Neural Networks 161 (2023) 185–201

Contents lists available at ScienceDirect

Neural Networks

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

Accelerating gradient descent and Adam via fractional gradients

Yeonjong Shin [a,*], Jérôme Darbon [b], George Em Karniadakis [b,c]

[a] Department of Mathematical Sciences, KAIST, Daejeon 34141, South Korea
[b] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA
[c] School of Engineering, Brown University, Providence, RI 02912, USA

ARTICLE INFO

Article history:
Received 1 July 2022
Received in revised form 6 November 2022
Accepted 4 January 2023
Available online 11 January 2023

Keywords:
Caputo fractional derivative
Non-local calculus
Optimization
Adam
Neural networks

ABSTRACT

We propose a class of novel fractional-order optimization algorithms. We define a fractional-order gradient via the Caputo fractional derivatives that generalizes integer-order gradient. We refer it to as the Caputo fractional-based gradient, and develop an efficient implementation to compute it. A general class of fractional-order optimization methods is then obtained by replacing integer-order gradients with the Caputo fractional-based gradients. To give concrete algorithms, we consider gradient descent (GD) and Adam, and extend them to the Caputo fractional GD (`CfGD`) and the Caputo fractional Adam (`CfAdam`). We demonstrate the superiority of `CfGD` and `CfAdam` on several large scale optimization problems that arise from scientific machine learning applications, such as ill-conditioned least squares problem on real-world data and the training of neural networks involving non-convex objective functions. Numerical examples show that both `CfGD` and `CfAdam` result in acceleration over GD and Adam, respectively. We also derive error bounds of `CfGD` for quadratic functions, which further indicate that `CfGD` could mitigate the dependence on the condition number in the rate of convergence and results in significant acceleration over GD.

© 2023 Elsevier Ltd. All rights reserved.

1. Introduction

The gradient descent (GD) method to optimize a function dates back to Cauchy in 1800s (Lemaréchal, 2012) and is one of the most fundamental approaches in optimization. It is an iterative algorithm to find a stationary point to an objective function. Due to its simplicity and scalability, GD and its variants have been widely used in many research fields, in particular, machine learning (LeCun, Bengio, & Hinton, 2015; Ruder, 2016). Countless works have been devoted to GD-based methods, and the amount of literature is huge due to its importance. Interested readers can consult the large number of textbooks on the basics of optimization, e.g. Bonnans, Gilbert, Lemarechal, and Sagastizabal (2006), Nesterov (2004) and Nocedal and Wright (2006).

Fractional calculus has been successfully employed in describing physical phenomena, e.g., anomalous transport, which classical models cannot capture (see D'Elia, Du, Glusa, Gunzburger, Tian, & Zhou, 2020; D'Elia, Gulian, Olson, & Karniadakis, 2021 and references therein). More recently, fractional calculus has been adapted in optimization algorithms (Chen, Gao, Wei, & Wang, 2017; Pu, Zhou, Zhang, Zhang, Huang, & Siarry, 2013; Sheng, Wei, Chen, & Wang, 2020; Wang, Wen, Gou, Ye, & Chen, 2017; Wei, Kang, Yin, & Wang, 2020). Since fractional derivatives are extensions of integer-order derivatives, one may naturally consider fractional gradient descent (FGD) as a generalization of gradient descent method. Many variants of FGD have been proposed and shown to be effective in some applications (Cheng, Wei, Chen, Li, & Wang, 2017; Khan, Naseem, Malik, Togneri, & Bennamoun, 2018; Sheng et al., 2020). Yet, many theoretical questions remain elusive. For example, GD seeks to find an optimum by taking discrete steps in the direction of steepest descent. However, which direction FGD follows is not well understood. The present work is motivated by the lack of mathematical understanding on the use of fractional calculus on optimization.

We briefly review the existing literature on fractional calculus-based optimization methods. It has been pointed out in Pu et al. (2013) and Wang et al. (2017) that the set of stationary points of fractional gradient is different from the one of integer-order gradient. In practice, integer-order stationary points are often sought. To remedy this issue, Wei et al. (2020) proposed several heuristic variants of FGD. These heuristics are useful for designing FGD algorithms, yet no theoretical guarantees were provided. While Wang et al. (2017) provided a convergence analysis, the results rely on multiple crude assumptions, such as uniform boundedness, which are not generally satisfied for many applications. Another approach is to generalize the gradient flow, a continuum version of GD, to fractional time scale gradient flow (Hai & Rosenfeld, 2020; Liang, Wang, & Yin, 2020). This approach requires one to discretize the continuous flow appropriately to yield a numerical method. Finally, we note that Nagaraj (2020) presents an abstract framework for convergence analysis

* Corresponding author.
E-mail address: yeonjong_shin@kaist.ac.kr (Y. Shin).

https://doi.org/10.1016/j.neunet.2023.01.002
0893-6080/© 2023 Elsevier Ltd. All rights reserved.

of certain non-local calculus (Mengesha & Spector, 2015) based optimization.

In this work, motivated by a mathematical analysis (Theorem 2.5), we propose a class of optimization algorithms that uses the Caputo fractional derivatives. The Caputo fractional derivative (Caputo, 1967) is one of the most popular fractional derivatives and is widely employed in modeling various physical phenomena, especially for initial value problems.

The main findings are summarized as follows:

- In Theorem 2.5, we show that the steepest descent direction of a smoothing of the objective function can be expressed in terms of Caputo fractional derivatives. This serves as our theoretical motivation on using the Caputo fractional derivatives in optimization.
- By extending the Caputo fractional derivative with respect to the Cartesian coordinate (Tarasov, 2008), *the Caputo fractional-based gradient* (6) is defined, which requires fractional orders $\boldsymbol{\alpha}$, degree of smoothing $\boldsymbol{\beta}$, and integral terminals $\mathbf{c}$ to be specified. An efficient implementation using the Gauss–Jacobi quadrature (9) is also developed.
- We propose a class of fractional-order optimization methods that replaces integer-order gradients with the Caputo fractional-based gradients. In particular, we present the Caputo fractional GD (`CfGD`) and the Caputo fractional Adam (`CfAdam`) that generalize GD and Kingma and Ba (2014), respectively.
- We analyze `CfGD` for quadratic functions. Three choices of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{c}$ are considered, and the resulting three versions are referred to as `CfGD-v1`, `CfGD-v2`, and `CfGD-v3`, respectively.

For quadratic objective functions, it is well known that GD converges linearly and the rate of convergence critically depends on the condition number of the objective function. A natural question is whether `CfGD` can mitigate the dependence on the condition number in the rate of convergence. In Fig. 1, we illustrate the performance of `CfGD-v2` for a simple quadratic objective function $f(x, y) = 5x^2 + 0.5y^2$, whose minimizer is the origin, marked as $(*)$. The trajectories of (top) GD and (bottom) `CfGD-v2` are reported and they both start at $(1, -10)$. We see that `CfGD-v2` finds the optimal solution within the error of machine precision in merely four iterations (Theorem 3.4). GD behaves as expected showing a linear convergence. However, since its rate depends on the condition number, more iterations are needed to reach the machine precision error. Combined with more examples in Section 4, we found that `CfGD-v2` effectively mitigates the dependence on the condition number in the convergence rate and yields a significant acceleration.

For non-convex objective functions, we consider two tasks of training neural networks. One is the function approximation and the other is the physics-informed neural networks (PINNs) for the inverse Helmholtz equation. We found that `GfCD` outperforms GD in function approximation, and `CfAdam` outperforms Adam in the inverse PINN problem.

The rest of this paper is organized as follows. After describing the problem setup and introducing some preliminaries, the Caputo fractional-based gradient is defined and then a general class of fractional-order optimization methods is introduced in Section 2. A convergence analysis of `CfGD` is presented Section 3. Numerical examples are provided in Section 4 to verify our theoretical findings and to demonstrate the effectiveness of the proposed fractional-order optimization methods.

2. Problem setup and method

We consider the general unconstrained minimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}),$$

where f is a real-valued function.

The standard gradient descent method (GD) commences with an initial starting point $\mathbf{x}^{(0)}$ and updates the kth iterated solution according to the following rule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \eta_k \cdot \nabla f(\mathbf{x}^{(k)}), \qquad k = 0, 1, \ldots$$

where $\eta_k > 0$ is the learning rate (or stepsize) at the kth iteration. The method is theoretically and practically well-understood. It is well-known that GD converges linearly to a stationary point for many convex- and non-convex objective functions as long as the learning rates are appropriately chosen (Bonnans et al., 2006; Nesterov, 2004; Nocedal & Wright, 2006).

2.1. Caputo fractional derivative

Since the fractional derivatives are not defined in a unified manner, there exist multiple definitions. In this paper, we focus on the fractional derivative in the sense of Caputo (1967) whose definition is given below.

Definition 2.1. For $n \in \mathbb{N}$, let $\alpha \in (n-1, n)$. For $c, b \in \mathbb{R}$, let ${}_cI = (c, \infty)$ and $I_b = (-\infty, b)$ be intervals. Let ${}_c\mathcal{D}^\alpha$ and $\mathcal{D}^\alpha_b$ be the sets of functions in $C({}_cI)$ and $C(I_b)$, respectively, such that

$${}_c\mathcal{D}^\alpha = \{f \in C({}_cI) : {}_c^C D_x^\alpha f \text{ exists and is finite in } {}_cI\},$$
$$\mathcal{D}^\alpha_b = \{f \in C(I_b) : {}_x^C D_b^\alpha f \text{ exists and is finite in } I_b\}.$$

Here ${}_c^C D_x^\alpha f$ is the left Caputo fractional derivative of f of order α at x with the lower integral terminal c, and ${}_x^C D_b^\alpha f$ is the right Caputo fractional derivative of f of order α at x with the upper integral terminal b, defined respectively by

$$\text{(Left)} \quad {}_c^C D_x^\alpha f := \frac{1}{\Gamma(n-\alpha)} \int_c^x \frac{f^{(n)}(t)}{(x-t)^{\alpha-n+1}} dt, \qquad \forall x \in {}_cI,$$
$$\text{(Right)} \quad {}_x^C D_b^\alpha f := \frac{(-1)^n}{\Gamma(n-\alpha)} \int_x^b \frac{f^{(n)}(t)}{(t-x)^{\alpha-n+1}} dt, \qquad \forall x \in I_b,$$

where $f^{(n)}$ is the nth derivative of f and Γ is the Gamma function. For $c \in \mathbb{R}$, we define a class $\mathcal{D}^\alpha(c)$ of functions which admit both left and right Caputo fractional derivatives of order α with the integral terminal c:

$$\mathcal{D}^\alpha(c) = {}_c\mathcal{D}^\alpha \cap \mathcal{D}^\alpha_c. \tag{1}$$

Remark 2.2. For notational convenience, ${}_b^C D_x^\alpha f$ is understood as the right Caputo fractional derivative if $x < b$. Some basic calculations of the Caputo fractional derivatives are presented in Appendix A. These calculations will be used for our theoretical results.

Remark 2.3. If $\alpha = n \in \mathbb{N} \cup \{0\}$, the left and right Caputo fractional derivatives are defined by ${}_c^C D_x^n f = f^{(n)}(x)$ and ${}_x^C D_b^n f = (-1)^n f^{(n)}(x)$, respectively. However, since we are interested in optimization, for the sake of notational simplicity, we will use ${}_c^C D_x^1 f = f'(x)$ for any $c \in \mathbb{R}$ with slight abuse of notation.

2.2. Smoothing effect via Caputo fractional derivatives

Although some generalized gradient descent methods that use fractional derivatives have been proposed and studied in several

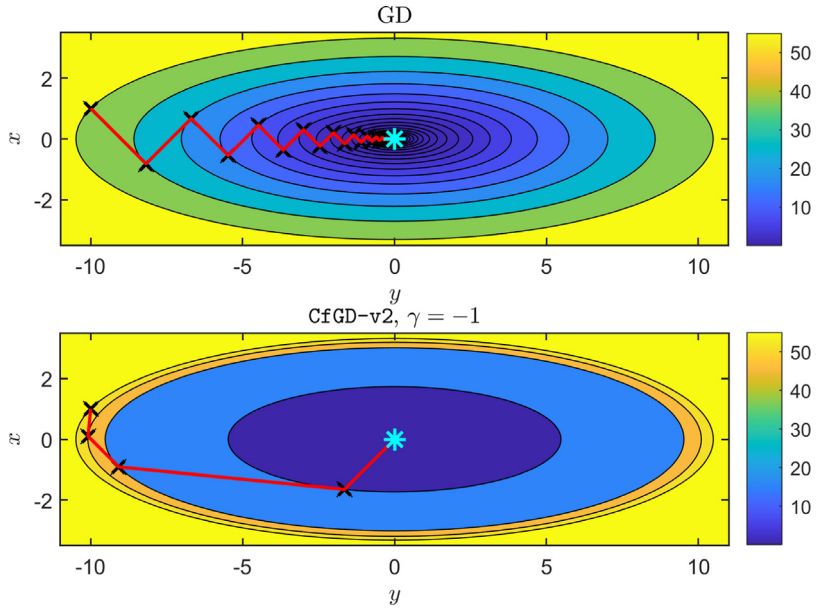

Fig. 1. A contour graph of the objective function $f(x, y) = 5x^2 + 0.5y^2$ (black) along with trajectories of (top) GD and (bottom) `CfGD-v2`. Each trajectory is shown as red solid line with the cross marks ($\times$). Both methods start at $\mathbf{x}^{(0)} = (1, -10)$. The horizontal and vertical axes represent the y- and x- axes, respectively.

works (Wang et al., 2017; Wei et al., 2020), the motivation of using fractional derivatives has been elusive in the context of optimization, except they are natural extensions of the integer order derivative. In this section, we provide a mathematical justification of using the Caputo fractional derivatives for optimization.

Let us first introduce a function class $\mathcal{F}$ that imposes sufficient regularity and decay conditions. These conditions will lead to an infinite series that is related to the Caputo fractional derivatives.

Definition 2.4. Let $\mathcal{F}$ be a class of analytic functions $C^\omega(\mathbb{R})$ satisfying the following property: For $f \in \mathcal{F}$ and for every $c \in \mathbb{R}$, there exist $\rho > 0$ such that for any $x \in B_\rho(c) := (c - \rho, c + \rho)$,

$$\sup_{y\in[c,x]} |f^{(j)}(y)| \le \mathcal{O}\left(\frac{\Gamma(j+1)}{\rho^{j-1}}\right), \qquad j \gg 1. \tag{2}$$

An example function that belongs to $\mathcal{F}$ is a Gaussian function, $f(x) = \exp(-x^2)$. It can be checked that for any interval K,

$$\sup_{y\in K} |f^{(j)}(y)| = \sup_{y\in K} |H_j(y)e^{-y^2}| \le \mathcal{O}\left(2^j \Gamma(\frac{j+1}{2})\right), \qquad j \gg 1,$$

where H_j is the jth-order (physicist's) Hermite polynomial. Thus, a Gaussian function satisfies (2).

In the following theorem, we show that the steepest descent direction of a certain smoothing of the objective function f can be expressed in terms of the Caputo fractional derivatives.

Theorem 2.5. *For $\alpha \in (0, 1)$ and $c \in \mathbb{R}$, suppose $f \in \mathcal{D}^\alpha(c) \cap \mathcal{D}^{1+\alpha}(c) \cap \mathcal{F}$. Let $\rho > 0$ be the radius determined by Definition 2.4. For $\beta \in \mathbb{R}$, we (formally) define a smoothing ${}_cF_{\alpha,\beta}$ of f by*

$${}_cF_{\alpha,\beta}(z) := f(c) + f'(c)(z - c) + \sum_{k=2}^{\infty} C_{k,\alpha,\beta} f^{(k)}(c)(z - c)^k,$$

where $C_{k,\alpha,\beta} = \frac{1}{k}\left(\frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} + \beta\frac{\Gamma(2-\alpha)}{\Gamma(k-\alpha)}\right)$. Then, the infinite series ${}_cF_{\alpha,\beta}(z)$ is convergent for every $z \in B_\rho(c)$. Furthermore, for any $x \in B_\rho(c)\backslash\{c\}$, the steepest descent direction of ${}_cF_{\alpha,\beta}$ at x is

$${}_cF'_{\alpha,\beta}(x) = \frac{{}_c^C D_x^\alpha f}{{}_c^C D_x^\alpha I} + \beta|x - c| \cdot \frac{{}_c^C D_x^{1+\alpha} f}{{}_c^C D_x^\alpha I}, \tag{3}$$

where $I : \mathbb{R} \ni x \mapsto x \in \mathbb{R}$ is the identity map.

Proof. The proof can be found in Appendix B. □

The parameters α and β control the degree of smoothing through the expansion coefficients $C_{k,\alpha,\beta}$. If $\alpha = 1$ and $\beta = 0$, the smoothing ${}_cF_{\alpha,\beta}$ becomes the Taylor series of f at c, hence, the standard steepest descent direction of f is recovered. Note that α and c correspond to fractional order and integral terminal, respectively, in the Caputo fractional derivative (Definition 2.1).

In Fig. 2, we provide an illustration of the smoothing ${}_cF_{\alpha,\beta}$ for a particular objective function $f(z) = (z-6)(z+4)(7z^2+10z+24)$. We also plot the linear approximation $f_{\text{lin}}(z) = f(x)+f'(x)(z-x)$ of f, and the linear approximation $F_{\text{lin}}(z) = {}_cF_{\alpha,\beta}(x) + {}_cF'_{\alpha,\beta}(x)(z-x)$ of ${}_cF_{\alpha,\beta}$ both at $x = 3.5$. Note that the slope of f_{lin} corresponds to the gradient of f while the slope of F_{lin} corresponds to the gradient of the smoothing ${}_cF_{\alpha,\beta}$ of f, which can be expressed by the Caputo fractional derivatives (3). On the left, we set $\alpha = 0.66$, $\beta = -1$, $c = 2.5$, while on the right, we set $\alpha = 0.99$, $\beta = 0$, $c = 2.5$. As expected by Theorem 2.5, the slope of F_{lin} is very close to the slope of f_{lin} when $\alpha \approx 1$ and $\beta = 0$.

