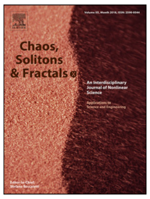
Chaos, Solitons and Fractals 153 (2021) 111530

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

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



Frontiers

Fractional Chebyshev deep neural network (FCDNN) for solving differential models

Zeinab Hajimohammadi[a], Fatemeh Baharifard[b], Ali Ghodsi[d], Kourosh Parand[a,c,d,*,*]

[a] Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
[c] Institute for Cognitive and Brain Sciences, Shahid Beheshti University, Tehran, Iran
[d] Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Canada

A R T I C L E I N F O

Article history:
Received 20 June 2021
Revised 13 October 2021
Accepted 17 October 2021
Available online 25 November 2021

Keywords:
Deep neural network
Chebyshev polynomials
Collocation method
Fractional partial differential equations (FPDEs)
Fractional Fredholm integral equations (FFIEs)

A B S T R A C T

Differential and integral equations have been used vastly in modeling engineering and science problems. Solving these equations has been always an active and important area of research. In this paper, we propose the Fractional Chebyshev Deep Neural Network (FCDNN) for solving fractional differential models. Chebyshev orthogonal polynomials are basic functions in spectral methods. These functions are used as activation functions in FCDNN. The marching in time technique and the Gaussian method are applied in the fractional operations to simplify the calculations. We show how FCDNN can be used to solve fractional Fredholm integral equations (FFIEs). We also propose a solution to the extension of fractional time order partial differential equations (FPDEs). In this approach, fractional PDEs are first discretized by the finite difference and the marching in time methods and then are solved using FCDNN. Fractional Fredholm integral equations are also first approximated by the numerically Gaussian quadrature method and then are solved using FCDNN. A comparison between the results from FCDNN and some other methods is presented to validate the effectiveness and advance of the proposed method.

1. Introduction

Neural networks as well as deep networks are part of a wider family of machine learning methods and have been used to solve many problems in various scientific fields. Some of the considered areas of application are image processing, speech processing and medical diagnosis [1–4]. The methods have achieved great success in these areas in obtaining good and accurate answers. But, one of the theoretical results most frequently considered by these methods is the approximation of the functions. From a mathematical point of view, these methods can be used to efficiently approximate a function, derived from the universal approximation theorems, stating that a neural network with a single hidden-layer can approximate a wide class of functions [5,6]. In fact, unlike the classical methods for approximating functions which are additive, the neural network approach is a compositional method in which a complicated function is obtained by combining several simple functions.

Neural networks and deep learning methods can also be used for solving various types of differential equations which have an arbitrary dimension and geometric complexity [7–12]. Differential equation models are one of the most important tools for modeling many problems in nature as dynamic problems. Actually, the problems occur in a wide range of physical sciences, engineering, neuroscience, financial sciences, etc., can be turned into one of the dynamic models such as partial differential equations (PDEs) or integral differential equations (IDEs) [13,14]. Recently, this field has received a lot of attention and various networks have been introduced for solving these models [8,12,15]. Moreover, the fractional models of differential equations have been widely applicable to introduce the better models for engineering and sciences problems [16–18]. Finance modeling [19–21], biological systems [22–25], physical and chemical processes [26–29] are various fields of fractional dynamic models. Fractional differential equations are obtained by replacing integer-order derivatives of differential equations with fractional-order derivatives. The two most common types of definition of fractional derivative are Riemann-Liouville and Caputo, which differ in the order of evaluation [30]. Solving fractional problems is one of the main topics of recent research and researchers introduced various methods to solve approximately these problems [31–33].

* Corresponding author.
E-mail addresses: Z_Hajimohammadi@sbu.ac.ir (Z. Hajimohammadi), f.baharifard@ipm.ir (F. Baharifard), aghodsib@uwaterloo.ca (A. Ghodsi), k_parand@sbu.ac.ir (K. Parand).

https://doi.org/10.1016/j.chaos.2021.111530
0960-0779/© 2021 Elsevier Ltd. All rights reserved.

Nomenclature

$T_n(\eta)$	n-th Chebyshev polynomial
$\chi(\varphi)$	Differential integral equation in general
$\kappa(\varphi)$	Initial or boundary condition in general
φ	Solution of the equation
AD	Automatic Differentiation
CP	Chebyshev Polynomials
FCDNN	Fractional Chebyshev Deep Neural Network
FFIE	Fractional Fredholm Integral Equation
FPDE	Fractional Partial Differential Equation
F	Fractional
MSE	Mean Square Error
NF	Non-Fractional
NN	Neraul Network
FFN/$\mathcal{FFN}$	Feed Forward Network
ObjFun	Objective Function
PINN	Physics Informed Neural Network
RCP	Root of Chebyshev Polynomial
RN/$\mathcal{RN}$	Residual Network
α	Fractional derivative order
Γ	Gamma function
${}_0^cD^\alpha$	Caputo fractional order differential operator
${}_0^cD_\tau^\alpha$	Caputo fractional order differential operator with respect to τ
D^p	Differential operator of order p
$y \in C_\zeta$	Continuity of the y function in ζ domain
$y \in C_\zeta^m$	Continuity of the y function and its derivatives up to the m-th order in ζ domain
δ	Hyperbolic tangent function
$\mathcal{A}_i$	Layers of the network
$\mathcal{B}_i$	Bias parameter of the network
$\mathcal{R}_i$	Residual function
$\mathcal{R}_{\text{IE}}$	Residual of integral equation
$\mathcal{R}_{\text{PDE}}$	Residual of partial differential equation
$\mathcal{W}_i$	Weight parameter of the network
φ_p	Predicted value of the function
φ_t	Target value of the function
$K(x, s, \varphi(s))$	Kernel of Fredholm equation
f, λ, a, b	Function/parameter of Fredholm equation
w_i	Chebyshev quadratic weight
s_i	Chebyshev quadratic point
m_i	Number of basis functions in Gaussian integration
$\bar{x}$	Vector of x_i
$\Delta\tau$	$(\tau_j - \tau_{j-1})$
φ^j	$\varphi(\bar{x}, \tau_j)$
∇	$\sum_i \frac{\partial^2}{\partial x_i^2}$
$\psi(\varphi)$	Nonlinear/linear differential operator in telegraph equation
F, ξ, ρ	Function/parameter of telegraph equation

The purpose of this paper is to present a new algorithm for solving a wide category of fractional models including fractional partial differential and fractional integral equations. We consider a combination of deep neural network (as a machine learning technique), Chebyshev spectral method (as a numerical method) and power of fractional calculus. Our method can be introduced as a new class of scientific machine learning approaches for solving the fractional problems in the standard differential form or in the integral form. To show the power of our method, we will apply it to the two famous fractional problems named Fredholm and telegraph equations and compare our result with some other methods.

In the next section, we bring a summary of related works that corresponding to the numerical method for solving differential models, learning method applied to differential equations and fractional Fredholm/telegraph equations, respectively and finish the section with describing of our result in details.

2. Related works

In this section first we present a brief history of research that are corresponding to our presented model in three parts: numerical methods, learning methods and fractional partial/Integral equations. Then we explain our method by categorizing its highlights.

2.1. Numerical methods

The development of various theoretical and numerical algorithms for solving differential equations (DEs) has been studied and researched for a long time. The proposed methods can be divided into general categories including analytical, semi-analytical and numerical methods. In the analytical and semi-analytical methods, we can mention variational iteration method (VIM) [34], homotopy analysis method (HAM) [35] and Adomian decomposition method [36] which have been used to solve various problems such as partial and fractional differential equations. However, some equations cannot be solved explicitly and so numerical methods must be used to approximate them. The numerical solutions, as a long-term challenge, have led to the presentation of various methods. Many of these methods fall into two categories: mesh-based and meshfree methods. The mesh-based methods include the finite difference methods (FDM) [37], the finite element methods (FEM) [38] and finite volume methods (FVM) [39]. And some common meshfree methods are smoothing particle hydrodynamic (SPH) [40], Meshless local Petrov-Galerkin (MLPG) [41] and Kansa method (KM) [42]. Also, Runge-Kutta methods [43] and spectral methods [44] are the efficient numerical methods. Runge-Kutta method is one of the most famous numerical methods which is considered as two types of implicit and explicit iterative methods [45]. Spectral methods have shown very successful results in the numerical solution of various types of DEs. Their main appeal relies on their superior rate of convergence for sufficiently smooth functions. The collocation and Tau approaches are kinds of classical spectral method, in which orthogonal functions such as Chebyshev and Legendre polynomials are usually used as the basis of these methods [46,47]. Extensive research has also recently been applied to these methods to solve variety of more complex types of equations such as higher-order problems [48,49], integro-differential equations [50,51] fractional differential equations [52–54].

2.2. Learning methods

Artificial intelligence methods such as neural networks have recently been used to solve differential equations. One of the classic attempts to solve partial differential equations by the neural networks was studied in [55]. Sun et al. [56] introduced the Bernstein neural network for solving PDEs which the network is applied extreme learning machine algorithm to set the parameters of the Bernstein network. Chen et al. [57] applied the auto-reservoir neural network (ARNN) to compute the data predictions on time series. Another method is based on using the orthogonal polynomials as activation functions in the neural network to have orthogonal neural networks [58]. In these networks, the gradient descent method was applied as optimization method. But in some other networks, meta-heuristic optimizers have been used, which has led to having meta-heuristic networks to solve differential equations [59]. Another approach is related to the support vector machine (SVM) method. This approach is based on regression and minimizing the mean squared errors between the approximated function and the training data and applied to some differential equations [60]. Moreover, least square support vector machine (LS_SVM) method was developed. In this method quadratic programming of the optimization problem converted to a system of linear equations which leads to improve the accuracy of the approximation solutions of DEs [61,62]. Recently this method is used to solve some problems, like the time-fractional sub diffusion model on a semi-infinite domain [63] and fractional Volterras population model [64].

More recently, another approach for solving differential equations has emerged, which relies on deep neural networks [65]. These networks are constructed based on minimizing the residual of the differential equations to approximate the solutions of many types of equations such as linear/nonlinear and fractional differential equations [66]. These deep networks with many layers work well in complex data set modeling and can solve high-dimensional problems [67]. The physics informed neural networks (PINN) is one of these networks, which introduces a new way to find solutions for complex dynamic systems using automatic differentiation and also uses the proposed network to discover differential equations [10]. Following this, several networks were proposed that extend PINN to examine other types of equations. For example fractional PINNs (fPINNs) solves space-time fractional equations by applying a hybrid approach of automatic differentiation and numerical discretization [11] and DeepXDE is a network in which a new residual-based adaptive refinement (RAR) method used to improve the training efficiency of PINNs in solving forward and inverse problems with complex-geometry domains [12]. Moreover, the bayesian physics-informed neural network (BPINN) [68], variational physics-informed neural networks (VPINN) [7] and several other networks are presented in other articles, e.g. [69–71] among many other references. Furthermore, the activation function and its effect on network performance have been investigated by some researches [72,73] and very recently the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs are studied in [74].

2.3. Fractional telegraph/Fredholm equations

The telegraph problem has a variety of applications, including signal analysis, propagation and transmission of electrical signals and viscous,uid, etc [75,76]. Recently, various numerical methods have been proposed to solve telegraph equations, which are even in fractional order. The authors of [77] solved the space-fractional telegraph equation by homotopy analysis transform method (HATM) and Chen et al. [78] investigated the time-fractional telegraph equation by separating variables method. Hosseini et al. [79] presented the radial basis functions for solving a type of time-fractional telegraph equation which defined by Caputo order definition. Dehghan et al. [80] solved the one-dimensional hyperbolic telegraph equation using collocation points and splines radial basis function and Heydari et al. [81] applied Legendre wavelets method to solve this fractional problem. This fractional equation was also solved by tau approximation method [82,83] and Eltayeb et al. [84] studied the linear and nonlinear fractional telegraph equations by natural transform decomposition method (NTDM) which is a combined form of the natural transform and the Adomian decomposition methods. More recently, local meshless method [85], modified extended cubic *B*-spline function [86], transcendental Bernstein series [87], wavelet technique [88], finite difference method [89], neural network method [90] and another methods were applied to solve the fractional telegraph equations.

Fredholm problem, first introduced by Erik Ivar Fredholm, also has wide applications in science, physics, and chemistry [91]. The fractional type of integral Fredholm problem has been studied by many researchers. For example, Zhu et al. [92] solved this problem by using the second kind Chebyshev wavelet method via operational matrix of fractional integration. Sharma et al. [93] applied a collocation method with Legendre polynomials to solve the generalized fractional integro-differential equations. Doha et. al [94] used Shifted JacobiGauss-collocation method for solving these problems. Recently, analysing the convergence of the Chebyshev Legendre spectral method in solving Fredholm fractional integro-differential equations was investigated in [95] and using Lucas wavelets (LWs) and the Legendre Gauss quadrature rule to solve the fractional FredholmVolterra integro-differential equations was presented in [96].

2.4. Our results

In this paper, we introduce the Fractional Chebyshev Deep Neural Network (FCDNN) for solving differential models includes fractional Fredholm equations and the extension of time order fractional partial differential equations. Fractional Fredholm equations are first approximated by the numerically Gaussian quadrature method and then are solved using FCDNN. Fractional telegraph equations are first discretized by the marching in time and finite difference methods and then are solved using FCDNN. FCDNN is a deep neural network, consists of a pair of networks, $\mathcal{FFN}$ and $\mathcal{RN}$; the first network produces the approximation of the solution and the latter is applied in order to generate the residual function based on a differential model. Chebyshev polynomials are introduced as the activation functions of $\mathcal{FFN}$. The operators such as derivatives, integrals and functions are used as nodes in $\mathcal{RN}$. Automatic differentiation and Guassian integration make network performance better. The fractional telegraph PDEs and fractional Fredholm IEs are implemented on FCDNN to verify the efficiency of this network. The results show that FCDNN can obtain the solution of fractional differential models with high accuracy.

The main contributions of this article are as follows.

- Introducing the Fractional Chebyshev Deep Neural Network (FCDNN) for solving fractional differential models.
- Solving fractional different models (fractional Fredholm equations, fractional Telegraph equations) via a combination of deep learning and Chebyshev collocation method.
- Chebyshev orthogonal polynomials are applied as activation functions in the structure of the FCDNN.
- FCDNN is a new and efficient numerical learning method to solve nonlinear/complex fractional differential equations.
- The spectral, the time marching and Gaussian methods as efficient techniques in scientific computation are used in the network to improve the accuracy and efficiency of the proposed method.

