Design of generalized fractional order gradient descent method

Yiheng Wei, Yu Kang, Weidi Yin, and Yong Wang

Abstract—This paper focuses on the convergence problem of the emerging fractional order gradient descent method, and proposes three solutions to overcome the problem. In fact, the general fractional gradient method cannot converge to the real extreme point of the target function, which critically hampers the application of this method. Because of the long memory characteristics of fractional derivative, fixed memory principle is a prior choice. Apart from the truncation of memory length, two new methods are developed to reach the convergence. The one is the truncation of the infinite series, and the other is the modification of the constant fractional order. Finally, six illustrative examples are performed to illustrate the effectiveness and practicability of proposed methods.

Index Terms—Fractional gradient, convergence design, truncation, fixed memory step, variable fractional order.

I. INTRODUCTION

GRADIENT descent method is prevalently used in research fields, such as optimization [1], [2], machine learning [3], [4] and image denoising [5], [6]. Through plenty of studies and experiments, it is known that gradient method is one of the most effective and efficient way to find the optimal solution of optimization problems. Nowadays, one of the key point of gradient method is how to improve the performance further [7]. As an important branch of mathematics, fractional calculus is believed to be a good tool to improve the traditional gradient descent method, mainly because of its special long memory characteristics and nonlocality [8], [9].

Some remarkable progress in studies of fractional gradient method not only reveals some interesting properties, but also gives practical suggestions for future research. In [10], the authors proposed a fractional gradient method by using Caputo nabla difference with an order no more than 1 as the iterative order, instead of first order difference. Though this method could ensure the convergence, the convergence speed becomes slow. Interesting, a similar idea can be found in [11], where Riemann–Liouville fractional derivative was initially used as a substitute for the first order gradient. With the adoption of fractional calculus, this newly developed method manifests distinct properties. For example, its iterative search process can easily pass over the local extreme points. However, one cannot guarantee that the extreme point can be found using the method in [11] even if the algorithm is indeed convergent. Additionally, it is difficult to calculate the needed fractional derivative online.

This shortcoming has been partially overcome in [12]. Despite some minor errors with the calculation procedure, the developed method has been successfully applied in speech enhancement [12] and channel equalization [13]. After that, by approximating the exact fractional derivative, a new fractional gradient method was proposed by us [14]. Hereinafter, this method was used in system identification [15]. Research indicated that introduction of fractional gradient method enhanced the performance of classical algorithm. Additionally, the promising fractional gradient method has been widely used in many applications, such as least mean square algorithm [16], back propagation neural networks [17], recommender systems [18], etc.

It is worth pointing out that fractional gradient method has been used successfully, but the related research is still in its infancy and deserves further investigation. The immediate problem is the convergence problem. On one hand, [11] has found that the fractional extreme value is not equal to the real extreme value of target function, which would make fractional gradient method loss practicability. However, the main reason for nonconvergence is unclear. The effective solutions for realizing convergence are still desirable. On the other hand, if the update equation is not searching at the right direction to the real extreme point of the cost function, the convergence speed could not be fast enough. Therefore, for a considerable performance, only convergence can not meet the requirements, and convergence speed should also be taken into account.

Although there are many problems and work to be completed with this issue, we have reasons to believe that these subsequent studies can break a new ground in the future. Inspired by the discussions above, the objective of this paper are: i) investigating the extreme point and value of fractional gradient method; ii) designing the solutions to solve the convergence problem; and iii) the convergence speed should be considered and improved.

The remainder of the paper is organized as follows. Section II is devoted to math preparation and problem formulation. Solutions to the convergence problem of fractional gradient method are introduced in Section III. Section IV shows some numerical examples to verify the proposed methods. At last, conclusion is presented in Section V.

II. PRELIMINARIES

This section presents a brief introduction to the mathematical background of fractional calculus and the fatal flaw of fractional order gradient descent method.
A. Fractional calculus

The commonly used definitions of fractional calculus are Grünwald–Letnikov, Riemann–Liouville and Caputo [19]. Because the first two definitions are identical under certain conditions and Grünwald–Letnikov definition is usually used in numerical calculation, only Riemann–Liouville and Caputo definitions are considered in this study.

