Engineered Science
DOI: https://dx.doi.org/10.30919/es8d789

Deep Ritz Method for Solving High-Dimensional Fractional Differential Equations

Juan Yang,[1] Jiarong Zuo,[1,*] Yu Tian[1] and Ming Lei[2]

Abstract

In this paper, we approximate the solutions of high-dimensional fractional order differential equations involving the right Riemann-Liouville fractional derivatives, left Caputo fractional derivatives and boundary value conditions. Once the problem's variational structure has been identified, solving the equation can be stated as an optimal control problem. We introduce a deep learning-based numerical scheme which can avoid the curse of dimensionality for this optimal control problem. The deep Ritz method and point-taking method play an important role. We show by examples that the proposed numerical scheme produces accurate results of fractional order differential equations of low, medium and high dimensions. Some tables and figures are used to show the results of our experiments so that the effects of different methods can be intuitively displayed. The new point-taking method reduces the L_2 error between exact solution and approximate solution of the equation in low-dimensional case to 10^{-6}, smaller than 10^{-5} achieved by the previous point-taking method. In medium-dimensional and high-dimensional cases, the approximate solutions obtained by two kinds of loss functions are compared and the loss function obtained by the variational method performs better that it makes the L_2 error between exact solution and approximate solution in two cases as low as 10^{-12} and 10^{-7}, respectively, which shows the superiority of the variational method in approximating differential equations of fractional order. Compared to the medium-dimensional fractional order differential equations, the deep Ritz method performs slightly less well on the high-dimensional case, but the L_2 error down to 10^{-7} is still a good result.

Keywords: Deep Ritz method; Fractional order differential equations; High-dimensional; Optimal control; Variational method.
Received: 09 September 2022; Revised: 12 November 2022; Accepted: 19 November 2022.
Article type: Research article.

1. Introduction

Fractional order calculus emerged almost simultaneously with classical calculus. The singularity and non-locality of fractional order calculus are more suitable for describing materials and processes with memorability and heredity than integer order calculus. As an extension of integer order differential equations, fractional order differential equations play an important role in physical and engineering fields such as simulation of complex media, and are also widely used in modeling physical processes and complex systems such as heat conduction engineering, statistical mechanics, electromagnetism, control systems, and so on.[1-7] The difficulty in finding the solutions of fractional order differential equations consists of two main aspects: one is that the solutions of fractional order differential equations usually contain functions that are difficult to handle, and the other is that the solutions of fractional order differential equations cannot be given in the explicit form. There have been many kinds of numerical solutions to differential equations, mainly divided into two general categories: numerical solutions with grid and numerical solutions without grid, other methods can be referred to Liu *et al*.[8] The commonly used numerical solutions with grid include the finite difference method and the finite element method.

The finite difference method is currently the most used method because of its ease of programming and stability of numerical calculation. Its main idea is to approximate the value of the fractional order derivative of the point by the value of the function at a finite number of points. For

[1] *Department of Mathematics, School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China.*
[2] *School of Integrated Circuits, Beijing University of Posts and Telecommunications, Beijing 100876, China.*
* *E-mail:* zuojr1997@bupt.edu.cn (J. Zuo)

Riemann-Liouville fractional order differential equations, the Grunwald-Letnikov derivative and its deformations are usually used to discretize the differential equation by finite differences. However, subsequent studies showed that this method was unstable, so Meerschaert *et al.* studied the biased Grunwald-Letnikov formula based on the equivalence of the Riemann-Liouville fractional order derivative and the Grunwald fractional order derivative, and verified that its difference format is first-order accurate and stable in 2006.[9] Meanwhile, Ortigueira studied the fractional order central difference format with second-order accuracy for the Riesz fractional order derivatives, and the convergence of this format was proved in detail by Celik *et al.* in 2012.[10]

The finite element method is more adaptable to the complex domain of definition and has higher convergence accuracy than the finite difference method. Ervin and Roop were the first to apply the finite element method to fractional order differential equations.[11] They defined the fractional order derivative space for the first time and proved the equivalence of the fractional order derivative space to the fractional order Sobolev space in some sense. After that, they applied the finite element method to the fractional order convection-diffusion equation and gave a complete theoretical analysis. The finite element method for fractional order partial differential equations containing Riesz fractional order derivatives was studied by Bu.[12] Deng studied the finite element method for the space-time fractional order Fokker-Planck equation, proved the stability of the method and performed convergence analysis.[13] Li studied the time-fractional order diffusion equation and the diffusion wave equation, and proved the unconditional stability and convergence of the fully discrete format.[14]

Numerical solutions with grid generally require the use of grid to approximate the definition space of partial differential equations, and the more detailed the grid, the finer the solution will be, so numerical solutions with grid require higher computational cost and larger storage space. For instance, in the finite difference method and the finite element method, if we set N grid points or basis functions in each direction to attain $O\left(\frac{1}{N}\right)$ accuracy, we will need $O\left(N^d\right)$ degree of freedom for a d-dimensional problem. When d becomes large, the exponential growth $O\left(N^d\right)$ makes numerical solutions with grid hard to handle. Due to the curse of dimensionality (CoD) which means the computational complexity grows exponentially with the increase of the dimension of the equation, such methods are limited to dealing with low-dimensional equations. We need to explore an algorithm that is independent of the curse of dimensionality for solving high-dimensional partial differential equations, i.e., the numerical solutions without grid, which consist of two main categories: the Monte Carlo method and the deep learning method.

The Monte Carlo method is widely used for solving differential equations.[15-17] In some cases, the Feynman-Kac formula can be used to represent the function to be solved in the equation as the mathematical expectation of a random variable in some stochastic process, and then the partial differential equation can be solved by the Monte Carlo method of finding the expectation. In 2005, Kebaier proposed a two-level Monte Carlo method and applied it to the Asian option problem.[18] In 2008, Giles extended Kebaier's two-level Monte Carlo method to a multilevel Monte Carlo method for option pricing problems[19] and partial differential equations.[20-25] The convergence of the Monte Carlo method depends on the number of points sampled in the space, and the more points sampled, the more accurate the solution can be. However, the more sampling points, the longer the computation time and the larger the storage space. Therefore, a large computational effort is still required in solving higher dimensional partial differential equations, which severely limits the practical application of the Monte Carlo method. We thus need a method to solve partial differential equations with a smaller number of parameters and less computational consumption, and that is the deep learning method we will introduce next.

Deep learning methods make seeking solutions of high-dimensional problems feasible as an important advantage of deep learning is that it provides a gridless algorithm, which avoids the problem of the grid division in high-dimensional partial differential equations and makes model training less time-consuming. The Kolmogorov-Arnold superposition theorem proves that, for general continuous functions, the curse of dimensionality can be avoided by three-layer neural networks with advanced activation functions and the networks merely require $O(d)$ parameters.[26] The network computation was usually expensive that limited the applications of deep neural networks. However, the development of GPU-based parallel computing over the last two decades greatly boosts the network computation, especially for high-dimensional problems. Also, due to the current popularity of deep learning frameworks, solving partial differential equations using deep learning will be less difficult to program than using numerical solutions with grid.

In applications, deep learning methods have been successfully applied to data mining, autonomous driving, computer vision, natural language processing,[22,23] *etc*. In addition, neural networks have many different structures that can be applied to different equations. Deep networks have been applied on control problem,[24] Schrodinger equation,[25]

nonlinear parabolic partial differential equation,[27-39] variational problem,[33] parametric partial differential equation,[28] *etc*. The workflow of the deep learning method applied to the partial differential equation is described by Sirignano,[29] where the deep learning method transforms solving the partial differential equation into an optimization problem, so that the network is trained with the given initial and boundary conditions, and when the training is completed, the numerical solution predicted by the neural network can be treated as an approximate solution of the partial differential equation, and when the number of parameters tends to infinity, the convergence of the method has been demonstrated.

Especially, for fractional order differential equations, researchers have proposed neural networks that use algorithms based on gradient descent.[30,31] A class of Gaussian process kernels can also be added to the neural network to solve space fractional order differential equations.[32-43] In cases where the finite difference method or the finite element method are difficult to handle, this method can also be applied to variable-order fractional order differential equations.