The rest of this article is organized as follows. Section 3 deals with the preliminaries. Section 4 describes the structure of our deep network and in Section 5 main algorithm for solving fractional partial/integral differential models is presented. Section 6 is devoted to some examples which the proposed method applied to them and finally Section 7 concludes this article.

3. Preliminaries

3.1. Differential models

The general form of a differential model is in the following form:

$$\begin{aligned} &\chi(\varphi(\bar{x}),\bar{x}) = \beta(x) \ \ or \\ &\chi(\varphi(\bar{x},\tau),\bar{x},\tau) = \beta(x), \\ &\bar{x} = (x_1, x_2, \cdots). \end{aligned} \tag{1}$$

$$\kappa(\varphi) = \begin{Bmatrix} \kappa_1(\varphi) = k_1, \\ \kappa_2(\varphi) = k_2, \dots, \\ \kappa_M(\varphi) = k_M \end{Bmatrix} \tag{2}$$

where χ is the functions of the operators consist of the derivatives or integrals. Eq. (2) is presented the initial or boundary conditions. For example consider a telegraph equation in the following form:

$$\begin{aligned}
&{}_0^cD_\tau^2\varphi(\bar{x},\tau) + 2\,{}_0^cD_\tau\varphi(\bar{x},\tau) + 3\varphi(\bar{x},\tau) - \nabla\varphi(\bar{x},\tau) = 0,\\
&x_i \in [0,1],\ \bar{x} = (x_1,x_2),\ \tau \in [0,1],\\
&\varphi(\bar{0},\tau) = 0,\\
&\varphi(\bar{1},\tau) = e^{-\tau}\sinh(1)\sinh(1),\\
&\varphi(\bar{x},0) = \sinh(x_1)\sinh(x_2),\\
&\frac{\partial\varphi(\bar{x},0)}{\partial\tau} = -\sinh(x_1)\sinh(x_2).
\end{aligned}$$

where $\chi(\varphi(\bar{x},\tau)) = {}_0^cD_\tau^2\varphi(\bar{x},\tau) + 2{}_0^cD_\tau\varphi(\bar{x},\tau) + 3\varphi(\bar{x},\tau) - \nabla\varphi(\bar{x},\tau)$, $\beta(x) = 0$ and $\kappa(\varphi) = \{\varphi(\bar{0},\tau) = 0, \varphi(\bar{1},\tau) = e^{-\tau}\sinh(1)\sinh(1), \varphi(\bar{x},0) = \sinh(x_1)\sinh(x_2), \frac{\partial\varphi(\bar{x},0)}{\partial\tau} = -\sinh(x_1)\sinh(x_2)\}$. Moreover, the residual functions are introduced as follows:

$$\begin{aligned}
&\mathcal{R}_1 = \chi(\varphi(\bar{x}),\bar{x}) - \beta(x) \quad or \quad \mathcal{R}_1 = \chi(\varphi(\bar{x},\tau),\bar{x},\tau) - \beta(x),\\
&\qquad\qquad \bar{x} = (x_1,x_2,\cdots),\\
&R_2 = \kappa(\varphi) - \{\kappa_1(\varphi) = k_1, \kappa_2(\varphi) = k_2, \cdots, \kappa_M(\varphi) = k_M\}.
\end{aligned}$$

The purpose of this paper is to set a network and its parameters to obtain the minimum value for the residual functions and to learn a differential model by considering the target values. This network could be applied to predict the other values in a domain and obtain the general solution.

3.2. Chebyshev polynomials

Chebyshev polynomials (CPs) of the first kind [97], denoted by $T_n(\eta)$, are defined as:

$$T_n(\eta) = \cos(n\theta), \quad \eta = \cos(\theta) \in [-1,1]. \tag{4}$$

The CPs could be defined by the recursive formula in the following form:

$$\begin{aligned}
&T_{n+1}(\eta) = 2\eta T_n(\eta) - T_{n-1}(\eta), \quad n \ge 1,\\
&T_0(\eta) = 1,\ \ T_1(\eta) = \eta.
\end{aligned} \tag{5}$$

The CPs are orthogonal polynomials with weighted function $w = \frac{1}{\sqrt{1-\eta^2}}$ in $[-1,1]$ domain. The orthogonal relation of CPs is defined as follows:

$$\int_{-1}^{1} T_n(\eta)T_m(\eta)w(\eta)\mathrm{d}\eta = \gamma\delta_{n,m}, \tag{6}$$

where $\gamma = \frac{\pi c_i}{2}$, $c_0 = 2$, $c_i = 1, i \ge 1$ and $\delta_{n,m}$ is a delta Kronecker function.

The CP of degree n has n real roots. The roots of CPs are formulated as:

$$\eta_k = \cos(\frac{(2k-1)\pi}{2n}), \quad k = 1,2,\cdots n. \tag{7}$$

CPs have the following useful properties:

$$\begin{aligned}
&T_{m+n}(\eta) + T_{|m-n|}(\eta) = 2T_n(\eta)T_m(\eta),\\
&T_m(T_n(\eta)) = T_n(T_m(\eta)) = T_{mn}(\eta), \forall n,m \in N.
\end{aligned} \tag{8}$$

In our method, CPs will be applied as activation functions because they can approximate the functions. They are differentiable, stable, adjustable with the number of network parameters and they have high predictive efficiency.

3.3. Fractional calculations

In this part, we will focus on the some definitions and a lemma related to the fractional calculation used in our method.

Definition 1. Suppose $t > 0$ and $y(t)$ be a real function. $y(t)$ is in space C_ζ, $\zeta \in \mathbb{R}$, if there is a real number $\varrho > \zeta$ such that $y(t) = t^\varrho y_1(t)$ and $y_1(t) \in C(0,\infty)$. [98]

Definition 2. Suppose $t > 0$ and $y(t)$ be a real function. $y(t)$ is in space C_ζ^m if and only if $y^{(m)} \in C_\zeta$, $m \in \mathbb{N}$ where $y^{(m)}$ the m-th order derivative of $y(t)$. [98]

Definition 3. By applying the RiemannLiouville fractional integral operator of order α, can have the Caputo fractional derivative for $y(t)$ where $t > 0$ as [98,99]:

$${}_0^cD^\alpha y(t) = \begin{cases} \frac{1}{\Gamma(p-\alpha)}\int_0^t (t-s)^{p-\alpha-1}D^py(s)\mathrm{d}s, & \alpha > 0\\ \frac{\partial^p y(t)}{\partial t^p}, & \alpha = p \end{cases} \tag{9}$$

for $p-1 < \alpha \le p$ where $p \in \mathbb{N}$ and is the smallest integer greater than α and also $y \in C_{-1}^p$.

Four important properties of the operator ${}_0^cD^\alpha$ are provided in the following cases for $y \in C_\zeta$, $\zeta \ge -1$, α, $\rho \ge 0$, $\gamma \ge -1$, $a_j \in \mathbb{R}$ and constant C:

(I) ${}_0^cD^\alpha C = 0$,

(II) ${}_0^cD^\alpha\, {}_0^cD^\rho y(t) = {}_0^cD^{\alpha+\rho}y(t)$,

(III) ${}_0^cD^\alpha t^\gamma = \begin{cases} 0, & \gamma \in \mathbb{N}\cup\{0\} \text{ and } \gamma < \lceil\alpha\rceil\\ \frac{\Gamma(\gamma+1)}{\Gamma(\gamma-\alpha+1)}t^{\gamma-\alpha}, & \gamma \in \mathbb{N}\cup\{0\} \text{ and } \gamma \ge \lceil\alpha\rceil, \\ & \text{or } \gamma \notin \mathbb{N} \text{ and } \gamma > \lfloor\alpha\rfloor \end{cases}$

(IV) ${}_0^cD^\alpha(\sum_{j=1}^n a_jy_j(t)) = \sum_{j=1}^n a_jD^\alpha y_j(t)$.

Theorem 1. *If $y(t) \in C[0,\eta]$ and $D^{l\alpha}y(t) \in C[0,\eta]$ for $l = 0,1,.,n$, $0 < \alpha < 1$ and $\eta > 0$, the following expansion is established for $y(t)$:*

$$\begin{aligned}
&y(t) = \textstyle\sum_{i=0}^{n-1}\frac{t^{i\alpha}}{\Gamma(i\alpha+1)}D^{i\alpha}y(0^+) + \frac{t^{n\alpha}}{\Gamma(n\alpha+1)}D^{n\alpha}y(\xi),\\
&s.t.\ \xi \in [0,t], \forall t \in [0,\eta].
\end{aligned} \tag{10}$$

So, we have

$$|y(t) - \sum_{i=0}^{n-1}\frac{t^{i\alpha}}{\Gamma(i\alpha+1)}D^{i\alpha}y(0^+)| \le N_\alpha\frac{t^{n\alpha}}{\Gamma(n\alpha+1)}, \quad s.t.\ N_\alpha \ge |D^{n\alpha}y(\xi)|. \tag{11}$$

Proof. See Ref. [100]. □

The above theorem relates to generalized version of Taylors series and if considered α equals to 1, the classical Taylor's formula obtained.

Theorem 2. *Let us suppose $\psi'(t) \in C^2[0,T]$, $\alpha \in [1,2]$ and $\Delta t = \frac{T}{n+1}$, it holds that*

$$\begin{aligned}
&\int_0^{t=t_{n+1}}\frac{\psi'(s)}{(t_{n+1}-s)^{\alpha-1}}\mathrm{d}s\\
&= \sum_{j=0}^{n}\frac{\psi(t_{j+1})-\psi(t_j)}{\Delta t}\int_{t_j}^{t_{j+1}}(t_{n+1}-s)^{1-\alpha}\mathrm{d}s + R^n,\ 1 \le n+1 \le N
\end{aligned} \tag{12}$$

and

$$|R^n| \le \left(\frac{1}{2(2-\alpha)} + 0.5\right)\Delta t^{3-\alpha}\max_{0\le t\le t_{n+1}}|\psi''(t)|. \tag{13}$$

Proof. See Ref [85]. □

4. Structure of our network

In this section, we present a new deep neural network, namely FCDNN for solving fractional partial differential equations (FPDEs) and fractional Fredholm integral equations (FFIEs). The aim of FCDNN is to present a new approach which it takes advantages of deep learning as well as spectral methods to obtain a better performance in solving differential models.

Recently, the use of deep networks to solve equations has expanded. PINN [71], DeepXDE [12], fPINN [11], ResNet [101] and etc. are all deep networks used to solve equations. PINN [71] is a main popular neural network that is used to solve a class of nonlinear partial differential equations which are introduced by the laws of physics models. In PINN, a neural network is constructed as an approximation of the solution of the equation. Then the training set is specified which includes two important classes of the points for the points on domain of the equation and the points of the initial or boundary conditions of the equation. In the next step the special function which is named the loss function, is determined. The loss function is consist of the summation of the L^2 norm errors on the residuals of the equation and the initial/ boundary conditions. Finally, the evaluation parameter, for example the L^2 norm errors, is determined to minimize the loss function and search the proper parameters of network. This procedure is named "train network". The minimization of the loss function is performed by applying the Adam [102] and L-BFGS [103] methods. Is is possible the train network is performed the several times to achieve the proper parameters of network because the loss function is the nonlinear and non convex.

In this paper, a deep neural network is combined with the Chebyshev collocation method and fractional calculus. Collocation method is a type of spectral methods to solve differential equations. This method approximates the solution of equations by expansion of basic functions such as Chebyshev polynomials [97]. This approximation includes the unknown coefficients which are obtained by replacing the collocation points in the residual functions. The collocation points are selected so that the solution satisfies the equation in these points and minimizes the residual function [97].

FCDNN consists of a pair of consecutive networks. The first network is a feed forward network (FFN) with different activation functions includes Chebyshev polynomials of different degrees. This network approximate the solution of the equation. The use of the Chebyshev polynomials lead to approximate the solution of the equation by some orthogonal bases, similar to the collocation method. The nodes of the second network usually are operations such as derivatives, integrals and functions related to the model. The output of the second network is the residual function of the model. The Chebyshev Gaussian quadrature method and the marching in time technique are used for the discretization of the fractional models in the second network. Indeed, FCDNN is an extension of both PINN network and Chebyshev collocation method and has a new numerical approach to solve some of the fractional models.

Before explaining about the detail of the network, we bring a brief comparison between FCDNN, PINN and Chebyshev collocation methods which is presented in Table 1. We have compared these three methods based on the main functions that are effective in calculating the answer, general method, unknown parameters, solving method, type of training points, general and specific type of equations that they can solve.

Now, we present the detail of our network. In our network, we use deep FFN because it is proper for our considered models. However, it is possible to apply the other types of networks such as recurrent and convolutional neural networks, whichever is more appropriate for the model. Moreover, Chebyshev polynomials and frequently used functions such as hyperbolic tangent are applied to FFN as activation functions. Chebyshev's polynomials in the initial layers calculate the equation solution approximation as well as the expansion approximation of the collocation method. Also applying the functions such as hyperbolic tangent creates a more complex basis of the orthogonal space than the orthogonal space based on polynomials. So, assume that $\mathcal{FFN}$ is a $\mathcal{H}$-layer FFN, which is the first network and defined as follows:

$$\mathcal{A}_0 = T_0(\bar{x}), \quad \bar{x} \in R^m,$$
$$A_i = T_i(\mathcal{W}_i\mathcal{A}_{i-1} + \mathcal{B}_i), \quad 1 \le i \le \mathcal{N},$$
$$A_i = \delta(\mathcal{W}_i\mathcal{A}_{i-1} + \mathcal{B}_i), \quad \mathcal{N}+1 \le i \le \mathcal{H}-1,$$
$$A_{\mathcal{H}} = \mathcal{W}_{\mathcal{H}}\mathcal{A}_{\mathcal{H}-1} + \mathcal{B}_{\mathcal{H}}.$$

where $\mathcal{A}_0$ is the input layer with m dimension. $\mathcal{A}_i$, $1 \le i \le \mathcal{H}-1$ are $\mathcal{N}$ hidden layers which have T_i as the Chebyshev activation function and $(\mathcal{H}-\mathcal{N}+1)$ hidden layers which have δ as the hyperbolic tangent activation function. $\mathcal{W}_i$ and $\mathcal{B}_i$, $1 \le i \le \mathcal{H}$ are the weight and bias parameters, respectively. Indeed the weight and bias parameters are similar to the unknown coefficients in the collocation method which the calculating of these parameters causes to find the equation solution approximation. $\mathcal{A}_{\mathcal{H}}$ is the output layer produces φ in Eqs. (1) and (2). At this point, the second network, namely $\mathcal{RN}$ is applied to obtain the differential model by using some operators. This network produces the residual function of the equation. Automatic differentiation (AD) [104], Chebyshev Gaussian integration [97] and fractional calculations techniques are applied to calculate the operator nodes. For example, if there is a second derivative in the equation ($\frac{\partial^2\varphi}{\partial x_i^2}$), an operational derivative node is defined to calculate the desired derivative of φ_p (output of the first network) by applying AD technique. In the same way, other operational nodes help to finally form the residual function of the equation. This technique is used to obtain the efficient scientific computations and to make a better structure of differential models. It is notable that applying AD technique causes the network to be more compatible with the constraints of the model such as invariances, conservation principles, any symmetries and etc. The architecture of FCDNN is presented in Fig. 1.