2.3. Caputo fractional-based gradient

We first define the Caputo fractional gradient by naturally extending the Caputo fractional derivative with respect to the Cartesian coordinate (Tarasov, 2008). Let $f(\cdot)$ be a real-valued sufficiently smooth function defined on $\mathbb{R}^d$. For $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $j = 1, \ldots, d$, let us define the function $f_{j,\mathbf{x}} : \mathbb{R} \to \mathbb{R}$ by

$$f_{j,\mathbf{x}}(y) = f(\mathbf{x} + (y - x_j)e_j), \tag{4}$$

where e_j denotes the vector in $\mathbb{R}^d$ with a 1 in the jth coordinate and 0's elsewhere. Note that $f_{j,\mathbf{x}}(x_j) = f(\mathbf{x})$. For vectors $\mathbf{c} = (c_1, \ldots, c_d) \in \mathbb{R}^d$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in (0, 1]^d$, following Tarasov (2008), we define the Caputo fractional gradient of f at $\mathbf{x}$ by

$${}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{\boldsymbol{\alpha}} f := \left({}_{c_1}^{C}D_{x_1}^{\alpha_1} f_{1,\mathbf{x}} \quad \cdots \quad {}_{c_d}^{C}D_{x_d}^{\alpha_d} f_{d,\mathbf{x}}\right)^\top \in \mathbb{R}^d. \tag{5}$$

Here $f_{j,\mathbf{x}}$ is assumed to be in $\mathcal{D}^{\alpha_j}(c_j)$ (1) and ${}_{c_j}^{C}D_{x_j}^{1} f_{j,\mathbf{x}}$ is understood as $\partial_{x_j} f(\mathbf{x})$ for all j.

Remark 2.6. The definition (5) can be generalized to use any coordinates by simply replacing the standard basis vectors $\{e_j\}$ to

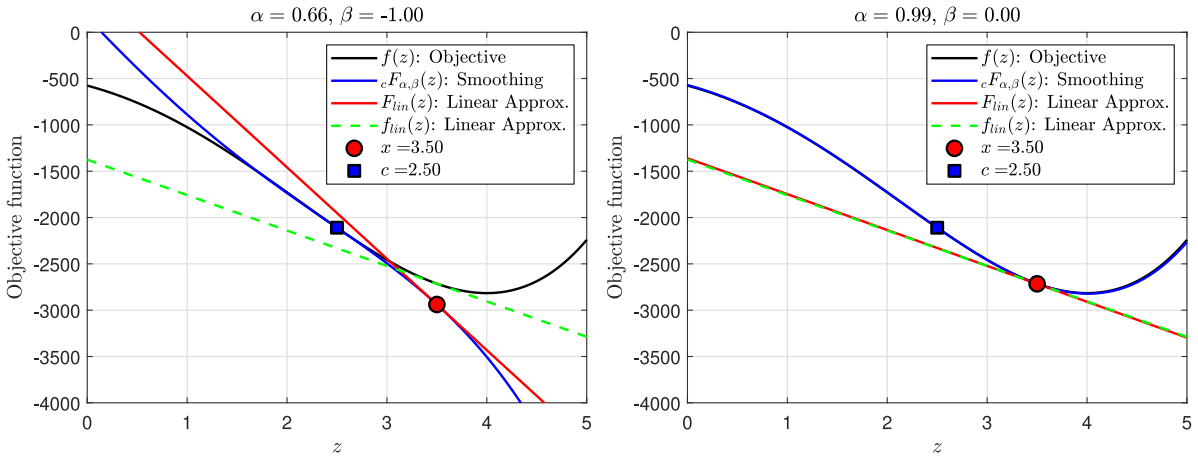

Fig. 2. The graphs of the objective function f (black), its corresponding smoothing ${}_cF_{\alpha,\beta}$ (blue), the linear approximation F_{lin} of ${}_cF_{\alpha,\beta}$ at x (red) and the linear approximation f_{lin} of f at x (green). Here $c = 2.5$ is the integral terminal and $x = 3.5$ is the point where both fractional and integer-order derivatives are evaluated at. (Left) $\alpha = 0.66, \beta = -1$. (Right) $\alpha = 0.99, \beta = 0$.

any orthonormal basis vectors $\{\hat{e}_j\}$. This generalization, however, may require the Hessian of f in the computation of the Caputo fractional derivative of $\hat{f}_{j,\mathbf{x}}(y) := f(\mathbf{x} + (y - x_j)\hat{e}_j)$. See also Theorem 2.7 and (8). The calculation of the Hessian makes the Caputo fractional gradient (5) using a general coordinate computationally demanding in practice. Yet, if the standard basis vectors $\{e_j\}$ are used as in (4), the diagonal components of the Hessian (as can be seen in (8)) are only need, which allows an efficient computation. Therefore, this work only focuses on the version (5) that uses the standard basis vectors.

We are now in a position to define *the Caputo fractional-based gradient*. For a vector $\mathbf{v} = (v_j) \in \mathbb{R}^d$, let either $\text{diag}(\mathbf{v})$ or $\text{diag}(v_j)$ be the diagonal matrix of size $d \times d$ whose (j, j) component is v_j. For smoothing parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$, by combining Theorem 2.5 and (5), we define

$$
{}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x}) := \text{diag}\left(\frac{1}{(1+|\beta_j|)\,{}_{c_j}^{C}D_{x_j}^{\alpha_j} I}\right) \times \left[{}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{\boldsymbol{\alpha}} f + \text{diag}\big(\beta_j|x_j - c_j|\big)\,{}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{1+\boldsymbol{\alpha}} f\right], \tag{6}
$$

which is referred to as the Caputo fractional-based gradient. Here $f_{j,\mathbf{x}}$ is assumed to be in $\mathcal{D}^{\alpha_j}(c_j) \cap \mathcal{D}^{1+\alpha_j}(c_j)$ (1) and the factor $\frac{1}{1+|\beta_j|}$ is additionally introduced as a normalization constant of the two directions.

Note that the Caputo fractional-based gradient ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ of (6) can be viewed as a generalization of the standard gradient as ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x}) = \nabla f(\mathbf{x})$ if $\alpha_j = 1, \beta_j = 0$ for all j.

Theorem 2.7. *Let $\boldsymbol{\alpha} = (\alpha_j) \in (0, 1]^d$, $\boldsymbol{\beta} = (\beta_j)$, $\mathbf{c} = (c_j) \in \mathbb{R}^d$ be given. For any $\mathbf{x} = (x_j) \in \mathbb{R}^d$, let f be a real-valued function defined on $\mathbb{R}^d$ whose associated functions $f_{j,\mathbf{x}}(\cdot)$, $j = 1, \ldots, d$, defined in (4), are in $\mathcal{D}^{\alpha_j}(c_j) \cap \mathcal{D}^{1+\alpha_j}(c_j) \cap \mathcal{F}$, respectively. Then, for $j = 1, \ldots, d$, we have*

$$
\left({}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})\right)_j = C_j \int_{-1}^{1} \left[f'_{j,\mathbf{x}}({}^{+}\Delta_j u + {}^{-}\Delta_j) + \beta_j|x_j - c_j| f''_{j,\mathbf{x}}({}^{+}\Delta_j u + {}^{-}\Delta_j)\right] \times (1-u)^{-\alpha_j} du, \tag{7}
$$

where ${}^{\pm}\Delta_j = (x_j \pm c_j)/2$ and $C_j = \frac{1-\alpha_j}{(1+|\beta_j|)2^{1-\alpha_j}}$.

Proof. The proof can be found in Appendix C. □

Theorem 2.7 shows that the evaluation of ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ requires one to compute the first and the second-order derivatives of $f_{j,\mathbf{x}}(\cdot)$, whose computation generally requires the gradient and the Hessian of the objective function f. However, thanks to the use of the standard basis in (4), it suffices to have the diagonal entries of the Hessian as shown below:

$$
\begin{aligned}
f'_{j,\mathbf{x}}(y) &= \langle e_j, \nabla f(\mathbf{x} + (y - x_j)e_j)\rangle \\
&= \frac{\partial f}{\partial x_j}(x_1, \ldots, x_{j-1}, y, x_{j+1}, \ldots, x_d), \\
f''_{j,\mathbf{x}}(y) &= \langle e_j, \nabla^2 f(\mathbf{x} + (y - x_j)e_j)e_j\rangle \\
&= \frac{\partial^2 f}{\partial x_j^2}(x_1, \ldots, x_{j-1}, y, x_{j+1}, \ldots, x_d),
\end{aligned} \tag{8}
$$

where $\nabla^2 f$ is the Hessian of f.

In general, computing fractional gradients is more expensive than computing integer-order gradients. From the observation that (7) of Theorem 2.7 involves integrals that can be accurately approximated by the Gauss–Jacobi quadrature, we propose an efficient way of evaluating ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$. Let $\{(u_{j,l}, w_{j,l})\}_{l=1}^{s}$ be the Gauss–Jacobi quadrature rule of s points, corresponding to the order α_j. Then, by replacing the integrals in (7) with the corresponding quadrature rules, ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ can be approximated by ${}_c\mathbf{Q}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ whose jth component is defined by

$$
\left({}_c\mathbf{Q}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})\right)_j = C_j \sum_{l=1}^{s} w_{j,l} \times \left[f'_{j,\mathbf{x}}({}^{+}\Delta_j u_{j,l} + {}^{-}\Delta_j) + \beta_j|x_j - c_j| f''_{j,\mathbf{x}}({}^{+}\Delta_j u_{j,l} + {}^{-}\Delta_j)\right]. \tag{9}
$$

This allows the computation of fractional gradients to be embarrassingly parallel, and hence, the Caputo fractional-based gradient of f at $\mathbf{x}$ can be efficiently evaluated.

We briefly describe the complexity of ${}_c\mathbf{Q}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$. Suppose s quadrature points are used and the evaluations of $f'_{j,\mathbf{x}}$ and $f''_{j,\mathbf{x}}$ take $\mathcal{O}(K)$ FLOPS each for all j. Then it can be checked that computing ${}_c\mathbf{Q}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ requires $\mathcal{O}(sKd)$ FLOPS. We note that the vanilla gradient takes $\mathcal{O}(Kd)$ FLOPS. Thus, the computational complexity of ${}_c\mathbf{Q}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ grows only linearly in both the dimension d and the number of quadrature points s.

Example. In machine learning, the objective function is often defined through a set of training data and a parametric

model such as neural networks. Suppose we have m training data $\{(z_i, y_i)\}_{i=1}^m$, where $z_i, y_i \in \mathbb{R}$, and use a two-layer neural network $N(z; \mathbf{x}) = \sum_{j=1}^n a_{3,j}\phi(a_{1,j}z + a_{2,j})$ to fit these data. Here ϕ is a nonlinear activation function (e.g., tanh, sigmoid, Gaussian), and $\mathbf{x} = \{a_{1,j}, a_{2,j}, a_{3,j}\}_{j=1}^n \in \mathbb{R}^{3n}$ is the set of network parameters. The goal is to find the optimal network parameter $\mathbf{x}^*$ that minimizes the following objective (loss) function:

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^m (N(z_i; \mathbf{x}) - y_i)^2 .$$

We discuss the computational complexity of the evaluation of ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ (9). It suffices to calculate the complexities of $f'_{j,\mathbf{x}}$ and $f''_{j,\mathbf{x}}$, which turns out to be at most $K = \mathcal{O}(mn)$ FLOPS. The detailed complexity calculation is reported in Appendix H. Hence, if s quadrature points are used for the evaluation of ${}_c\mathbf{D}_\beta^\alpha f(\mathbf{x})$ (7), the computational cost for the evaluation of ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ (9) is $\mathcal{O}(smn^2)$ FLOPS.

Remark 2.8. Another approach of evaluating fractional derivatives is to utilize a modern machine learning technique. In Lu, Jin, Pang, Zhang, and Karniadakis (2021), the authors demonstrated that neural networks can learn linear and nonlinear operators, and specifically they have two examples of learning the Caputo fractional derivative and a fractional Laplacian. The resulting neural network is called a deep operator network (DeepONet). We can pre-train a DeepONet for the purpose of evaluating Caputo fractional gradients and utilize it in calculating the Caputo fractional-based gradient (7). This deep learning approach will significantly lessen the computational cost of computing fractional gradients. We will pursue this direction in future study.

2.4. Caputo fractional order optimization methods

Numerous first-order optimization methods have been proposed, analyzed and used routinely in many research areas. A general form of them may be written as follows: Starting at an initial point $\mathbf{x}^{(0)}$, the $(k+1)$th iterated solution is defined according to the rule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \eta_k \cdot \Phi(\boldsymbol{\mu}^{(k)}), \qquad k = 0, 1, \ldots$$

where $\eta_k > 0$ is the learning rate, $\Phi(\cdot)$ is a method-dependent function, and $\boldsymbol{\mu}^{(k)}$ may represent all the first-order information, i.e., $\boldsymbol{\mu}^{(k)} = (\nabla f(\mathbf{x}^{(0)}), \ldots, \nabla f(\mathbf{x}^{(k)}))$. For example, if $\Phi(\boldsymbol{\mu}^{(k)}) = -\nabla f(\mathbf{x}^{(k)})$, the gradient descent method is recovered.

By replacing the integer-order gradient to the Caputo fractional-based gradient, a natural class of Caputo fractional-order optimization algorithms is obtained. An analogous general form is given by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \eta_k \cdot \Phi(\boldsymbol{\mu}_C^{(k)}) \qquad k = 0, 1, \ldots$$

where $\boldsymbol{\mu}_C^{(k)}$ represents all the Caputo fractional-based gradient information, i.e.,

$$\boldsymbol{\mu}_C^{(k)} = \left({}_{c_0}\mathbf{D}_{\beta_0}^{\alpha_0} f(\mathbf{x}^{(0)}), \ldots, {}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)})\right),$$

with $\{\boldsymbol{\alpha}_i\} \subset (0, 1]^d$, $\{\boldsymbol{\beta}_i\}, \{\mathbf{c}_i\} \subset \mathbb{R}^d$.

For the rest of the paper, we focus on Φ that corresponds to gradient descent (GD), $\Phi_{\mathtt{GD}}$, and the Adam optimization method (Kingma & Ba, 2014), $\Phi_{\mathtt{Adam}}$.

Caputo Fractional Gradient Descent. Since $\Phi_{\mathtt{GD}}(\boldsymbol{\mu}_C^{(k)}) = -{}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)})$, the Caputo fractional gradient descent (`CfGD`) updates the kth iterated solution by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \eta_k \cdot \Phi_{\mathtt{GD}}(\boldsymbol{\mu}_C^{(k)}), \qquad k = 0, 1, \ldots \tag{10}$$

Caputo Fractional Adam. The Adam optimization algorithm (Kingma & Ba, 2014) is one of the most popular gradient-based optimization methods in machine learning. There are three hyperparameters, β_1, β_2 and ϵ whose default values are 0.9, 0.999 and 10^{-8}, respectively. By replacing the gradient to the Caputo fractional-based gradient, the Caputo fractional Adam (`CfAdam`) is obtained. `CfAdam` commences with an initial point $\mathbf{x}^{(0)}$ with $\boldsymbol{m}^{(0)} = 0$ and $\boldsymbol{s}^{(0)} = 0$, and updates $\mathbf{x}^{(k)}$ according to

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \eta_k \cdot \Phi_{\mathtt{Adam}}(\boldsymbol{\mu}_C^{(k)}), \qquad k = 0, 1, \ldots \tag{11}$$

where $\Phi_{\mathtt{Adam}}(\boldsymbol{\mu}_C^{(k)}) = -\frac{1}{\sqrt{\boldsymbol{s}^{(k)}/(1-\beta_2^k)}+\epsilon} \odot \frac{\boldsymbol{m}^{(k)}}{1-\beta_1^k}$,

$$\boldsymbol{m}^{(k)} = \beta_1 \boldsymbol{m}^{(k-1)} + (1-\beta_1)\left[{}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)})\right],$$

$$\boldsymbol{s}^{(k)} = \beta_2 \boldsymbol{s}^{(k-1)} + (1-\beta_2)\left[{}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)}) \odot {}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)})\right].$$

Here the operator $\odot$ is the elementwise multiplication. The default value of the learning rate is $\eta_k = 10^{-3}$.

The use of the Caputo fractional-based gradient requires one to choose $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k$ appropriately at the kth iteration. Consequently, the performance of the Caputo fractional-order optimization methods highly depends on them. For example, if $\boldsymbol{\alpha}_k = (1, \ldots, 1)$, $\boldsymbol{\beta}_k = (0, \ldots, 0)$ for all k, the Caputo fractional method is identical to the first-order method as ${}_{c_k}\mathbf{D}_{\beta_k}^{\alpha_k} f(\mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k)})$. In general, it is a challenging task to find an optimal configuration and that is not the focus of the present work. Yet, numerical examples show that certain configurations lead to superior performance over the corresponding first-order methods.

As discussed in Section 2.3, the Caputo fractional-based gradient ${}_c\mathbf{D}_\beta^\alpha f(\mathbf{x})$ can be approximated by ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ of (9) using the Gauss–Jacobi quadrature rules. However, the computational cost of ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ is roughly s-times more expensive than those of $\nabla f(\mathbf{x})$, where s is the number of quadrature points. How large the number s should be is an important question to be addressed. In Section 4.3, we investigate the sensitivity of `CfGD` with respect to s. For small s values (including $s = 1$), while ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ may no longer be an accurate approximation to ${}_c\mathbf{D}_\beta^\alpha f(\mathbf{x})$, we empirically found that `CfGD` still results in acceleration over GD. See Section 4.3 for more details.

3. Convergence analysis: Quadratic functions

We present a convergence analysis of `CfGD` (10) for the minimization of quadratic objective functions:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x}, \tag{12}$$

where $\mathbf{x}, b \in \mathbb{R}^d$, and $A = (a_{ij}) \in \mathbb{R}^{d\times d}$. Here, A is assumed to be symmetric positive definite. It can be checked that the unique minimizer is $\mathbf{x}^* = -A^{-1}b$.