In this paper we will study the fractional order differential equation in the following form[44]:

$$\begin{cases} \sum_{i=1}^{N} {}_{x_i} D_T^{\alpha_i}({}_0^C D_{x_i}^{\alpha_i} u)(x) + u(x) = -f(x), \\ \qquad x = (x_1, x_2, \cdots, x_N) \in \Omega \\ u(x) = 0, \ x \in \partial\Omega \end{cases} \tag{1}$$

where $\Omega = [0,T]^N$, $T \in R$, $\frac{1}{2} \le \alpha \le 1$ ${}_{x_i}D_T^{\alpha_i}$, ${}_0^C D_{x_i}^{\alpha_i}$, denote right Riemann-Liouville and left Caputo fractional derivatives respectively. From Theorem 4.1 and Theorem 5.2 in Jiao *et al.*,[44] under suitable conditions, there exists a solution of Equation (1).

The residual networks (ResNets) have been used to approximate partial differential equations such as Poisson equation,[33] Navier-Stokes equation, Burgers equation, Korteweg–de Vries equation,[45] *etc*. Some relative works also employ the variational method to solve partial differential equations.[34-36] To obtain a more accurate approximate solution of the above fractional order differential equation, we combine the variational method with the ResNets, which means energy functional is chosen as the loss function in the ResNet. The similar method called the deep Ritz method was first proposed by E *et al*.[33] This method is used to solve variational problems and is tested on Poisson equation. The test results show that the deep Ritz method has the potential to work in rather high dimensions. In this article, we extend this method to fractional order differential equations. Different from E *et al.*,[33] we adopt a new point-taking method. After changing the point-taking method, we get a better approximate solution, which results in a significant improvement in the efficiency of model training. Intuitively, in finding the approximate solution of the high-dimensional fractional order differential equation, using L_2 error between the approximate solution and the explicit solution as the loss function gives a better approximate solution than using the energy functional as the loss function. However, the opposite is true. In our experiments, we found that using the energy functional as the loss function works better. To make results clear and concise, a large number of figures and tables will be shown to compare the data obtained by different methods.

2. Basic knowledge of fractional derivatives

The fractional derivative is an extension of the normal derivative D^n which transforms n to arbitrary values such as rational, irrational, or complex. In this section, to fully understand the fractional order differential equations, we will introduce some basic notions of fractional derivatives.

Definition 2.1 (Left and Right Riemann-Liouville Fractional Derivatives)[40]

Let f be a function defined on [a, b]. The left and right Riemann-Liouville fractional derivatives of order $\gamma > 0$ for function f denoted by ${}_aD_t^{\gamma} f(t)$ and ${}_tD_b^{\gamma} f(t)$, respectively, are defined by:

$$\begin{aligned} {}_aD_t^{\gamma} f(t) &= \frac{d^n}{dt^n}\, {}_aD_t^{\gamma-n} f(t) \\ &= \frac{1}{\Gamma(n-\gamma)} \frac{d^n}{dt^n} \left(\int_a^t (t-s)^{n-\gamma-1} f(s) ds \right) \end{aligned} \tag{2}$$

and

$$\begin{aligned} {}_tD_b^{\gamma} f(t) &= (-1)^n \frac{d^n}{dt^n}\, {}_tD_b^{\gamma-n} f(t) = \\ & \frac{1}{\Gamma(n-\gamma)} (-1)^n \frac{d^n}{dt^n} \left(\int_t^b (s-t)^{n-\gamma-1} f(s) ds \right) \end{aligned} \tag{3}$$

where $t \in [a, b]$, $n-1 \le \gamma < n$ and $n \in N$. In particular, if $0 \le \gamma < 1$, then

$$\begin{aligned} {}_aD_t^{\gamma} f(t) &= \frac{d}{dt}\, {}_aD_t^{\gamma-1} f(t) \\ &= \frac{1}{\Gamma(1-\gamma)} \frac{d}{dt} \left(\int_a^t (t-s)^{-\gamma} f(s) ds \right), \quad t \in [a,b] \end{aligned} \tag{4}$$

and

$$\begin{aligned} {}_tD_b^{\gamma} f(t) &= -\frac{d}{dt}\, {}_tD_b^{\gamma-1} f(t) = \\ & -\frac{1}{\Gamma(1-\gamma)} \frac{d}{dt} \left(\int_t^b (s-t)^{-\gamma} f(s) ds \right), \quad t \in [a,b] \end{aligned} \tag{5}$$

Remark 2.1 For $n \in N$, if γ becomes an integer $n-1$, according to Definition 1, the left and right Riemann-Liouville fractional derivatives become the usual definitions, namely

$aD_t^{n-1}f(t) = f^{n-1}(t)$ and $tD_b^{n-1}f(t) = (-1)^{n-1}f^{(n-1)}(t)$, $t \in [a,b]$

where $f^{(n-1)}(t)$ is the usual derivative of order $n-1$.

Definition 2.2 (Left and Right Caputo Fractional Derivatives) [40] Let $\gamma \geq 0$ and $n \in N$.

(i) If $\gamma \in (n-1,n)$ and $f \in AC^n[a,b]$, then the left and right Caputo fractional derivatives of order γ for function f denoted by ${}_a^cD_t^\gamma f(t)$ and ${}_t^cD_b^\gamma f(t)$, respectively, exist almost everywhere on [a, b]. ${}_a^cD_t^\gamma f(t)$ and ${}_t^cD_b^\gamma f(t)$ are represented by

$$ {}_a^cD_t^\gamma f(t) = {}_aD_t^{\gamma-n}f^{(n)}(t) = \frac{1}{\Gamma(n-\gamma)}\left(\int_a^t (t-s)^{n-\gamma-1} f^{(n)}(s)ds\right) \quad (6)$$

and

$$ {}_t^cD_b^\gamma f(t) = (-1)^n {}_tD_b^{\gamma-n}f^{(n)}(t) = \frac{(-1)^n}{\Gamma(n-\gamma)}\left(\int_t^b (s-t)^{n-\gamma-1} f^{(n)}(s)ds\right) \quad (7)$$

respectively, where $t \in [a,b]$. In particular, if $0 < \gamma < 1$, then

$$ {}_a^cD_t^\gamma f(t) = {}_aD_t^{\gamma-1}f'(t) = \frac{1}{\Gamma(1-\gamma)}\left(\int_a^t (t-s)^{-\gamma} f'(s)ds\right), \quad t \in [a,b] \quad (8)$$

and

$$ {}_t^cD_b^\gamma f(t) = -{}_tD_b^{\gamma-1}f'(t) = -\frac{1}{\Gamma(1-\gamma)}\left(\int_t^b (s-t)^{-\gamma} f'(s)ds\right), \quad t \in [a,b] \quad (9)$$

(ii) If $\gamma = n-1$ and $f \in AC^{n-1}[a,b]$, then ${}_a^cD_t^{n-1}f(t)$ and ${}_t^cD_b^{n-1}f(t)$ are represented by

${}_a^cD_t^{n-1}f(t) = f^{(n-1)}(t)$ and ${}_t^cD_b^{n-1}f(t) = (-1)^{n-1} f^{(n-1)}(t)$, $t \in [a,b]$

In particular, ${}_a^cD_t^0 f(t) = {}_t^cD_b^0 f(t) = f(t)$ $t \in [a,b]$

3. Numerical scheme

In this section, the method will be shown. The algorithm we use is based on the following ideas:

1. Approximating the solution of the fractional order differential equation will be transformed into an optimization problem by variational principle.
2. Minimizing the energy functional $I(u, \beta)$ by deep neural networks.

3.1 Variational principle

Theorem 3.1 Solving Equation (1) is transformed to the following optimization problem

$$\min_{u,\beta} I(u,\beta) \quad (10)$$

where the energy functional

$$I(u,\beta) = \int_\Omega \left(\frac{1}{2}\left|u(x)\right|^2 + \frac{1}{2}\sum_{i=1}^N \left|{}_0^cD_{x_i}^{\alpha_i}u(x)\right|^2\right)dx + \int_\Omega f(x)u(x)dx + \frac{1}{2}\beta\int_{\partial\Omega}|u(x)|^2 dx \quad (11)$$

and the norm on the separable Hilbert space $L^2(\Omega)$ is denoted by $|\cdot|$. Then u which minimizes $I(u, \beta)$ is the solution of Equation (1).