Table 1
A brief comparison between FCDNN, PINN and Chebyshev collocation methods. FCDNN can apply to solve the fractional/non-fractional models. Training points of FCDNN can be chosen from both scattered and adaptive (randomly distributed Gaussian) points. (CP=Chebyshev Polynomials, NN= Neural Network, RCP= Roots of Chebyshev Polynomials, F= Fractional, NF= Non-Fractional).

	FCDNN	PINN	Chebyshev Collocation
Basic functions	CP + NN	NN	CP
Method	nonlinear + mesh-free	nonlinear + mesh-free	nonlinear + mesh
Parameters	weights and biases	weights and biases	coefficients of polynomials
Embedding the solution	optimize objective function	optimize objective function	solve the system of equations
Training points	scattered + adaptive	scattered	RCP
Models	F + NF	NF	F + NF
Equations	FPDEs/PDEs + FIEs/IEs	PDEs	FPDEs/PDEs + FIEs/IEs

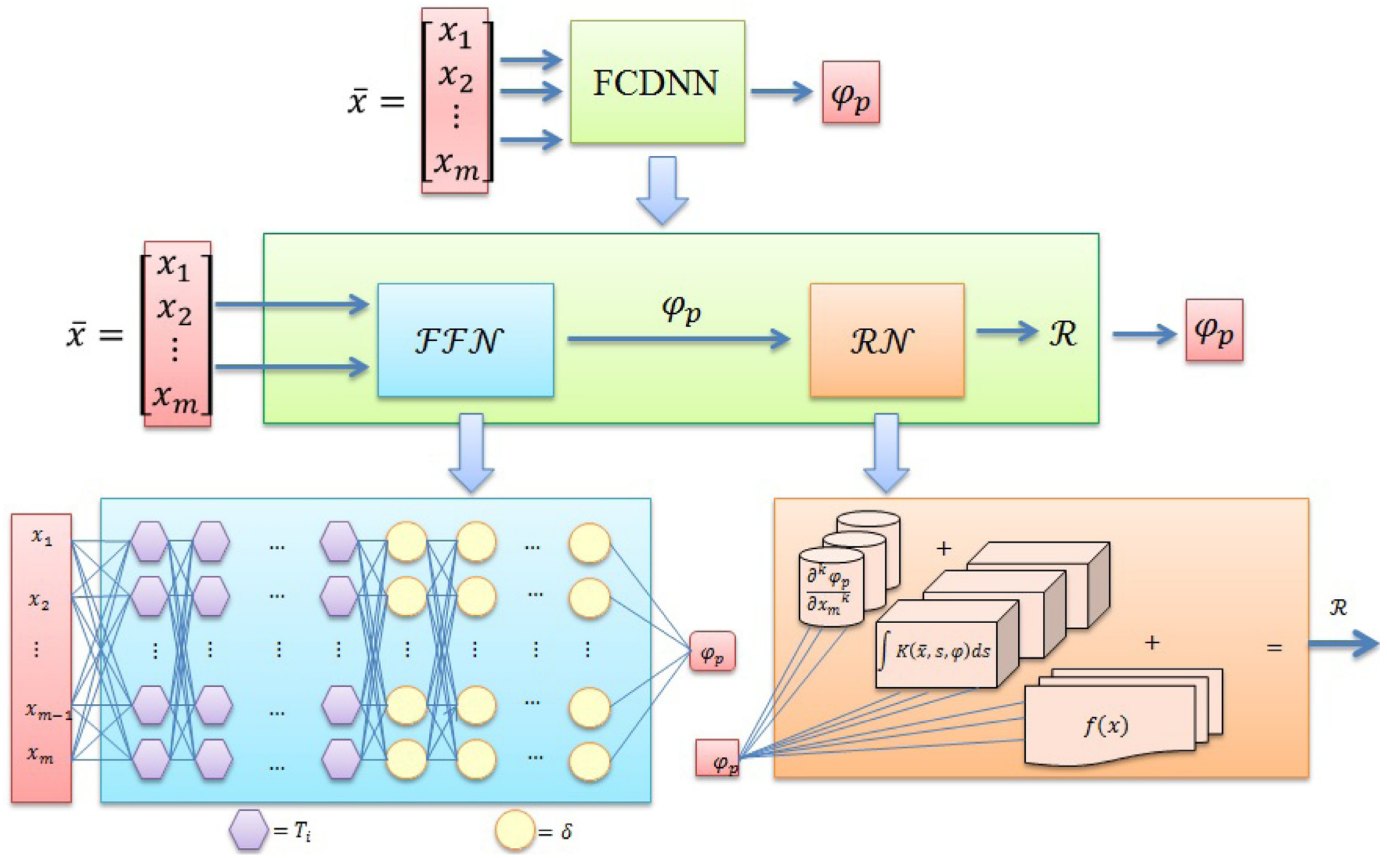

$\bar{x}=[x_1, x_2, \vdots, x_m]$
FCDNN
φ_p
$\mathcal{FFN}$
$\mathcal{RN}$
$\mathcal{R}$
$\frac{\partial^k \varphi_p}{\partial x_m{}^k}$
$\int K(\bar{x}, s, \varphi)ds$
$f(x)$
+
=
$= T_i$
$= \delta$

Fig. 1. The architecture of FCDNN. The output of the $\mathcal{FFN}$ network is an approximation of φ. The first $\mathcal{N}$-layers have Chebyshev activation functions. Indeed, i-th-layer has T_i as the activation function. The later layers have δ as an activation function such as hyperbolic tangent. The residual function of the differential model is obtained by the output of the $\mathcal{RN}$. The nodes of $\mathcal{RN}$ network are operators such as derivatives, integrals and functions related to the model.

As mentioned before, the output of $\mathcal{FFN}$ network is an approximation of φ and the presence of $\mathcal{RN}$ network leads to obtain the proper φ which has been satisfied in the differential model. In the next step, we need to define an appropriate objective function to obtain the solution of the equation by minimizing the objective function and optimizing the values of the network parameters. Therefore, the objective function of the network is defined as follows:

$$\text{ObjFun} = \min(\varphi_t - \varphi_p) + \min(\mathcal{R}_1) + \min(\mathcal{R}_2), \quad (14)$$

where φ_t is a target value and φ_p is a predicted value of the FCDNN. The minimization of $\mathcal{R}_1$ calculates the proper residual function which has the least possible value and the minimum value of $\mathcal{R}_2$ satisfies the initial/boundary conditions for models. The mean squared errors (MSE) over the training data set are considered to minimize the objective functions (ObjFun). Therefore the objective function can be convert to the following form:

$$\text{ObjFun} = \text{MSE}(\varphi_t - \varphi_p) + \text{MSE}(\mathcal{R}_1) + \text{MSE}(\mathcal{R}_2), \quad (15)$$

where the MSE is calculated as follows:

$$\text{MSE}(f) = \frac{1}{N}\sum_{i=1}^{N}(f_i)^2, \quad (16)$$

where N is the size of the training data set and f_i is the value of f on the i-th member of the training data set. The main method for minimizing is Adam algorithm [102], while the L-BFGS method [103] as a quasi-Newton method is applied to obtain a better solution. Adam algorithm is calculated the initial values for L-BFGS method and L-BFGS method applied the second order of the derivatives and caused to be the faster convergence.

In summary, $\mathcal{FFN}$ network obtained an expansion of the solution and $\mathcal{RN}$ network obtained the residual function. The points of the training data set are similar to collocation points and the network parameters are as unknown coefficients in collocation method. So, optimizing the parameters of FCDNN network leads to obtain the solution of the model. The fractional Fredholm integral equations and a type of fractional PDE are investigated and learned by FCDNN in this research and in the next section the algorithms for solving them are described in detail.

5. Algorithm of our method

In this section we bring the general form of fraction Fredholm/telegraph problems and present the main algorithm for solving these problems with the help of our proposed network which was explained in the previous section.

5.1. Fractional fredholm integral equations and FCDNN

Fractional Fredholm integral equations have the general form as follows [91]:

$${}_0^cD^\alpha\varphi(x) = f(x) + \lambda\int_a^b K(x, s, \varphi(s))\mathrm{d}s, \quad 0 \le \alpha \le 1 \quad (17)$$

where the function $f(x)$, the kernel $K(x, s, \varphi(s))$ and the parameter λ are given; the constants of a, b are fixed and the aim is to find the function $\varphi(x)$. When the $\alpha \neq 0$, the initial condition is defined $\varphi(a) = \varphi_0$. This model is a fractional inhomogeneous Fredholm equation of the second kind. By considering Gaussian and deficit calculations, we will have the following results:

$$\begin{cases} \varphi(x) = f(x) + \lambda \int_a^b K(x,s,\varphi(s))ds, & \alpha = 0, \\ \frac{1}{\Gamma(1-\alpha)} \int_0^x \frac{\partial\varphi}{\partial\varepsilon} \frac{d\varepsilon}{(x-\varepsilon)^\alpha} = f(x) + \lambda \int_a^b K(x,s,\varphi(s))ds, & 0<\alpha<1, \\ \frac{\partial\varphi}{\partial x} = f(x) + \lambda \int_a^b K(x,s,\varphi(s))ds, & \alpha = 1. \end{cases} \quad (18)$$

The output of $\mathcal{FFN}$ is $\varphi(x)$ and then the output of the second network is $\mathcal{R}_{\text{IE}}$ function as the following form [97]:

$$\mathcal{R}_{\text{IE}} = \begin{cases} \varphi(x) - f(x) - \alpha \int_a^b K(x,s,\varphi(s))ds, & \alpha = 0, \\ \frac{1}{\Gamma(1-\alpha)} \int_0^x \frac{\partial\varphi}{\partial\varepsilon} \frac{d\varepsilon}{(x-\varepsilon)^\alpha} - f(x) + \lambda \int_a^b K(x,s,\varphi(s))ds, & 0<\alpha<1, \\ \frac{\partial\varphi}{\partial x} - f(x) + \lambda \int_a^b K(x,s,\varphi(s))ds, & \alpha = 1. \end{cases} \quad (19)$$

Integral operator is approximated by Chebyshev Gaussian quadrature method and therefore, have the following results:

$$\int_a^b K(x,s,\varphi(s))ds \approx \sum_{i=0}^{m_1} w_i K(x,s_i,\varphi(s_i)), \quad (20)$$

$$\int_0^x \frac{\partial\varphi}{\partial\varepsilon} \frac{d\varepsilon}{(x-\varepsilon)^\alpha} \approx \sum_{i=0}^{m_2} w_i \frac{\partial\varphi}{\partial\varepsilon}\Big|_{\varepsilon=s_i} (x-s_i)^{-\alpha}, \quad (21)$$

where m_i, w_i and s_i are number of basis functions in Gaussian integration, Chebyshev quadrature weights and Chebyshev quadrature points, respectively [97] in a desire domain [97]. It is remarkable that utilizing the $\mathcal{R}_{\text{IE}}$ satisfied the residual functions in Eq. (3) for FFIEs. Algorithm 1 represents the steps of application of FCDNN for solving fractional Fredholm integral equations.