As a special case, we consider the following regression problem. Let

$$W = \left[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\right] \in \mathbb{R}^{d\times m}, \qquad \mathbf{w}_k = \begin{pmatrix} w_{k1} \\ \vdots \\ w_{kd} \end{pmatrix} \in \mathbb{R}^d,$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m.$$

The least squares problem is formulated as follows

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\|W^\top \mathbf{x} - y\|^2, \tag{13}$$

where $\|\cdot\|$ is the Euclidean norm. Assuming W is of full rank with $m \geq d$, the least square solution is explicitly written as $\mathbf{x}^* = (WW^\top)^{-1}Wy$. This is a special case of (12) with $A = WW^\top$ and $b = -Wy$.

For quadratic objective functions, an explicit formula of ${}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x})$ can be obtained.

Corollary 3.1. *Let f be a quadratic function of the form (12). Then,*

$$ {}_c\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x}) = diag\left(\frac{1}{1+|\beta_j|}\right)\Big(A\mathbf{x} + b + diag(\gamma_{\boldsymbol{\alpha},\boldsymbol{\beta}})diag(A)(\mathbf{x}-\mathbf{c})\Big), \tag{14} $$

where $\gamma_{\boldsymbol{\alpha},\boldsymbol{\beta}} = (\beta_j - \frac{1-\alpha_j}{2-\alpha_j}) \in \mathbb{R}^d$, and $diag(A)$ is the diagonal matrix from A.

Proof. The proof is readily followed from Theorem 2.7. See also Appendix D. □

Note that the integer-order gradient of the quadratic objective function at $\mathbf{x}$ is $A\mathbf{x} + b$. Hence, the computational cost of CfGD is roughly the same as the one of GD in this case. Also, the fractional order $\boldsymbol{\alpha}$ and the smoothing parameter $\boldsymbol{\beta}$ depend only through $\gamma_{\boldsymbol{\alpha},\boldsymbol{\beta}}$. We often write $\gamma_{\alpha_j,\beta_j}$ as γ_j if the context is clear.

Both GD and CfGD require one to determine appropriate learning rates (stepsizes). For quadratic functions, an optimal learning rate could be obtained by the line search:

$$ \min_{\eta} \frac{1}{2}(\mathbf{x}^{(k+1)})^\top A\mathbf{x}^{(k+1)} + b^\top \mathbf{x}^{(k+1)}, $$

where $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \eta_k \cdot \vec{\mathbf{d}}_k$ for some direction vector $\vec{\mathbf{d}}_k$ (either the integer-order gradient or the Caputo fractional-based gradient (14) at $\mathbf{x}^{(k)}$). It then can be checked that the first-order optimality condition yields the optimal stepsize

$$ \eta_k^* = \frac{\langle A\mathbf{x}^{(k)} + b, \vec{\mathbf{d}}_k\rangle}{\vec{\mathbf{d}}_k^\top A \vec{\mathbf{d}}_k}. \tag{15} $$

CfGD requires one to choose $\{\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k\}$ to give a specific Caputo fractional-order optimization algorithm. For the purpose of convergence analysis, we consider the following three configurations:

- CfGD-v1 is a version of CfGD that uses $\boldsymbol{\alpha}_k = (\alpha, \ldots, \alpha)$, $\boldsymbol{\beta}_k = (\beta, \ldots, \beta)$ and $\mathbf{c}_k = \mathbf{c}$ for all k for some $\alpha \in (0,1)$, $\beta \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}^d$. See Section 3.1.
- CfGD-v2 is a version of CfGD that uses $\boldsymbol{\alpha}_k = (\alpha, \ldots, \alpha)$, $\boldsymbol{\beta}_k = (\beta, \ldots, \beta)$ and $\mathbf{c}_k = \mathbf{x}^{(k-L)}$ for all k for some $\alpha \in (0,1)$, $\beta \in \mathbb{R}$ and $L \in \mathbb{N}$. See Section 3.2.
- CfGD-v3 is a version of CfGD that uses $\boldsymbol{\alpha}_k = (\alpha_k, \ldots, \alpha_k)$, $\boldsymbol{\beta}_k = (\beta_k, \ldots, \beta_k)$ and $\mathbf{c}_k$ for all k where $\alpha_k \in (0,1]$ and $\beta_k \in \mathbb{R}$ satisfy $\lim_{k\to\infty}(\beta_k - \frac{1-\alpha_k}{2-\alpha_k}) = 0$. See Section 3.3.

In what follows, we analyze the three versions of CfGD. We note that the purpose of considering the three versions of CfGD is mainly to theoretically understand the effects of $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k$ through the lens of the quadratic objective functions. As we shall show in Section 3.1, CfGD-v1 does converge and the rate of convergence can be significantly improved with appropriate choices of β. However, in general, the limit does not coincide with the integer-order stationary point, which is often sought in practice. This motivates us to further consider and investigate CfGD-v2 and CfGD-v3.

3.1. Caputo fractional gradient descent: Version 1

Let us consider CfGD of (10) with $\boldsymbol{\alpha}_k = (\alpha, \ldots, \alpha)$, $\boldsymbol{\beta}_k = (\beta, \ldots, \beta)$ and $\mathbf{c}_k = \mathbf{c}$ for all k for some $\alpha \in (0,1)$, $\beta \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}^d$, which is referred to as CfGD-v1 throughout.

In Theorem 3.2, we show that the stationary point of CfGD-v1 is the solution to a Tikhonov regularization. For reader's convenience and completeness, we recall Tikhonov regularization (Golub, Hansen, & O'Leary, 1999):

$$ \min_{\mathbf{x}} \|W^\top \mathbf{x} - y\|^2 + \gamma \|R^\top(\mathbf{x} - \bar{\mathbf{x}})\|^2, $$

where R is some suitably chosen Tikhonov matrix. It can be checked that the solution is explicitly written as

$$ \mathbf{x}^*_{\mathrm{Tik}}(\gamma) = \bar{\mathbf{x}} + \left(WW^\top + \gamma RR^\top\right)^{-1} W(y - W^\top \bar{\mathbf{x}}), \tag{16} $$

assuming $WW^\top + \gamma RR^\top$ is invertible.

We are now in a position to present our convergence analysis of CfGD-v1.

Theorem 3.2. *Let f be the quadratic function of the form (13). For $\alpha \in (0,1)$ and $\beta \in \mathbb{R}$, let $\tilde{A}_{\alpha,\beta}$ be the matrix whose (i,j)-component is $(\beta + \frac{1}{2-\alpha})\sum_{k=1}^m w_{ki}^2$ if $i = j$ and $\sum_{k=1}^m w_{ki}w_{kj}$ if $i \neq j$. Suppose $\tilde{A}_{\alpha,\beta}$ is a positive definite matrix and $\sigma_{\max}$ is its largest singular value. For some $0 < \eta < 2$, let $\eta_k = \frac{1+|\beta|}{\sigma_{\max}}\eta$. For $\mathbf{c} \in \mathbb{R}^d$, the kth iterated solution of CfGD-v1 satisfies*

$$ \|\mathbf{x}^{(k)} - \mathbf{x}^*_{Tik}(\gamma)\| \leq \|\mathbf{x}^{(0)} - \mathbf{x}^*_{Tik}(\gamma)\| \cdot |1 - \eta/\kappa_{\alpha,\beta}|^{k/2}, $$

where $\kappa_{\alpha,\beta}$ is the condition number of $\tilde{A}_{\alpha,\beta}$ and $\mathbf{x}^*_{Tik}$ is the solution to the Tikhonov regularization (16) with $\bar{\mathbf{x}} = \mathbf{c}$, $\gamma = \beta - \frac{1-\alpha}{2-\alpha}$ and $R = diag(\sqrt{\sum_{k=1}^m w_{kj}^2})$.

Proof. The proof can be found in Appendix D. □

Theorem 3.2 shows that CfGD-v1 converges linearly to the stationary point, which is the solution to a certain Tikhonov regularization. The convergence rate depends only on A and $\beta + \frac{1}{2-\alpha}$. We note that when $\alpha = 1$ and $\beta = 0$, it can be checked from Corollary 3.1 that we recover the standard convergence analysis of GD. If $\beta + \frac{1}{2-\alpha} > 1$, CfGD-v1 expects to converge faster than GD as the condition number of $\kappa_{\alpha,\beta}$ is smaller than $\kappa_{1,0}$. However, the fractional stationary point is not the same as the integer-order stationary point, which is often sought in practice. With the goal of finding the integer-order stationary point, in the following two subsections, we consider two adaptive versions of CfGD.

3.2. Caputo fractional gradient descent: Version 2

Let us consider CfGD of (10) with $\boldsymbol{\alpha}_k = (\alpha, \ldots, \alpha)$, $\boldsymbol{\beta}_k = (\beta, \ldots, \beta)$ and $\mathbf{c}_k = \mathbf{x}^{(k-L)}$ for all k for some $\alpha \in (0,1)$, $\beta \in \mathbb{R}$ and $L \in \mathbb{N}$, which is referred to as CfGD-v2 throughout. The main difference between CfGD-v1 and CfGD-v2 is that CfGD-v2 uses adaptive integral terminal $\mathbf{c}_k = \mathbf{x}^{(k-L)}$ instead of a given fixed integral terminal $\mathbf{c}$. Thus, it commences with $(L+1)$ initial points $\{\mathbf{x}^{(-j)}\}_{j=0}^L \subset \mathbb{R}^d$.

We first present an error bound of CfGD-v2 which also shows that the convergence to the integer-order stationary point $\mathbf{x}^*$ of (12) under some conditions.

Theorem 3.3. *Let f be the objective function of the form (12) and $\mathbf{x}^*$ be the corresponding optimal solution. For $\alpha \in (0,1)$, $\beta \in \mathbb{R}$ and $L \in \mathbb{N}$, let $\gamma_{\alpha,\beta} = \beta - \frac{1-\alpha}{2-\alpha}$ and $\{\mathbf{x}^{(-j)}\}_{j=0}^L \subset \mathbb{R}^d$ be initial points. Then, the kth iterated solution of CfGD-v2 satisfies*

$$ \|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \sum_{j=0}^{L} \|\mathcal{A}_{k,j}(\mathbf{x}^{(-j)} - \mathbf{x}^*)\|, $$

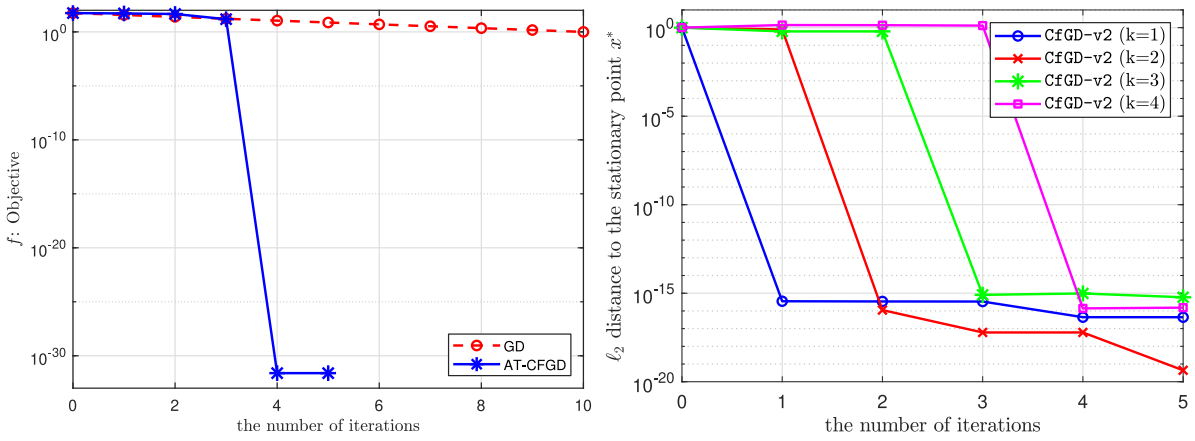

Fig. 3. (Left) The same example of Fig. 1 is considered. The objective function is $f(x, y) = 5x^2 + 0.5y^2$. The objective values versus the number of iterations. CfGD-v2 finds the global minimum within merely 4 iterations, while GD requires a large number of iterations to reach the same level. (Right) An illustration of Theorem 3.4. The convergence to the optimal point is observed within at most four iterations.

where $\mathcal{A}_{k,j} \in \mathbb{R}^{d\times d}$ are matrices defined recursively by

$$\mathcal{A}_{k,0} = \mathcal{A}_{k-1,0}\mathcal{A}_{1,0} + \mathcal{A}_{k-1,1}, \quad \mathcal{A}_{k,L} = \mathcal{A}_{k-1,0}\mathcal{A}_{1,L},$$
$$\mathcal{A}_{k,j} = \mathcal{A}_{k-1,j+1},$$

for $1 \le j < L$ and $k = 2, \dots$ starting with $\mathcal{A}_{1,0} = I - \frac{\eta}{1+|\beta|}(A + \gamma_{\alpha,\beta}\mathrm{diag}(A))$, $\mathcal{A}_{1,L} = \frac{\eta}{1+|\beta|}\gamma_{\alpha,\beta}\mathrm{diag}(A)$ and $\mathcal{A}_{1,j} = 0$ for $1 \le j < L$. Furthermore, if $\lim_{k\to\infty} \|\mathcal{A}_{k,0}(\mathbf{x}^{(0)} - \mathbf{x}^*)\| = 0$, we have $\lim_{k\to\infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0$.

Proof. The proof can be found in Appendix E. □

Although Theorem 3.3 provides an error bound with respect to the integer-order stationary point, further investigation is needed in understanding the dynamics of $\mathcal{A}_{k,0}$ for convergence. Yet, we empirically found that CfGD-v2 not only converges but also converges significantly faster than GD. We remark that the convergence *rate* does not explain such a significant acceleration. As shown in Fig. 1, the first two iterations of CfGD-v2 do not show a significant improvement (actually no better than GD). This indicates that linear convergence (uniform rate) does not explain the accelerated convergence of CfGD-v2. More numerical tests are reported in Section 4, while we provide details of Fig. 1 below.

In Fig. 1, we employ CfGD-v2 with $L = 1$, $\gamma = -1$, $\mathbf{x}^{(-1)} = (-1, -1)$, and the optimal stepsize of (15). Note that α and β depend only through $\gamma_{\alpha,\beta}$ (Corollary 3.1). We further report the objective values versus the number of iterations on the left of Fig. 3. We see that (as expected) GD converges linearly to the optimum whose rate depends on the condition number of the model matrix. While CfGD-v2 does not exhibit a linear convergence, it finds the optimal solution within the error of machine accuracy in merely four iterations.

Theorem 3.4. *Let f be the objective function of the form given in (12) where A is a diagonal matrix of size 2×2. Let $\mathbf{x}^{(k)}$ be the kth iterated solution of CfGD-v2 with $L = 1$, $\gamma = -1$ and η_k being the optimal learning rate (15). Let $\mathbf{x}^*$ be the optimal solution and let $\mathcal{E}_k = \mathbf{x}^{(k)} - \mathbf{x}^*$. Suppose the initial two starting points $\mathbf{x}_{-1}, \mathbf{x}_0$ satisfy one of the four following conditions:*

$$\langle \mathcal{E}_{-1}, A^3\mathcal{E}_0\rangle = 0 \quad or \quad \langle \mathcal{E}_{-1}, PA\mathcal{E}_0\rangle = 0 \quad or \quad \langle \mathcal{E}_{-1}, A\mathcal{E}_0\rangle = 0 \quad or$$
$$\langle \mathcal{E}_{-1}, AP\mathcal{E}_0\rangle = 0,$$

where $P = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$. Then, there exists $k \in \{1, 2, 3, 4\}$ such that $\mathbf{x}^{(k)} = \mathbf{x}^*$.

Proof. The proof can be found in Appendix F. □

Theorem 3.4 exactly predicts the numerical experiment shown in Figs. 1 and 3. As a matter of fact, since $\mathbf{x}^{(-1)} = (-1, -1)$, $\mathbf{x}^{(0)} = (1, -10)$, and $\mathbf{x}^* = (0, 0)$, it can be checked that $\langle \mathcal{E}_{-1}, A\mathcal{E}_0\rangle = 0$ holds, which guarantees $\mathbf{x}^{(4)} = \mathbf{x}^*$ by Theorem 3.4. The right of Fig. 3 further illustrates Theorem 3.4, which shows the ℓ_2 distance to the optimum, i.e., $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|$ with respect to the number of iterations at varying initial points satisfying the given conditions in Theorem 3.4. As expected, the optimal point is reached (up to machine accuracy) within four iterations.

While Theorem 3.4 provides some insights on when and why CfGD-v2 could lead to a significantly fast convergence, a further investigation is required for general cases.

3.3. Caputo fractional gradient descent: Version 3

Lastly, let us consider CfGD of (10) with $\boldsymbol{\alpha}_k = (\alpha_k, \dots, \alpha_k)$, $\boldsymbol{\beta}_k = (\beta_k, \dots, \beta_k)$ and $\mathbf{c}_k \in \mathbb{R}^d$ for all k, where $\alpha_k \in (0, 1]$, $\beta_k \in \mathbb{R}$ satisfy $\beta_k - \frac{1-\alpha_k}{2-\alpha_k} \to 0$ as $k \to \infty$. We refer to it as CfGD-v3.

The main feature of CfGD-v3 is the use of *adaptive* parameters in a scheduled manner. Let $\{k_t\}_{t\ge1}$ be a sequence of positive integers that serves as the update frequency for $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k$, and let $\{\tilde{\alpha}_t, \tilde{\beta}_t, \tilde{\mathbf{c}}_t\}_{t\ge1}$ be a sequence that serves as the update rules satisfying $\lim_{t\to\infty} \gamma_t := \tilde{\beta}_t - \frac{1-\tilde{\alpha}_t}{2-\tilde{\alpha}_t} = 0$ with $\tilde{\alpha}_t \in (0, 1]$ and $\tilde{\beta}_t \in [B_\ell, B_u]$ for some $B_\ell \le B_u$ such that $0 \in (B_\ell - 0.5, B_u]$.