Proof If $\min_{\mu,\beta} I(\mu,\beta)$ exists, then for every $v \in C_0^\infty(\Omega)$, $\langle \frac{\partial}{\partial u} I, v\rangle = 0$ and $\frac{\partial}{\partial \beta} I = 0$, where $\langle\cdot,\cdot\rangle$ denotes the inner product in $L^2(\Omega)$. Specifically,

$$\frac{\partial}{\partial\beta} I(u,\beta) = \frac{1}{2}\int_{\partial\Omega}|u(x)|^2 dx = 0$$

implies u (x) = 0 when $x \in \partial\Omega$. That means boundary condition of Equation (1) is satisfied. Moreover, if

$$0 = \langle \frac{\partial}{\partial u} I(u,\beta), v\rangle$$

$$= \int_\Omega \left[u(x)v(x) + \sum_{i=1}^N {}_0^cD_{x_i}^{\alpha_i}u(x).\,{}_0^cD_{x_i}^{\alpha_i} v(x)\right]dx + \int_\Omega f(x)v(x)dx + \beta\int_{\delta\Omega} u(x)\,v(x)\,dx$$

$$= \int_\Omega \left[u(x)v(x) + \sum_{i=1}^N {}_0^cD_{x_i}^{\alpha_i}u(x).\,{}_0^cD_{x_i}^{\alpha_i} v(x)\right]dx + \int_\Omega f(x)v(x)dx$$

$$= \int_\Omega \left[u(x) + \sum_{i=1}^N {}_{x_i}D_T^{\alpha_i}\left({}_0^cD_{x_i}^{\alpha_i}u\right) + f(x)\right]v(x)\,dx$$

Then Equation (1) is satisfied. The prove is completed.

3.2 Deep neural network

The deep Ritz method is to transform the input x to $y_\theta(x) \in R^m$ by a nonlinear transformation which is realized by a deep neural network. θ denotes the weights in the network and is useful to define the nonlinear transformation. The network we use is shown in Fig. 1, which consists of several stacked blocks, one block is composed of two linear transformations, two activation functions and a residual connection. In this network, the input is a set of random points sampled from Ω and $\partial\Omega$, and the outputs are u and the Caputo fractional derivatives of u.

The dimension of input x is transformed to m after the first fully connected layer, so that the weights in the network can be increased to improve the accuracy of the network.

$$x^1 = W_0 \cdot x + b_0 \quad (12)$$

It corresponds to the first fully connected layer in the network.

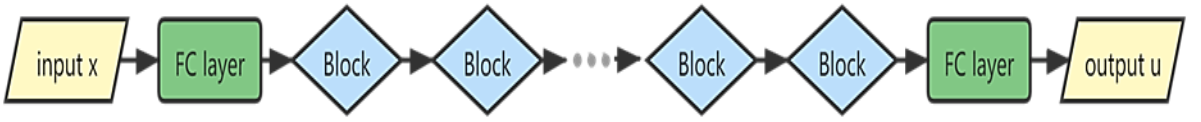

Fig. 1 A network including several blocks and two fully connected layers. The input x enters each part in turn and then transforms into the output u.

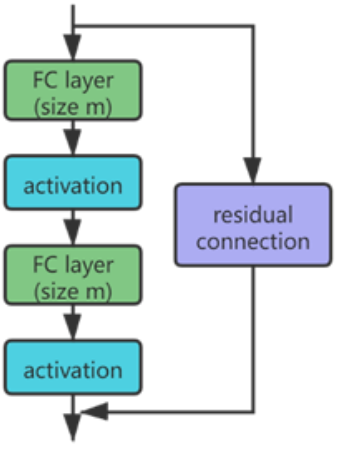

Fig. 2 The block in the network. Each block consists of two fully connected layers and a residual connection. The input of the ith block is x^i and the output is $f_i(x^i)$.

Fig. 2 shows how a block works. The ith block can be expressed as such equation:

$$f_i(x^i) = \phi(W_{i,2} \cdot \phi(W_{i,1}x^i + b_{i,1}) + b_{i,2}) + x^i \quad (13)$$

where vectors $x^i \in R^m$ is the input of the block, $\emptyset$ is the activation function, $W_{i,1}, W_{i,2} \in R^{m\times m}$ and $b_{i,1}, b_{i,2} \in R^m$ are weights in the block. The last term in Equation (13) is the residual connection which helps to avoid the problem of vanishing gradient to let the network much easier to train.[37] The activation function is usually applied to neural networks to make the network nonlinear and has a significant impact on the algorithm's accuracy. To achieve a balance between simplicity and precision, we use Equation (14), Equation (15) as activation functions in the 2-dimensional case, high-dimensional case, respectively.

$$\phi(x) = max\{x^3, 0\} \quad (14)$$

$$\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (15)$$

The full k-block network can be expressed as:

$$y_\theta(x^1) = f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1(x^1) \quad (16)$$

where θ denotes the weights in the whole network. After having to $y_\theta\ (x^1), u$ can be obtained by

$$u(x;\theta) = W_1 \cdot y_\theta(x^1) + b_1 \quad (17)$$

it corresponds to the output linear layer (the second FC layer in Fig. 1). The dimension of input becomes m after several blocks, so we need a layer to change the dimension of output from m to what we want. After substituting $u(x;\theta)$ into the form of I, a function of θ which we should minimize is obtained, that is *I (u,β)* (Equation (11)). The suitable u which makes *I (u, β)* take the infimum and optimizes the parameters θ, is the object we want, which we also call the approximate solution of fractional order differential equation.

Different from typical grid-based quadrature rules such as the Trapezoidal rule and the Simpson's rule, the Monte Carlo algorithm does not suffer from CoD and works well in high-dimensional situations. Computing the integral in I is quite impossible so we use the Monte Carlo algorithm to discrete.

$$I = \frac{1}{M}\sum_{k=1}^{M} I^k \quad (18)$$

where I^k is the value of functional at I the point I^k. For instance,

$$I^1 = \begin{cases} T^N[\frac{1}{2}|u(x^1)|^2 + \frac{1}{2}\sum_{i=1}^{N} |{}_0^C D_{x_i}^{\alpha_i} u(x)|_{x=x^1}{}^2 + \\ \qquad f(x^1)u(x^1)], x^1 \in \Omega \\ \beta NT|u(x^1)|^2, x^1 \in \partial\Omega \end{cases} \quad (19)$$

M is the number of data points and is usually quite huge. The method we use to update the weights θ is stochastic gradient descent algorithm, which can be described below,

$$\theta^{k+1} = \theta^k - \eta \nabla I_{\gamma^k}(\theta^k) \quad (20)$$

where $\{\gamma^k\}$ are i.i.d random variables uniformly distributed over *{1,2,L, M}*. θ^k is the weight at the kth iteration and θ^{k+1} is the weight at the $(k+1)th$ iteration. η is the learning rate. Different from gradient descent, at the $(k+1)th$ iteration, we simply randomly choose one term $I_{\gamma^K}(\theta^K)$ which is computed by one random point. The loss function is used to estimate the difference between the output of the neural network and the true value, and to give direction to the optimization of the parameters in the neural network. We choose the Adam optimizer[38] during the neural network back propagation to update parameters layer by layer to make the loss function gradually decrease until convergence.

4. Numerical tests

In this section, we test three fractional order differential equations in two dimensions, ten dimensions and one hundred dimensions. All numerical tests were performed on a Windows 10 computer with an Intel Core i7-11800H processor, using the Pytorch framework.

4.1 Example

Let $\alpha_1 = \alpha_2 = \frac{1}{2}$ and $T = 1$

4.1.1 Example 1

Give a continuous function $-f(x_1,x_2) = x_2(1-x_2)\,g(x_1) + x_1(1-x_1)g(x_2) + x_1x_2(1-x_1)$,

$g(x) = -[\Gamma(\frac{1}{2})]^{-2}(2I_1 - \frac{1}{3}I_2)$

where, $I_1 = \frac{1}{\sqrt{1-x}} + A - \frac{\log(x)}{2}$, $I_2 = \frac{4-12x}{\sqrt{1-x}} + 12xA - 6x\log(x)$ with $A = Log\left(\sqrt{1-x}-1\right)$.