Algorithm 1 Applying FCDNN for Solving FFIEs.

```
input: Training data set
output: $\varphi_p$, MSE$_{train}$ and MSE$_{test}$
1: Shift the training data set to a specific domain
2: Specify the structure of the network
3: Construct the $\mathcal{FFN}$ and initial the parameters
4: Construct the $\mathcal{RN}$ based on $\mathcal{R}_{\text{IE}}$
5: Instance a new sample of FCDNN as the model
6: Train the model by applying Adam and L-BFGS methods
7: Calculate MSE$_{train}$
8: Specify a test data set and predict the model on it
9: Calculate MSE$_{test}$
10: Return $\varphi_p$, MSE$_{train}$ and MSE$_{test}$
```

5.2. Fractional partial differential equations and FCDNN

A general type of fractional partial differential equations is considered in this research which has the following form:

$${}_0^cD_\tau^\alpha \varphi(\bar{x},\tau) + \xi\, {}_0^cD_\tau^{\alpha-1} \varphi(\bar{x},\tau) = \psi(\varphi(\bar{x},\tau)), \quad \bar{x} = (x_1,x_2,\cdots) \quad (22)$$

where ${}_0^cD_\tau^\alpha$ is a Caputo fractional derivative of order α with respect to τ, $\alpha \in [1,2]$, ξ is the constant coefficient of equation and ψ is a nonlinear/linear differential operator. This equation has initial/boundary conditions similar to Eq. (2). This equation first discretized by the approximation of mentioned formula in Section 3.3 and central explicit finite difference method [37]:

$${}_0^cD_\tau^\alpha \varphi = \begin{cases} \frac{1}{\Gamma(1-\alpha)} \int_0^\tau \frac{\partial\varphi}{\partial s} \frac{ds}{(\tau-s)^\alpha}, & \alpha \in [0,1] \\ \frac{1}{\Gamma(2-\alpha)} \int_0^\tau \frac{\partial^2\varphi}{\partial s^2} \frac{ds}{(\tau-s)^{\alpha-1}}, & \alpha \in [1,2] \end{cases} \quad (23)$$

$$\frac{\partial^2\varphi}{\partial\tau^2} = \frac{\varphi^{n+1} - 2\varphi^n + \varphi^{n-1}}{(\Delta\tau)^2}, \quad (24)$$

$$\frac{\partial\varphi}{\partial\tau} = \frac{\varphi^{n+1} - \varphi^{n-1}}{(2\Delta\tau)}, \quad (25)$$

$$\frac{1}{\Gamma(2-\alpha)} \int_0^{\tau=\tau_{n+1}} \frac{\partial^2\varphi}{\partial s^2} \frac{ds}{(\tau_{n+1}-s)^{\alpha-1}} = \frac{1}{\Gamma(2-\alpha)} \sum_{j=0}^{n} \frac{\varphi^{j+1} - 2\varphi^j + \varphi^{j-1}}{(\Delta\tau)^2} \int_{\tau_j}^{\tau_{j+1}} (\tau_{n+1}-s)^{1-\alpha} ds. \quad (26)$$

$$\frac{1}{\Gamma(2-\alpha)} \int_0^{\tau=\tau_{n+1}} \frac{\partial\varphi}{\partial s} \frac{ds}{(\tau_{n+1}-s)^{\alpha-1}} = \frac{1}{\Gamma(2-\alpha)} \sum_{j=0}^{n} \frac{\varphi^{j+1} - \varphi^{j-1}}{(2\Delta\tau)} \int_{\tau_j}^{\tau_{j+1}} (\tau_{n+1}-s)^{1-\alpha} ds. \quad (27)$$

The mentioned above formulas are substituted in Eq. (22), the following results are obtained:

$$c_1 \sum_{j=0}^{n} (\varphi^{j+1} - 2\varphi^j + \varphi^{j-1})([\tau_{n+1}-\tau_j]^{2-\alpha} - [\tau_{n+1}-\tau_{j+1}]^{2-\alpha}) + \xi c_2 \sum_{i=0}^{n} (\varphi^{i+1} - \varphi^{i-1})([\tau_{n+1}-\tau_i]^{2-\alpha} - [\tau_{n+1}-\tau_{i+1}]^{2-\alpha}) = \psi(\varphi^{n+1}), \quad (28)$$

where $\varphi^i = \varphi(\bar{x},\tau_i)$ is shown a value of φ in τ_i step $0 \le i \le n+1$, $c_1 = \frac{1}{(\Delta\tau)^2\Gamma(3-\alpha)}$ and $c_2 = \frac{1}{2(\Delta\tau)\Gamma(3-\alpha)}$. The main output of $\mathcal{FFN}$ is φ^{n+1} and the output of the second network is $\mathcal{R}_{\text{PDE}}$ function as the following form:

$$\begin{aligned} \mathcal{R}_{PDE} &= \left[(c_1+\xi c_2)\varphi^{n+1} - 2c_1\varphi^n + (c_1-\xi c_2)\varphi^{n-1}\right] \left[(\tau_{n+1}-\tau_n)^{2-\alpha}\right] - \psi\left(\varphi^{n+1}\right) + H^n, \\ H^n &= \sum_{j=0}^{n-1} \left[(c_1+\xi c_2)\varphi^{j+1} - 2c_1\varphi^j + (c_1-\xi c_2)\varphi^{j-1}\right] \left[\left(\tau_{n+1}-\tau_j\right)^{2-\alpha} - \left(\tau_{n+1}-\tau_{j+1}\right)^{2-\alpha}\right]. \end{aligned} \quad (29)$$

It is considerable that using the $\mathcal{R}_{\text{PDE}}$ satisfied the residual function in Eq. (3) for $\mathcal{R}_1$ and utilizing the initial/boundary conditions in form $\mathcal{R}_2$ completed the residual functions. Algorithm 2 represents the steps of application of FCDNN for solving FPDEs.

Algorithm 2 Applying FCDNN for Solving FPDEs.

```
input: Training data set
output: $\varphi_p^{n+1}$, MSE$_{train}$ and MSE$_{test}$
1: Shift the training data set to a specific domain
2: Perform the fractional formulas and finite method on a FPDE
3: Specify $\varphi_p^{n-1}$, $\varphi_p^n$, $\varphi_p^{n+1}$ and $H_n$ on the training data set
4: Specify the structure of the network
5: Construct the $\mathcal{FFN}$ and initial the parameters
6: Construct the $\mathcal{RN}$ based on $\mathcal{R}_{\text{PDE}}$ and $\mathcal{R}_2$
7: Instance a new sample of FCDNN as the model
8: Train the model by applying Adam and L-BFGS methods
9: Calculate MSE$_{train}$
10: Specify a test data set and predict the model on it
11: Calculate MSE$_{test}$
12: Return $\varphi_p^{n+1}$, MSE$_{train}$ and MSE$_{test}$
```

6. Experiments

We performed several experiments to verify the performance of FCDNN for solving differential models. The error of numerical

Table 2
The FCDNN parameters and error of the numerical results for all the examples. The notation $\ell * k$ indicates that there are ℓ layers with k neurons in each layer.

	Model	α	$\mathcal{H}$-Layers	N_1	MSE $_{train}$	N_2	MSE $_{test}$
FFIE	Example 1	$\alpha = 0$	$[1, 10*10, 30, 20, 10, 1]$	70	1.339266e−06	20	1.501477e−06
	Example 2	$\alpha = 0$	$[1, 10*10, 30, 20, 10, 1]$	70	1.478133e−06	20	1.227377e−05
	Example 3	$0 < \alpha \leq 1$	$[1, 10*10, 30, 20, 10, 1]$	70	1.553263e−06	20	2.227289e−05
FPDE	Example 4	$\alpha = 2$	$[2, 10*10, 4*100, 1]$	1200	6.079306e−06	400	8.718456e−06
	Example 5	$\alpha = 2$	$[2, 10*10, 4*100, 1]$	1200	1.414510e−06	400	2.243810e−06
	Example 6	$\alpha = 2$	$[3, 10*10, 4*100, 1]$	24000	1.816772e−06	8000	3.394147e−06
	Example 7	$\alpha = 1.1$	$[1, 10*10, 100, 50, 10, 1]$	1200	5.727525e − 07	200	1.232703e−06
	Example 7	$\alpha = 1.4$	$[1, 10*10, 100, 50, 10, 1]$	1200	4.164674e − 06	200	1.059138e−05
	Example 8	$\alpha = 1.75$	$[2, 10*10, 4*100, 1]$	1200	1.023663e−05	200	4.266165e−05
	Example 8	$\alpha = 1.95$	$[2, 10*10, 4*100, 1]$	1200	1.015015e−06	200	4.012831e−05

results is calculated as follows:

$$\text{MSE}_{train} = \frac{1}{N_1}\sum_{i=1}^{N_1}(\varphi_t - \varphi_p)^2, \tag{30}$$

where N_1 is the size of training data set.

$$\text{MSE}_{test} = \frac{1}{N_2}\sum_{i=1}^{N_2}(\varphi_t - \varphi_p)^2, \tag{31}$$

where N_2 is the size of test data set. The Training data set consists of the Chebyshev quadrature points up to degree N_1 in a specific domain. Also, test data set includes N_2 randomly sample points in the same specific domain. The error of the numerical results and considered parameters for all the examples, are reported in Table 2. The codes of the examples are written by Tensorflow package of Python version 3.7.0. It is notable that the results of figures have been performed on the test data set. The maximum iteration of Adam algorithm is up to 10000 times and L-BFGS method is repeated until it converges.

Other researchers [75,76,85,91,96] are investigated the following examples as numerical results. Chebyshev collocation method (Cheb CM) and fractional deep neural network (FDNN) have also been run on the several examples and the numerical results are reported in tables to compare the performance of the FCDNN network. It is notable that FDNN is an extended version of the PINN network for solving fractional PDEs and fractional IEs; so, the Chebyshev polynomials are not used in its structure. Also, by comparing the results of FDNN and FCDNN networks, the role of Chebyshev polynomials in the network structure can be evaluated. As the results representation, the use of Chebyshev polynomials improves the results compared to the results of the FDNN network.

6.1. Fractional Fredholm integral equations

Example 1. Inserting $f(x) = -2 - 3x$, $\alpha = 0$, $\lambda = 1$, $a = 0$, $b = 1$ and $K(x, s, \varphi(s)) = (3x + s)\varphi(s)$ into Eq. (17), leads to obtain Fredholm integral model as follows:

$$\varphi(x) = -2 - 3x + \int_0^1 (3x + s)\varphi(s)\mathrm{d}s. \tag{32}$$

The exact solution of this model is $6x$. The obtained solution of it by FCDNN and comparison with the exact solution is illustrated in Fig. 2. FCDNN for this example has one input, 13-hidden layers and one output. The first 10-layers have 10 neurons in each layer. The Chebyshev activation function T_i, $1 \leq i \leq 10$ is considered as the activation function of the i-th layer. The last 3-layers have 30, 20 and 10 neurons with hyperbolic tangent activation function, respectively. $m_1 = 50$ is the number of shifted Chebyshev quadrature points.The training data set includes the Chebyshev quadrature points up to degree 70, which are shifted in [0,1] domain. The size of test data set is 20 which are random points in [0,1] domain.

Table 3
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network and the predicted value φ_p of FCDNN network in several test points on [0,1] domain for Example 1.

x	Exact (φ_t)	Cheb CM	FDNN	FCDNN (φ_p)
0.0	0.0	0.0	0.0	0.0
0.1	0.6	0.599999983	0.599991044	0.599994044
0.2	1.2	1.199999997	1.199991804	1.199995069
0.3	1.8	1.799999997	1.799992687	1.799998977
0.4	2.4	2.399999997	2.399990046	2.399998744
0.5	3.0	2.999999999	3.000005450	2.999998437
0.6	3.6	3.600000001	3.599991107	3.600001579
0.7	4.2	4.200000001	4.199997784	4.200006662
0.8	4.8	4.800000001	4.799991960	4.800001934
0.9	5.4	5.399999998	5.399997512	5.399998999
1.0	6.0	5.999999996	5.999997407	5.999998277

Table 4
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network and the predicted value φ_p of FCDNN network in several test points on [0,1] domain for Example 2.

x	Exact (φ_t)	Cheb CM	FDNN	FCDNN (φ_p)
0.0	1.0	1.000037523	0.999950238	0.999992214
0.1	1.791824698	1.791858981	1.791768938	1.791821491
0.2	2.825540928	2.825523000	2.825481567	2.825535359
0.3	4.220116923	4.220095810	4.220116580	4.220111084
0.4	6.153032424	6.153074397	6.152976903	6.153029673
0.5	8.889056099	8.889056802	8.889053935	8.889053790
0.6	12.82317638	12.82313242	12.82312614	12.82316733
0.7	18.54464677	18.54467304	18.54459525	18.54464445
0.8	26.93253020	26.93255451	26.93251096	26.93252632
0.9	39.29823444	39.29818903	39.29820642	39.29823350
1.0	57.59815003	57.59809594	57.59810956	57.59814936

The comparison of the numerical result on the several test points are reported in Table 3.

Example 2. Putting $f(x) = 3x + \mathrm{e}^{4x} - \frac{17+3\mathrm{e}^4}{16}$, $\alpha = 0$, $\lambda = 1$, $a = 0$, $b = 1$ and $K(x, s, \varphi(s)) = s\varphi(s)$ into Eq. (17), the Fredholm integral model is obtained in the following form:

$$\varphi(x) = 3x + \mathrm{e}^{4x} - \frac{17 + 3\mathrm{e}^4}{16} + \int_0^1 s\varphi(s)\mathrm{d}s. \tag{33}$$

The result of FCDNN for this model and the exact solution is presented in Fig. 3. The exact solution for this model is $3x + \mathrm{e}^{4x}$. $m_1 = 50$ is the number of shifted Chebyshev quadrature points. The structure of FCDNN, the training data set and the test data set applied to this example, are similar to Example 1. The comparison of the numerical result on the several test points are reported in Table 4.

Example 3. Putting $f(x) = \varphi(x) - \frac{\mathrm{e}^{x+1}-1}{x+1}$, $0 < \alpha \leq 1$, $\lambda = 1$, $a = 0$, $b = 1$ and $K(x, s, \varphi(s)) = \mathrm{e}^{xs}\varphi(s)$ into Eq. (17), the Fredholm integral model is obtained in the following form:

$$D_0^{c\,\alpha}\varphi(x) = \varphi(x) - \frac{\mathrm{e}^{x+1} - 1}{x + 1} + \int_0^1 \mathrm{e}^{xs}\varphi(s)\mathrm{d}s. \quad 0 < \alpha \leq 1. \tag{34}$$

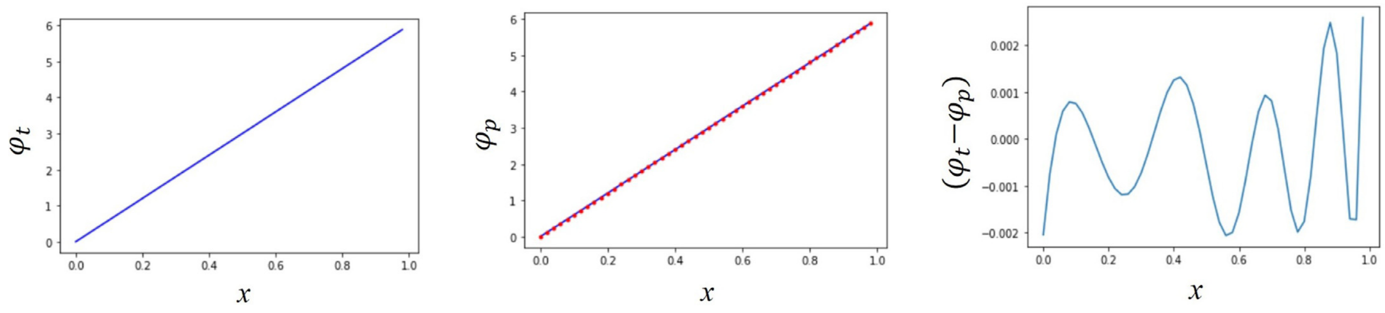


Fig. 2. Results of Example 1, a Fredholm integral equation. Exact solution $\varphi_t(x) = 6x$, predicted solution $\varphi_p(x)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $(\varphi_t - \varphi_p)$.

