- $N_{tp} = 51$ and number of iterations= 30000.

to obtain the best results.

5. Conclusion

The objective of this work is to develop a vectorized algorithm based on DFNN-2H for solving FFDEPC. The proposed method is performed with Adam optimization technique for network training. A basic framework of the algorithm include the designation of approximate solution (AS) and approximation of fractal–fractional derivative with power law kernel in Caputo sense (FFDPC). Results from the five examples indicate that proposed scheme produced AS with accurate result for various value of fractional-order, ν and fractal-order, η through observation of absolute error with low computational time. The network parameters obtained are mostly reached the optimal ones with cost functions decreases to nearly zero until the iteration ended. Our method also can be solve for system of FFDEPC in which a good choice of learning rate, training points and number of iterations should carefully decided to get the better output of the results. For future studies, this method can be implement to solve not only examples provided here but also variety types of FFDEPC can be founded in literature. There exist another form of neural network known as Hopfield neural network (HNN) which interested readers could read [64] on its capability on solving FDEs. This form of HNN also could be extended on solving this type of FFDEPC.

CRediT authorship contribution statement

Mohd Rashid Admon: Conceptualization, Methodology, Software, Writing – original draft. **Norazak Senu:** Supervision, Validation, Reviewing original draft. **Ali Ahmadian:** Supervision, Validation, Reviewing original draft. **Zanariah Abdul Majid:** Validation of data, Final editing. **Soheil Salahshour:** Validation of data, Final editing.

Acknowledgements

The authors are very thankful to Malaysia Ministry of Education for awarded Fundamental Research Grant Scheme (Ref. No. FRGS/1/2022/STG06/UPM/02/2) and Fellow Scheme from Universiti Teknologi Malaysia for supporting this work.