We consider the PDE for $(x_1,x_2) \in [0,1] \times [0,1]$,

$$ {}_{x_1}D_T^{\frac{1}{2}}\left({}_0^C D_{x_1}^{\frac{1}{2}}u\right)(x_1,x_2) + {}_{x_2}D_T^{\frac{1}{2}}\left({}_0^C D_{x_2}^{\frac{1}{2}}u\right)(x_1,x_2) + u(x_1,x_2) = -f(x_1,x_2) $$
$$ u(0,x_2) = u(1,x_2) = u(x_1,0) = u(x_1,1) = 0 \quad (21) $$

Then $u(x_1,x_2) = x_1(1-x_1)\,x_2(1-x_2)$ is the solution of Equation (21).

Consider the following high-dimensional case,

$$ \begin{cases} \sum_{i=1}^{N} {}_{x_i}D_T^{\frac{1}{2}}({}_0^C D_{x_i}^{\frac{1}{2}}u)(x) + u(x) = -f(x),\ x = (x_1,x_2,\cdots,x_N) \in [0,1]^N \\ u(x) = 0,\ x \in \partial([0,1]^N) \end{cases} \quad (22) $$

where $-f(x) = \sum_{i=1}^{N}\left[g(x_i)\prod_{j\neq 1}x_j(1-x_j)\right] + \prod_{i=1}^{N}x_i(1-x_i)$ then $u(x) = \prod_{i=1}^{N}x_i(1-x_i)$ is the solution of Equation (22).

4.1.2 Example 2

Give a continuous function $-f(x_1,x_2) = x_2^2(1-x_2)\,g(x_1) + x_1^2(1-x_1)g(x_2) + x_1^2x_2^2(1-x_1)(1-x_2)$

$g(x) = -[\Gamma(\frac{1}{2})]^{-2}\left(\frac{1}{3}I_2 - 3I_3\right)$, where $I_3 = x^2(2A - \log(x)) + \frac{4+10x(1-3x)}{15\sqrt{1-x}}$

We consider the PDE for $(x_1,x_2) \in [0,1] \times [0,1]$,

$$ \begin{cases} {}_{x_1}D_T^{\frac{1}{2}}({}_0^C D_{x_1}^{\frac{1}{2}}u)(x_1,x_2) + {}_{x_2}D_T^{\frac{1}{2}}({}_0^C D_{x_2}^{\frac{1}{2}}u)(x_1,x_2) + u(x_1,x_2) = -f(x_1,x_2) \\ u(0,x_2) = u(1,x_2) = u(x_1,0) = u(x_1,1) = 0 \end{cases} \quad (23) $$

Then $u(x_1,x_2) = x_1^2(1-x_1)\,x_2^2(1-x_2)$ is the solution of Equation (23).

Consider the following high-dimensional case,

$$ \begin{cases} \sum_{i=1}^{N} {}_{x_i}D_T^{\frac{1}{2}}\left({}_0^C D_{x_i}^{\frac{1}{2}}u\right)(x) + u(x) = -f(x) \\ u(x) = 0,\ x \in \partial([0,1]^N) \end{cases} \quad (24) $$

where, $x = (x_1,x_2,\cdots,x_N) \in [0,1]^N$

$-f(x) = \sum_{i=1}^{N}\left[g(x_i)\prod_{j\neq 1}x_j^2(1-x_j)\right] + \prod_{i=1}^{N}x_i^2(1-x_i)$.

Then $u(x) = \prod_{i=1}^{N}x_i^2(1-x_i)$ is the solution of Equation (24).

4.1.3 Example 3

Give a continuous function $-f(x_1,x_2) = x_2(1-x_2)^2g(x_1) + x_1(1-x_1)^2g(x_2) + x_1x_2(1-x_1)^2(1-x_2)^2$

$g(x) = -[\Gamma(\frac{1}{2})]^{-2}(2I_1 - \frac{2}{3}I_2 + 3I_3)$,

We consider the PDE for $(x_1,x_2) \in [0,1] \times [0,1]$,

$$ {}_{x_1}D_T^{\frac{1}{2}}({}_0^C D_{x_1}^{\frac{1}{2}}u)(x_1,x_2) + {}_{x_2}D_T^{\frac{1}{2}}({}_0^C D_{x_2}^{\frac{1}{2}}u)(x_1,x_2) + u(x_1,x_2) = -f(x_1,x_2) $$
$$ u(0,x_2) = u(1,x_2) = u(x_1,0) = u(x_1,1) = 0 \quad (25) $$

Then $u(x_1,x_2) = x_1(1-x_1)^2\,x_2(1-x_2)^2$ is the solution of Equation (25).

Consider the following high-dimensional case,

$$ \begin{cases} \sum_{i=1}^{N} {}_{x_i}D_T^{\frac{1}{2}}\left({}_0^C D_{x_i}^{\frac{1}{2}}u\right)(x) + u(x) = -f(x) \\ u(x) = 0,\ x \in \partial([0,1]^N) \end{cases} \quad (26) $$

where, $x = (x_1,x_2,\cdots,x_N) \in [0,1]^N$

$f(x) = \sum_{i=1}^{N}[g(x_i)\prod_{j\neq 1}x_j(1-x_j)^2$

Then $u(x) = \prod_{i=1}^{N}x_i(1-x_i)^2$ is the solution of Equation (26).

4.2 Numerical results

4.2.1 2-dimensional fractional order differential equations

The network we use to solve these 2-dimensional fractional order differential equations is a stack of three blocks (six fully connected layers), an input layer and an output layer. Table 1 shows that the deep Ritz method gives a more accurate solution when more blocks are used. However, due to the complexity of the equations, overfitting occurs during the iterations when the number of blocks increases to 4. The total error is $I(u,\beta)$ and the L_2 error which can be approximated by the mean squared error, is between u_* and, u_h with u_* being the exact solution and u_h being the approximate solution, resulting in the smallest $I(u,\beta)$. Mean square error is one of the most commonly used loss functions in regression tasks, whose basic form is $\frac{1}{N}\sum_{i=1}^{N}(u_n - u_*)^2$. We set $\beta = 500$ in $I(u,\beta)$.

In Table 1, the L_2 error for different numbers of blocks are presented. The L_2 error of three examples decrease to the minimum values: 7.3119e-06, 1.7677e-06, 1.8367e-06, respectively, as the number of blocks increases to 3, but as the number of blocks reaches 4, overfitting occurs. As a result, we choose 3 blocks for accuracy.

Table 1. L_2 error for the deep Ritz method (d = 2).

Blocks No.	L_2 error (Example 1)	L_2 error (Example 2)	L_2 error (Example 3)
2	1.0453e-05	1.5997e-05	1.0747e-05
3	7.3119e-06	1.7677e-06	1.8367e-06
4	8.7139e-05	5.2528e-05	2.1828e-05

The value of m (width of layers that from blocks) is set to be 10. Table 2 displays total errors and L_2 errors of the deep Ritz method in terms of m. Both of them decrease to the minimum values: -4.833e-03 and 7.3119e-06 (Example 1), -1.586e-03 and 1.7677e-06 (Example 2), -1.074e-03 and 1.8367e-06 (Example 3) as the value of m increases to 10, as expected. Therefore, we set m = 10 for simplicity. This means a 2-dimensional vector becomes a 10-dimensional vector through all of the blocks.

At each step of the stochastic gradient descent iteration, we sample 1000 points in Ω and 100 points at each hyperplane of $\partial\Omega$ to do numerical integration. Next part, we will show two different ways of sampling points. The first way is to take points randomly within the definition domain and make the coordinate values of them uniformly distributed, while the second way is to divide the definition domain into several equal parts and then take points randomly in each small part and make the coordinate values of them uniformly distributed. In the following, we will compare the effect of these two ways of taking points in terms of the minimum L_2 error during the iteration.