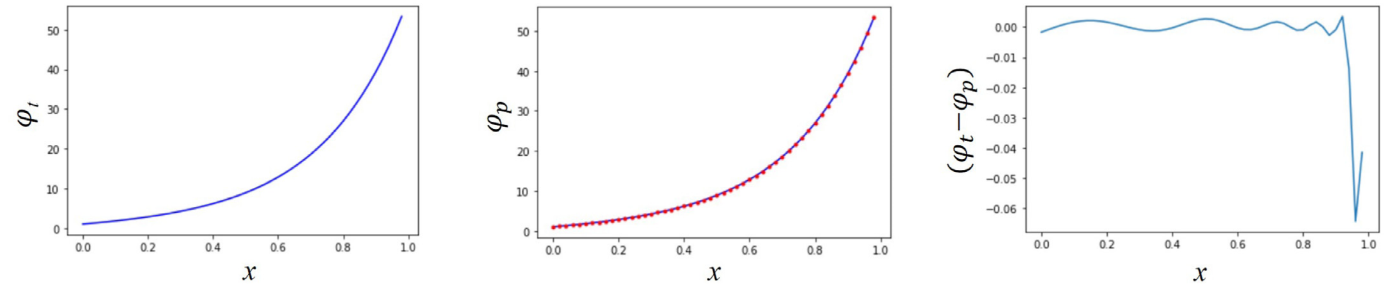


Fig. 3. Results of Example 2, a Fredholm integral equation. Exact solution $\varphi_t(x) = 3x + e^{4x}$, predicted solution $\varphi_p(x)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $(\varphi_t - \varphi_p)$.

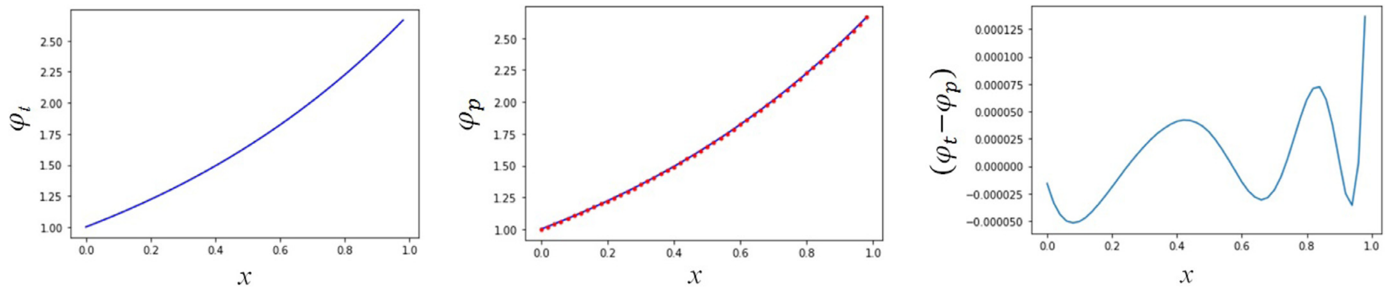


Fig. 4. Results of Example 3, a fractional Fredholm integral equation. Exact solution $\varphi_t(x) = e^x$, for $\alpha = 1$, predicted solution $\varphi_p(x)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $(\varphi_t - \varphi_p)$.

Table 5
The value of $|\mathcal{R}_{IE}|$ in some test points for several different values of α in Example 3.

x	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 0.99$	$\alpha = 1$
0.0	1.7276953e−02	3.1776580e − 03	2.8831480e − 04	1.5735626e − 06
0.2	2.0982331e−02	2.1676415e−03	1.3853300e − 04	2.6804206e − 05
0.4	1.5759063e−02	1.1547971e−02	5.6379680e − 03	3.8052226e − 05
0.6	3.2027840e − 02	1.3071540e − 02	6.1823730e − 03	4.2595270e − 05
0.8	1.4430243e−02	1.7493522e−02	6.5169300e − 04	4.3594615e − 05
1.0	3.4341680e − 01	3.4956790e − 02	1.2740254e−03	1.3648828e − 05

with the initial condition: $\varphi(0) = 1$. The exact solution for this model is e^x for specific value of $\alpha = 1$. The result of FCDNN for this model and the exact solution for $\alpha = 1$ is presented in Fig. 4. The structure of FCDNN, the training data set and the test data set applied to this example, are similar to Example 1 and Example 2. m_1 and m_2 are equal 50 which are the number of shifted Chebyshev quadrature points. Table 5 reports the value of $|\mathcal{R}_{IE}|$ in some test points for several different values of α. The comparison of the numerical result on the several test points are reported in Table 6.

6.2. Fractional telegraph equations

We considered a telegraph equation to evaluate the FCDNN in solving FPDE. A general form of the fractional telegraph equation

Table 6
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network, the predicted value of the Lucas wavelets method (LW) [96] and the predicted value φ_p of FCDNN network in several test points on [0,1] domain with $\alpha = 1$ for Example 3.

x	Exact (φ_t)	Cheb CM	FDNN	LW [96]	FCDNN (φ_p)
0.05	1.0512710	1.0512367	1.0512713	1.0512710	1.0512711
0.15	1.1618342	1.1617365	1.1618346	1.1618342	1.1618344
0.25	1.2840254	1.2838626	1.2840257	1.2840254	1.2840254
0.35	1.4190675	1.4188317	1.4190673	1.4190675	1.4190674
0.45	1.5683121	1.5679921	1.5683124	1.5683121	1.5683122
0.55	1.7332530	1.7328354	1.7332533	1.7332530	1.7332530
0.65	1.9155408	1.9150106	1.9155405	1.9155408	1.9155407
0.75	2.1170000	2.1163401	2.1170004	2.1170000	2.1170001
0.85	2.3396468	2.3388379	2.3396465	2.3396468	2.3396467

is as follows:

$$\begin{aligned}
&{}^{c}_{0}D^{\alpha}_{\tau}\varphi(\bar{x},\tau) + \xi\, {}^{c}_{0}D^{\alpha-1}_{\tau}\varphi(\bar{x},\tau) = \psi(\varphi(\bar{x},\tau)),\\
&\psi(\varphi(\bar{x},\tau)) = F(\bar{x},\tau) - \rho^2\varphi(\bar{x},\tau) + \nabla\varphi(\bar{x},\tau),\\
&x_i \in [c,d],\ \bar{x} = (x_1, x_2, \cdots, x_m),\ \tau \in [0,\mathcal{T}],\\
&\varphi(\bar{c},\tau) = f_1(\tau),\\
&\varphi\left(\bar{d},\tau\right) = f_2(\tau),\\
&\varphi(\bar{x},0) = g_1(\bar{x}),\\
&\frac{\partial\varphi(\bar{x},0)}{\partial\tau} = g_2(\bar{x}).
\end{aligned} \tag{35}$$

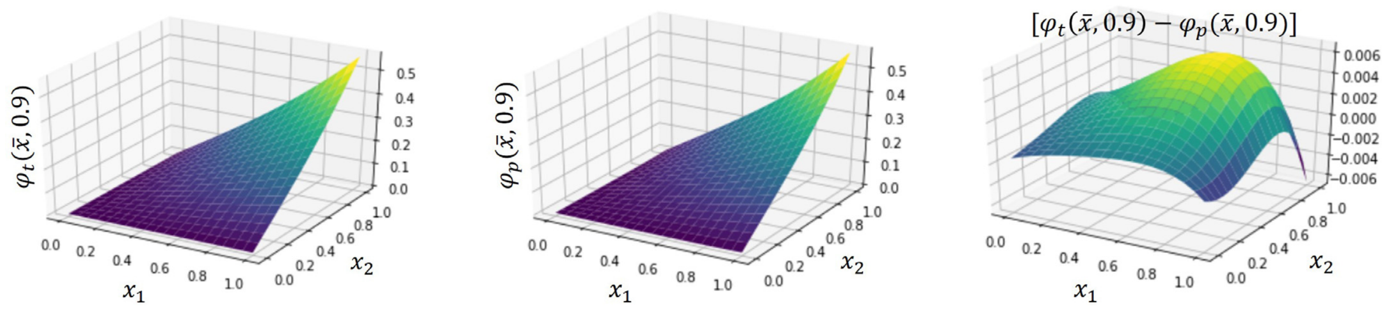


Fig. 5. Results of Example 4, a 2D-telegraph equation. Exact solution $\varphi_t(x_1, x_2, \tau) = e^{-\tau}\sinh(x_1)\sinh(x_2)$, predicted solution $\varphi_p(x_1, x_2, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, x_2, \tau) - \varphi_p(x_1, x_2, \tau)]$, where consider $\tau = 0.9$.

F, f_i and g_i are given functions, ρ is a constant and $\nabla = \sum_i \frac{\partial^2 \varphi(\bar{x},\tau)}{\partial x_i^2}$. Some significant applications of the above mentioned model are viscous fluids, electrical signals, acoustic waves and porous media. Algorithm 2 is performed on Eq. (35) and holds the following results:

$$\left[\left(1+\xi\frac{\Delta\tau}{2}\right)\varphi^{n+1} - 2\varphi^n + \left(1-\xi\frac{\Delta\tau}{2}\right)\varphi^{n-1}\right](\tau_{n+1}-\tau_n)^{2-\alpha} - \frac{1}{c_1}\psi\left(\varphi^{n+1}\right) + \frac{1}{c_1}H^n = 0,$$

$$\frac{1}{c_1}H^n = \sum_{j=0}^{n-1}\left[\left(1+\xi\frac{\Delta\tau}{2}\right)\varphi^{j+1} - 2\varphi^j + \left(1-\xi\frac{\Delta\tau}{2}\right)\varphi^{j-1}\right]\left[\left(\tau_{n+1}-\tau_j\right)^{2-\alpha} - \left(\tau_{n+1}-\tau_{j+1}\right)^{2-\alpha}\right],$$

$$\psi\left(\varphi^{n+1}\right) = F(\bar{x}, \tau_{n+1}) - \rho^2\varphi(\bar{x}, \tau_{n+1}) + \nabla\varphi(\bar{x}, \tau_{n+1}),$$

$$\Theta_p^{n+1} = \left[\left(1+\xi\frac{\Delta\tau}{2}\right)\varphi^{n+1}\right](\tau_{n+1}-\tau_n)^{2-\alpha} - \frac{1}{c_1}\psi\left(\varphi^{n+1}\right),$$

$$\Theta_t^{n+1} = \left[2\varphi^n + \left(\xi\frac{\Delta\tau}{2} - 1\right)\varphi^{n-1}\right](\tau_{n+1}-\tau_n)^{2-\alpha} - \frac{1}{c_1}H^n,$$

$$R_{\text{PDE}} = \Theta_p^{n+1} - \Theta_t^{n+1}. \tag{36}$$

The numerical results are investigated in the following examples.

Example 4. We assume that $\alpha = 2$, $F = 0$, $\xi = 2$, and $\rho = \sqrt{3}$.

$$\begin{aligned}
&{}_0^cD_\tau^2\varphi(\bar{x},\tau) + 2\,{}_0^cD_\tau\varphi(\bar{x},\tau) + 3\varphi(\bar{x},\tau) - \nabla\varphi(\bar{x},\tau) = 0,\\
&x_i \in [0,1],\ \bar{x} = (x_1, x_2),\ \tau \in [0,1],\\
&\varphi(\bar{0},\tau) = 0,\\
&\varphi(\bar{1},\tau) = e^{-\tau}\sinh(1)\sinh(1),\\
&\varphi(\bar{x},0) = \sinh(x_1)\sinh(x_2),\\
&\frac{\partial\varphi(\bar{x},0)}{\partial\tau} = -\sinh(x_1)\sinh(x_2).
\end{aligned} \tag{37}$$

The above model is a 2D-time-fractional telegraph equation with initial and boundary conditions. The exact solution of this model is $\varphi(x_1, x_2, \tau) = e^{-\tau}\sinh(x_1)\sinh(x_2)$. Fig. 5 represents the numerical results for Example 3. Considering the values of the function in times $\tau = 0.1$ and $\tau = 0.5$, and $\Delta\tau = 0.4$, FCDNN predicts the value of the function in $\tau = 0.9$. The training data set in each dimension is selected randomly as Chebyshev quadrature points up to degree 20, which are shifted to [0,1] domain. FCDNN for this example has two inputs, 14-hidden layers and one output. The first 10-layers have 10 neurons in each layer with Chebyshev activation function T_i, $1 \le i \le 10$ for the i-th layer. The last 4-layers have 100 neurons in each layer with hyperbolic tangent activation function. The size of test data set is 20 in each dimension which are random points in [0,1] domain. The comparison of the numerical result on the several test points are reported in Table 7. Also, the result of mean squared errors of the proposed method and the generalized finite difference method (GFDM) [75] are compared into Table 8

Table 7
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network, and the predicted value φ_p of FCDNN network in several test points at $\tau = 0.9$ for Example 4.

(x_1, x_2)	Exact (φ_t)	Cheb CM	FDNN	FCDNN (φ_p)
(0.0,0.0)	0.0	0.000299960	0.000000450	0.0
(0.1,0.1)	0.004079267	0.004061273	0.004079376	0.004079048
(0.2,0.2)	0.016480783	0.016429409	0.016480971	0.016480796
(0.3,0.3)	0.037702265	0.037673190	0.037702453	0.037702284
(0.4,0.4)	0.068595405	0.068505869	0.068595712	0.068595399
(0.5,0.5)	0.110400054	0.110373881	0.110400142	0.110400033
(0.6,0.6)	0.164793979	0.164716177	0.164794003	0.164793985
(0.7,0.7)	0.233960198	0.233953611	0.233961128	0.233960102
(0.8,0.8)	0.320674596	0.320602243	0.320674661	0.320674596
(0.9,0.9)	0.428417326	0.428385748	0.428417379	0.428417319

Table 8
The comparison of mean squared errors between FCDNN and GFDM [75] methods for Examples 4-6.