The Riemann-Liouville derivative and Caputo derivative of order $\alpha$, are expressed as

$$\frac{\text{RL}}{c} \mathcal{D}_x^\alpha f(x) = \frac{1}{\Gamma(n - \alpha)} \int_c^x \frac{f^{(n)}(r)}{(x - r)^{\alpha + n - 1}} \, dr, \quad (1)$$

$$\frac{\text{C}}{c} \mathcal{D}_x^\alpha f(x) = \frac{1}{\Gamma(n - \alpha)} \int_c^x \frac{f^{(n)}(r)}{(x - r)^{\alpha + n - 1}} \, dr, \quad (2)$$

respectively, where $n - 1 < \alpha < n$, $n \in \mathbb{Z}_+$, $c$ is the lower terminal, $\Gamma(\alpha) = \int_0^{+\infty} e^{-t} t^{\alpha-1} \, dt$ is the Gamma function. Notably, the fractional derivatives in (1) and (2) are actually special nonlocality, which will play a pivotal role in fractional optimization algorithm for finding the minimum of a cost function $f(x)$. For the well-known gradient descent method is a first-order iterative optimization algorithm for finding the minimum of a function. To this end, one typically takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. For example, $x_k$ is updated by the following law

$$x_{k+1} = x_k - \mu \nabla f(x_k), \quad (5)$$

where $x_k$ is the current position, $x_{k+1}$ is the next position, $\mu$ is the learning rate and $\nabla f(x_k)$ is the first-order gradient at $x = x_k$. When the classical gradient is replaced by the fractional one, it follows that

$$x_{k+1} = x_k - \mu \nabla_\alpha f(x_k). \quad (6)$$

For $f(x) = (x-5)^2$, its exact extreme point is $x^* = 5$. Its first-order derivative is $f^{(1)}(x) = 2x - 10$, while for $\alpha \in (0, 1)$, its fractional order derivative satisfies

$$\frac{\text{RL}}{c} \mathcal{D}_x^\alpha f(x) = \frac{2}{\Gamma(3-\alpha)} (x-c)^{2-\alpha} + \frac{2(\alpha-5)}{\Gamma(2-\alpha)} (x-c)^{1-\alpha} + \frac{(\alpha-5)^2}{\Gamma(1-\alpha)} (x-c)^{-\alpha}, \quad (7)$$

$$\frac{\text{C}}{c} \mathcal{D}_x^\alpha f(x) = \frac{2}{\Gamma(3-\alpha)} (x-c)^{2-\alpha} + \frac{2(\alpha-5)}{\Gamma(2-\alpha)} (x-c)^{1-\alpha}. \quad (8)$$

Consider the following three cases

1. case 1: using (6) with first-order derivative;
2. case 2: using (6) with Riemann–Liouville derivative;
3. case 3: using (6) with Caputo derivative,

and set $c = 0$, $x_0 = 1$, $\mu = 0.5$ and $\alpha = 0.7$. On this basis, the simulation results are shown in Fig. 1.

![Fig. 1: Search process of gradient descent method.](image)

It is clearly observed that all the three cases can realize the convergence within 20 steps, while only case 1 is able to converge to the exact extreme point. If the algorithm is convergent for case 2, the convergent point satisfies $\frac{\text{RL}}{c} \mathcal{D}_x^\alpha f(x) = 0$. With the exception of $x = c$, one obtains

$$x = c - (c - 5) \frac{(2-\alpha) \pm \sqrt{(2-\alpha)^2 - 2(2-\alpha)(1-\alpha)}}{2} \approx 3.25 \pm 2.5\sqrt{0.91} \approx 5.6348 \text{ or } 0.8652, \quad (9)$$

which matches the green dot dash line well.

Similarly, when case 3 is convergent, the final value can be calculated by $\frac{\text{C}}{c} \mathcal{D}_x^\alpha f(x) = 0$, which leads to

$$x = c - (c - 5) (2 - \alpha) = 6.5. \quad (10)$$

This just coincides with the blue dotted line.