References

[1] I. Podlubny, Fractional Differential Equations: an Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications, Elsevier, 1998.
[2] K. Oldham, J. Spanier, The Fractional Calculus Theory and Applications of Differentiation and Integration To Arbitrary Order, Elsevier, 1974.
[3] H. Sun, Y. Zhang, D. Baleanu, W. Chen, Y. Chen, A new collection of real world applications of fractional calculus in science and engineering, Commun. Nonlinear Sci. Numer. Simul. 64 (2018) 213–231.
[4] V.E. Tarasov, No nonlocality. No fractional derivative, Commun. Nonlinear Sci. Numer. Simul. 62 (2018) 157–163.
[5] H. Qu, M.U. Rahman, M. Arfan, G. Laouini, A. Ahmadian, N. Senu, S. Salahshour, et al., Investigating fractal-fractional mathematical model of tuberculosis (tb) under fractal-fractional Caputo operator, Fractals (fractals) 30 (05) (2022) 1–14.
[6] J.F. Gómez-Aguilar, H. Yépez-Martínez, C. Calderón-Ramón, I. Cruz-Orduña, R.F. Escobar-Jiménez, V.H. Olivares-Peregrino, Modeling of a mass-spring-damper system by fractional derivatives with and without a singular kernel, Entropy 17 (9) (2015) 6289–6303.
[7] A. Atangana, A novel model for the lassa hemorrhagic fever: deathly disease for pregnant women, Neural Comput. Appl. 26 (8) (2015) 1895–1903.
[8] P. Kumar, M. Vellappandi, Z.A. Khan, S. Sivalingam, A. Kaziboni, V. Govindaraj, A case study of monkeypox disease in the United States using mathematical modeling with real data, Math. Comput. Simulation (2023).
[9] S. Sivalingam, P. Kumar, V. Govindaraj, A novel optimization-based physics-informed neural network scheme for solving fractional differential equations, Eng. Comput. (2023) 1–11.
[10] W. Chen, Time–space fabric underlying anomalous diffusion, Chaos Solitons Fractals 28 (4) (2006) 923–929.
[11] W. Chen, H. Sun, X. Zhang, D. Korošak, Anomalous diffusion modeling by fractal and fractional derivatives, Comput. Math. Appl. 59 (5) (2010) 1754–1758.
[12] H. Sun, M.M. Meerschaert, Y. Zhang, J. Zhu, W. Chen, A fractal richards' equation to capture the non-Boltzmann scaling of water transport in unsaturated media, Adv. Water Resour. 52 (2013) 292–295.
[13] Y. Liang, Q.Y. Allen, W. Chen, R.G. Gatto, L. Colon-Perez, T.H. Mareci, R.L. Magin, A fractal derivative model for the characterization of anomalous diffusion in magnetic resonance imaging, Commun. Nonlinear Sci. Numer. Simul. 39 (2016) 529–537.
[14] A. Atangana, Fractal-fractional differentiation and integration: connecting fractal calculus and fractional calculus to predict complex system, Chaos Solitons Fractals 102 (2017) 396–406.
[15] H. Srivastava, K.M. Saad, Numerical simulation of the fractal-fractional ebola virus, Fractal Fract. 4 (4) (2020) 49.
[16] E.A. Algehyne, M. Ibrahim, Fractal-fractional order mathematical vaccine model of COVID-19 under non-singular kernel, Chaos Solitons Fractals 150 (2021) 111150.
[17] N.R. Babu, P. Balasubramaniam, Master–slave synchronization for glucose–insulin metabolism of type-1 diabetic mellitus model based on new fractal–fractional order derivative, Math. Comput. Simulation 204 (2023) 282–301.
[18] K.A. Abro, A. Atangana, Numerical study and chaotic analysis of meminductor and memcapacitor through fractal–fractional differential operator, Arab. J. Sci. Eng. 46 (2) (2021) 857–871.
[19] N.R. Babu, P. Balasubramaniam, K. Ratnavelu, Existence and uniqueness for a new perturbed chaotic jerk circuit model based on fractal-fractional derivative, in: AIP Conference Proceedings, Vol. 2756, No. 1, AIP Publishing, 2023.
[20] Z. Li, Z. Liu, M.A. Khan, Fractional investigation of bank data with fractal-fractional Caputo derivative, Chaos Solitons Fractals 131 (2020) 109528.
[21] A. Atangana, S. Qureshi, Modeling attractors of chaotic dynamical systems with fractal–fractional operators, Chaos Solitons Fractals 123 (2019) 320–337.
[22] A. Atangana, M.A. Khan, et al., Modeling and analysis of competition model of bank data with fractal-fractional Caputo-Fabrizio operator, Alex. Eng. J. 59 (4) (2020) 1985–1998.
[23] K.A. Abro, A. Atangana, Mathematical analysis of memristor through fractal-fractional differential operators: a numerical study, Math. Methods Appl. Sci. 43 (10) (2020) 6378–6395.
[24] M. Arfan, H. Alrabaiah, M.U. Rahman, Y.-L. Sun, A.S. Hashim, B.A. Pansera, A. Ahmadian, S. Salahshour, Investigation of fractal-fractional order model of COVID-19 in Pakistan under Atangana-Baleanu Caputo (ABC) derivative, Results Phys. 24 (2021) 104046.
[25] K.M. Saad, Fractal-fractional brusselator chemical reaction, Chaos Solitons Fractals 150 (2021) 111087.
[26] T. Mekkaoui, A. Atangana, S.İ. Araz, Predictor–corrector for non-linear differential and integral equation with fractal–fractional operators, Eng. Comput. 37 (3) (2021) 2359–2368.
[27] A. Rayal, S.R. Verma, Numerical analysis of pantograph differential equation of the stretched type associated with fractal-fractional derivatives via fractional order Legendre wavelets, Chaos Solitons Fractals 139 (2020) 110076.
[28] M.H. Heydari, A. Atangana, Z. Avazzadeh, Chebyshev polynomials for the numerical solution of fractal–fractional model of nonlinear Ginzburg–Landau equation, Eng. Comput. 37 (2) (2021) 1377–1388.
[29] A. Shloof, N. Senu, A. Ahmadian, S. Salahshour, An efficient operation matrix method for solving fractal–fractional differential equations with generalized Caputo-type fractional–fractal derivative, Math. Comput. Simulation 188 (2021) 415–435.
[30] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[31] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[32] S. Chakraverty, S. Mall, Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations, CRC Press, 2017.
[33] M. Pakdaman, A. Ahmadian, S. Effati, S. Salahshour, D. Baleanu, Solving differential equations of fractional order using an optimization technique based on training artificial neural network, Appl. Math. Comput. 293 (2017) 81–95.
[34] M.A.Z. Raja, R. Samar, M.A. Manzar, S.M. Shah, Design of unsupervised fractional neural network model optimized with interior point algorithm for solving Bagley–Torvik equation, Math. Comput. Simulation 132 (2017) 139–158.
[35] S. Panghal, M. Kumar, Neural network method: delay and system of delay differential equations, Eng. Comput. (2021) 1–10.
[36] T.T. Dufera, Deep neural network for system of ordinary differential equations: vectorized algorithm and simulation, Mach. Learn. Appl. 5 (2021) 100058.
[37] A. Shloof, N. Senu, A. Ahmadian, M. Pakdaman, S. Salahshour, A new iterative technique for solving fractal-fractional differential equations based on artificial neural network in the new generalized Caputo sense, Eng. Comput. (2022) 1–11.
[38] N.R. Babu, P. Balasubramaniam, Master-slave synchronization of a new fractal-fractional order quaternion-valued neural networks with time-varying delays, Chaos Solitons Fractals 162 (2022) 112478.
[39] M.R. Admon, N. Senu, A. Ahmadian, Z.A. Majid, S. Salahshour, A new efficient algorithm based on feedforward neural network for solving differential equations of fractional order, Commun. Nonlinear Sci. Numer. Simul. 117 (2023) 106968.
[40] P. Dixit, S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Comp. Sci. Rev. 39 (2021) 100317.