From Table 3, we can find that the minimum L_2 errors of the second way that can be achieved during the iteration is smaller than that of the first way by 3.1411e-06 in Example 1, 6.88e-06 in Example 2, 3.7194e-06 in Example 3, which means that the accuracy of the second method is much higher than that of the first method. The second point-taking method not only preserves the randomness of the data, but also increases the stability of the data. We improve the point-taking method in E *et al.* [33] and explore a new point-taking method. Recently, various efforts have been made to exploit better point-taking techniques to increase effectiveness. The importance of point-taking methods is showed in Daw *et al.* [41-43]

Moreover, the results are shown in Fig. 3 in which (a), (d), (g) are the approximate solutions using the first point-taking method and (b), (e), (h) are the approximate solutions using the second point-taking method. The explicit solutions of Example 1, Example 2, Example 3 are (c), (f), (i) which are drawn by Python. It is obvious that the second point-taking method is better than the first one, especially in terms of symmetry, which further confirms the superiority of the second point-taking method.

4.2.2 10-dimensional fractional order differential equations

In this subsection, the performance of the deep Ritz method will be investigated in a relatively high dimension. We will find that the deep Ritz method is likewise effective in high dimensions and the superiority of the variational method in the high-dimensional case.

In this network structure, a stack of 3 blocks (six fully connected layers) and an output layer in which m = 10 are employed to solve the 10-dimensional fractional order differential equations. At each step of the stochastic gradient descent iteration, we sample 1000 points in Ω, 100 points at each hyperplane of $\partial\Omega$ and make them uniformly distributed for numerical integration. We set $\beta = 1000$ in $I(u, \beta)$. The L_2 and u_h is shown in Table 4. We can see that the L_2 error of Example 1 drops to a minimum of 7.9703e-12 when using 3 blocks, while the overfitting phenomenon still occurs with more blocks used, the same as what we have learned from Table 1. As a result, we choose 3 blocks for accuracy.

Table 2. Total errors and L_2 errors with different m.

	Example 1		Example 2		Example 3	
m	Total error	L_2 error	Total error	L_2 error	Total error	L_2 error
6	-0.003638	0.0001087	0.000292	7.3361e-05	-0.000320	1.7782e-05
8	-0.004281	4.5578e-05	-0.000537	1.2421e-05	-0.000523	1.2851e-05
10	-0.004833	7.3119e-06	-0.001586	1.7677e-06	-0.001074	1.8367e-06

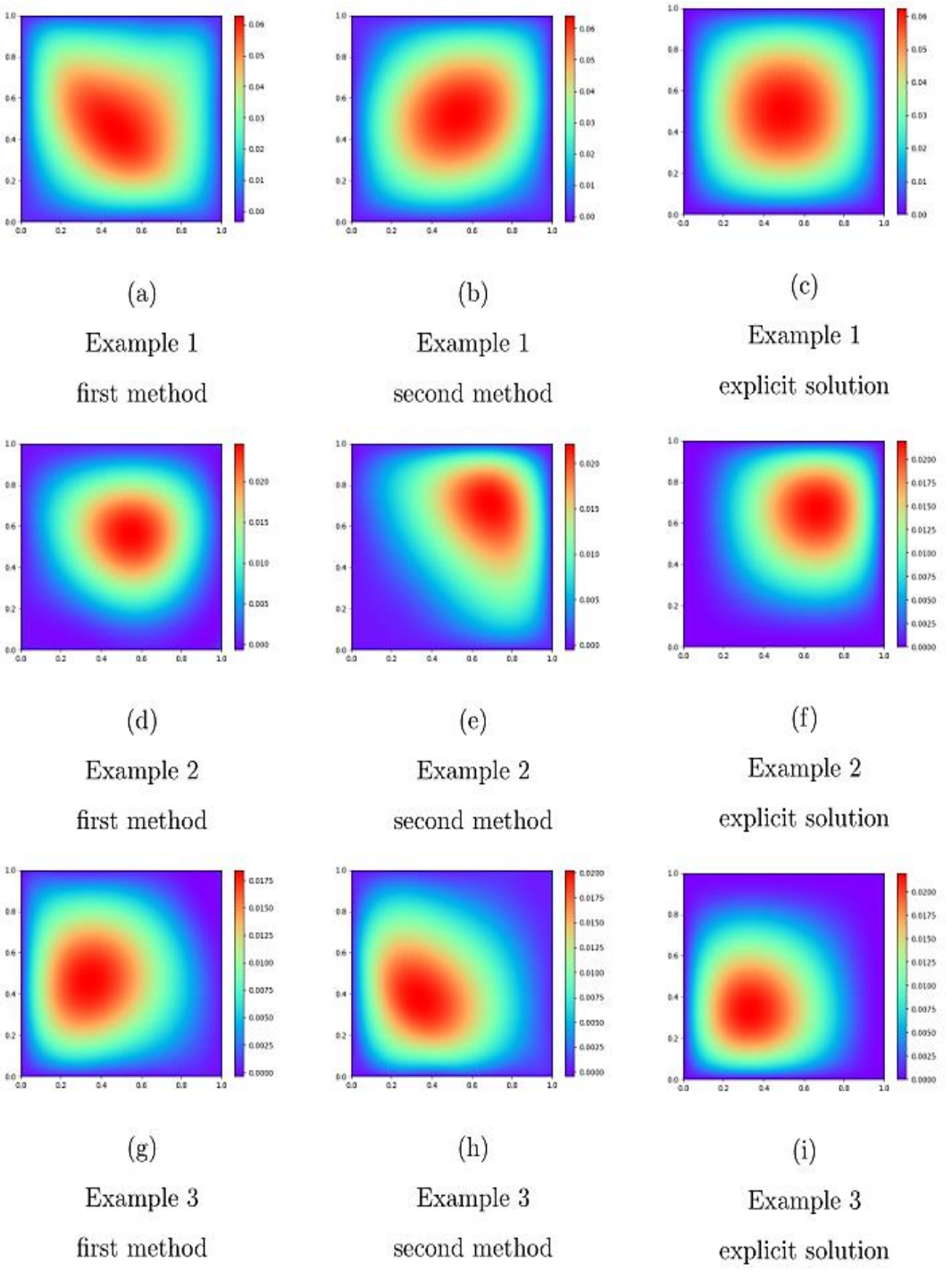

Fig. 3 Approximate solutions and explicit solutions of examples. (a), (d), (g) are the approximate solutions using the first point-taking method. (b), (e), (h) are the approximate solutions using the second point-taking method. (c), (f), (i) are the explicit solutions.

Table 3. The minimum L_2 errors of two different ways for taking points.

	L_2 error (the first way)	L_2 error (the second way)
Example 1	1.0453e-05	7.3119e-06
Example 2	8.6477e-06	1.7677e-06
Example 3	5.5561e-06	1.8367e-06

Table 4. L_2 error of Example 1 for deep Ritz method (d = 10).

Blocks No.	L_2 error
2	5.9512e-11
3	7.9703e-12
4	2.3824e-10

When we choose a loss function for a neural network, the most straightforward choice is the L_2 error. L_2 error is the most commonly used loss function. No matter what kind of

machine learning model we train, our goal is to minimize the loss function. When the approximate solution is exactly equal to the true solution, L_2 error reaches a minimum value of 0. When L_2 error reaches to a minimum value during iteration, this means that the approximate solution at that point is the best approximate solution to the true solution. However, we find that using total error as the loss function works better than using L_2 error as the loss function, that is, the minimum L_2 error that can be achieved during the iteration when using total error as loss function is smaller than the minimum L_2 error that can be achieved when using L_2 error as loss function. Here we conduct five experiments for each of the three examples, and present the experimental data in tables and display them as follows, where lf_1, lf_2 mean total error and L_2 error, respectively.

From Table 5, we can learn that the minimum L_2 error that can be achieved during the iteration when using lf_1 is smaller than that when using lf_2 by a maximum of 3.92311e-07 in Example 1, 2.51945e-08 in Example 2, 2.5367e-08 in Example 3, and the minimum L_2 error achieved by choosing lf_1 as the loss function is at most five orders of magnitude smaller than that achieved by choosing lf_2. The ability to obtain better approximate solutions using energy functional as a loss function implies the superiority of the variational method in approximating the solution of high-dimensional partial differential equations.