To sum up, fractional extreme points are different from the actual extreme point, since the fractional derivative of $f(x)$ depends on the variable $x$ and is also connected with the order $\alpha$ and the initial instant $c$. It is not easy to ensure $\frac{\text{RL}}{c} \mathcal{D}_x^\alpha f(x) = 0$ and $\frac{\text{C}}{c} \mathcal{D}_x^\alpha f(x) = 0$ at the actual extreme point. Without loss of generality, consider $f(x) = a(x - x^*)^2 + f_m$, $a > 0$, $x^* \in \mathbb{R}, f_m \in \mathbb{R}$ and then similar conclusions can be drawn. If $\alpha \in (0, 1)$ and $c \neq x^*$, then case 2 may not converge. Even if it is convergent, it will never converge to $x^*$. Besides, case
3 will never converge to $x^*$ either. This kind of convergence problem inevitably will, to some extent, cause performance deterioration when using fractional gradient method in some practical applications. Therefore, it is necessary to deal with the convergence problem mentioned above.

III. MAIN RESULTS

To solve the convergence problem, three viable solutions are proposed by fully considering the nonlocality of fractional derivative. Additionally, the convergence and implementation are described in detail.

A. Fixed memory step

From the discussion in the previous section, it can be found that the gradient descent method cannot converge to the real extreme point as the order extends to the non-integer case. To substantially study the short memory characteristics of classical derivative and modify the long memory characteristics of fractional derivative, an intuitive idea is to replace the constant initial instant $c$ with the varying initial instant $x_k-K$.

The resulting method follows immediately

$$x_{k+1} = x_k - \mu x_k C \mathcal{D}_x^\alpha f(x)|_{x=x_k},$$

where $K \in \mathbb{Z}_+$. This design is inspired by our previous work in [14], [20]. When the algorithm is convergent, it will converge to its actual extreme point. Now, let us prove that $x_k$ converges to $x^*$ by contradiction. Supposing that $x_k$ converges to a point $X$ different from $x^*$ and $f^{(1)}(X) \neq 0$, then $\lim_{k \to +\infty} x_k = X$. Therefore, for any small positive scalar $\varepsilon$, there exists a sufficient large number $n \in \mathbb{N}$ such that $|x_k - X| < \varepsilon < |x^* - X|$ for any $k > n$.

By combining formulas (3) and (11), one has the following inequality

$$|x_{k+1} - x_k| = \mu |x_k-C \mathcal{D}_x^\alpha f(x)|_{x=x_k} = |\mu| \sum_{i=1}^{+\infty} \frac{\Gamma[(2-\alpha)]}{(i+1-\alpha)(i-1)} \left| f^{(i)}(x_k) \right|^\alpha |

\geq \mu \sigma \sum_{i=0}^{+\infty} |x_k - x_k-K|^i |x_k - x_k-K|^{1-\alpha} = d |x_k - x_k-K|^{1-\alpha},$$

where $\sigma = \sup_{k>0,i \in \mathbb{N}} \left( \frac{(i-1)}{(i+1-\alpha)} \right) |f^{(i+2)}(x_k)|^{1-\alpha}$ and $d = \frac{\mu \sigma}{1-\varepsilon}$.

Actually, one can always find an $\varepsilon$ satisfying $d > (K\varepsilon)^\alpha$.

From the assumption on $|x_k - X| < \varepsilon < |x^* - X|$, it becomes $|x_k - x_k-K| = |x_k - x_k-K + x_k-1 + \cdots - x_k-K| < K\varepsilon$. Hence, $|x_{k+1} - x_k| > d |K\varepsilon|^{-\alpha}$ can be derived, which contradicts to the fact $|x_{k+1} - x_k| < K\varepsilon$. This completes the proof of convergence.