For a given positive integer T, CfGD-v3 consists of T stages. The first stage starts with an initial point $\mathbf{x}_1^{(0)} = \mathbf{x}^{(0)}$ and apply CfGD of (10) with $\boldsymbol{\alpha}_k = (\tilde{\alpha}_1, \dots, \tilde{\alpha}_1)$, $\boldsymbol{\beta}_k = (\tilde{\beta}_1, \dots, \tilde{\beta}_1)$, $\mathbf{c}_k = \tilde{\mathbf{c}}_1$ for the first k_1 iterations. Let us denote the k_1th iterated solution by $\mathbf{x}_1^{(k_1)}$. For $2 \le t \le T$, the tth stage starts with $\mathbf{x}_t^{(0)} := \mathbf{x}_{t-1}^{(k_{t-1})}$ and apply CfGD with $\boldsymbol{\alpha}_k = (\tilde{\alpha}_t, \dots, \tilde{\alpha}_t)$, $\boldsymbol{\beta}_k = (\tilde{\beta}_t, \dots, \tilde{\beta}_t)$, $\mathbf{c}_k = \tilde{\mathbf{c}}_t$ for the next k_t iterations. The solution of CfGD-v3 after the t stages is denoted by $\mathbf{x}_t^{(k_t)}$.

In Theorem 3.5, we present an error bound of CfGD-v3 with respect to the optimal solution $\mathbf{x}^*$.

Theorem 3.5. *Let f be the objective function of the form (13). At the tth stage, suppose $\eta_t \in (0, 2)$ is chosen as in Theorem 3.2. Suppose*

$$M_{\max} = \sup_{\gamma\in(B_\ell-0.5,B_u]} \|\left(WW^\top + \gamma RR^\top\right)^{-1}\| < \infty,$$

where R is defined in Theorem 3.2. Then, the following error bound of `CfGD-v3` *holds:*

$$\|\mathbf{x}_T^{(k_T)} - \mathbf{x}^*\| \le \left(\prod_{j=0}^{T-1} r_{T-j}\right) \|\mathbf{x}^{(0)} - \mathbf{x}^*_{Tik}(\gamma_1)\| + C|\gamma_T|\|\tilde{\mathbf{c}}_T - \mathbf{x}^*\| + C\sum_{i=1}^{T-1}\left(\prod_{j=0}^{i-1} r_{T-j}\right) \times \left[|\gamma_{T-i} - \gamma_{T-i+1}|\|\tilde{\mathbf{c}}_{T-i+1} - \mathbf{x}^*\| + |\gamma_{T-i}|\|\tilde{\mathbf{c}}_{T-i+1} - \tilde{\mathbf{c}}_{T-i}\|\right],$$

where $\mathbf{x}^*_{Tik}(\gamma_1)$ is the solution to the Tikhonov regularization (16) with $\bar{\mathbf{x}} = \tilde{\mathbf{c}}_1$, $C = M_{\max}^2\|W\|^2\|R\|^2$, $\gamma_t = \tilde{\beta}_t - \frac{1-\tilde{\alpha}_t}{2-\tilde{\alpha}_t}$, $r_t = |1 - \eta_t/\kappa_t|^{k_t/2}$, and κ_t is the condition number of $\tilde{A}_{\tilde{\alpha}_t,\tilde{\beta}_t}$ defined in Theorem 3.2.

Proof. The proof can be found in Appendix G. □

The error bound of Theorem 3.5 reveals the trade-off between the accelerated rate of convergence and the convergence toward the integer-order stationary point $\mathbf{x}^*$. In an extreme case of $\tilde{\mathbf{c}}_t = \mathbf{x}^*$ for all t, the bound becomes

$$\|\mathbf{x}_t^{(k_t)} - \mathbf{x}^*\| \le \left(\prod_{j=0}^{t-1} r_{t-j}\right) \|\mathbf{x}^{(0)} - \mathbf{x}^*_{\text{Tik}}(\gamma_1)\|.$$

This implies that not only `CfGD-v3` converges to $\mathbf{x}^*$ but also its rate of convergence can be significantly improved if κ_t is much smaller than the condition number of $A = WW^\top$ for all t, which, as a matter of fact, can be achieved by setting $\gamma_t \gg 1$. However, assuming $\tilde{\mathbf{c}}_t \neq \mathbf{x}^*$, one may not set $\gamma_t \gg 1$ due to the term $|\gamma_t|\|\tilde{\mathbf{c}}_t - \mathbf{x}^*\|$. This term remains unchanged even when the number k_t of iterations is sufficiently large enough so that $r_t \approx 0$.

Remark 3.6. If $T = 1$ and $k_1 = k$, *`CfGD-v3`* becomes *`CfGD-v1`*. Furthermore, if $\tilde{\mathbf{c}}_1 = \mathbf{x}^*$, the error bound is identical to Theorem 3.2 as $\mathbf{x}^*_{\text{Tik}} = \mathbf{x}^*$.

The idea of using adaptive fractional order appeared in Wei et al. (2020), where multiple heuristic adaptation strategies were also presented. While some promising empirical results were reported in Wei et al. (2020), finding an optimal strategy requires more investigation and remains a challenging problem. Due to these reasons, we do not consider `CfGD-v3` in numerical tests. Yet, we employ a specific *random* configuration of $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k$ and use them with `CfAdam` in Section 4.4.

4. Numerical examples

We present numerical examples to verify our theoretical findings and demonstrate the performance of `CfGD-v1`, `CfGD-v2`, and `CfAdam`.

Note that there is no rule of thumb for the choices of the parameters $\{\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \mathbf{c}_k\}$ for the Caputo fractional-based gradient (6). Hence, in all the numerical examples, we have attempted various choices and reported results that show reasonably good performance. For quadratic objective functions, unless otherwise stated, the optimal learning rate (stepsize) of (15) is used.

4.1. Quadratic objective function: Random data

We consider quadratic objective functions of the form of (13). We generate a matrix W of size $d \times m$ by sampling each entry independently from a normal distribution $\mathcal{N}(0, 1/m)$. Similarly, we randomly generate the vector y of size m from $\mathcal{N}(0, I_m)$, where I_m is the identity matrix of size m. In terms of (12), the model matrix A is $WW^\top$ and b is $-Wy$.

We first verify the convergence of `CfGD-v1` to the solution to Tikhonov regularization. We set the integral terminal vector $\mathbf{c}$ to the vector whose elements are all ones and the initial starting point $\mathbf{x}^{(0)}$ to a random vector from the standard normal distribution. In Fig. 4, we show the results for $d = m = 100$. On the left, the ℓ_2 distance to the solution $\mathbf{x}^*_{\text{Tik}}(\gamma)$ (16) is reported with respect to the number of iterations at varying $\gamma \in \{0.15, 0.25, 0.5, 0.75, 1, 10\}$. Note that gradient descent corresponds to the case of $\gamma = 0$. As expected from Theorem 3.2, we clearly see that `CfGD-v1` converges to $\mathbf{x}^*_{\text{Tik}}$ at a significantly faster rate compared to those of GD. This is not a surprise as Tikhonov regularization induces a smaller condition number of the model matrix, that results in a fast convergence. On the right, the objective values are reported. Since the Tikhonov solution does not necessarily minimize f, we see that `CfGD-v1` does not effectively decrease the objective function, while it already reaches to its own fractional stationary point. This suggests that using constant parameters in `CfGD` may not be an effective way of minimizing f as the integer-order stationary point cannot reached unless $\gamma = 0$. We note that in this case, the model matrix $A = WW^\top$ is ill-conditioned, whose condition number is greater than 10^5. This explains the extremely slow convergence by GD.

Next, we demonstrate the performance of `CfGD-v2`, which enables to reach the integer-order stationary point $\mathbf{x}^*$. We consider an ill-conditioned model matrix generated by the following procedure. First, we generate a random matrix W_0 according to the aforementioned manner. After computing a singular value decomposition $W_0 = US_0V^\top$, we consider the matrix S obtained from S_0 by replacing the largest singular value to 10 and the smallest singular value to 0.1. We then obtain a matrix defined by $W = USV^\top$ whose condition number is at least 10^2. Therefore, the condition number of the model matrix $A = WW^\top$ is at least 10^4. In what follows, we set $d = m = 20$. For both GD and `CfGD-v2`, $\mathbf{x}^{(0)}$ is randomly chosen from the uniform distribution of $[-1, 1]^d$. The initial points $\{\mathbf{x}^{(-j)}\}_{j=1}^L$ for `CfGD-v2` are randomly chosen from the standard normal distribution.

In Fig. 5, we show the ℓ_2 distance to the integer-order stationary point $\mathbf{x}^*$ with respect to the number of iterations at varying parameters (L, γ) of `CfGD-v2`. Specifically, we report the results of $\gamma \in \{\pm 1, \pm 0.75, \pm 0.5, \pm 0.25, 0\}$. Again, GD corresponds to the special case of $\gamma = 0$. On the top, the results for $L = 1, 2$ are shown, and on the bottom, the results for $L = 3, 4$ are shown. When $L = 1$, we found that `CfGD-v2` with negative γ outperforms those with positive γ and GD ($\gamma = 0$). When $L = 2, 3, 4$, however, for all values of γ, `CfGD-v2` significantly outperforms GD. Yet, the best result is obtained by `CfGD-v2` with $L = 1, \gamma = -0.25$, which reaches to the stationary point within the machine accuracy in 4×10^4 iterations. GD exhibits a linear convergence (as expected), however, since the model matrix A is ill-conditioned (the condition number of A is 90,053), the convergence speed is too slow to see a significant improvement within 10^5 iterations. This demonstrates the effectiveness of `CfGD-v2` in mitigating the dependence on the condition number in the rate of convergence.

4.2. Quadratic objective function: Real data

We employ the dataset from UCI Machine Learning Repository's "Gas Sensor Array Drift at Different Concentrations" (Rodriguez-Lujan, Fonollosa, Vergara, Homer, & Huerta, 2014; Vergara, Vembu, Ayhan, Ryan, Homer, & Huerta, 2012). Specifically, we used the datasets Ethanol problem a scalar regression task with 2565 examples, each comprising 128 features (one of the largest numeric regression tasks in the repository). The input and output data sets are normalized to have zero mean and

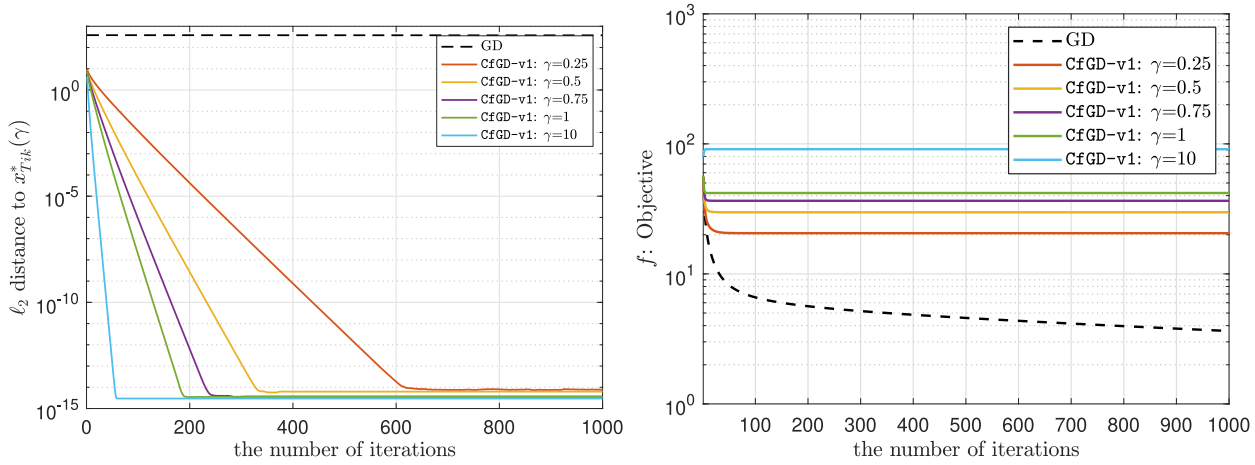

Fig. 4. (To be viewed in color) (Left) The ℓ_2 distance to the solution $\mathbf{x}^*_{\text{Tik}}(\gamma)$ (16) to Tikhonov regularization with respect to the number of iterations at varying γ. (Right) The objective values versus the number of iterations. `CfGD-v1` does not effectively decrease the objective function as it does not converge to the integer-order stationary point $\mathbf{x}^*$ of the objective.

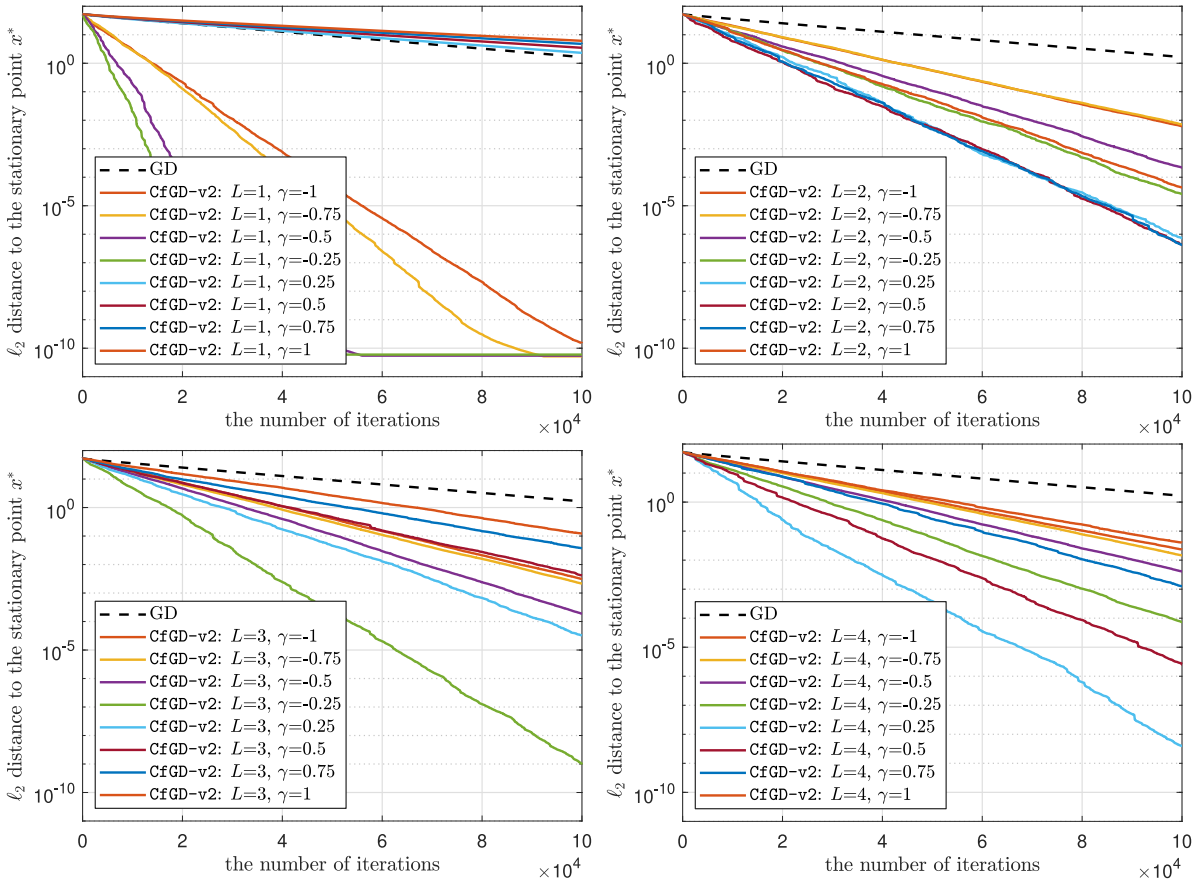

Fig. 5. (To be viewed in color) The ℓ_2 distance to the stationary point $\mathbf{x}^*$ with respect to the number of iterations at varying γ and L. (Top) $L = 1, 2$ and (Bottom) $L = 3, 4$ from left to right. GD corresponds to the case of $\gamma = 0$. The condition number of the model matrix A is 90,053.

unit variance. After the normalization, the condition number of the input data matrix W is 70,980, yielding a gigantic condition number of the model matrix $A = WW^\top$ of $(70,980)^2 \approx 5 \times 10^9$.

In Fig. 6, we report the ℓ_2 distance to the stationary point $\mathbf{x}^*$ versus the number of iterations. We employ `CfGD-v2` with $L = 1$ at varying $\gamma \in [-100, -20]$. The initial point $\mathbf{x}^{(-1)}$ is set to zero and $\mathbf{x}^{(0)}$ is randomly chosen from the uniform distribution in the hypercube $[-10, 10]^d$. GD corresponds to the case of $\gamma = 0$ and its trajectories are shown as black dashed-lines. We clearly see that `CfGD-v2` converges significantly faster than GD. In almost all cases of γ, `CfGD-v2` converges to the optimum $\mathbf{x}^*$ within the ℓ_2 error of 10^{-5} in merely 3×10^5 iterations. For GD, as expected, we

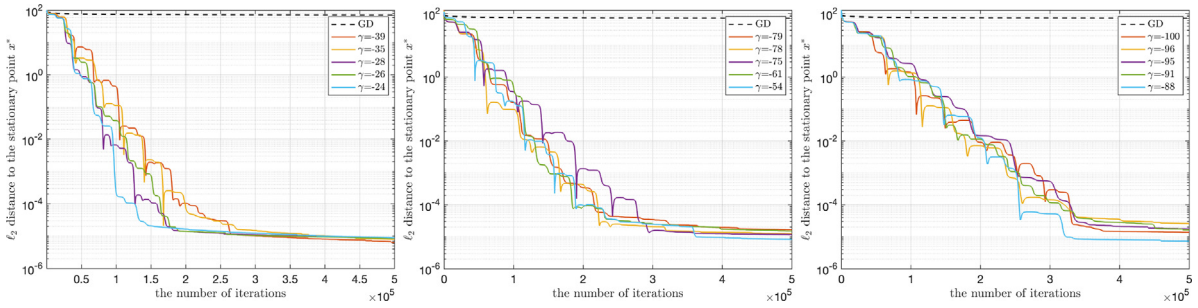

Fig. 6. (To be viewed in color) The regression results for the UCI Machine Learning Repository's dataset of 2565 examples, which leads to a gigantic the condition number of $(70,980)^2 \approx 5 \times 10^9$. The ℓ_2 distance to the stationary point $\mathbf{x}^*$ with respect to the number of iterations at varying γ and $L = 1$. Here γ's are randomly chosen from $\{-100, \ldots, -20\}$. Both GD and CfGD-v2 employ the optimal stepsize (15).

cannot see any significant improvement within 5×10^5 iterations. Again, this clearly demonstrates that CfGD-v2 can effectively mitigate the dependence on the condition number in the rate of convergence, resulting a significant acceleration over GD.