Next, we select the best one from the five experiments of each example for presentation. Example 1 reaches to its minimum total error 5.7006e-08 at the 48900th iteration, when the corresponding L_2 error is 5.1271e-12. Example 2 reaches to its minimum total error 5.0535e-08 at the 48000th iteration, when the corresponding L_2 error is 4.5078e-12. Example 3 reaches to its minimum total error 1.0800e-07 at the 49000th iteration, when the corresponding L_2 error is 1.0050e-11. Fig. 4 shows the training processes.

Next, we compute many different values to measure the effectiveness of the deep Ritz method, and the results show that the deep Ritz method works well on all three examples and gets better as the number of iteration steps increases. Details are as follows. Table 6 shows important numerical results of solving 10-dimensional fractional order differential equations by the deep Ritz method, including the mean of minimum total error, the standard deviation of minimum total error, the mean of L_2error between u_* and u_h and the standard deviation of L_2 error between u_* and u_h. The calculated results presented in the table are based on data from five independent trials. Since each trial is randomized, we take the first 48,000 iterations for the calculation. The table below shows the rate of decline.

Table 5. The minimum L_2 errors of two different loss functions.

Example 1		Example 2		Example 3	
L_2 error lf_1	L_2 error lf_2	L_2 error lf_1	L_2 error lf_2	L_2 error lf_1	L_2 error lf_2
7.9703e-12	6.8242e-11	1.2220e-11	5.4052e-11	4.5863e-11	2.5368e-11
5.1271e-12	1.8529e-10	6.6265e-12	2.0951e-10	6.2266e-11	1.1110e-09
9.4363e-12	3.5121e-09	8.3893e-12	4.4110e-10	1.0240e-11	2.0903e-09
9.6136e-12	2.5377e-08	4.5078e-12	2.0085e-08	2.4830e-11	9.2478e-09
9.4925e-12	3.9232e-07	2.7545e-11	2.5222e-08	1.0050e-11	2.5377e-08

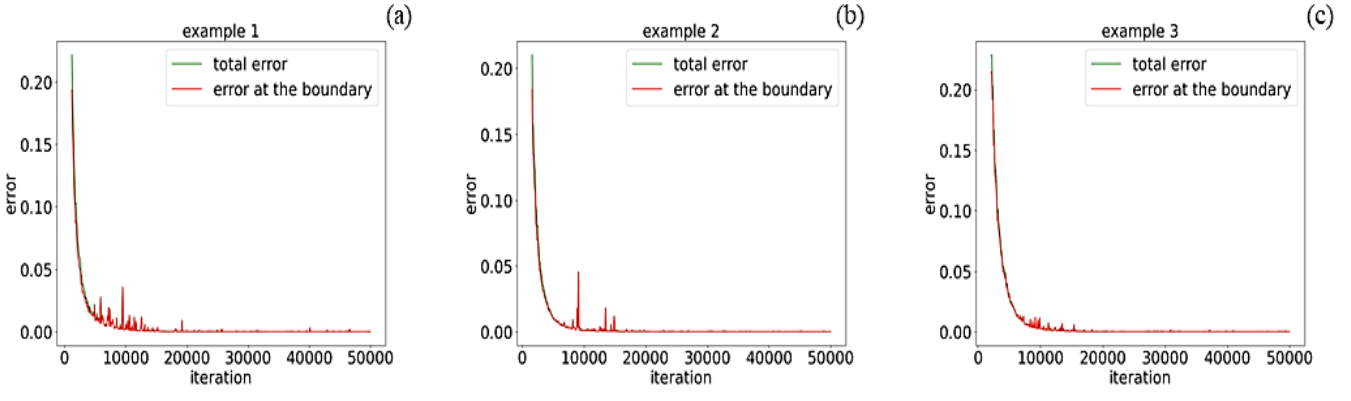

Fig. 4 The total error and the error at the boundary of three 10-d examples during the training process. (a), (b), (c) correspond to example 1, example 2, example 3, respectively. The x-axis represents the iteration steps. The green curves show the total error. The red curves show the error on the boundary. Example 1 reaches to its minimum total error 5.7006e-08 at the 48900th iteration, when the corresponding L_2 error is 5.1271e-12. Example 2 reaches to its minimum total error 5.0535e-08 at the 48000th iteration, when the corresponding L_2 error is 4.5078e-12. Example 3 reaches to its minimum total error 1.0800e-07 at the 49000th iteration, when the corresponding L_2 error is 1.0050e-11.

Table 6. Numerical results of deep Ritz method (10d).

	Iteration	Mean of minimum total error	Standard deviation of minimum total error	Mean of L_2 error	Standard deviation of L_2 error
Example 1	10000	5.0825e-03	2.8322e-03	4.7157e-07	2.6186e-07
	20000	1.5019e-03	1.5889e-03	1.4763e-07	1.5619e-07
	30000	3.5727e-05	3.1844e-05	3.5292e-09	3.1789e-09
	40000	1.2101e-06	1.2891e-06	1.1670e-10	1.2638e-10
	48000	5.7802e-07	3.7673e-07	5.4011e-11	3.5817e-11
Example 2	10000	6.0194e-03	7.0328e-03	5.6924e-07	6.9069e-07
	20000	3.8639e-04	5.4615e-04	3.7007e-08	5.4113e-08
	30000	3.8032e-05	6.2686e-05	3.6926e-09	6.2077e-09
	40000	8.0303e-06	1.2582e-05	7.8525e-10	1.2476e-09
	48000	4.9442e-07	3.8223e-07	4.3862e-11	3.4025e-11
Example 3	10000	1.5026e-02	9.2314e-03	1.4270e-06	8.8211e-07
	20000	5.0197e-04	5.5411e-04	4.5708e-08	5.0241e-08
	30000	1.7547e-04	1.9689e-04	1.7378e-08	1.9635e-08
	40000	5.0838e-06	6.2816e-06	4.8588e-10	6.1729e-10
	48000	5.4062e-07	2.9343e-07	5.1037e-11	2.5406e-11

The mean values of minimum total error and the mean values of L_2 error for all three examples decrease to 10^{-7} and 10^{-11}, respectively, after 48,000 iterations, which reflects the effectiveness of the method, and the standard deviation of the two values also decrease to 10^{-7} and 10^{-11}, respectively, after 48,000 iterations, which reflects the stability of the method.

4.2.3 100-dimensional fractional order differential equations

We expand the dimension of the fractional order differential equation to 100. A stack of 3 blocks (six fully connected layers) and an output layer in which m = 100 are employed to solve the fractional order differential equations in one hundred dimensions. At each step of the stochastic gradient descent iteration, we sample 1000 points in Ω, 100 points at each hyperplane of $\partial\Omega$ and make them uniformly distributed for numerical integration. β is set to be 1000 in $I\ (u, \beta)$.

The L_2 error between u_* and u_h is shown in Table 7. We can see that the L_2 error of Example 1 decreases to the minimum value of 1.8057e-07 when using 3 blocks, while increases to 4.5320e-06 with 4 blocks used. The deep Ritz method gives more accurate solutions of 100-dimensional fractional order differential equations when using more blocks while the overfitting phenomenon still occurs with more blocks used, the same as what we have learned from the 10-dimensional case. Therefore, we choose 3 blocks for accuracy.

When discussing the 10-dimensional fractional order differential equation, we find that using total error as the loss function gives a better approximate solution than using L_2 error as the loss function. We wonder if the same is true for the 100-dimensional fractional order differential equation, thus we conduct experiments for each of the three examples and find that using total error as the loss function still gives a better approximate solution in the 100-dimensional case, which shows that the variational method is still superior. The experimental data are presented in Table 8 and we can find that the minimum L_2 errors of three examples with lf_1 used are all one order of magnitude smaller than those with lf_2 used.

Table 7. L_2 error of Example 1 for deep Ritz method (d = 100).

Blocks No.	L_2 error
2	3.5622e-07
3	1.8057e-07
4	4.5320e-06

Table 8. The minimum L_2 errors of two different loss functions.