[41] S. Dong, P. Wang, K. Abbas, A survey on deep learning and its applications, Comp. Sci. Rev. 40 (2021) 100379.
[42] S. Minaee, Y.Y. Boykov, F. Porikli, A.J. Plaza, N. Kehtarnavaz, D. Terzopoulos, Image segmentation using deep learning: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[43] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), Vol. 2, Ieee, 2004, pp. 985–990.
[44] N. Yadav, A. Yadav, M. Kumar, et al., An Introduction To Neural Network Methods for Differential Equations, Vol. 1, Springer, 2015.
[45] E. Guresen, G. Kayakutlu, Definition of artificial neural networks with comparison to other networks, Procedia Comput. Sci. 3 (2011) 426–433.
[46] P. Kim, Matlab deep learning, with machine learning, Neural Netw. Artif. Intell. 130 (21) (2017).
[47] R. Demidov, A.I. Pechenkin, P.D. Zegzhda, M.O. Kalinin, Application model of modern artificial neural network methods for the analysis of information systems security, Autom. Control Comput. Sci. 52 (8) (2018) 965–970.
[48] J. Durodola, N. Li, S. Ramachandra, A. Thite, A pattern recognition artificial neural network method for random fatigue loading life prediction, Int. J. Fatigue 99 (2017) 55–67.
[49] Z. Abduh, E.A. Nehary, M.A. Wahed, Y.M. Kadah, Classification of heart sounds using fractional fourier transform based mel-frequency spectral coefficients and traditional classifiers, Biomed. Signal Process. Control 57 (2020) 101788.
[50] P. Mani, R. Rajan, L. Shanmugam, Y.H. Joo, Adaptive control for fractional order induced chaotic fuzzy cellular neural networks and its application to image encryption, Inform. Sci. 491 (2019) 74–89.
[51] J. Bruna, L. Dec, Mathematics of deep learning, 2018,
[52] S. Sivalingam, P. Kumar, V. Govindaraj, A neural networks-based numerical method for the generalized Caputo-type fractional differential equations, Math. Comput. Simulation (2023).
[53] M.J. Kochenderfer, T.A. Wheeler, Algorithms for Optimization, Mit Press, 2019.
[54] S. Ruder, An overview of gradient descent optimization algorithms, 2016, arXiv preprint arXiv:1609.04747.
[55] G.F. Corliss, Application of Differentiation Arithmetic, Volume 19 of Perspectives in Computing, Academic Press, Boston, 1988.
[56] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Marchine Learn. Res. 18 (2018) 1–43.
[57] S. Wright, J. Nocedal, et al., Numerical optimization, Springer Sci. 35 (67–68) (1999) 7.
[58] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[59] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[60] S. Mall, S. Chakraverty, Artificial neural network approach for solving fractional order initial value problems, 2018, arXiv preprint arXiv:1810.04992.
[61] J. Gómez-Aguilar, A. Atangana, New chaotic attractors: Application of fractal-fractional differentiation and integration, Math. Methods Appl. Sci. 44 (4) (2021) 3036–3065.
[62] A. Akgül, Analysis and new applications of fractal fractional differential equations with power law kernel, Discr. Contin. Dyn. Syst.-S 14 (10) (2021) 3401.
[63] S. Kumar, A. Kumar, Z.M. Odibat, A nonlinear fractional model to describe the population dynamics of two interacting species, Math. Methods Appl. Sci. 40 (11) (2017) 4134–4148.
[64] S. Rezapour, P. Kumar, V.S. Erturk, S. Etemad, A study on the 3D hopfield neural network model via nonlocal Atangana–Baleanu operators, Complexity 2022 (2022).