4.3. Nonconvex objective functions: Neural networks

We consider the training of neural network by CfGD-v2 in function approximation tasks. The test functions are

$$\begin{aligned} h_1(z) &= \sin(5\pi z), \\ h_2(z) &= \sin(2\pi z)e^{-z^2}, \\ h_3(z) &= \mathbb{I}_{z>0}(z) + 0.2\sin(2\pi z), \end{aligned} \tag{17}$$

whose graphs are shown in the top row of Fig. 7. We employ the univariate hyperbolic tangent two-layer neural network $N(\cdot;\mathbf{x}) : \mathbb{R} \ni z \to N(z;\mathbf{x}) \in \mathbb{R}$ where $\mathbf{x}$ is a network parameter, defined as follows:

$$N(z;\mathbf{x}) = \sum_{j=1}^{n} a_{3,j}\tanh(a_{1,j}z + a_{2,j}), \qquad \mathbf{x} = \{a_{1,j}, a_{2,j}, a_{3,j}\}_{j=1}^{n} \in \mathbb{R}^d,$$

where $d = 3n$. Given a set of training data $\{(z_i, h(z_i))\}_{i=1}^m$, the objective (loss) function is defined by

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(N(z_i;\mathbf{x}) - h(z_i))^2, \tag{18}$$

and the learning goal is to find an appropriate network parameter $\mathbf{x}^*$ that minimizes f so that $N(\cdot;\mathbf{x}^*) \approx h$. Thanks to Theorem 2.7, the Caputo fractional gradient of f can be efficiently computed by using the Gauss–Jacobi quadrature (Ralston & Rabinowitz, 2001).

For fixed $\{a_{1,j}, a_{2,j}\}$, the objective function f with respect to the coefficients $\{a_{3,j}\}$ is quadratic. Hence, we employ the optimal learning rate (13) for the coefficients $\{a_{3,j}\}$. For the weights and the biases $\{a_{1,j}, a_{2,j}\}$, we select the best learning rate among 32 selections – $\{t \times 10^{-l} : l = 1, \ldots, 8, t = \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \frac{4}{4}\}$ in every iteration. Here the best learning rate is the one that yields the largest decrease in the loss. For comparison, we also report the results of gradient descent (GD). Similarly, we employ the optimal learning rate (13) for the coefficients and the best one among 32 selections for the weights and biases. We remark that while the considered learning task is univariate function approximation, the resulting optimization problem is of dimension $d = 3n$.

In the following tests, we set $m = 100$, $n = 50$ and use 10 quadrature points. The 100 training data points are randomly uniformly drawn from $(-1, 1)$. To measure the performance of the trained neural networks, we also report the test error, which is the mean square error on another 100,000 points uniformly randomly drawn from $(-1, 1)$. CfGD-v2 requires two parameters – α is a fractional order and γ is a parameter that determines β such that $\beta = \gamma + \frac{1-\alpha}{2-\alpha}$.

Fig. 7 shows the training loss and the test error trajectories by GD and CfGD-v2 with respect to the number of iterations. For CfGD-v2, we set $\alpha = 0.7$ for h_1, and $\alpha = 0.4$ for h_2 and h_3. For each test, we choose three different values of γ. On the left, the approximation results for h_1 are reported. We observe that CfGD-v2 ($\alpha = 0.70, \gamma = 20, 40, 70$) reaches the loss of 2×10^{-4} at the end of the training, while GD landed at the loss level of 6×10^{-4}. At around 2×10^4 iterations, we see that CfGD-v2 already reduces the loss function to the level of 10^{-3}, while GD has not effectively decreased the loss staying at the level of 10^{-1}. This clearly shows the faster convergence of CfGD. In the middle and right, we show the results for h_2 and h_3. Again, similar behavior is observed. We clearly see that CfGD-v2 consistently converges faster than GD in all cases. Furthermore, we found that CfGD-v2 not only converges faster but also produces neural networks that generalize well. We see that the test errors are generally smaller than or equal to the ones by GD.

We investigate how the number of quadrature points affects the performance of CfGD-v2. In Fig. 8, we report the results of CfGD-v2 implemented by using s quadrature points with $s = 1, \ldots, 10$. For h_1, we set $\alpha = 0.70, \gamma = 70$. For h_2 and h_3, we set $\alpha = 0.4$ for both and $\gamma = 10$ and $\gamma = -150$, respectively. We clearly see that for any choices of s including $s = 1$, CfGD-v2 outperforms GD. As discussed in Section 3, the computational cost of CfGD-v2 is approximately s-times higher than those of GD. Hence, even when the computational cost is taken into account (e.g. $s = 1$), we see that CfGD-v2 converges faster than GD. We note that when s is small, the direction ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ (9) obtained by a quadrature rule may no longer be an accurate approximation to ${}_c\mathbf{D}_\beta^\alpha f(\mathbf{x})$ (6). Yet, we empirically found that regardless of the number of quadrature points, CfGD-v2 implemented by ${}_c\mathbf{Q}_\beta^\alpha f(\mathbf{x})$ still produces good directions for the purpose of minimizing the loss function. We defer further investigation to future work.

Lastly, we found that while CfGD-v2 seems not very sensitive to the choice of the parameter γ, there is a range of γ that makes CfGD-v2 more effective. Understanding the effect of γ also requires further investigation, which is deferred to future research.

4.4. Nonconvex objective function: Physics informed neural networks

Let us consider the inhomogeneous two-dimensional Helmholtz equation:

$$u_{xx} + u_{yy} + (\nu^*)^2 u = -g \quad \text{on} \quad \Omega = (-1, 1)^2, \tag{19}$$

with appropriate Dirichlet boundary conditions. We consider the analytic solution of $u^*(x, y) = \sin(\pi x)\sin(2\pi y)$ with $\nu^* = 1$; g is determined accordingly. Given some sparse measurements in the domain, we are interested in inferring both the solution u^* and the wave number ν^* simultaneously using neural networks.

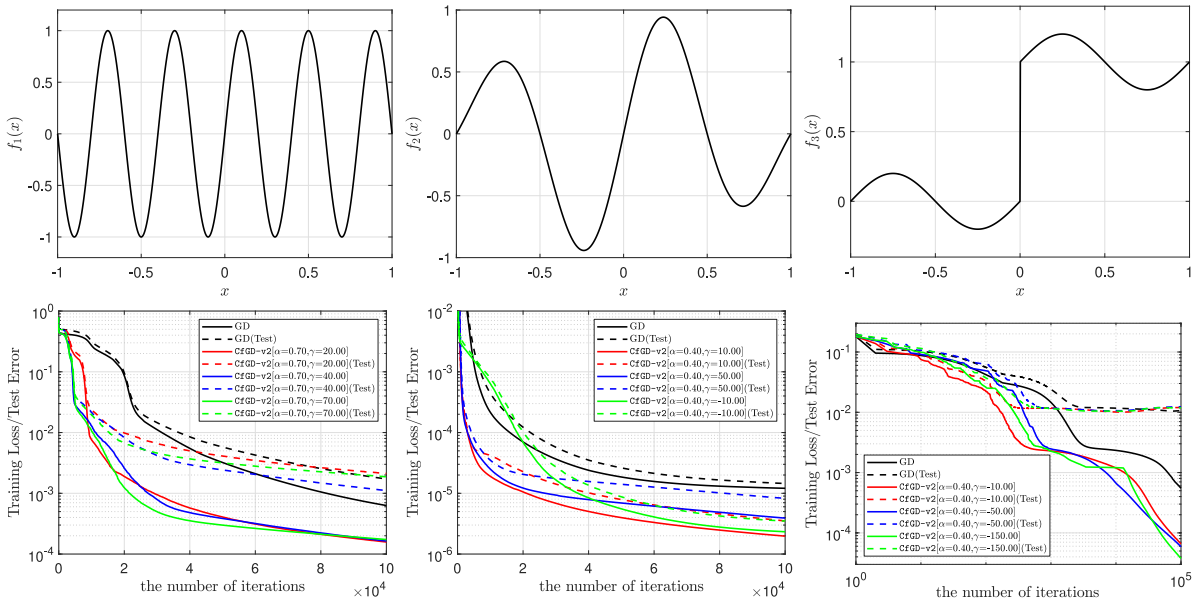

Fig. 7. (To be viewed in color) Approximation results for the three test functions (17) – (left) $h_1(x) = \sin(5\pi x)$, (middle) $h_2(x) = \sin(2\pi x)e^{-x^2}$, and (right) $h_3(x) = \mathbb{I}_{x>0}(x) + 0.2\sin(2\pi x)$. (Top) The graphs of the three test functions. (Bottom) The training loss and the test error trajectories are shown with respect to the number of iterations.

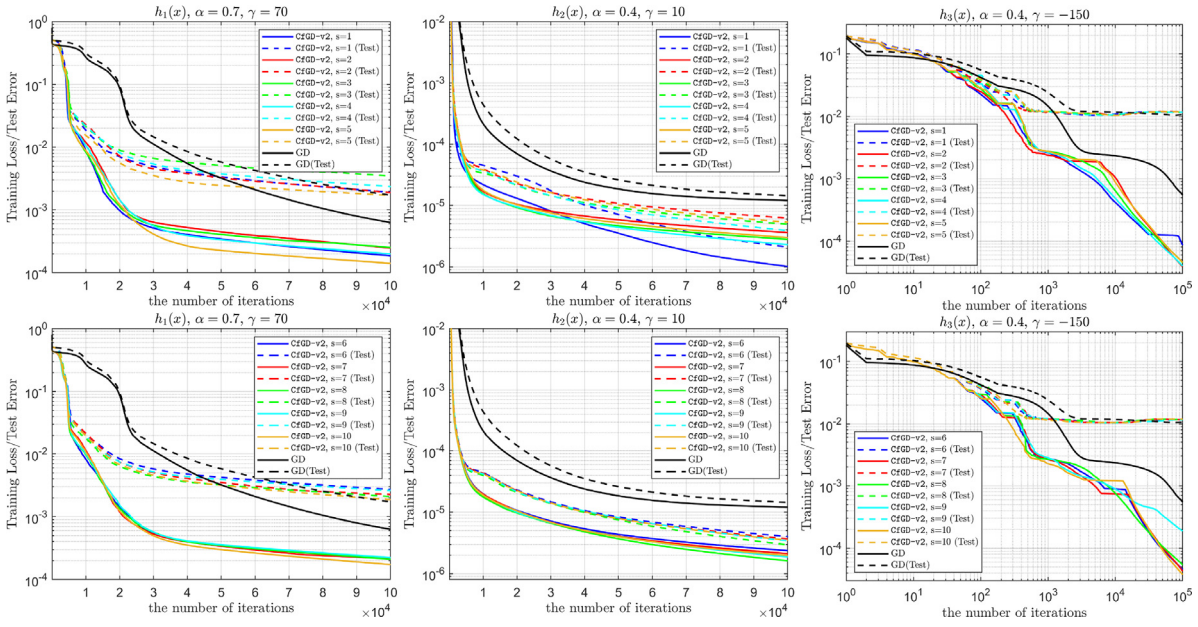

Fig. 8. (To be viewed in color) Approximation results for the three test functions (17) – (left) $h_1(x) = \sin(5\pi x)$, (middle) $h_2(x) = \sin(2\pi x)e^{-x^2}$, and (right) $h_3(x) = \mathbb{I}_{x>0}(x) + 0.2\sin(2\pi x)$. The training loss and the test error trajectories are shown with respect to the number of iterations at varying the number s of the Gauss–Jacobi quadrature points. (Top) From $s = 1$ to $s = 5$. (Bottom) From $s = 6$ to $s = 10$.

Let us consider a two-layer hyperbolic tangent neural network defined as follow:

$$N(\boldsymbol{z};\theta_{\mathrm{NN}}) = \sum_{j=1}^{n} a_{4,j}\tanh(a_{1,j}x + a_{2,j}y + a_{3,j}),$$

where $\boldsymbol{z} = (x, y)$ and $\theta_{\mathrm{NN}} = \{a_{1,j}, a_{2,j}, a_{3,j}, a_{4,j}\}_{j=1}^{n} \in \mathbb{R}^{4n}$. The goal is to find the parameters $\theta := \theta_{\mathrm{NN}} \cup \{\nu\} \in \mathbb{R}^d$ with $d = 4n + 1$ that minimizes the physics-informed loss function defined by

$$f(\theta) = \frac{1}{|D_{\mathrm{rd}}|} \times \sum_{\boldsymbol{z}\in D_{\mathrm{rd}}} \left(N_{xx}(\boldsymbol{z};\theta_{\mathrm{NN}}) + N_{yy}(\boldsymbol{z};\theta_{\mathrm{NN}}) + \nu^2 N(\boldsymbol{z};\theta_{\mathrm{NN}}) + g(\boldsymbol{z})\right)^2 + \frac{1}{|D_{\mathrm{bd}}|}\sum_{\boldsymbol{z}\in D_{\mathrm{bd}}} \left(N(\boldsymbol{z};\theta_{\mathrm{NN}}) - u^*(\boldsymbol{z})\right)^2 + \frac{1}{|D_{\mathrm{sd}}|}\sum_{\boldsymbol{z}\in D_{\mathrm{sd}}} \left(N(\boldsymbol{z};\theta_{\mathrm{NN}}) - u^*(\boldsymbol{z})\right)^2,$$

where $D_{\mathrm{rd}}, D_{\mathrm{sd}} \subset \Omega$, $D_{\mathrm{bd}} \subset \partial\Omega$ are the sets of residual, solution, and boundary data points, respectively. D_{rd}, D_{sd}, D_{bd} are randomly

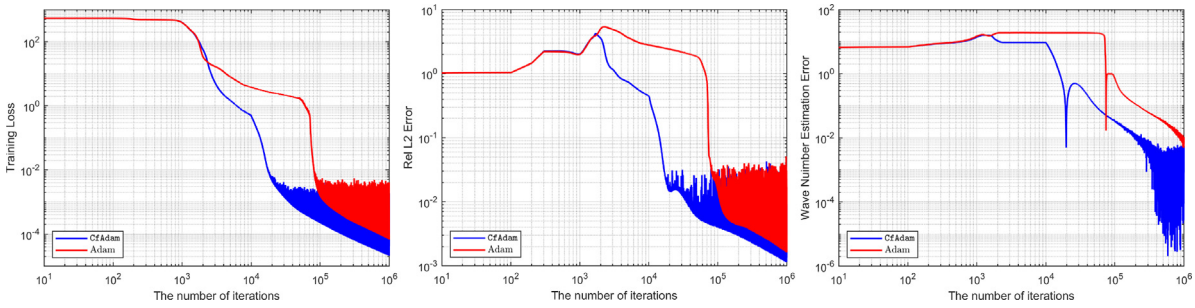

Fig. 9. (To be viewed in color) The training loss (left), the relative ℓ_2 error (middle), the wave number estimation error (right) versus the number of iterations.

uniformly generated and their cardinalities are 1000, 600 and 400, respectively.

For a comparison, we employ the Adam optimization algorithm (Kingma & Ba, 2014), one of the most popular gradient-based optimization methods for neural networks. Since Adam typically outperforms GD, we consider `CfAdam` (11). Instead of replacing all the gradient to the Caputo fractional one, `CfAdam` updates the parameters of the hidden layer, $\{a_{1,i}, a_{2,i}, a_{3,i}\}_{i=1}^n$, and ν via the Caputo fractional gradient with random adaptive fractional orders, while updates the remaining parameters, $\{a_{4,1}, \ldots, a_{4,n}\}$, using the standard gradient. Specifically, if we let

$$(\theta)_i = \begin{cases} a_{1,i} & \text{if } 1 \le i \le n, \\ a_{2,i-n} & \text{if } n < i \le 2n, \\ a_{3,i-2n} & \text{if } 2n < i \le 3n, \\ a_{4,i-3n} & \text{if } 3n < i \le 4n, \\ \nu & \text{if } i = 4n+1, \end{cases}$$

the kth iteration of `CfAdam` uses the fractional orders $\boldsymbol{\alpha}_k = (\alpha_1^{(k)}, \ldots, \alpha_d^{(k)})$ that are set to $\alpha_i^{(k)} = \alpha^{(k)}$ if $1 \le i \le 3n, i = 4n+1$, and $\alpha_i^{(k)} = 1$ if $3n < i \le 4n$, where $\alpha^{(k)}$ is randomly sampled from the uniform distribution on $(\alpha_k^{\text{low}}, 1)$ with $\alpha_k^{\text{low}} = \min\{1, 0.8 + 0.02\lfloor k/10^4 \rfloor\}$. We set $\mathbf{c}_k = \theta^{(k-1)}$ and $\boldsymbol{\beta}_k = \boldsymbol{Z}_k \odot (1 - \boldsymbol{\alpha}_k)$, where $\boldsymbol{Z}_k$ is a standard normal random vector and $\odot$ is the elementwise multiplication. The learning rate is set to 10^{-3}. The Caputo fractional-based gradient is computed using (9) with $s = 1$. The width of the network is set to $n = 200$.

In Fig. 9, we plot the training loss (left), the relative ℓ_2 error (middle) and the wave number estimation error (right) with respect to the number of iterations. The relative ℓ_2 errors were computed based on another 10^5 uniformly randomly sampled points from Ω. The wave number estimation error is measured by $|(\nu^*)^2 - (\nu^{(k)})^2|$. It is clear that `CfAdam` outperforms Adam. It is worth noting that since $\alpha_k^{\text{low}} = 1$ if $k \ge 10^5$, the acceleration of `CfAdam` mainly comes from the first 10^5 iterations. As a matter of fact, it can be observed that `CfAdam` exhibits a lot faster convergence in the first 10^5 iterations, while it shows a similar rate compared to those of Adam in the rest of the iterations.