	L_2 error lf_1	L_2 error lf_2
Example 1	1.8057e-07	1.25928e-06
Example 2	3.7236e-07	4.0062e-06
Example 3	4.5324e-07	1.1656e-06

When the loss function is energy functional, we conduct five experiments for each of the three examples. Then, we select the best one from the five experiments of each example for presentation. Example 1 reaches to its minimum total error 0.0181 at the 14800th iteration, when the corresponding L_2 error is 1.8057e-07. Example 2 reaches to its minimum total error 0.0372 at the 13400th iteration, when the corresponding L_2 error is 3.7236e-07. Example 3 reaches to its minimum total error 0.0193 at the 14400th iteration, when the corresponding L_2 error is 1.9251e-07. Fig. 5 shows the

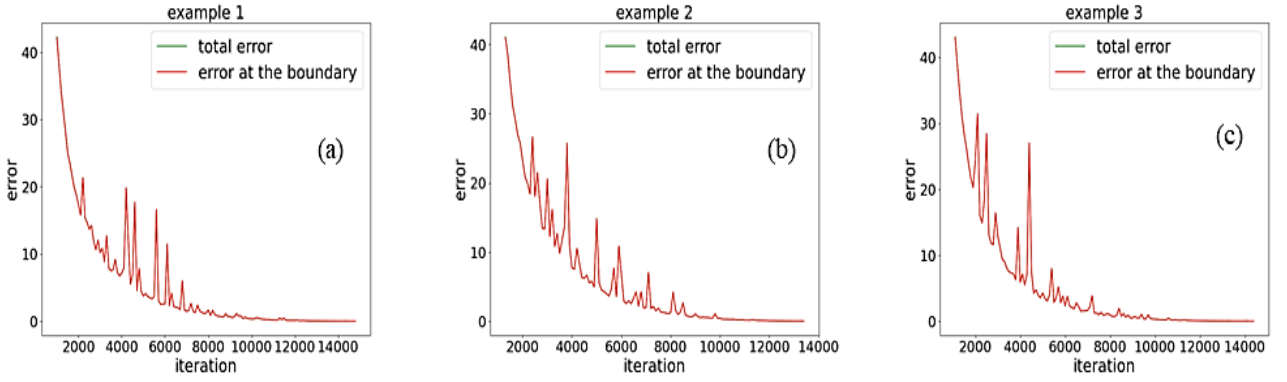

Fig. 5 The total error and the error at the boundary of three 100-d examples during the training process. (a), (b), (c) correspond to example 1, example 2, example 3, respectively. The x-axis represents the iteration steps. The green curves show the total error. The red curves show the error on the boundary. Example 1 reaches to its minimum total error 0.0181 at the 14800th iteration, when the corresponding L_2 error is 1.8057e-07. Example 2 reaches to its minimum total error 0.0372 at the 13400th iteration, when the corresponding L_2 error is 3.7236e-07. Example 3 reaches to its minimum total error 0.0193 at the 14400th iteration, when the corresponding L_2 error is 1.9251e-07.

Table 9. Numerical results of deep Ritz method (100d).

	Iteration	Mean of minimum total error	Standard deviation of minimum total error	Mean of L_2 error	Standard deviation of L_2 error
Example 1	2400	1.5959e+01	1.3592e+00	1.5905e-04	1.3758e-05
	4800	4.8299e+00	1.5803e+00	4.8234e-05	1.5776e-05
	7200	1.9787e+00	6.2171e-01	1.9752e-05	6.2488e-06
	9600	5.0477e-01	3.5089e-01	5.0368e-06	3.5058e-06
	12000	8.5169e-02	1.4944e-02	8.4910e-07	1.4988e-07
Example 2	2400	1.5809e+01	2.5068e+00	1.5744e-04	2.5324e-05
	4800	4.3958e+00	8.9669e-01	4.3828e-05	8.9114e-06
	7200	1.4456e+00	3.9722e-01	1.4424e-05	3.9590e-06
	9600	4.9195e-01	1.3495e-01	4.9097e-06	1.3518e-06
	12000	7.8470e-02	2.7055e-02	7.8262e-07	2.7052e-07
Example 3	2400	1.4108e+01	2.3338e+00	1.4047e-04	2.3282e-05
	4800	4.1684e+00	7.3563e-01	4.1523e-05	7.3358e-06
	7200	1.3782e+00	1.9085e-01	1.3751e-05	1.9078e-06
	9600	5.3214e-01	3.4050e-01	5.3249e-06	3.4166e-06
	12000	8.1113e-02	1.8958e-02	8.0764e-07	1.8801e-07

training processes.

Then, we still compute many different values to measure the effectiveness of the deep Ritz method, and the results show that the deep Ritz method still works well on 100-dimensional equations and gets better as the number of iteration steps increases. Details are as follows. Table 9 shows the important numerical results of solving the 100-dimensional fractional order differential equations by the deep Ritz method, including the mean of minimum total error, the standard deviation of minimum total error, the mean of L_2 error between u_* and u_h and the standard deviation of L_2 error between u_* and u_h. The calculated results presented in the table are based on data from five independent trials. Since each trial is randomized, we take the first 12000 iterations for the calculation. The table below shows the rate of decline.

The mean values of minimum total error and the mean values of L_2 error for all three examples decrease to 10^{-2} and 10^{-7}, respectively, after 12,000 iterations, which shows the effectiveness of the method, and the standard deviation of the two values also decrease to 10^{-2} and 10^{-7}, respectively, after 12,000 iterations, which shows the stability of the method in the 100-dimensional case. Compared to the 10-dimensional fractional order differential equations, the deep Ritz method performs slightly less well on the 100-dimensional fractional order differential equations, but the L_2 error for the three examples still reaches to 10^{-7} at more than 10,000 steps.

5. Conclusion

For solving fractional order differential equations, we combine the variational method with ResNet. To show the

excellent fitness of the variational method with ResNet, we do a comparison between using energy functional as loss function and using L_2 error as loss function, and the results show that the variational method performs better in approximating solutions of equations. Additionally, we use a new point-taking method which makes the approximate solution more accurate. Some tables and figures are used to show the results of our experiments so that the effects of different methods can be intuitively displayed. We provide a deep learning-based method for approximating solutions of fractional order differential equations in even quite high dimensions. Owing to the curse of dimensionality, there are very few practical high dimensional schemes that have been proposed in literature. Our method can also be applied to different forms of fractional order differential equations, such as those containing only Riemann-Liouville fractional derivatives or Caputo fractional derivatives.

Conflict of Interest

There is no conflict of interest.

Supporting Information

Not applicable.

References

[1] J. Sabatier, O. Agrawal, J. Machado, Advances in fractional calculus: theoretical developments and applications in physics and engineering, *Biochemical Journal*, 2007, **36**, 97-103, doi: 10.1007/978-1-4020-6042-7.

[2] T. Atanacković, S. Pilipovic, B. Stankovic, D. Zorica, Fractional calculus with applications in mechanics: vibrations and diffusion processes, Publisher, Drug Dev Ind Pharm, 2014.

[3] D. Baleanu, J. Machado, A. Luo, Fractional dynamics and control, Publisher, Springer, 2012.

[4] F. Gao, General fractional calculus in non-singular power-law kernel applied to model anomalous diffusion phenomena in heat transfer problems, *Thermal Science*, 2017, **21**, 11-18, doi: 10.2298/tsci170310194g.

[5] F. Mainardi, Fractional calculus: some basic problems in continuum and statistical mechanics, Publisher, Springer-Verlag, 2012.

[6] D. Baleanu, M. Inc, A. Yusuf, A. I. Aliyu, Time fractional third-order evolution equation: symmetry analysis, explicit solutions, and conservation laws, *Journal of Computational and Nonlinear Dynamics*, 2018, **13**, 021011, doi: 10.1115/1.4037765.

[7] D. Baleanu, A. Jajarmi, M. Hajipour, A new formulation of the fractional optimal control problems involving mittag–leffler nonsingular kernel, *Journal of Optimization Theory and Applications*, 2017, **175**, 718-737, doi: 10.1007/s10957-017-1186-0.

[8] F. W. Liu, P. H. Zhuang, Q. X. Liu, Numerical methods of fractional partial differential equations and its application; Publisher, Science Press, 2015.