Example	FCDNN	GFDM [75]
Example 4	8.718456e−06	4.71e−05
Example 5	2.243810e−06	9.41e−04
Example 6	3.394147e−06	5.30e − 06

Example 5. Let $\alpha = 2$, $F = 2\sin(x_1)\sin(x_2)(\cos(\tau) - \sin(\tau))$, $\xi = 2$ and $\rho = 1$. Then the model with initial and boundary conditions is as follows:

$$\begin{aligned}
&{}_0^cD_\tau^2\varphi(\bar{x},\tau) + 2{}_0^cD_\tau\varphi(\bar{x},\tau) + \varphi(\bar{x},\tau) - \nabla\varphi(\bar{x},\tau)\\
&\quad - 2\sin(x_1)\sin(x_2)(\cos(\tau) - \sin(\tau)) = 0,\\
&x_i \in [0,1],\ \bar{x} = (x_1, x_2),\ \tau \in [0,1],\\
&\varphi\left(\bar{0},\tau\right) = 0,\\
&\varphi\left(\bar{1},\tau\right) = \cos(\tau)\sin(1)\sin(1),\\
&\varphi(\bar{x},0) = \sin(x_1)\sin(x_2),\\
&\frac{\partial\varphi(\bar{x},0)}{\partial\tau} = 0.
\end{aligned} \tag{38}$$

The exact solution of this model is $\varphi(x_1, x_2, \tau) = \cos(\tau)\sin(x_1)\sin(x_2)$. The numerical results of Example 5 are indicated in Fig. 6. FCDNN predicts the value of the function in $\tau = 0.9$ by considering the values of the function in times $\tau = 0.1$ and $\tau = 0.5$, and fixing $\Delta\tau = 0.4$. The structure of FCDNN, the training data set and the test data set apply to this example, are similar to Example 4. The comparison of the numerical result on

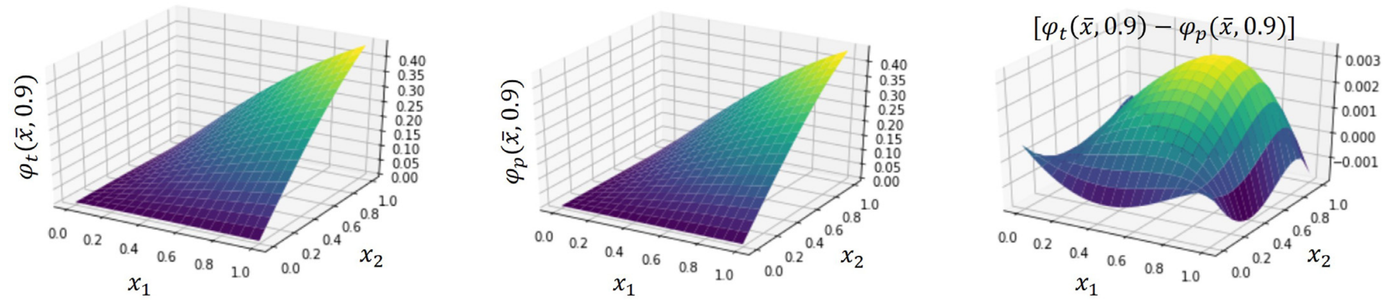


Fig. 6. Results of Example 5, a 2D-telegraph equation. Exact solution $\varphi_t(x_1, x_2, \tau) = \cos(\tau)\sinh(x_1)\sinh(x_2)$, predicted solution $\varphi_p(x_1, x_2, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, x_2, \tau) - \varphi_p(x_1, x_2, \tau)]$, where consider $\tau = 0.9$.

Table 9
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network, and the predicted value φ_p of FCDNN network in several test points at $\tau = 0.9$ for Example 5.

(x_1, x_2)	Exact (φ_t)	Cheb CM	FDNN	FCDNN (φ_p)
(0.0,0.0)	0.0	0.000310260	0.000003300	0.0
(0.1,0.1)	0.006195406	0.006173455	0.006195728	0.006195394
(0.2,0.2)	0.024534636	0.024334273	0.024534674	0.024534619
(0.3,0.3)	0.054286561	0.054223771	0.054286601	0.054286592
(0.4,0.4)	0.094265066	0.094223780	0.094265032	0.094265074
(0.5,0.5)	0.142876334	0.142829270	0.142876732	0.142876389
(0.6,0.6)	0.198182388	0.198089050	0.198182429	0.198182336
(0.7,0.7)	0.257978349	0.257952770	0.257978568	0.257978352
(0.8,0.8)	0.319880341	0.319852801	0.319880295	0.319880367
(0.9,0.9)	0.381420527	0.381373025	0.381420603	0.381420589

Table 10
The exact value φ_t, the predicted value of the Chebyshev collocation method (Cheb CM), the predicted value of the FDNN network, and the predicted value φ_p of FCDNN network in several test points at $\tau = 0.9$ for Example 6.

(x_1, x_2, x_3)	Exact (φ_t)	Cheb CM	FDNN	FCDNN (φ_p)
(0.0,0.0,0.0)	0.0	−0.004223879	0.000000335	0.0
(0.1,0.1,0.1)	0.000408607	0.002402271	0.000408120	0.000408482
(0.2,0.2,0.2)	0.003318175	0.006612490	0.003317906	0.003317959
(0.3,0.3,0.3)	0.011481104	0.010039021	0.011481085	0.011481089
(0.4,0.4,0.4)	0.028175722	0.020009749	0.028175455	0.028175468
(0.5,0.5,0.5)	0.057528950	0.055561553	0.057528302	0.057528697
(0.6,0.6,0.6)	0.104916677	0.104831190	0.104916573	0.104916502
(0.7,0.7,0.7)	0.177478393	0.175920827	0.177477652	0.177478202
(0.8,0.8,0.8)	0.284793027	0.277418339	0.284792887	0.284792970
(0.9,0.9,0.9)	0.439777551	0.434983788	0.439777108	0.439777349

the several test points are reported in Table 9. Also, the result of mean squared errors of the proposed method and the generalized finite difference method (GFDM) [75] are compared into Table 8.

Example 6. Here, we increase the dimension of the telegraph equation and evaluate FCDNN. Consider $\alpha = 2$, $F = 0$, $\xi = 2$ and $\rho = 1$. Then a 3D-telegraph equation with initial and boundary conditions is obtained in the following form:

$$\begin{aligned}
&{}_0^cD_\tau^2\varphi(\bar{x}, \tau) + 2\,{}_0^cD_\tau\varphi(\bar{x}, \tau) + \varphi(\bar{x}, \tau) - \nabla\varphi(\bar{x}, \tau) = 0,\\
&x_i \in [0, 1],\ \bar{x} = (x_1, x_2, x_3),\ \tau \in [0, 1],\\
&\varphi(\bar{0}, \tau) = 0,\\
&\varphi(\bar{1}, \tau) = e^{-\tau}\sinh(1)\sinh(1)\sinh(1),\\
&\varphi(\bar{x}, 0) = \sinh(x_1)\sinh(x_2)\sinh(x_3),\\
&\frac{\partial\varphi(\bar{x}, 0)}{\partial\tau} = -\sinh(x_1)\sinh(x_2)\sinh(x_3).
\end{aligned} \tag{39}$$

The exact solution of this model is $\varphi(x_1, x_2, x_3, \tau) = e^{-\tau}\sinh(x_1)\sinh(x_2)\sinh(x_3)$. Fig. 7 represents the numerical results for Example 6. FCDNN predicts the value of the function in $\tau = 0.9$ by considering the values of the function in times $\tau = 0.1$ and $\tau = 0.5$ and selecting $\Delta\tau = 0.4$. The Chebyshev quadrature points up to degree 20, which are shifted in [0,1] domain, are selected randomly as the training data set in each dimension. FCDNN for this example has three inputs. The other cases are similar to Example 4 and Example 5. The size of the test data set is 20 in each dimension which are random points in [0,1] domain. The comparison of the numerical result on the several test points are reported in Table 10. Also, the result of mean squared errors of the proposed method and the generalized finite difference method (GFDM) [75] are compared into Table 8.

Example 7. In this example, we investigate the 1D-time-fractional telegraph equation to evaluate FCDNN. Consider $F = 2\sin(x_1)[\frac{\tau^{2-\alpha}}{\Gamma(3-\alpha)} + \frac{\tau^{2-\frac{\alpha}{2}}}{\Gamma(3-\frac{\alpha}{2})} + \tau^2]$, $\xi = 1$ and $\rho = 1$. Then a 1D-telegraph equation with initial and boundary conditions is obtained in the following form:

$$\begin{aligned}
&{}_0^cD_\tau^\alpha\varphi(\bar{x}, \tau) + {}_0^cD_\tau^{\alpha-1}\varphi(\bar{x}, \tau) + \varphi(\bar{x}, \tau) - \nabla\varphi(\bar{x}, \tau)\\
&\quad -2\sin(x_1)\left[\frac{\tau^{2-\alpha}}{\Gamma(3-\alpha)} + \frac{\tau^{2-\frac{\alpha}{2}}}{\Gamma\left(3-\frac{\alpha}{2}\right)} + \tau^2\right] = 0,\\
&x_i \in [0, 1],\ \bar{x} = (x_1),\ \tau \in [0, 1],\\
&\varphi(0, \tau) = 0,\\
&\varphi(1, \tau) = \tau^2\sin(1),\\
&\varphi(\bar{x}, 0) = 0,\\
&\frac{\partial\varphi(\bar{x}, 0)}{\partial\tau} = 0.
\end{aligned} \tag{40}$$

The exact solution of this model is $\varphi(x_1, \tau) = \tau^2\sin(x_1)$. We solve this equation with some values of α. Figs. 8 and 9 represent the numerical results of Example 7 with $\alpha = 1.1$ and 1.4, respectively. FCDNN predicts the value of the function in time $\tau = 1.0$ by considering all the values of the function in the previous times when $\Delta\tau = 2^{-8}$. The Chebyshev quadrature points up to degree 50, which are shifted in [0,1] domain, are selected the training data set. The training data set is randomly 1000 points in [0,1] domain and the test data set is 200 random points in [0,1] domain. FCDNN for this example has one input and the other cases are similar to the previous examples. The result of mean squared errors of the proposed method and the radial basis functions method (RBF) [76] are compared into Table 11.

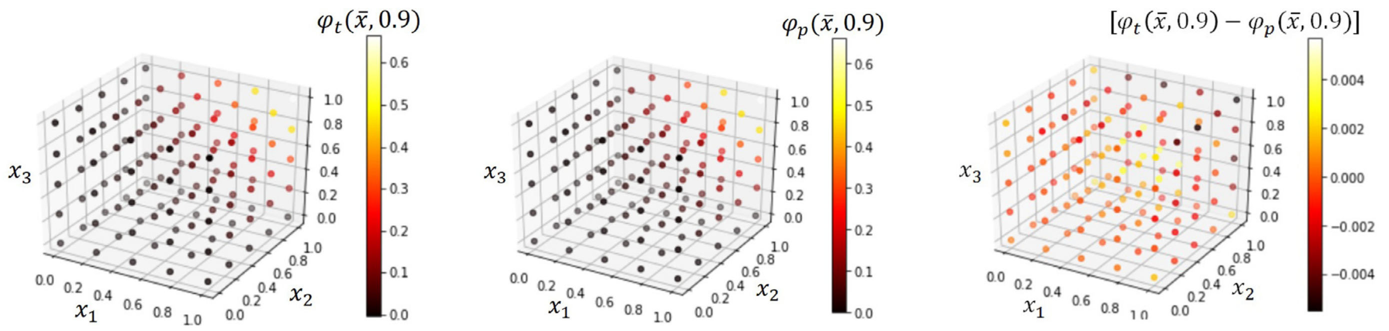


Fig. 7. Results of Example 6, a 3D-telegraph equation. Exact solution $\varphi_t(x_1, x_2, x_3, \tau) = e^{-\tau}\sinh(x_1)\sinh(x_2)\sinh(x_3)$, predicted solution $\varphi_p(x_1, x_2, x_3, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, x_2, x_3, \tau) - \varphi_p(x_1, x_2, x_3, \tau)]$, where consider $\tau = 0.9$.

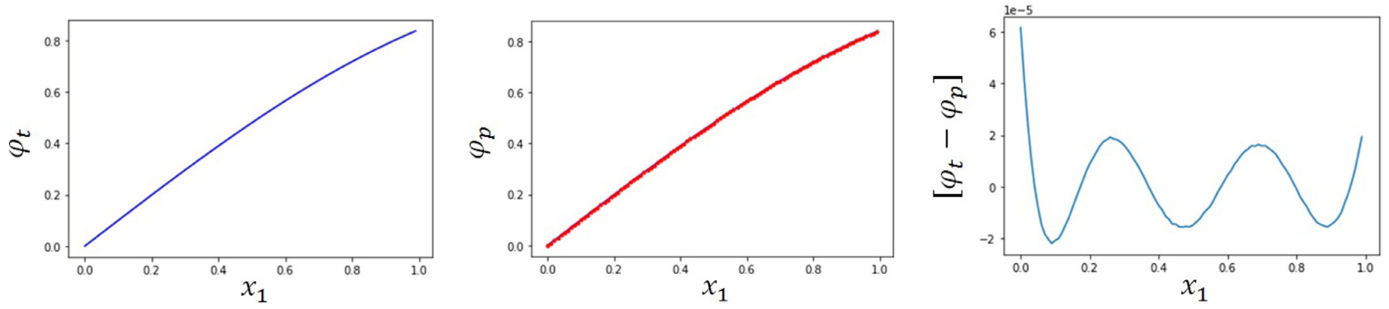


Fig. 8. Results of Example 7, a 1D-time-fractional telegraph equation for $\alpha = 1.1$. Exact solution $\varphi_t(x_1, \tau) = \tau^2\sin(x_1)$, predicted solution $\varphi_p(x_1, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, \tau) - \varphi_p(x_1, \tau)]$, where consider $\tau = 1$.