4.5. Comparative experiment

We compare the performance of the proposed method (`CfGD-v2`) with those of the two methods from Wang et al. (2017) and Wei et al. (2020). We note that since each method requires one to determine multiple parameters (e.g. $\alpha, \beta, c, \epsilon, L, s$) and the performance is highly sensitive to them, fair and meaningful comparative experiments require extensive and well-designed numerical tests, which is beyond the scope of the present work. While we defer thorough comparative investigations to future work, here we provide one numerical test. Since the method from Wang et al. (2017) is only applicable for training of neural networks, we consider a task of learning the target function h using a two-layer tanh neural network of width 50. The target function h is defined by $h(z) = \frac{\sin(10\pi(z+1.5))}{2(z+1.5)} + (z+0.5)^4$, known as the Gramacy and Lee function. 100 training data were generated randomly from the uniform distribution $[-1, 1]$ and the test MSE were computed based on another 10^5 uniform random samples. (See Section 4.3 for more detailed experiment setups). Fig. 10 shows the training loss and the test MSE with respect to the number of iterations for gradient descent (GD), our method (`CfGDv2`), FGD from Wang et al. (2017) (`FGD` Wang et al., 2017), and FGD from Wei et al. (2020) (`FGD` Wei et al., 2020). It can be seen that our method outperforms all the others.

5. Summary

We proposed a general class of fractional-order optimization algorithms via the Caputo fractional derivatives. Theorem 2.5 serves as the theoretical motivation on using the Caputo fractional derivatives in optimization. Based on Theorem 2.5, we defined the Caputo fractional-based *gradient* (6), which generalizes the standard integer-order gradient. An efficient implementation was developed from an equivalent formulation derived in Theorem 2.7. By replacing integer-order gradients with the Caputo fractional-based ones, we proposed the Caputo fractional gradient descent (`CfGD`) and the Caputo fractional Adam (`CfAdam`) that generalize GD and Adam, respectively. We demonstrated the performance of `CfGD` and `CfAdam` on ill-conditioned least squares, function approximation and an inverse PINN problem. We analyzed `CfGD` for quadratic functions. Theorems 3.4 and 3.5 indicate that `CfGD` could mitigate the dependence on the condition number in the rate and results in significant acceleration over GD.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was funded by the PhILMS, USA grant DE-SC0019453, the ARO MURI, USA W911NF-15-1-0562 and the AFOSR, USA MURI FA9550-20-1-0358.

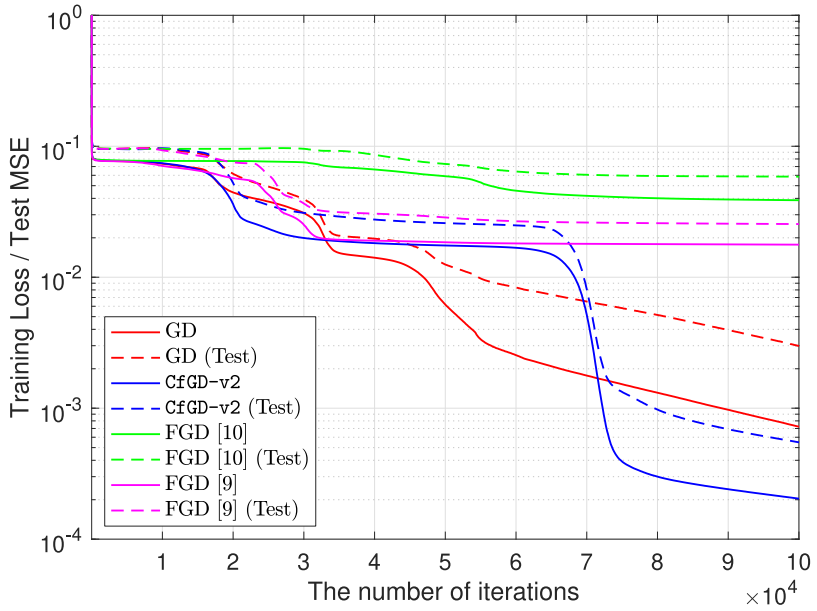

Fig. 10. The training loss and the test MSE with respect to the number of iterations for GD, CfGD-v2, FGD (Wang et al., 2017) and FGD (Wei et al., 2020). For CfGD-v2, we set $\alpha = 0.7$, $\gamma = 70$, $L = 1$, and. $s = 1$. For FGD (Wang et al., 2017), we set $\alpha = 0.7$. For FGD (Wei et al., 2020), we set $\alpha = 0.7$ and $\epsilon = 0$.

Appendix A. Basic calculations

Here we collected all the necessary calculations involving the Caputo fractional derivative.

Lemma A.1. *Let $I : \mathbb{R} \to \mathbb{R}$ be the identity map defined by $I(x) = x$. For $0 < \alpha < 1$ and for any c,*

$$ {}_c^C D_x^\alpha I = \frac{sign(x-c)}{\Gamma(2-\alpha)}|x-c|^{1-\alpha}, \qquad {}_c^C D_x^\alpha I^2 = 2({}_c^C D_x^\alpha I)(\gamma_\alpha(x-c)+x), $$

where $\gamma_\alpha = -\frac{1-\alpha}{2-\alpha}$. Also, we have

$$ {}_c^C D_x^{1+\alpha} I = 0, \qquad {}_c^C D_x^{1+\alpha} I^2 = 2\, sign(x-c)\, {}_c^C D_x^\alpha I. $$

Proof. Direct calculations lead to the results. □

Proposition A.2. *For $0 < \alpha < 1$ and $c \in \mathbb{R}$, let $f \in \mathcal{F} \cap \mathcal{D}^\alpha(c) \cap \mathcal{D}^{1+\alpha}(c)$ where $\mathcal{F}$ is the function class defined in Definition 2.4. Then, there exists $\rho > 0$ such that for all $x \in B_\rho(c) := (c-\rho, c+\rho)$,*

$$ {}_c^C D_x^\alpha f = ({}_c^C D_x^\alpha I) \sum_{k=1}^{\infty} \frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} f^{(k)}(c)(x-c)^{k-1}, $$

$$ |x-c|\, {}_c^C D_x^{1+\alpha} f = ({}_c^C D_x^\alpha I) \sum_{k=2}^{\infty} \frac{\Gamma(2-\alpha)}{\Gamma(k-\alpha)} f^{(k)}(c)(x-c)^{k-1}. $$

Proof. For $x > c$, by definition and applying integration by parts, we obtain

$$ \begin{aligned} {}_c^C D_x^\alpha f &= \frac{1}{\Gamma(1-\alpha)} \int_c^x f'(s)(x-s)^{-\alpha} ds \\ &= \frac{f'(c)}{\Gamma(2-\alpha)}(x-c)^{1-\alpha} + \frac{1}{\Gamma(2-\alpha)} \int_c^x f^{(2)}(s)(x-s)^{1-\alpha} ds. \end{aligned} $$

By repeating integration by parts, we have

$$ \begin{aligned} {}_c^C D_x^\alpha f &= \sum_{k=1}^{n} \frac{f^{(k)}(c)}{\Gamma(k+1-\alpha)}(x-c)^{k-\alpha} \\ &\quad + \frac{1}{\Gamma(n+1-\alpha)} \int_c^x f^{(n+1)}(s)(x-s)^{n-\alpha} ds \\ &= ({}_c^C D_x^\alpha I) \frac{\Gamma(2-\alpha)}{(x-c)^{1-\alpha}} \sum_{k=1}^{n} \frac{f^{(k)}(c)}{\Gamma(k+1-\alpha)}(x-c)^{k-\alpha} + R_n(x,c) \\ &= ({}_c^C D_x^\alpha I) \sum_{k=1}^{n} \frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} f^{(k)}(c)(x-c)^{k-1} + R_n(x,c), \end{aligned} $$

where $R_n(x,c) = \frac{1}{\Gamma(n+1-\alpha)} \int_c^x f^{(n+1)}(s)(x-s)^{n-\alpha} ds$ and the second equality uses Lemma A.1. Similarly, if $x < c$, one can check that

$$ \begin{aligned} {}_x^C D_c^\alpha f &= \frac{-1}{\Gamma(1-\alpha)} \int_x^c f'(s)(s-x)^{-\alpha} ds \\ &= -\frac{f'(c)}{\Gamma(2-\alpha)}(c-x)^{1-\alpha} + \frac{1}{\Gamma(2-\alpha)} \int_x^c f^{(2)}(s)(s-x)^{1-\alpha} ds \\ &= ({}_x^C D_c^\alpha x) f'(c) + \frac{f''(c)}{\Gamma(3-\alpha)}(c-x)^{2-\alpha} \\ &\quad - \frac{1}{\Gamma(3-\alpha)} \int_x^c f^{(3)}(s)(s-x)^{2-\alpha} ds \\ &= ({}_x^C D_c^\alpha x) \left[f'(c) + \frac{\Gamma(2-\alpha)}{\Gamma(3-\alpha)} f''(c)(x-c)^1 \right] \\ &\quad - \frac{1}{\Gamma(3-\alpha)} \int_x^c f^{(3)}(s)(s-x)^{2-\alpha} ds \\ &= \cdots = ({}_x^C D_c^\alpha x) \sum_{k=1}^{n} \frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} f^{(k)}(c)(x-c)^k + R_n'(x,c), \end{aligned} $$

where $R_n'(x,c) = \frac{(-1)^{n+1}}{\Gamma(n+1-\alpha)} \int_x^c f^{(n+1)}(s)(s-x)^{n-\alpha} ds$. Note that

$$ \begin{aligned} |R_n(x,c)| &\le \frac{\sup_{y\in[c,x]} |f^{(n+1)}(y)|}{\Gamma(n+1-\alpha)} \int_c^x (x-s)^{n-\alpha} ds \\ &= \frac{\sup_{y\in[c,x]} |f^{(n+1)}(y)|}{\Gamma(n+2-\alpha)}(x-c)^{n+1-\alpha} \\ &= ({}_c^C D_x^\alpha I) \frac{\Gamma(2-\alpha)}{\Gamma(n+2-\alpha)}(x-c)^n \sup_{y\in[c,x]} |f^{(n+1)}(y)| \\ &\le ({}_c^C D_x^\alpha I) \frac{\Gamma(2-\alpha)}{\Gamma(n+2-\alpha)}(x-c)^n C \frac{\Gamma(n+2)}{\rho^n} \\ &= \mathcal{O}(r^n (n+2)^\alpha), \end{aligned} $$

where $r = |x-c|/\rho$, and $\frac{\Gamma(n+1+1)}{\Gamma(n+1+(1-\alpha))} \le (n+2)^\alpha$ is used in the last inequality. Since $|r| < 1$, we have $\lim_{n\to\infty} |R_n(x,c)| = 0$. Similarly, one can show that $\lim_{n\to\infty} |R_n'(x,c)| = 0$.

The other equation can be checked similarly. Observe that if $x > c$, we have

$$
\begin{aligned}
{}_c^C D_x^{1+\alpha} f &= \frac{1}{\Gamma(1-\alpha)} \int_c^x f''(s)(x-s)^{-\alpha} ds \\
&= \left[\frac{-f''(s)(x-s)^{1-\alpha}}{\Gamma(2-\alpha)} \right]_c^x + \frac{1}{\Gamma(2-\alpha)} \int_c^x f^{(3)}(s)(x-s)^{1-\alpha} ds \\
&= ({}_c^C D_x^\alpha I) f''(c) + \left[\frac{-f^{(3)}(s)(x-s)^{2-\alpha}}{\Gamma(3-\alpha)} \right]_c^x \\
&\quad + \frac{1}{\Gamma(3-\alpha)} \int_c^x f^{(4)}(s)(x-s)^{2-\alpha} ds \\
&= \cdots = ({}_c^C D_x^\alpha I) \sum_{k=1}^{n} \frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} f^{(k+1)}(c)(x-c)^{k-1} + \widetilde{R}_n(x,c),
\end{aligned}
$$

where $\widetilde{R}_n(x,c) := \frac{\int_c^x f^{(n+2)}(s)(x-s)^{n-\alpha} ds}{\Gamma(n+1-\alpha)}$. Also,

$$
\begin{aligned}
|\widetilde{R}_n(x,c)| &\le \frac{\sup_{y\in[c,x]} |f^{(n+2)}(y)|}{\Gamma(n+2-\alpha)} (x-c)^{n+1-\alpha} \\
&\le C \frac{\Gamma(n+3)}{\Gamma(n+2-\alpha)} \frac{(x-c)^{n+1-\alpha}}{\rho^{n+1}} \\
&\le ({}_c^C D_x^\alpha I) C \frac{\Gamma(2-\alpha)(n+2)}{\rho} \frac{\Gamma(n+2)}{\Gamma(n+2-\alpha)} r^n \\
&= \mathcal{O}(r^n (n+2)^{1+\alpha}),
\end{aligned}
$$

which shows $\lim_{n\to\infty} |\widetilde{R}_n(x,c)| = 0$. We omit the proof for the case $x < c$ as it can be done very similarly. □

Appendix B. Proof of Theorem 2.5

Proof. Let ${}_cF_{\alpha,\beta}$ be a smoothing of f, formally defined by

$$
{}_cF_{\alpha,\beta}(z) := \sum_{k=0}^{\infty} a_k (z-c)^k, \qquad a_k = C_{k,\alpha,\beta} f^{(k)}(c),
$$

where $C_{k,\alpha,\beta} = 1$ if $k \le 1$ and $C_{k,\alpha,\beta} = \frac{1}{k}\left(\frac{\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)} + \beta \frac{\Gamma(2-\alpha)}{\Gamma(k-\alpha)} \right)$ if $k \ge 2$. Note that for $k \ge 2$,

$$
|C_{k,\alpha,\beta}| \le \frac{2\max\{1,|\beta|\}\Gamma(2-\alpha)}{\Gamma(k+1-\alpha)}.
$$

Let $C = 2\max\{1,|\beta|\}\Gamma(2-\alpha)$. It then follows from the root test that

$$
\begin{aligned}
|a_k (z-c)^k|^{1/k} &\le \left| \frac{C}{\Gamma(k+1-\alpha)} f^{(k)}(c) |z-c|^k \right|^{1/k} \\
&\le \left| \frac{C}{\Gamma(k+1-\alpha)} \frac{\Gamma(k+1)}{\rho^{k-1}} |z-c|^k \right|^{1/k} \\
&\le (C\rho(k+1)^\alpha)^{1/k} \cdot r \to r \text{ as } k \to \infty,
\end{aligned}
$$

where $r = |z-c|/\rho$. Since $r < 1$ for any $z \in B_\rho(c)$, the series converges if $z \in B_\rho(c)$.

The linear approximation of ${}_cF_{\alpha,\beta}(z)$ at $x \ne c$ is given by ${}_c\ell_{\alpha,\beta}(z) = {}_cF_{\alpha,\beta}(x) + {}_cF'_{\alpha,\beta}(x)(z-x)$. Hence, the steepest descent direction of ${}_cF_{\alpha,\beta}(z)$ at $x \in B_\rho(c)\backslash\{c\}$ is ${}_cF'_{\alpha,\beta}(x) = \frac{{}_c^C D_x^\alpha f}{{}_c^C D_x^\alpha I} + \beta |x-c| \frac{{}_c^C D_x^{1+\alpha} f}{{}_c^C D_x^\alpha I}$, where the second equality holds from Proposition A.2. □

Appendix C. Proof of Theorem 2.7

The proof is readily followed by the following Lemma.

Lemma C.1. *For $\alpha \in (0,1)$ and $\mathbf{c} = (c_j) \in \mathbb{R}^d$, let $f \in \mathcal{D}_\alpha(\mathbf{c})$ be a real-valued function defined on $\mathbb{R}^d$. Let $\Delta_j = \frac{x_j - c_j}{2}$. Then, for $j = 1, \dots, d$, we have*

$$
\begin{aligned}
({}_{c_j}^C D_{x_j}^\alpha I)^{-1} \left({}_{\mathbf{c}}^C \nabla_{\mathbf{x}}^\alpha f \right)_j &= C_\alpha \int_{-1}^{1} f'_{j,\mathbf{x}}(\Delta_j(1+u)+c_j)(1-u)^{-\alpha} du, \\
(|{}_{c_j}^C D_{x_j}^\alpha I|)^{-1} \left({}_{\mathbf{c}}^C \nabla_{\mathbf{x}}^{1+\alpha} f \right)_j &= C_\alpha \int_{-1}^{1} f''_{j,\mathbf{x}}(\Delta_j(1+u)+c_j)(1-u)^{-\alpha} du,
\end{aligned}
$$

where $C_\alpha = (1-\alpha)2^{-(1-\alpha)}$ and $f_{j,\mathbf{x}}$'s are defined in (5).