The main idea of this method is called as fixed memory principle. $x_k-C \mathcal{D}_x^\alpha f(x)|_{x=x_k}$ can be calculated with the help of (4). Then the corresponding fractional gradient method can be expressed as

$$x_{k+1} = x_k - \mu \sum_{i=1}^{+\infty} \frac{\Gamma[(2-\alpha)]}{(i+1-\alpha)} f^{(i)}(x_k) |x_k - x_k-K|^{i-\alpha}.$$

B. Higher order truncation

Recalling the classical gradient descent method, it can be obtained that for any positive $\mu$, if it is small enough, one has $f(x_k - \mu) \approx f(x_k) - \mu^2 f^{(1)}(x_k)$. Moreover, it becomes

$$f(x_{k+1}) = f(x_k) - \mu \left[ f^{(1)}(x_k) \right]^2 \leq f(x_k).$$

When the dominant factor $f^{(1)}(x_k) = 0$ emerges, the iteration completes, which confirms the fact that the first-order derivative is equal to 0 at the extreme point. From this point of view, only the relevant term is reserved and the other terms are omitted, resulting a new fractional gradient descent method.

$$x_{k+1} = x_k - \mu f^{(1)}(x_k) |x_k-c|^{1-\alpha}.\tag{15}$$

To avoid the appearance of a complex number, the update law in (15) can be rewritten as

$$x_{k+1} = x_k - \mu f^{(1)}(x_k) |x_k-c|^{1-\alpha}.$$

To prevent the emergence of a denominator of 0, i.e., $x_k = c$, a small nonnegative number $\varepsilon$ is introduced to modify the update law further as

$$x_{k+1} = x_k - \mu f^{(1)}(x_k) |x_k-c+\varepsilon|^{1-\alpha}.\tag{17}$$

If $f(x)$ is a convex function with a unique extreme point and it has a Lipschitz continuous gradient with the Lipschitz constant $\rho$, $0 < \mu < \frac{2\Gamma(2-\alpha)}{\rho^2}$ can guarantee the convergence, where $d = \sup_k |x_k-c| + \varepsilon$.

Assuming that $x_k$ is the exact extreme point, one obtains

$$f^{(1)}(x_k) = 0$$

and then

$$x_{k+1} = x_k - \mu f^{(1)}(x_k) |x_k-c|^{1-\alpha}$$

$$= (x_k - x^*) \left[ 1 - \frac{\mu K}{\Gamma(2-\alpha)} \right]^{1-\alpha},$$

where $f^{(1)}(x_k) = f^{(1)}(x^*) = f^{(1)}(x_k)$. Then the convergence condition of the sequence $x_k$ becomes

$$0 < \kappa |x_k-c|^{1-\alpha} \leq 2.$$

C. Variable fractional order

It is well known that the traditional gradient method could converge to the exact extreme point. For this reason, adjusting the order $\alpha$ with $x$ is an alternative method. If the target function satisfies $f(x) \geq 0$ and $f(x^*) = 0$, one can design the variable fractional order as follows

$$\alpha(x) = \frac{1}{1+3\beta J(x)}; \tag{20}$$

$$\alpha(x) = \frac{2}{1+2\beta J(x)}; \tag{21}$$

$$\alpha(x) = 1 - \tanh(\beta J(x)); \tag{22}$$
where the loss function $J(x) = f(x)$ and the constant $\beta > 0$. Beside, it is noticed that

$$0 < \alpha(x) < 1, \beta \neq 0, x \neq x^*, \quad (23)$$

$$\lim_{\beta \to 0} \alpha(x) = \lim_{x \to x^*} \alpha(x) = 1. \quad (24)$$

At the beginning of learning, $J(x)$ is a relative large value and then $\alpha(x) \ll 1$, which results in a quick learning. Subsequently, $x$ get close to $x^*$ gradually and then $\alpha(x) \approx 1$, which leads to an accurate learning. In the end, $x \to x^*$ is expected.

However, the order $\alpha(x)$ is constructed with the assumption on $f(x)$. If the minimum value of $f(x)$ is nonzero or even negative, the designed orders in (20)-(22) will no longer work. In this case, the function will be redefined

$$J(x) = \left| f^{(1)}(x) \right|^2, \quad (25)$$

and thereby the designed orders are revived. At this point, the corresponding method can be expressed as

$$x_{k+1} = x_k - \mu \sum_{i=1}^{+\infty} \left( \frac{(\alpha(x)-1)}{i-1} \frac{f^{(i)}(x_k)}{\Gamma((i+1-\alpha(x)))} (x_k - c)^{i-\alpha(x)} \right). \quad (26)$$

The description of variable fractional order gradient descent method is given in Algorithm 3.  