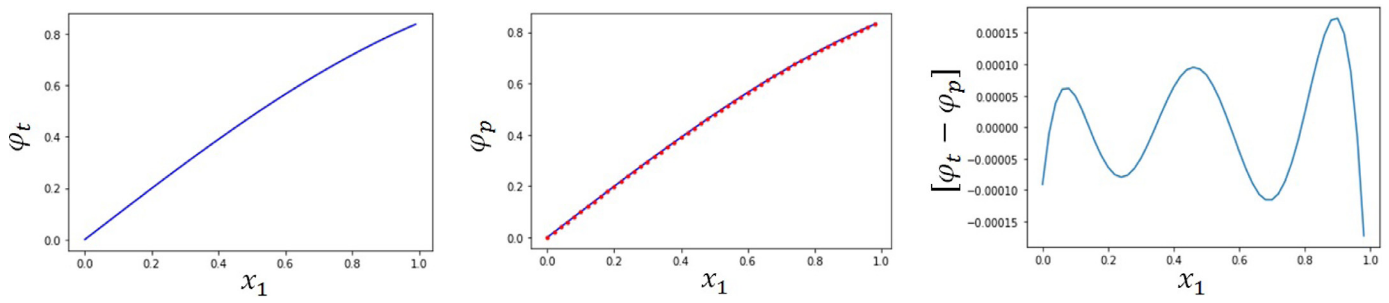


Fig. 9. Results of Example 7, a 1D-time-fractional telegraph equation for $\alpha = 1.4$. Exact solution $\varphi_t(x_1, \tau) = \tau^2\sin(x_1)$, predicted solution $\varphi_p(x_1, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, \tau) - \varphi_p(x_1, \tau)]$, where consider $\tau = 1$.

Table 11
The comparison of mean squared errors between FCDNN and RBF [76] methods at the time $\tau = 1$ for Example 7.

α	FCDNN	RBF [76]
1.1	1.232703e−06	8.4405e−05
1.4	1.059138e−05	1.0242e−05

Example 8. In this example, we investigate the 2D-time-fractional telegraph equation with nonuniform domain boundary to evaluate FCDNN. Consider $F = \frac{24\tau^{4-\alpha}}{\Gamma(5-\alpha)} + \frac{24\tau^{5-\alpha}}{\Gamma(6-\alpha)} + (x_1^2 + x_2^2 + \tau^2) - 4$, $\xi = 1$ and $\rho = 1$. Then a 2D-telegraph equation with initial and boundary conditions is obtained in the following form:

$$
\begin{aligned}
&{}_0^cD_\tau^\alpha\varphi(\bar{x}, \tau) + {}_0^cD_\tau^{\alpha-1}\varphi(\bar{x}, \tau) + \varphi(\bar{x}, \tau) - \nabla\varphi(\bar{x}, \tau) \\
&\quad - \left(\frac{24\tau^{4-\alpha}}{\Gamma(5-\alpha)} + \frac{24\tau^{5-\alpha}}{\Gamma(6-\alpha)} + \left(x_1^2 + x_2^2 + \tau^2\right) - 4\right) = 0, \\
&x_i \in [0, 1],\ \bar{x} = (x_1, x_2),\ \tau \in [0, 1], \\
&\varphi\left(\bar{0}, \tau\right) = \tau^4, \\
&\varphi\left(\bar{1}, \tau\right) = \tau^4 + 2, \\
&\varphi(\bar{x}, 0) = x_1^2 + x_2^2, \\
&\frac{\partial\varphi(\bar{x}, 0)}{\partial\tau} = x_1^2 + x_2^2.
\end{aligned}
\qquad (41)
$$

The exact solution of this model is $\varphi(x_1, x_2, \tau) = x_1^2 + x_2^2 + \tau^4$. The nonuniform domain has a circular shape $(x_1 - 0.5)^2 + (x_2 - 0.5)^2 = (0.5)^2$. We solve this equation with some values of α. Figs. 10 and 11 represent the numerical results of Example 8 with $\alpha = 1.75$ and 1.95, respectively. FCDNN predicts the value of the function in time $\tau = 1$ by considering all the values of the function in the previous times when $\Delta\tau = \frac{1}{100}$. Therefore the Chebyshev quadrature points up to degree 99, which are shifted in [0,1] domain, are selected for the training data set. The boundary training data set includes 200 randomly points on the boundary specific circular domain and 1000 randomly points inside the specific circular domain selects for other members of the training data set. The result of mean squared errors of the proposed method and the local meshless method [85] are compared into Table 12. The FCDNN performs for the several times to predict the value of the telegraph function at time $\tau = 1$ by considering $\Delta\tau = \frac{1}{80}$, $\Delta\tau = \frac{1}{100}$ and $\Delta\tau = \frac{1}{160}$. It is obvious that the result is more better when the $\Delta\tau$ is incremented.

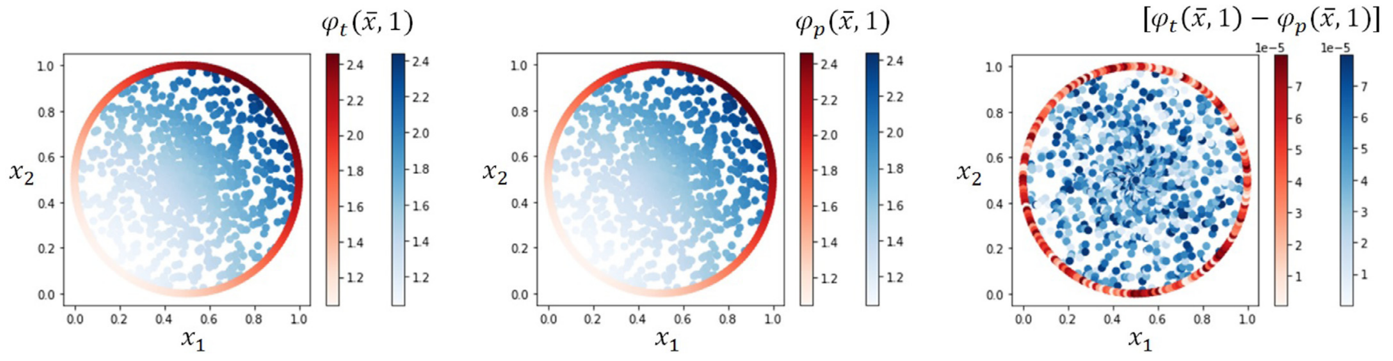


Fig. 10. Results of Example 8, a 2D-time-fractional telegraph equation for $\alpha = 1.95$. Exact solution $\varphi_t(x_1, x_2, \tau) = x_1^2 + x_2^2 + \tau^4$, predicted solution $\varphi_p(x_1, x_2, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, x_2, \tau) - \varphi_p(x_1, x_2, \tau)]$, where consider $\tau = 1$.

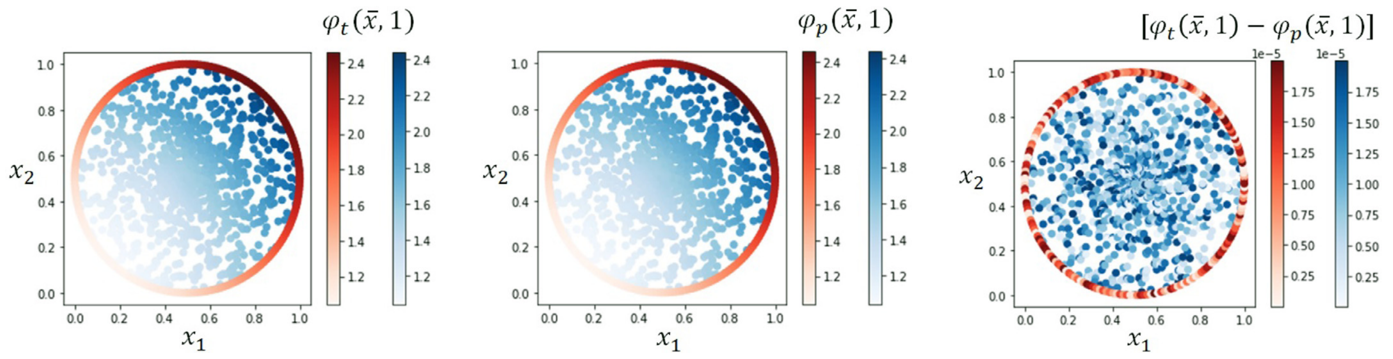


Fig. 11. Results of Example 8, a 2D-time-fractional telegraph equation for $\alpha = 1.75$. Exact solution $\varphi_t(x_1, x_2, \tau) = x_1^2 + x_2^2 + \tau^4$, predicted solution $\varphi_p(x_1, x_2, \tau)$ by FCDNN and comparison of the exact and the predicted solutions by computing the absolute error $[\varphi_t(x_1, x_2, \tau) - \varphi_p(x_1, x_2, \tau)]$, where consider $\tau = 1$.

Table 12
The comparison of mean squared errors between FCDNN and local meshless [85] methods at the time $\tau = 1$ for Example 8.

α	$\Delta\tau$	FCDNN	Local Meshless [85]
1.75	$\frac{1}{80}$	5.234763e−04	1.6185e−03
1.75	$\frac{1}{100}$	4.266165e−05	-
1.75	$\frac{1}{160}$	3.694852e−05	6.6630e−04
1.95	$\frac{1}{80}$	5.726188e−04	5.5439e−03
1.95	$\frac{1}{100}$	4.012831e−05	-
1.95	$\frac{1}{160}$	3.962347e−05	2.6748e−03

7. Conclusion

We presented Fractional Chebyshev Deep Neural Network (FCDNN) for solving frractional differential models in this paper. FCDNN is a deep learning network which combined with the Chebyshev collocation method. The structure of the FCDNN network consists of two networks that are connected in series. In the first network which is called $\mathcal{FFN}$, the approximate solution of the equation is obtained. So the output of this network is φ_p as an approximation of φ in the main equation. In the structure of the $\mathcal{FFN}$ network, Chebyshev polynomials are used as activation functions. Chebyshev polynomials are used to expand the approximate solution of the equation with orthogonal bases. Fast convergence, reduction of computational cost, simplicity of calculations, including derivative and integral calculations are some of the advantages of using orthogonal bases. Particularly, one of the reasons for the lower cost of computation, could be the use of Chebyshev polynomial roots in the training set. In fact, due to the orthogonal property of these polynomials, in less training steps, more appropriate approximations for the network parameters are calculated. In addition, it is notable that the use of the hyperbolic tangent function in some layers of the network, causes to approximate the solution of the equation by the functions of Chebyshev (not just with Chebyshev polynomials themselves and without any expansions). This property expands the feature space and the solution space, resulting in more complex approximations for solving the equation. The second network of our structure is called $\mathcal{RN}$. In this network, by using operational nodes such as derivative and integral nodes and applying them to the obtained approximation solution from the first network, the residual function is formed. In the proposed FCDNN network, Automatic differentiation (AD) is used for derivative calculations. Applying AD allows derivatives to be performed quickly and accurately. In this network, fractional computation techniques such as time marching and Gaussian integral have been proposed to solve a significant range of fractional partial differential equations and fractional integral equations. In the objective function of the network, which is used to determine the parameters of the network, the boundary and initial conditions of the equation are considered as well as the residual function. To optimize the objective function, the mean squared errors on the training data set are considered. The training data set includes the points on domain of the equation and the points of the initial or boundary conditions of the equation. Also, to find more suitable parameters, Adam method is applied first and then the result of the parameters obtained from this method is used as the initial parameters in the L-BFGS Newton method. In this way, near-optimal parameters are obtained to approximate the equation. Finally, using a random process to select training points causes both using suitable points in the deep network and having points which have the necessary features for the collocation method. The proposed network was applied to some fractional problems and accurate results were obtained.

For future works, one can expand the proposed network to solve more complex equations such as higher order equations,

equations and systems with discontinuities, fractional Voltera integral equations or fractional Voltera-Fredholm integral equations. Solving equations defined in infinite or semi-finite domain could be another important future research. Applying developed method like rational Chebyshev fractional collocation method or using particular orthogonal bases like Laguerre or Hermit functions in the structure of the network, could probably be helpful to solve this category of equations.

Declaration of Competing Interest

None.

CRediT authorship contribution statement

Zeinab Hajimohammadi: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft. **Fatemeh Baharifard:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – review & editing. **Ali Ghodsi:** Supervision, Writing – review & editing. **Kourosh Parand:** Supervision, Writing – review & editing.