[9] M. M. Meerschaert, Finite difference approximations for fractional advection-dispersion flow equations, *Journal of Computational and Applied Mathematics*, 2004, **172**, 65-77, doi: 10.1016/j.cam.2004.01.033.

[10] C. Çelik, Crank-Nicolson method for the fractional diffusion equation with the Riesz fractional derivative, *Journal of Computational Physics*, 2012, **231**, 1743-1750, doi: 10.1016/j.jcp.2011.11.008.

[11] V. J. Ervin, J. P. Roop, Variational formulation for the stationary fractional advection dispersion equation, *Numerical Methods for Partial Differential Equations*, 2006, **22**, 558-576, doi: 10.1002/num.20112.

[12] W. Bu, Crank-Nicolson ADI Galerkin finite element method for two-dimensional fractional FitzHugh-Nagumo monodomain model, *Applied Mathematics and Computation*, 2015, **257**, 355-364, doi: 10.1016/j.amc.2014.09.034.

[13] W. Deng, Finite element method for the space and time fractional Fokker-Planck equation, *SIAM Journal on Numerical Analysis*, 2009, **47**, 204-226, doi: 10.1137/080714130.

[14] M. Li, C. Huang, W. Ming, Mixed finite-element method for multi-term time-fractional diffusion and diffusion-wave equations, *Computational and Applied Mathematics*, 2018, **37**, 2309-2334, doi: 10.1007/s40314-017-0447-8.

[15] P. Henry-Labordère, N. Oudjane, X. Tan, N. Touzi, X. Warin, Branching diffusion representation of semilinear PDEs and Monte Carlo approximation, *Annales De l'Institut Henri Poincaré, Probabilités et Statistiques*, 2019, **55**, 184-210, doi: 10.1214/17-aihp880.

[16] X. Warin, Nesting Monte Carlo for high-dimensional non-linear PDEs, *Monte Carlo Methods and Applications*, 2018, **24**, 225-247, doi: 10.1515/mcma-2018-2020.

[17] B. F. Vajargah, Monte Carlo method for finding the solution of dirichlet partial differential equations, *Applied Mathematical Sciences*, 2007, **1**, 453-462.

[18] A. Kebaier, Statistical Romberg extrapolation: a new variance reduction method and applications to option pricing, *The Annals of Applied Probability*, 2005, **15**, 2681-2705, doi: 10.1214/105051605000000511.

[19] M. B. Giles, Multilevel Monte Carlo path simulation, *Operations Research*, 2008, **56**, 607-617, doi: 10.1287/opre.1070.0496.

[20] M. B. Giles, C. Reisinger, Stochastic finite differences and multilevel Monte Carlo for a class of SPDEs in finance, *SIAM Journal on Financial Mathematics*, 2012, **3**, 572-592, doi: 10.1137/110841916.

[21] K. A. Cliffe, M. B. Giles, R. Scheichl, A. L. Teckentrup, Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients, *Computing and Visualization in Science*, 2011, **14**, 3-15, doi: 10.1007/s00791-011-0160-x.

[22] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, 2016.

[23] G. Hinton, L. Deng, D. Yu, Deep neural networks for acoustic modeling in speech recognition, *Signal Processing Magazine*, 2012, **29**, 82-97.

[24] J. Han, E. Weinan, Deep learning approximation for stochastic control problems, Deep Reinforcement Learning Workshop, 2016.

[25] G. Carleo, M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science*, 2017, **355**, 602-606, doi: 10.1126/science.aag2302.

[26] N. J. Guliyev, Approximation capability of two hidden layer feedforward neural networks with fixed weights, Neurocomputing, *Neurocomputing*, 2018, **316**, 262-269, doi: 10.1016/j.neucom.2018.07.075.

[27] C. Beck, E. Weinan, A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, *Journal of Nonlinear Science*, 2019, **29**, 1563-1619, doi: 10.1007/s00332-018-9525-3.

[28] Y. Khoo, L. Ying, SwitchNet: A neural network model for forward and inverse scattering problems, *SIAM Journal on Scientific Computing*, 2019, **41**, A3182-A3201, doi: 10.1137/18m1222399.

[29] J. Sirignano, DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, 2018, **375**, 1339-1364, doi: 10.1016/j.jcp.2018.08.029.

[30] H. Qu, X. Liu, A numerical method for solving fractional differential equations by using neural network, *Advances in Mathematical Physics*, 2015, **2015**, 1-12, doi: 10.1155/2015/439526.

[31] A. Jafarian, M. Mokhtarpour, D. Baleanu, Artificial neural network approach for a class of fractional ordinary differential equation, *Neural Computing and Applications*, 2017, **28**, 765-773, doi: 10.1007/s00521-015-2104-8.

[32] M. Gulian, M. Raissi, P. Perdikaris, G. Karniadakis, Machine learning of space-fractional differential equations, *SIAM Journal on Scientific Computing*, 2019, **41**, A2485-A2509, doi: 10.1137/18m1204991.

[33] E. Weinan, B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, *Communications in Mathematics and Statistics*, 2018, **6**, 1-12, doi: 10.1007/s40304-018-0127-z.

[34] K. Li, K. Tang, T. Wu, Q. Liao, D3M: A deep domain decomposition method for partial differential equations, *IEEE Access*, 2019, **8**, 5283-5294, doi: 10.1109/ACCESS.2019.2957200.

[35] E. Kharazmi, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, *Computer Methods in Applied Mechanics and Engineering*, 2021, **374**, 113547, doi: 10.1016/j.cma.2020.113547.

[36] Y. Gu, M. K. Ng, Deep ritz method for the spectral fractional Laplacian equation using the caffarelli: silvestre extension, *SIAM Journal on Scientific Computing*, 2022, **44**, A2018-A2036, doi: 10.1137/21m1442516.

[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 27-30, 2016, Las Vegas, NV, USA. IEEE, 2016, 770-778, doi: 10.1109/CVPR.2016.90.

[38] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014.

[39] E. Weinan, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Communications in Mathematics and Statistics*, 2017, **5**, 349-380, doi: 10.1007/s40304-017-0117-6.

[40] A. A. Kilbas, H. M. Srivastava, J. J. Trujillo, Preface. Theory and applications of fractional differential equations. Amsterdam, *Elsevier*, 2006, doi: 10.1016/s0304-0208(06)80001-0.

[41] A. Daw, J. Bu, S. F. Wang, P. Perdikaris, A. Karpatne, Rethinking the importance of sampling in physics-informed neural networks, 2022.

[42] K. Tang, Adaptive deep density approximation for Fokker-Planck equations, *Journal of Computational Physics*, 2022, **457**, 111080, doi: 10.1016/j.jcp.2022.111080.

[43] C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, *Computer Methods in Applied Mechanics and Engineering*, 2023, **403**, 115671.

[44] F. Jiao, Y. Zhou, Existence results for fractional boundary value problem via critical point theory, *International Journal of Bifurcation and Chaos*, 2012, **22**, 1250086, doi: 10.1142/s0218127412500861.

[45] J. Bu, A. Karpatne, Quadratic residual networks: a new class of neural networks for solving forward and inverse problems in physics involving PDEs, *Proceedings of the 2021 SIAM International Conference on Data Mining*, 2021.

Author Information

Juan Yang *is an associate professor of Mathematics at School of Science, Beijing University of Posts and Telecommunications. She got her Ph. D supervision of Prof. Tusheng Zhang at Department of Mathematics, the University of Manchester. Her research interests lie at stochastic partial differential equations, fractional differential equations and related numerical analysis. More precisely, she focuses on the wellposedness for several types of stochastic partial differential equations with singular terms and fractional differential equations involving the existence, uniqueness and smoothness of the solutions, the long-time behaviour of the solutions for stochastic partial differential equations including invariant measures, large deviation principle as well as moderate deviation principle, numerical algorithms for the solutions of (stochastic) partial differential equations, fractional differential equations. The related papers are published in journals like Journal of Theoretical Probability, Electronic Journal of Probability, Electronic Communications in Probability, Stochastics and Dynamics, Acta Mathematicae Applicatae Sinica, Journal of Mathematical Analysis and Applications, and so on.*