**Remark 1:** In this paper, three solutions are developed to solve the non convergence problem of fractional order gradient descent method. The first method is benefit from the long memory characteristics of fractional derivative, and then the fixed memory principle is adopted here. The second method is proposed to weaken the nonlocality of fractional derivative and keep only the first order term. The third method is to change the order with the loss function at each step. This work surely activates the existing method, which will make this method potential, potent and practical.

**Remark 2:** In general, it is difficult or even impossible to obtain the analytic form of fractional derivative for arbitrary functions. With regard to this work, the equivalent representation in formulas (3)-(4) plays a critical role. In particular, to avoid the implementation difficulties brought by the infinite series, the approximate finite sum is alternative. Furthermore, the following formulas (27)-(28) could also be equally adopted in fractional gradient method.

$$\frac{\Gamma(\alpha(x))}{\Gamma((1+1-\alpha(x)))} \frac{f^{(i)}(x_k)}{\Gamma((i+1-\alpha(x)))} (x_k - c)^{i-\alpha(x)}.$$ (27)

$$\frac{\Gamma(\alpha(x))}{\Gamma((1+1-\alpha(x)))} \frac{f^{(i)}(x_k)}{\Gamma((i+1-\alpha(x)))} (x_k - c)^{i-\alpha(x)}.$$ (28)

**Remark 3:** To convey the main contributions of this work clearly, some points are listed here.

i) In Algorithm 1, $0.5 < \alpha < 1.5$ is generally selected, when $(x_k - x_{k-K})^{i-\alpha}$ is used. When it is selectively replaced by $(x_k - x_{k-K})^{i-\alpha}$ or $(x_k - x_{k-K} + \epsilon)^{i-\alpha}$, the range of $\alpha$ can extend to $(0, 2)$.

ii) In Algorithm 2, combining with the fixed memory principle, the leading term in (17), i.e., $(x_k - c + \epsilon)^{i-\alpha}$ could be modified by $(x_k - x_{K})^{i-\alpha}$ and then the range of its application is enlarged to $0 < \alpha < 2$.

iii) In Algorithm 3, $(x_k - c)^{i-\alpha(x)}$ could also find its substitute as $|x_k - c|^{i-\alpha(x)}$ or $(|x_k - c| + \epsilon)^{i-\alpha(x)}$.

iv) Note that (20)-(22) are not the unique forms of the order and any valid forms are suitable here.

v) Only Caputo definition is considered in constructing the solutions, while the Riemann–Liouville case can still be handled similarly.

vi) Although this paper only focuses on the scalar fractional gradient method, the multivariate case can be also established with the similar treatment.

It is intended that in-depth studies in these directions will be undertaken as future course of work.

**Algorithm 1** Fixed memory step gradient descent method.

**Input:** $x_0, x_1, \cdots, x_{K-1}$

**Output:** $x_N$

**Initialization:**

$\alpha, \mu, N, K$: user defined value

for $k = K$ to $N - 1$

$h = \sum_{i=1}^{\infty} \frac{\Gamma((i+1-\alpha(x)))}{\Gamma((i+1-\alpha(x)))} (x_k - x_{k-K})^{i-\alpha}$

$x_{k+1} = x_k - \mu h$

end for

**Algorithm 2** Higher order truncation gradient descent method.

**Input:** $x_0$

**Output:** $x_N$

**Initialization:**

$\alpha, \mu, N, c, \epsilon$: user defined value

for $k = 0$ to $N - 1$

$h = \sum_{i=1}^{\infty} \frac{\Gamma((i+1-\alpha(x)))}{\Gamma((i+1-\alpha(x)))} (x_k - c)^{i-\alpha}$

$x_{k+1} = x_k - \mu h$

end for

**Algorithm 3** Variable fractional order gradient descent method.

**Input:** $x_0$

**Output:** $x_N$

**Initialization:**

$\mu, N, c, \beta$: user defined value

for $k = 1$ to $N - 1$

$h = \sum_{i=1}^{\infty} \frac{\Gamma((i+1-\alpha(x)))}{\Gamma((i+1-\alpha(x)))} (x_k - c)^{i-\alpha}$

$x_{k+1} = x_k - \mu h$

end for

**IV. SIMULATION STUDY**

In this section, several examples are provided to explicitly demonstrate the validity of the proposed solutions. Examples 1-3 aim at testifying the convergence design. Examples 4-5 consider a target function with nonzero minimum value and the notable Rosenbrock function, respectively. Example 6 gives an application according to LMS algorithm.