References

[1] Korshunova N, Jomo J, Lékó G, Reznik D, Balázs P, Kollmannsberger S. Image-based material characterization of complex microarchitectured additively manufactured structures. Computers & Mathematics with Applications 2020;80(11):2462–80.
[2] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25:1097–105.
[3] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.
[4] Vasilyeva M, Tyrylgin A. Machine learning for accelerating macroscopic parameters prediction for poroelasticity problem in stochastic media. Computers & Mathematics with Applications 2021;84:185–202.
[5] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems 1989;2(4):303–14.
[6] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359–66.
[7] Kharazmi E, Zhang Z, Karniadakis GE. Hp-VPINNs: variational physics-informed neural networks with domain decomposition. Comput Methods Appl Mech Eng 2021;374:113547.
[8] Sirignano J, Spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations. J Comput Phys 2018;375:1339–64.
[9] Berg J, Nyström K. A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing 2018;317:28–41.
[10] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[11] Pang G, Lu L, Karniadakis GE. Fpinns: fractional physics-informed neural networks. SIAM Journal on Scientific Computing 2019;41(4):A2603–26.
[12] Lu L, Meng X, Mao Z, Karniadakis GE. Deepxde: a deep learning library for solving differential equations. SIAM Rev 2021;63(1):208–28.
[13] Courant R, Hilbert D. Methods of mathematical physics: partial differential equations. John Wiley & Sons; 2008.
[14] Davis HT. Introduction to nonlinear differential and integral equations. US Government Printing Office; 1961.
[15] Meng X, Li Z, Zhang D, Karniadakis GE. PPINN: Parareal physics-informed neural network for time-dependent PDEs. Comput Methods Appl Mech Eng 2020;370:113250.
[16] Parand K, Delkhosh M. Accurate solution of the thomas–fermi equation using the fractional order of rational chebyshev functions. J Comput Appl Math 2017;317:624–42.
[17] Patnaik S, Hollkamp JP, Semperlotti F. Applications of variable-order fractional operators: a review. Proceedings of the Royal Society A 2020;476(2234):20190498.
[18] Heydari M, Atangana A, Avazzadeh Z. Chebyshev polynomials for the numerical solution of fractal–fractional model of nonlinear ginzburg–landau equation. Eng Comput 2019:1–12.
[19] Atangana A, Bonyah E, Elsadany A. A fractional order optimal 4d chaotic financial model with mittag-leffler law. Chin J Phys 2020;65:38–53.
[20] Korbel J, Luchko Y. Modeling of financial processes with a space-time fractional diffusion equation of varying order. Fractional Calculus and Applied Analysis 2016;19(6):1414.
[21] Atangana A, Khan MA, Fatmawati. Modeling and analysis of competition model of bank data with fractal-fractional caputo-fabrizio operator. Alexandria Engineering Journal 2020;59(4):1985–98.
[22] Frunzo L, Garra R, Giusti A, Luongo V. Modeling biological systems with an improved fractional gompertz law. Commun Nonlinear Sci Numer Simul 2019;74:260–7.
[23] Chakraverty S, Jena RM, Jena SK. Time-fractional order biological systems with uncertain parameters. Synthesis Lectures on Mathematics and Statistics 2020;12(1):1–160.
[24] Pourhashemi A, Ramezani A, Siahi M. Dynamic fractional-order sliding mode strategy to control and stabilize fractional-order nonlinear biological systems. IETE J Res 2020:1–11.
[25] Sweilam NH, Al-Mekhlafi SM, Assiri T, Atangana A. Optimal control for cancer treatment mathematical model using atangana–baleanu–caputo fractional derivative. Advances in Difference Equations 2020;2020(1):1–21.
[26] Riaz MB, Atangana A, Saeed ST. MHD-Free convection flow over a vertical plate with ramped wall temperature and chemical reaction in view of nonsingular kernel. Fractional Order Analysis: Theory, Methods and Applications 2020:253–82.
[27] Sheybak M, Tajadodi H. Numerical solutions of fractional chemical kinetics system. Nonlinear Dyn Syst Theory 2019;19(1):200–8.
[28] Ahmad I, Ahmad H, Thounthong P, Chu Y-M, Cesarano C. Solution of multi-term time-fractional PDE models arising in mathematical biology and physics by local meshless method. Symmetry (Basel) 2020;12(7):1195.
[29] Abro KA, Atangana A. A comparative study of convective fluid motion in rotating cavity via atangana–baleanu and caputo–fabrizio fractal–fractional differentiations. The European Physical Journal Plus 2020;135(2):226.
[30] Oldham K, Spanier J. The fractional calculus theory and applications of differentiation and integration to arbitrary order. Elsevier; 1974.
[31] Dassios I, Baleanu D. Optimal solutions for singular linear systems of caputo fractional differential equations. Math Methods Appl Sci 2021;44(10):7884–96.
[32] Abbaszadeh M, Dehghan M. Meshless upwind local radial basis function-finite difference technique to simulate the time-fractional distributed-order advection–diffusion equation. Eng Comput 2021;37(2):873–89.
[33] Sousa JVdC, Oliveira DS, de Oliveira EC. A note on the mild solutions of hilfer impulsive fractional differential equations. Chaos, Solitons & Fractals 2021;147:110944.
[34] He J-H, Wu X-H. Variational iteration method: new development and applications. Computers & Mathematics with Applications 2007;54(7–8):881–94.
[35] Liao S. Homotopy analysis method in nonlinear differential equations. Springer; 2012.
[36] Wazwaz A-M. A reliable modification of adomian decomposition method. Appl Math Comput 1999;102(1):77–86.
[37] Smith GD. Numerical solution of partial differential equations: finite difference methods. Oxford university press; 1985.
[38] Johnson C. Numerical solution of partial differential equations by the finite element method. Courier Corporation; 2012.
[39] Eymard R, Gallouët T, Herbin R. Finite volume methods. Handbook of numerical analysis 2000;7:713–1018.
[40] Takeda H, Miyama SM, Sekiya M. Numerical simulation of viscous flow by smoothed particle hydrodynamics. Progress of Theoretical Physics 1994;92(5):939–60.
[41] Atluri SN, Zhu T. A new meshless local petrov-galerkin (MLPG) approach in computational mechanics. Comput Mech 1998;22(2):117–27.
[42] Sharan M, Kansa E, Gupta S. Application of the multiquadric method for numerical solution of elliptic partial differential equations. Appl Math Comput 1997;84(2–3):275–302.
[43] Hairer E, Lubich C, Roche M. The numerical solution of differential-algebraic systems by runge-Kutta methods, vol 1409. Springer; 2006.
[44] Canuto C, Hussaini MY, Quarteroni A, Thomas Jr A. Spectral methods in fluid dynamics. Springer Science & Business Media; 2012.
[45] Dormand JR, Prince PJ. A family of embedded runge-kutta formulae. J Comput Appl Math 1980;6(1):19–26.
[46] Khater A, Temsah R, Hassan M. A chebyshev spectral collocation method for solving burgerstype equations. J Comput Appl Math 2008;222(2):333–50.
[47] Parand K, Razzaghi M. Rational legendre approximation for solving some physical problems on semi-infinite intervals. Phys Scr 2004;69(5):353.
[48] Agarwal P, Attary M, Maghasedi M, Kumam P. Solving higher-order boundary and initial value problems via chebyshev–spectral method: application in elastic foundation. Symmetry (Basel) 2020;12(6):987.
[49] Abdelhakem M, Ahmed A, El-Kady M. Spectral monic chebyshev approximation for higher order differential equations. arXiv preprint arXiv:210310343 2021.
[50] Ezz-Eldien SS, Doha EH. Fast and precise spectral method for solving pantograph type volterra integro-differential equations. Numer Algorithms 2019;81(1):57–77.
[51] Zaky MA. An accurate spectral collocation method for nonlinear systems of fractional differential equations and related integral equations with nonsmooth solutions. Appl Numer Math 2020;154:205–22.
[52] Delkhosh M, Parand K. A new computational method based on fractional lagrange functions to solve multi-term fractional differential equations. Numer Algorithms 2021:1–38.
[53] Li X, Xu C. A space-time spectral method for the time fractional diffusion equation. SIAM J Numer Anal 2009;47(3):2108–31.
[54] Singh H, Akhavan Ghassabzadeh F, Tohidi E, Cattani C. Legendre spectral method for the fractional bratu problem. Math Methods Appl Sci 2020;43(9):5941–52.
[55] Aarts LP, Van Der Veer P. Neural network method for solving partial differential equations. Neural Processing Letters 2001;14(3):261–71.

[56] Sun H, Hou M, Yang Y, Zhang T, Weng F, Han F. Solving partial differential equation based on bernstein neural network and extreme learning machine algorithm. Neural Processing Letters 2019;50(2):1153–72.
[57] Chen P, Liu R, Aihara K, Chen L. Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. Nat Commun 2020;11(1):1–15.
[58] Chakraverty S, Mall S. Artificial neural networks for engineers and scientists: solving ordinary differential equations. CRC Press; 2017.
[59] Raja MAZ, Mehmood J, Sabir Z, Nasab AK, Manzar MA. Numerical solution of doubly singular nonlinear systems using neural networks-based integrated intelligent computing. Neural Computing and Applications 2019;31(3):793–812.
[60] Vapnik V. The nature of statistical learning theory. Springer science & business media; 2013.
[61] Mehrkanoon S, Suykens JA. Learning solutions to partial differential equations using LS-SVM. Neurocomputing 2015;159:105–16.
[62] Hajimohammadi Z, Baharifard F, Parand K. A new numerical learning approach to solve general falkner–skan model. Eng Comput 2020:1–17.
[63] Hajimohammadi Z, Parand K. Numerical learning approximation of time-fractional sub diffusion model on a semi-infinite domain. Chaos, Solitons & Fractals 2021;142:110435.
[64] Parand K, Aghaei A, Jani M, Ghodsi A. Parallel LS-SVM for the numerical simulation of fractional volterras population model. Alexandria Engineering Journal 2021;60(6):5637–47.
[65] Baker N, Alexander F, Bremer T, Hagberg A, Kevrekidis Y, Najm H, Parashar M, Patra A, Sethian J, Wild S, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence Tech. Rep.. USDOE Office of Science (SC), Washington, DC (United States); 2019.
[66] Long Z, Lu Y, Ma X, Dong B. PDE-Net: Learning PDEs from data. In: International Conference on Machine Learning; 2018. p. 3208–16.
[67] Han J, Jentzen A, Weinan E. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences 2018;115(34):8505–10.
[68] Yang L, Meng X, Karniadakis GE. B-PINNS: bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. J Comput Phys 2021;425:109913.
[69] Mao Z, Jagtap AD, Karniadakis GE. Physics-informed neural networks for high-speed flows. Comput Methods Appl Mech Eng 2020;360:112789.
[70] Yang Y, Perdikaris P. Adversarial uncertainty quantification in physics-informed neural networks. J Comput Phys 2019;394:136–52.
[71] Raissi M. Deep hidden physics models: deep learning of nonlinear partial differential equations. The Journal of Machine Learning Research 2018;19(1):932–55.
[72] He J, Li L, Xu J, Zheng C. Relu deep neural networks and linear finite elements. arXiv preprint arXiv:180703973 2018.
[73] Molina A, Schramowski P, Kersting K. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In: International Conference on Learning Representations; 2019.
[74] Shin Y, Darbon J, Karniadakis GE. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. arXiv preprint arXiv:200401806 2020.
[75] Ureña F, Gavete L, Benito J, García A, Vargas A. Solving the telegraph equation in 2-d and 3-dusing generalized finite difference method (GFDM). Eng Anal Bound Elem 2020;112:13–24.
[76] Mohebbi A, Abbaszadeh M, Dehghan M. The meshless method of radial basis functions for the numerical solution of time fractional telegraph equation. International Journal of Numerical Methods for Heat & Fluid Flow 2014.
[77] Kumar S. A new analytical modelling for fractional telegraph equation via laplace transform. Appl Math Model 2014;38(13):3154–63.
[78] Chen J, Liu F, Anh V. Analytical solution for the time-fractional telegraph equation by the method of separating variables. J Math Anal Appl 2008;338(2):1364–77.
[79] Hosseini VR, Chen W, Avazzadeh Z. Numerical solution of fractional telegraph equation by using radial basis functions. Eng Anal Bound Elem 2014;38:31–9.
[80] Dehghan M, Shokri A. A numerical method for solving the hyperbolic telegraph equation. Numerical Methods for Partial Differential Equations: An International Journal 2008;24(4):1080–93.
[81] Heydari M, Hooshmandasl M, Mohammadi F. Two-dimensional legendre wavelets for solving time-fractional telegraph equation. Adv Appl Math Mech 2014;6(2):247–60.
[82] Bhrawy AH, Zaky MA, Machado JAT. Numerical solution of the two-sided space–time fractional telegraph equation via chebyshev tau approximation. J Optim Theory Appl 2017;174(1):321–41.
[83] Bonyadi S, Mahmoudi Y, Lakestani M, Rad MJ. A tau method based on jacobi operational matrix for solving fractional telegraph equation with riesz-space derivative. Computational and Applied Mathematics 2020;39(4):1–26.
[84] Eltayeb H, Abdalla YT, Bachar I, Khabir MH. Fractional telegraph equation and its solution by natural transform decomposition method. Symmetry (Basel) 2019;11(3):334.
[85] Kumar A, Bhardwaj A, Dubey S. A local meshless method to approximate the time-fractional telegraph equation. Eng Comput 2020:1–16.
[86] Akram T, Abbas M, Iqbal A, Baleanu D, Asad JH. Novel numerical approach based on modified extended cubic b-spline functions for solving non-linear time-fractional telegraph equation. Symmetry (Basel) 2020;12(7):1154.
[87] Hassani H, Avazzadeh Z, Machado JT. Numerical approach for solving variable-order space–time fractional telegraph equation using transcendental bernstein series. Eng Comput 2020;36(3):867–78.
[88] Srinivasa K, Rezazadeh H. Numerical solution for the fractional-order one-dimensional telegraph equation via wavelet technique. International Journal of Nonlinear Sciences and Numerical Simulation 2020.
[89] Kumar K, Pandey RK, Yadav S. Finite difference scheme for a fractional telegraph equation with generalized fractional derivative terms. Physica A 2019;535:122271.
[90] Ibrahim W, Bijiga LK. Neural network method for solving time-fractional telegraph equation. Mathematical Problems in Engineering 2021;2021.
[91] Wazwaz A-M. Linear and nonlinear integral equations, vol 639. Springer; 2011.
[92] Zhu L, Fan Q. Solving fractional nonlinear fredholm integro-differential equations by the second kind chebyshev wavelet. Commun Nonlinear Sci Numer Simul 2012;17(6):2333–41.
[93] Sharma S, Pandey RK, Kumar K. Collocation method with convergence for generalized fractional integro-differential equations. J Comput Appl Math 2018;342:419–30.
[94] Doha EH, Abdelkawy MA, Amin A, Lopes AM. Shifted jacobi–gauss-collocation with convergence analysis for fractional integro-differential equations. Commun Nonlinear Sci Numer Simul 2019;72:342–59.
[95] Yousefi A, Javadi S, Babolian E, Moradi E. Convergence analysis of the chebyshev–legendre spectral method for a class of fredholm fractional integro-differential equations. J Comput Appl Math 2019;358:97–110.
[96] Dehestani H, Ordokhani Y, Razzaghi M. Combination of lucas wavelets with legendre–gauss quadrature for fractional fredholm–volterra integro-differential equations. J Comput Appl Math 2021;382:113070.
[97] Shen J, Tang T, Wang L-L. Spectral methods: algorithms, analysis and applications, vol 41. Springer Science & Business Media; 2011.
[98] Parand K, Delkhosh M. Solving volterras population growth model of arbitrary order using the generalized fractional order of the chebyshev functions. Ricerche di Matematica 2016;65(1):307–28.
[99] Kilbas A, Srivastava H, Trujillo J. Theory and applications of fractional differential equations, vol 204. elsevier; 2006.
[100] Odibat Z, Momani S. An algorithm for the numerical solution of differential equations of fractional order. Journal of Applied Mathematics & Informatics 2008;26(1_2):15–27.
[101] Karumuri S, Tripathy R, Bilionis I, Panchal J. Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. J Comput Phys 2020;404:109120.
[102] Kingma DP, Ba J. Adam (2014), a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR), arXiv preprint arXiv, vol. 1412; 2015.
[103] Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Math Program 1989;45(1–3):503–28.
[104] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: a survey. The Journal of Machine Learning Research 2017;18(1):5595–637.