Proof. Suppose $c_j^* < x_j^*$. By definition, we have $\left({}_{\mathbf{c}^*}^C \nabla_{\mathbf{x}^*}^\alpha f \right)_j = \frac{1}{\Gamma(1-\alpha)} \int_{c_j^*}^{x_j^*} f'_{j,\mathbf{x}^*}(t)(x_j^* - t)^{-\alpha} dt$. Observe that

$$
\begin{aligned}
&\frac{1}{\Gamma(1-\alpha)} \int_{c_j^*}^{x_j^*} f'_{j,\mathbf{x}^*}(t)(x_j^* - t)^{-\alpha} dt \\
&= (1-\alpha) \frac{(x_j^* - c_j^*)^{1-\alpha}}{\Gamma(2-\alpha)} \int_{c_j^*}^{x_j^*} f'_{j,\mathbf{x}^*}(t) \left(\frac{x_j^* - t}{x_j^* - c_j^*} \right)^{-\alpha} \frac{dt}{x_j^* - c_j^*} \\
&= (1-\alpha) {}_{c_j^*}^C D_{x_j^*}^\alpha I \int_0^1 f'_{j,\mathbf{x}^*}(2\Delta_j s + c_j^*)(1-s)^{-\alpha} ds,
\end{aligned}
$$

where $\Delta_j = \frac{x_j^* - c_j^*}{2}$, $I(x) = x$ is the identity map and the change of variable with $s = \frac{t - c_j^*}{x_j^* - c_j^*}$ is used in the last equality. By further using the change of variable with $u = 2s - 1$, the above can be written as

$$
\frac{1-\alpha}{2^{1-\alpha}} {}_{c_j^*}^C D_{x_j^*}^\alpha I \int_{-1}^{1} f'_{j,\mathbf{x}^*}(\Delta_j(1+u) + c_j^*)(1-u)^{-\alpha} du.
$$

Now suppose $x_j^* < c_j^*$. By definition, we have $\left({}_{\mathbf{c}^*}^C \nabla_{\mathbf{x}^*}^\alpha f \right)_j = \frac{-1}{\Gamma(1-\alpha)} \int_{x_j^*}^{c_j^*} f'_{j,\mathbf{x}^*}(t)(t - x_j^*)^{-\alpha} dt$. Following a similar change of variables, we have

$$
\begin{aligned}
&\frac{-1}{\Gamma(1-\alpha)} \int_{c_j^*}^{x_j^*} f'_{j,\mathbf{x}^*}(t)(t - x_j^*)^{-\alpha} dt \\
&= (1-\alpha) \frac{-(c_j^* - x_j^*)^{1-\alpha}}{\Gamma(2-\alpha)} \int_{x_j^*}^{c_j^*} f'_{j,\mathbf{x}^*}(t) \left(\frac{t - x_j^*}{c_j^* - x_j^*} \right)^{-\alpha} \frac{dt}{c_j^* - x_j^*} \\
&= (1-\alpha) {}_{c_j^*}^C D_x^\alpha I(x_j^*) \int_0^1 f'_{j,\mathbf{x}^*}(-2\Delta_j s + x_j^*) s^{-\alpha} ds,
\end{aligned}
$$

where the change of variable with $s = \frac{t - x_j^*}{c_j^* - x_j^*}$ is used in the last equality. By further using the change of variable with $u = 1 - 2s$, the above can be written as

$$
\frac{1-\alpha}{2^{1-\alpha}} {}_{c_j^*}^C D_{x_j^*}^\alpha I \int_{-1}^{1} f'_{j,\mathbf{x}^*}(\Delta_j(1+u) + c_j^*)(1-u)^{-\alpha} du,
$$

which completes the first part of the proof.

Next, suppose $c_j^* < x_j^*$. Observe that $\left({}_{\mathbf{c}^*}^{C}\nabla_{\mathbf{x}^*}^{1+\alpha} f\right)_j = \frac{1}{\Gamma(1-\alpha)} \int_{c_j^*}^{x_j^*} f''_{j,\mathbf{x}^*}(t)(x_j^* - t)^{-\alpha} dt$. It then can be checked that

$$\begin{aligned}
&\frac{1}{\Gamma(1-\alpha)} \int_{c_j^*}^{x_j^*} f''_{j,\mathbf{x}^*}(t)(x_j^* - t)^{-\alpha} dt \\
&= (1-\alpha)\frac{(x_j^* - c_j^*)^{1-\alpha}}{\Gamma(2-\alpha)} \int_{c_j^*}^{x_j^*} f''_{j,\mathbf{x}^*}(t)\left(\frac{x_j^* - t}{x_j^* - c_j^*}\right)^{-\alpha} \frac{dt}{x_j^* - c_j^*} \\
&= (1-\alpha)\, {}_{c_j^*}^{C}D_{x_j^*}^{\alpha} I \int_0^1 f''_{j,\mathbf{x}^*}(2\Delta_j s + c_j^*)(1-s)^{-\alpha} ds.
\end{aligned}$$

By using the change of variable with $u = 2s - 1$, the above can be written as

$$\frac{1-\alpha}{2^{1-\alpha}}\, {}_{c_j^*}^{C}D_{x_j^*}^{\alpha} I \int_{-1}^1 f''_{j,\mathbf{x}^*}(\Delta_j(1+u) + c_j^*)(1-u)^{-\alpha} du.$$

Now suppose $x_j^* < c_j^*$. Similarly,it can be checked that

$$\begin{aligned}
\left({}_{\mathbf{c}^*}^{C}\nabla_{\mathbf{x}^*}^{1+\alpha} f\right)_j &= \frac{1}{\Gamma(1-\alpha)} \int_{x_j^*}^{c_j^*} f''_{j,\mathbf{x}^*}(t)(t - x_j^*)^{-\alpha} dt \\
&= -(1-\alpha)\frac{-(c_j^* - x_j^*)^{1-\alpha}}{\Gamma(2-\alpha)} \int_{x_j^*}^{c_j^*} f''_{j,\mathbf{x}^*}(t)\left(\frac{t - x_j^*}{c_j^* - x_j^*}\right)^{-\alpha} \frac{dt}{c_j^* - x_j^*} \\
&= -(1-\alpha)\, {}_{c_j^*}^{C}D_{x_j^*}^{\alpha} I \int_{-1}^1 f''_{j,\mathbf{x}^*}(2\Delta_j(1+u) + c_j^*)(1-u)^{-\alpha} ds. \quad \square
\end{aligned}$$

Appendix D. Proof of Theorem 3.2

Proof. Let $\mathbf{c} = (c_j)$ and $\mathbf{x} = (x_j)$. Let

$$M = \operatorname{diag}\left(\left[{}_{c_1}^{C}D_{x_1}^{\alpha} I \quad \cdots \quad {}_{c_d}^{C}D_{x_d}^{\alpha} I\right]\right) \in \mathbb{R}^{d\times d},$$

and $\tilde{\boldsymbol{w}} := (\sum_{k=1}^m w_{kj}^2)_j \in \mathbb{R}^d$. For $0 < \alpha < 1$, it can be checked that

$$\begin{aligned}
\left({}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{\alpha} f\right)_j &= \frac{1}{2}\sum_{i=1}^m {}_{c_j}^{C}D_{x_j}^{\alpha}(\mathbf{w}_i^T\mathbf{x} - y_i)^2 \\
&= \frac{1}{2}\sum_{i=1}^m {}_{c_j}^{C}D_{x_j}^{\alpha}\left[w_{ij}^2 x_j^2 + 2\Big(\sum_{l\neq j} w_{il}x_l - y_i\Big)w_{ij}x_j\right] \\
&= \frac{1}{2}\sum_{i=1}^m \left[w_{ij}^2 \cdot {}_{c_j}^{C}D_{x_j}^{\alpha} I^2 + 2\Big(\sum_{l\neq j} w_{il}x_l - y_i\Big)w_{ij} \cdot {}_{c_j}^{C}D_{x_j}^{\alpha} I\right] \\
&= {}_{c_j}^{C}D_{x_j}^{\alpha} I \sum_{i=1}^m \left(w_{ij}^2\left[\gamma_\alpha(x_j - c_j) + x_j\right] + w_{ij}\Big(\sum_{l\neq j} w_{il}x_l - y_i\Big)\right) \\
&= {}_{c_j}^{C}D_{x_j}^{\alpha} I \sum_{i=1}^m \left(w_{ij}^2\gamma_\alpha(x_j - c_j) + w_{ij}(\mathbf{w}_i^T\mathbf{x} - y_i)\right),
\end{aligned}$$

where the fourth equality uses Lemma A.1. Then, the Caputo fractional gradient of $f(\mathbf{x})$ is given by ${}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{\alpha} f = M\left[W(W^T\mathbf{x} - \mathbf{y}) + \gamma_\alpha \operatorname{diag}(\tilde{\boldsymbol{w}})(\mathbf{x} - \mathbf{c})\right]$. Similarly, $\left({}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{1+\alpha} f\right)_j = \operatorname{sign}(x_j - c_j)\, {}_{c_j}^{C}D_{x_j}^{\alpha} I\,(\sum_{i=1}^m w_{ij}^2)$, which gives ${}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{1+\alpha} f = M\operatorname{diag}(\operatorname{sign}(x_j - c_j))\tilde{\boldsymbol{w}}$. We have

$$\begin{aligned}
&{}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{\alpha} f + \operatorname{diag}(\beta|x_j - c_j|)\, {}_{\mathbf{c}}^{C}\nabla_{\mathbf{x}}^{1+\alpha} f \\
&= M\left[W(W^T\mathbf{x} - \mathbf{y}) + (\beta + \gamma_\alpha)\operatorname{diag}(\tilde{\boldsymbol{w}})(\mathbf{x} - \mathbf{c})\right].
\end{aligned}$$

Let $\gamma_{\alpha,\beta} = \beta + \gamma_\alpha$. We then have

$${}_{\boldsymbol{c}}\mathbf{D}_{\boldsymbol{\beta}}^{\boldsymbol{\alpha}} f(\mathbf{x}) = \frac{1}{1+|\beta|} WW^T\left[(I + \gamma_{\alpha,\beta}K)\mathbf{x} - (\mathbf{x}^* + \gamma_{\alpha,\beta}K\mathbf{c})\right]$$

where $K = (WW^T)^{-1}\operatorname{diag}(\tilde{\boldsymbol{w}})$. Let $\mathbf{x}_{\alpha,\beta}^* = (I + \gamma_{\alpha,\beta}K)^{-1}(\mathbf{x}^* + \gamma_{\alpha,\beta}K\mathbf{c})$. It then can be checked that ${}_{\boldsymbol{c}}\mathbf{D}_{\boldsymbol{\beta}}^{\boldsymbol{\alpha}} f(\mathbf{x}) = \frac{1}{1+|\beta|}(WW^T + \gamma_{\alpha,\beta}\operatorname{diag}(\tilde{\boldsymbol{w}}))(\mathbf{x} - \mathbf{x}_{\alpha,\beta}^*)$. Let $A_{\alpha,\beta} = WW^T + \gamma_{\alpha,\beta}\operatorname{diag}(\tilde{\boldsymbol{w}})$. Note that $[A_{\alpha,\beta}]_{ij} = (\beta + \frac{1}{2-\alpha})\sum_{k=1}^m w_{ki}^2$ if $i = j$ and $\sum_{k=1}^m w_{ki}w_{kj}$ otherwise. It follows from the Caputo fractional gradient descent that

$$\mathbf{x}^{(k+1)} - \mathbf{x}_{\alpha,\beta}^* = \left(I - \frac{\eta_k}{1+|\beta|}A_{\alpha,\beta}\right)(\mathbf{x}^{(k)} - \mathbf{x}_{\alpha,\beta}^*).$$

Since α and β are chosen to make $A_{\alpha,\beta}$ positive definite, let $\sigma_{\max}$ be the largest singular values of $A_{\alpha,\beta}$. Let κ be the condition number of $A_{\alpha,\beta}$. Suppose $\eta_k = \frac{1+|\beta|}{\sigma_{\max}}\eta$ for some $\eta \in (0, 2)$. Then, we have $\|\mathbf{x}^{(k)} - \mathbf{x}_{\alpha,\beta}^*\|^2 \leq \|\mathbf{x}^{(0)} - \mathbf{x}_{\alpha,\beta}^*\|^2\left|1 - \frac{\eta}{\kappa}\right|^k$. By observing that

$$\begin{aligned}
\mathbf{x}_{\alpha,\beta}^* &= \mathbf{c} + (I + \gamma_{\alpha,\beta}K)^{-1}(\mathbf{x}^* - \mathbf{c}) \\
&= \mathbf{c} + (WW^T + \gamma_{\alpha,\beta}\operatorname{diag}(\tilde{\boldsymbol{w}}))^{-1}WW^T(\mathbf{x}^* - \mathbf{c}) \\
&= \mathbf{c} + (WW^T + \gamma_{\alpha,\beta}\operatorname{diag}(\tilde{\boldsymbol{w}}))^{-1}W(\mathbf{y} - W^T\mathbf{c}) = \mathbf{x}_{\text{Tik}}^*,
\end{aligned}$$

the proof is completed. $\square$

Appendix E. Proof of Theorem 3.3

Proof. For a positive integer L, we observe that

$$(1 + |\beta|)_{\mathbf{x}^{(k-L)}}\mathbf{D}_{\boldsymbol{\beta}}^{\boldsymbol{\alpha}} f(\mathbf{x}^{(k)}) = A\mathbf{x}^{(k)} + b + \gamma_{\alpha,\beta}\operatorname{diag}(A)(\mathbf{x}^{(k)} - \mathbf{x}^{(k-L)}),$$

where $\gamma_{\alpha,\beta} = \beta - \frac{1-\alpha}{2-\alpha}$ and $\operatorname{diag}(A)$ is the diagonal matrix whose diagonal entries are from A. Since $\mathbf{x}^* = -A^{-1}b$, we have

$$\begin{aligned}
\mathbf{x}^{(k+1)} - \mathbf{x}^* = &\left[I - \tilde{\eta}(A + \gamma_{\alpha,\beta}\operatorname{diag}(A))\right](\mathbf{x}^{(k)} - \mathbf{x}^*) \\
&+ \tilde{\eta}\gamma_{\alpha,\beta}\operatorname{diag}(A)(\mathbf{x}^{(k-L)} - \mathbf{x}^*),
\end{aligned}$$

where $\tilde{\eta} = \frac{\eta}{1+|\beta|}$. Let $\mathcal{A} = I - \tilde{\eta}A - \mathcal{B}$, $\mathcal{B} = \tilde{\eta}\gamma_{\alpha,\beta}\operatorname{diag}(A)$ and $\Delta^k = \mathbf{x}^{(k)} - \mathbf{x}^*$. Then, the above can be written as $\mathcal{E}_{k+1} = M\mathcal{E}_k$ where

$$\mathcal{E}_k = \left[\Delta^k \quad \cdots \quad \Delta^{k-L}\right]^T, \quad M = \begin{bmatrix} \vec{\mathcal{A}} & \mathcal{B} \\ I_{Ld} & 0 \end{bmatrix},$$

$$\vec{\mathcal{A}} = \left[\mathcal{A} \quad 0_{d\times(L-1)d}\right].$$

Here $0_{d\times(L-1)d}$ is the zero matrix of size $d \times (L-1)d$ and I_{Ld} is the identity matrix of size Ld. We then obtain

$$\Delta^k = \left[I \quad 0_{d\times Ld}\right]\mathcal{E}_k = \left[I \quad 0_{d\times Ld}\right]M^k\mathcal{E}_0. \tag{E.1}$$

Let $M^k = \left[\mathcal{C}_k^0 \quad \mathcal{C}_k^1 \quad \cdots \quad \mathcal{C}_k^L\right]$ where $\mathcal{C}_k^j \in \mathbb{R}^{(L+1)d\times d}$. It then can be checked that

$$\begin{aligned}
&\left[\mathcal{C}_{k+1}^0 \quad \mathcal{C}_{k+1}^1 \quad \cdots \quad \mathcal{C}_{k+1}^{L-1} \quad \mathcal{C}_{k+1}^L\right] \\
&\quad = \left[\mathcal{C}_k^0\mathcal{A} + \mathcal{C}_k^1 \quad \mathcal{C}_k^2 \quad \cdots \quad \mathcal{C}_k^L \quad \mathcal{C}_k^0\mathcal{B}\right],
\end{aligned}$$

which gives the following recurrent relations: Let the first d rows of M^k be

$$\left[I \quad 0_{d\times Ld}\right]M^k = \left[\mathcal{A}_k^0 \quad \mathcal{A}_k^1 \quad \cdots \quad \mathcal{A}_k^L\right].$$

Then, starting with $\mathcal{A}_1^0 = \mathcal{A}$, $\mathcal{A}_1^L = \mathcal{B}$ and $\mathcal{A}_1^j = 0$ for $1 \leq j < L$, we have,

$$\begin{aligned}
&\mathcal{A}_k^0 = \mathcal{A}_{k-1}^0\mathcal{A}_1^0 + \mathcal{A}_{k-1}^1, \qquad \mathcal{A}_k^j = \mathcal{A}_{k-1}^{j+1}, \quad \forall 1 \leq j < L, \\
&\mathcal{A}_k^L = \mathcal{A}_{k-1}^0\mathcal{A}_1^L,
\end{aligned}$$

for $k = 2, \ldots$. It then follows from (E.1) that $\|\Delta^k\| \leq \sum_{j=0}^L \|\mathcal{A}_k^j\Delta^{-j}\|$.