**Example 1:** Recalling $f(x) = (x - 5)^2$ and setting the fractional order $\alpha = 0.7$, the learning rate $\mu = 0.5$, the initial
point \( x_0 = 1 \), then the results using Algorithm 1 with \( K = 5 \), \( K = 3 \) and \( K = 1 \) are given in Fig. 2. It is clearly seen that for any case, the expected convergence can reach within 20 steps. Additionally, the speed of convergence becomes more rapid as \( K \) decreases.

When the parameter \( K \) is set as 1, the learning rate varies from 0.1 to 0.5 and then the simulation with Algorithm 1 is conducted once again. Fig. 3 indicates that with the increase of the learning rate, the convergence gets faster and the overshoot emerges gradually.

**Example 2:** Let us continue to consider the target function \( f(x) = (x - 5)^2 \). Setting \( c = 0, \mu = 0.2, \epsilon = 0, x_0 = 1 \) and \( \alpha = 0.1, 0.3, \cdots, 1.9 \), Algorithm 2 is adopted to search the minimum value point. It can be clearly observed from Fig. 4 that the proposed method is effective and the convergence will accelerate along with the reduction of the order.

Similarly, provided \( \alpha = 0.7 \) and \( x_0 = 1.0, 1.5, \cdots, 6.0 \), the related simulation is performed and the results are shown in Fig. 5. This picture suggests that the algorithm with different initial points converges simultaneously.

**Example 3:** Look into the target function \( f(x) = (x - 5)^2 \) once again. The following three cases are considered

\[
\begin{align*}
\text{case 1}: & \quad \text{using (20) with } \beta = 0.03; \\
\text{case 2}: & \quad \text{using (21) with } \beta = 0.10; \\
\text{case 3}: & \quad \text{using (22) with } \beta = 0.10,
\end{align*}
\]

and other parameters are set as \( c = 0, \mu = 0.2 \) and \( x_0 = 1 \). The corresponding results are depicted in Fig. 6. It is shown that all the designed orders could achieve the convergence as expected while is still a valuable work to design suitable order for better performance.

In the previous simulation, the initial instant \( c \) is randomly selected and different from \( x_0 \). To test the influence of \( c \), a series of values 0.0, 0.1, \cdots, 0.9 are configured for \( c \) one after another. Besides, the aforementioned case 3 is still applied here and the results are shown in Fig. 7. It illustrates that when \( c \) increases, the search process gets slower, while all of them are convergent as expected.

**Example 4:** Consider the target function proposed in [11], i.e., \( f(x, y) = 2(x - 5)^2 + 3(y - 6)^2 + 10 \). The real extreme point of \( f(x, y) \) is \( x = 5, y = 6 \) and the extreme value is nonzero. Setting \( c = 0, x_0 = 1, y_0 = 1, \mu = 0.05, \epsilon = 0, \cdots, 6.0 \),
\( \alpha = 0.7, K = 3 \) and \( \beta = 0.005 \), then the results using three proposed methods are given in Figs. 8-10.

As revealed in Fig. 8 and Fig. 9, the proposed algorithms converge simultaneously in the \( x \) and \( y \) coordinate directions and the actual extreme points are reached, respectively. To give a more intuitive understanding, the convergence trajectories are displayed in Fig. 10. The iterative search process of \( y \) plays faster than that of \( x \), since the coefficient about \( y \) is larger than that \( x \) under the same condition. Extra simulation indicates that when the learning rates are set separately as \( \mu_x \) and \( \mu_y \) instead of \( \mu \), the convergence on \( x \) will speed up and the overshoot on \( y \) will disappear with properly defined \( \mu_x, \mu_y \).