Since $\mathcal{A}_{k+s,L-s+1} = \mathcal{A}_{k+1,L} = \mathcal{A}_{k,0}\mathcal{A}_{1,L}$ for $s = 1, \ldots, L$, if $\|\mathcal{A}_{k,0}\|$ converges to 0 as $k \to \infty$, `CfGD-v2` converges to the optimal solution to (12) and the proof is completed. $\square$

Appendix F. Proof of Theorem 3.4

Proof. Let $\mathbf{x}^* = -A^{-1}b$. Since A is diagonal and $\gamma = -1$, we have

$$(1+|\beta|)_{\mathbf{x}^{(k-1)}}\mathbf{D}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}} f(\mathbf{x}^{(k)}) := \vec{\mathbf{d}}_k = A\mathbf{x}^{(k)} + b - A(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) = A\Delta_{k-1},$$

where $\Delta_k = \mathbf{x}^{(k)} - \mathbf{x}^* = (\epsilon_1^{(k)}, \epsilon_2^{(k)})^\top$. Assuming $\Delta_{k-1} \neq 0$, the optimal learning rate (15) is

$$\eta_k^* = \frac{(A\mathbf{x}^{(k)} + b)^\top \vec{\mathbf{d}}_k}{\vec{\mathbf{d}}_k^\top A \vec{\mathbf{d}}_k} = \frac{(A\Delta_k)^\top (A\Delta_{k-1})}{(A\Delta_{k-1})^\top A (A\Delta_{k-1})} = \frac{(\Delta_{k-1})^\top A^2 \Delta_k}{(\Delta_{k-1})^\top A^3 \Delta_{k-1}}.$$

Hence, we have

$$\begin{aligned}\Delta_{k+1} &= \Delta_k - \eta_k^* \vec{\mathbf{d}}_k = \Delta_k - \eta_k^* A\Delta_{k-1} \\ &= \left(A^{-1} - A\Delta_{k-1}\frac{(\Delta_{k-1})^\top A}{(\Delta_{k-1})^\top A^3 \Delta_{k-1}}\right) A\Delta_k = \mathbf{M}_k A\Delta_k,\end{aligned}$$

where $\mathbf{M}_k = A^{-1} - A\Delta_{k-1}\frac{(\Delta_{k-1})^\top A}{(\Delta_{k-1})^\top A^3 \Delta_{k-1}}$. Let $A = \mathrm{diag}(a_j)$. It can be checked that the eigenvalue decomposition of $\mathbf{M}_k$ is $\mathbf{M}_k = V_k \Lambda_k V_k^\top$, where

$$\Lambda_k = \begin{bmatrix} \lambda(\Delta_{k-1}) & 0 \\ 0 & 0 \end{bmatrix},$$

$$V_k = \frac{1}{\sqrt{a_1^4(\epsilon_1^{(k-1)})^2 + a_2^4(\epsilon_2^{(k-1)})^2}} \begin{bmatrix} -a_2^2\epsilon_2^{(k-1)} & a_1^2\epsilon_1^{(k-1)} \\ a_1^2\epsilon_1^{(k-1)} & a_2^2\epsilon_2^{(k-1)} \end{bmatrix},$$

and $\lambda(x,y) = \frac{a_1^4x^2 + a_2^4y^2}{a_1a_2(a_1^3x^2 + a_2^3y^2)}$. Let $\lambda_k := \lambda(\Delta_{k-1})$ and let $v_j^{(k)}$ be the jth column of V_k. Note that

$$v_1^{(k)} = \frac{1}{\sqrt{a_1^4(\epsilon_1^{(k-1)})^2 + a_2^4(\epsilon_2^{(k-1)})^2}} PA^2\Delta_{k-1}.$$

Also, it follows from $APA^2 = a_1a_2PA$ and $(PA)(PA) = a_1a_2I$ that $(PA^2)(PA^2) = PA(APA^2) = (a_1a_2)PA(PA) = (a_1a_2)^2I$ where I is the identity matrix. Hence, $(PA^2)^{2k} = (a_1a_2)^{2k}I$. We then obtain

$$\begin{aligned}\Delta_{k+1} &= \mathbf{M}_k A\Delta_k = \lambda_k \langle v_1^{(k)}, A\Delta_k\rangle v_1^{(k)} \\ &= \frac{\langle PA^2\Delta_{k-1}, A\Delta_k\rangle}{a_1a_2(a_1^3(\epsilon_1^{(k-1)})^2 + a_2^3(\epsilon_2^{(k-1)})^2)} PA^2\Delta_{k-1} \\ &= \frac{\langle PA\Delta_{k-1}, \Delta_k\rangle}{a_1^3(\epsilon_1^{(k-1)})^2 + a_2^3(\epsilon_2^{(k-1)})^2} PA^2\Delta_{k-1},\end{aligned}$$

which implies that given Δ_{-1} and Δ_0,

$$\Delta_k \in \mathrm{span}\{\xi_k\}, \quad \xi_k = \begin{cases} PA^2\Delta_{-1} & \text{if } k \equiv 1 \pmod 4 \\ PA^2\Delta_0 & \text{if } k \equiv 2 \pmod 4 \\ \Delta_{-1} & \text{if } k \equiv 3 \pmod 4 \\ \Delta_0 & \text{if } k \equiv 0 \pmod 4. \end{cases}$$

Also, note that

$$PA\Delta_{k-1} \in \mathrm{span}\{\xi_{k-1}\}, \quad \xi_{k-1} = \begin{cases} PA\Delta_0 & \text{if } k \equiv 1 \pmod 4 \\ A\Delta_{-1} & \text{if } k \equiv 2 \pmod 4 \\ A\Delta_0 & \text{if } k \equiv 3 \pmod 4 \\ PA\Delta_{-1} & \text{if } k \equiv 0 \pmod 4. \end{cases}$$

By combining the above, we have

$$\langle PA\Delta_{k-1}, \Delta_k\rangle \propto \begin{cases} \langle \Delta_{-1}, A^3\Delta_0\rangle & \text{if } k \equiv 1 \pmod 4 \\ \langle \Delta_{-1}, PA\Delta_0\rangle & \text{if } k \equiv 2 \pmod 4 \\ \langle \Delta_{-1}, A\Delta_0\rangle & \text{if } k \equiv 3 \pmod 4 \\ \langle \Delta_{-1}, AP\Delta_0\rangle & \text{if } k \equiv 0 \pmod 4. \end{cases}$$

Since $\langle PA\Delta_{k-1}, \Delta_k\rangle = 0$ implies $\Delta_{k+1} = 0$, the proof is completed. □

Appendix G. Proof of Theorem 3.5

Proof. Let $\mathbf{x}_s^*$ be the solution to the Tikhonov regularization (16) with $\gamma_s = \beta_s - \frac{1-\alpha_s}{2-\alpha_s}$ and $\bar{\mathbf{x}} = \mathbf{c}_s$. Note that since $\lim_{s\to\infty}\gamma_s = 0$, $\lim_{s\to\infty}\mathbf{x}_s^* = \mathbf{x}^*$. Let $\tilde{A}_s := \tilde{A}_{\tilde{\alpha}_s,\tilde{\beta}_s}$ be the matrix defined in Theorem 3.2, and κ_s be its condition number. It follows from Theorem 3.2 that for $s \geq 1$,

$$\|\mathbf{x}_s^{(k_s)} - \mathbf{x}_s^*\|^2 \leq (1 - \eta/\kappa_s)^{k_s}\|\mathbf{x}_s^{(0)} - \mathbf{x}_s^*\|^2.$$

Let $\mathcal{E}_s = \|\mathbf{x}_s^{(0)} - \mathbf{x}_s^*\|$, $e_s = \|\mathbf{x}_s^* - \mathbf{x}_{s+1}^*\|$ and $r_s = (1 - \frac{\eta}{\kappa_s})^{k_s/2}$ for $s \geq 1$. Since $\mathbf{x}_{s-1}^{(k_{s-1})} = \mathbf{x}_s^{(0)}$ for all s, we have $\mathcal{E}_s \leq r_{s-1}\mathcal{E}_{s-1} + e_{s-1}$. By recursively applying it, we have $\mathcal{E}_s \leq \sum_{k=1}^s \left(\prod_{j=1}^{k-1} r_{s-j}\right) e_{s-k}$ where we set $e_0 := \mathcal{E}_1$ for notational convenience. It follows from $\|\mathbf{x}_s^{(k_s)} - \mathbf{x}^*\| \leq r_s\mathcal{E}_s + \|\mathbf{x}_s^* - \mathbf{x}^*\|$ that

$$\|\mathbf{x}_s^{(k_s)} - \mathbf{x}^*\| \leq \sum_{k=1}^s \left(\prod_{j=0}^{k-1} r_{s-j}\right) e_{s-k} + \|\mathbf{x}_s^* - \mathbf{x}^*\|.$$

Let $M_\gamma = (WW^\top + \gamma RR^\top)^{-1}$ and let $\mathbf{x}^*(\gamma;\mathbf{c})$ be the solution to the Tikhonov regularization (16) with γ and $\bar{\mathbf{x}} = \mathbf{c}$. Observe that for any $\gamma, \gamma' \in (B_\ell - 0.5, B_u)$ and $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^d$,

$$\begin{aligned}&\mathbf{x}^*(\gamma;\mathbf{c}) - \mathbf{x}^*(\gamma';\mathbf{c}') \\ &= \mathbf{c} - \mathbf{c}' + M_\gamma W(y - W^\top\mathbf{c}) - M_{\gamma'}W(y - W^\top\mathbf{c}') \\ &= (M_\gamma - M_{\gamma'})W(y - W^\top\mathbf{c}) + (I - M_{\gamma'}WW^\top)(\mathbf{c}' - \mathbf{c}) \\ &= (\gamma' - \gamma)M_\gamma RR^\top M_{\gamma'}WW^\top(\mathbf{x}^* - \mathbf{c}) + (M_0 - M_{\gamma'})WW^\top(\mathbf{c}' - \mathbf{c}) \\ &= (\gamma' - \gamma)M_\gamma RR^\top M_{\gamma'}WW^\top(\mathbf{x}^* - \mathbf{c}) \\ &\quad + \gamma' M_0 RR^\top M_{\gamma'}WW^\top(\mathbf{c}' - \mathbf{c}).\end{aligned}$$

Assuming $M_{\max} = \sup_{\gamma\in(B_\ell - 1/2, B_u)}\|M_\gamma\|$ is finite, we obtain

$$\|\mathbf{x}^*(\gamma;\mathbf{c}) - \mathbf{x}^*(\gamma';\mathbf{c}')\| \leq C\Big(|\gamma - \gamma'|\|\mathbf{x}^* - \mathbf{c}\| + |\gamma'|\|\mathbf{c}' - \mathbf{c}\|\Big),$$

where $C = M_{\max}^2\|W\|^2\|R\|^2$. By letting $\gamma' \to 0$, we have $\|\mathbf{x}_s^* - \mathbf{x}^*\| \leq C|\gamma_s|\|\mathbf{c}_s - \mathbf{x}^*\|$. Also, $e_s \leq C(|\gamma_s - \gamma_{s+1}|\|\mathbf{x}^* - \mathbf{c}_{s+1}\| + |\gamma_s|\|\mathbf{c}_{s+1} - \mathbf{c}_s\|)$, which gives

$$\begin{aligned}\|\mathbf{x}_s^{(k_s)} - \mathbf{x}^*\| &\leq \left(\prod_{j=0}^{s-1} r_{s-j}\right)\|\mathbf{x}^{(0)} - \mathbf{x}_1^*\| + C|\gamma_s|\|\mathbf{c}_s - \mathbf{x}^*\| \\ &+ C\sum_{k=1}^{s-1}\left(\prod_{j=0}^{k-1} r_{s-j}\right) \\ &\times \Big[|\gamma_{s-k} - \gamma_{s-k+1}|\|\mathbf{c}_{s-k+1} - \mathbf{x}^*\| + |\gamma_{s-k}|\|\mathbf{c}_{s-k+1} - \mathbf{c}_{s-k}\|\Big],\end{aligned}$$

and the proof is completed. □

Appendix H. Complexity calculations

Firstly, we note that given $\mathbf{x}$, the evaluation of the difference between the network prediction and the output data (misfit) costs $\mathcal{O}(mn)$ FLOPS as

$$\mathrm{MISFIT}_{\mathbf{x}} := W^\top \boldsymbol{a}_3 - \boldsymbol{y},$$

where $(W)_{ji} = \phi(a_{1,j}z_i + a_{2,j})$, $(\boldsymbol{y})_i = y_i$, and $(\boldsymbol{a}_3)_j = a_{3,j}$. For $j = 1, \dots, n$, let

$$\begin{aligned}
\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) &= \sum_{l=1,l\neq j}^{n} a_{3,l}\phi(a_{1,l}z_i + a_{2,l}) - y_i \\
&\quad + a_{3,j}\phi(uz_i + a_{2,j}), \ \text{if } t = j, \\
\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) &= \sum_{l=1,l\neq j}^{n} a_{3,l}\phi(a_{1,l}z_i + a_{2,l}) - y_i \\
&\quad + a_{3,j}\phi(a_{1,j}z_i + u), \ \text{if } t = n + j, \\
\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) &= \sum_{l=1,l\neq j}^{n} a_{3,l}\phi(a_{1,l}z_i + a_{2,l}) - y_i \\
&\quad + u\phi(a_{1,j}z_i + a_{2,j}), \ \text{if } t = 2n + j.
\end{aligned}$$

Observe that for $j = 1, \dots, n$,

$$\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) = \begin{cases} (\mathrm{MISFIT}_{\mathbf{x}})_i \\ \quad + a_{3,j}(\phi(uz_i + a_{2,j}) - \phi(a_{1,j}z_i + a_{2,j})) & \text{if } t = j, \\ (\mathrm{MISFIT}_{\mathbf{x}})_i \\ \quad + a_{3,j}(\phi(a_{1,j}z_i + u) - \phi(a_{1,j}z_i + a_{2,j})) & \text{if } t = n + j, \\ (\mathrm{MISFIT}_{\mathbf{x}})_i \\ \quad + (u - a_{3,j})\phi(a_{1,j}z_i + a_{2,j}) & \text{if } t = 2n + j. \end{cases}$$

This shows that if $\mathrm{MISFIT}_{\mathbf{x}}$ and W are already computed and stored, the function $\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u)$ can be evaluated with almost no computational cost. Assuming evaluation of ϕ takes 1 FLOPS, a single evaluation of $\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u)$ costs at most 6 FLOPS. With the function $\mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u)$, $f'_{j,\mathbf{x}}$ and $f''_{j,\mathbf{x}}$ are given as follows: For $t = j$ where $j = 1, \dots, n$,

$$\begin{aligned}
f'_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} \mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) a_{3,j}\phi'(uz_i + a_{2,j})z_i, \\
f''_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} \Big\{ \mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) a_{3,j}\phi''(uz_i + a_{2,j})z_i^2 \\
&\qquad + (a_{3,j}\phi'(uz_i + a_{2,j})z_i)^2 \Big\}.
\end{aligned}$$

For $t = n + j$ where $j = 1, \dots, n$,

$$\begin{aligned}
f'_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} \mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) a_{3,j}\phi'(a_{1,j}z_i + u), \\
f''_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} \Big\{ \mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) a_{3,j}\phi''(a_{1,j}z_i + u) \\
&\qquad + (a_{3,j}\phi'(a_{1,j}z_i + u))^2 \Big\}.
\end{aligned}$$

For $t = 2n + j$ where $j = 1, \dots, n$,

$$\begin{aligned}
f'_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} \mathrm{MISFIT}^{[i]}_{t,\mathbf{x}}(u) \phi(a_{1,j}z_i + a_{2,j}), \\
f''_{t,\mathbf{x}}(u) &= \sum_{i=1}^{m} (\phi(a_{1,j}z_i + a_{2,j}))^2.
\end{aligned}$$

Assuming the evaluations of ϕ, ϕ', ϕ'' take 1 FLOPS each, it can be checked that a single evaluation of $f'_{t,\mathbf{x}}(u)/f''_{t,\mathbf{x}}(u)$ takes at most 13 m/20 m FLOPS.

References

Bonnans, J.-F., Gilbert, J., Lemarechal, C., & Sagastizabal, C. (2006). *Numerical optimization*. Springer Berlin Heidelberg.

Caputo, M. (1967). Linear models of dissipation whose q is almost frequency independent—II. *Geophysical Journal International*, *13*(5), 529–539.

Chen, Y., Gao, Q., Wei, Y., & Wang, Y. (2017). Study on fractional order gradient methods. *Applied Mathematics and Computation*, *314*, 310–321.

Cheng, S., Wei, Y., Chen, Y., Li, Y., & Wang, Y. (2017). An innovative fractional order LMS based on variable initial value and gradient order. *Signal Processing*, *133*, 260–269.

D'Elia, M., Du, Q., Glusa, C., Gunzburger, M., Tian, X., & Zhou, Z. (2020). Numerical methods for nonlocal and fractional models. *Acta Numerica*, *29*, 1–124.

D'Elia, M., Gulian, M., Olson, H., & Karniadakis, G. E. (2021). Towards a unified theory of fractional and nonlocal vector calculus. *Fractional Calculus & Applied Analysis*, *24*(5), 1301–1355.

Golub, G. H., Hansen, P. C., & O'Leary, D. P. (1999). Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, *21*(1), 185–194.

Hai, P. V., & Rosenfeld, J. A. (2020). The gradient descent method from the perspective of fractional calculus. *Mathematical Methods in the Applied Sciences*.

Khan, S., Naseem, I., Malik, M. A., Togneri, R., & Bennamoun, M. (2018). A fractional gradient descent-based RBF neural network. *Circuits, Systems, and Signal Processing*, *37*(12), 5311–5332.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lemaréchal, C. (2012). Cauchy and the gradient method. *Documenta Mathematica Extra*, 251–254.

Liang, S., Wang, L., & Yin, G. (2020). Fractional differential equation approach for convex optimization with convergence rate analysis. *Optimization Letters*, *14*(1), 145–155.

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, *3*, 218–229.

Mengesha, T., & Spector, D. (2015). Localization of nonlocal gradients in various topologies. *Calculus of Variations and Partial Differential Equations*, *52*(1), 253–279.

Nagaraj, S. (2020). Optimization and learning with nonlocal calculus. arXiv preprint arXiv:2012.07013.

Nesterov, Y. (2004). *Applied Optimization*: *vol. 87*, *Introductory lectures on convex optimization* (p. xviii+236). Kluwer Academic Publishers, Boston, MA, A basic course.

Nocedal, S., & Wright, J. (2006). *Numerical optimization*. Springer New York.

Pu, Y.-F., Zhou, J.-L., Zhang, Y., Zhang, N., Huang, G., & Siarry, P. (2013). Fractional extreme value adaptive training method: fractional steepest descent approach. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(4), 653–662.

Ralston, A., & Rabinowitz, P. (2001). *A first course in numerical analysis*. Courier Corporation.

Rodriguez-Lujan, I., Fonollosa, J., Vergara, A., Homer, M., & Huerta, R. (2014). On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemometrics and Intelligent Laboratory Systems*, *130*, 123–134.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Sheng, D., Wei, Y., Chen, Y., & Wang, Y. (2020). Convolutional neural networks with fractional order gradient method. *Neurocomputing*, *408*, 42–50.

Tarasov, V. E. (2008). Fractional vector calculus and fractional Maxwell's equations. *Annals of Physics*, *323*(11), 2756–2778.

Vergara, A., Vembu, S., Ayhan, T., Ryan, M., Homer, M., & Huerta, R. (2012). Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B (Chemical)*, *166*, 320–329.

Wang, J., Wen, Y., Gou, Y., Ye, Z., & Chen, H. (2017). Fractional-order gradient descent learning of BP neural networks with Caputo derivative. *Neural Networks*, *89*, 19–30.

Wei, Y., Kang, Y., Yin, W., & Wang, Y. (2020). Generalization of the gradient method with fractional order gradient direction. *Journal of the Franklin Institute*, *357*(4), 2514–2532.