Notably, the variable fractional orders are designed individually, namely,
\[
\begin{align*}
\alpha(x) &= 1 - \tanh (\beta J_x(x, y)) , \\
\alpha(y) &= 1 - \tanh (\beta J_y(x, y)),
\end{align*}
\]
where \( J_x(x, y) = \left| \frac{\partial}{\partial x} f(x, y) \right|^2 \) and \( J_y(x, y) = \left| \frac{\partial}{\partial y} f(x, y) \right|^2 \).

Actually, the order can also be designed uniformly
\[
\alpha(x, y) = 1 - \tanh (\beta J(x, y)),
\]
where \( J(x, y) = \left| \frac{\partial}{\partial x} f(x, y) \right|^2 + \gamma \left| \frac{\partial}{\partial y} f(x, y) \right|^2 \) and \( \gamma > 0 \) is the weighting factor.
The solid cyan line manifests the general fractional order gradient descent method cannot converge to the exact extreme point, which just coincides with the claim in [11].

**Example 5:** Consider the famous Rosenbrock function [21], i.e., \( f(x, y) = (1-x)^2 + 100(y-x^2)^2 \). The real extreme point of \( f(x, y) \) is \( x = 1, y = 1 \). Choose the parameters as \( c = 0, x_0 = -0.2, y_0 = -0.2, \epsilon = 0, \alpha = 0.7, K = 2 \) and \( \beta = 0.01 \). Because the minimum value of the target function is 0, the formula (22) is adopted to calculate a common \( \alpha(x, y) \). The learning rates are selected separately for the three algorithms, i.e., \( \mu = 0.0182, \mu = 0.0018 \) and \( \mu = 0.002 \). On this basis, three proposed methods are implemented numerically and the simulation curves are recorded in Figs. 11-13.

As can be seen from Fig. 16, all of them accomplish the parameter estimation successfully. It is no exaggeration to say that Algorithm 3 is the best in the right direction. It is plain that the red dotted line, which denotes the result of Algorithm 1, in Fig. 13 would eventually reach to the extreme point \((1, 1)\).

**Example 6:** Consider a three order transverse filtering issue shown in Fig. 14. The optimal tap weight is \( \mathbf{w} = [w_1, w_2, w_3] = [2, -3, 1] \). The known input \( u \) and unknown noise \( v \) are given in Fig. 15. Select the parameters \( c = 0, \mathbf{w}(0) = [0.1, -0.1, 0.1], \mu = 0.02, \epsilon = 0, \alpha = 0.7, K = 3 \) and \( \beta = 0.005 \). \( \alpha(w_i), i = 1, 2, 3 \) are designed according to (22) with \( J(w_i) = \epsilon^2 \). Then, all the three proposed methods can be used to estimate parameters of the filter and simulation results are given in Fig. 16.

\[
e_k = [x_k, y_k] - [1, 1],
\]

then it can be calculated that \( e_{9000} = [0.0105, 0.0208] \) and \( e_{10000} = [0.0035, 0.007] \). Though Algorithm 1 performs bad in later period, it is surely ahead of the other algorithms within 5000th iteration and it is the first one entered the 2\% error band. According to Fig. 13, what can be seen is that all the curves get together as \( y = x^2 \) when they are away from the starting point soon. Although Algorithm 1 does not converge to the extreme point in the previous two figures, it converges in the right direction. It is plain that the red dotted line, which denotes the result of Algorithm 1, in Fig. 13 would eventually reach to the extreme point \((1, 1)\).

Rosenbrock function is a non-convex benchmark function, which is often used as a performance test problem for optimization algorithm [21]. The extreme point of Rosenbrock function is in a long, narrow and parabolic valley, which is difficult to reach. Fig. 11 and Fig. 12 are variation of \( x \) and \( y \) of Rosenbrock case, respectively. The general fractional case is not given in the figures because it also converges to a point different from the exact extreme point. The three proposed methods are given in figures, and it turns out that Algorithm 2 and Algorithm 3 converge to the extreme point at around 6000th iteration. Algorithm 1 could approximate the extreme point while small deviations still exist. More specially, defining the error as
to say that the three methods have considerable convergence speed. Fig. 16 also demonstrates that the proposed solutions can resolve the convergence problems of fractional gradient descent method and the resulting methods are implementable in practical case.

V. CONCLUSIONS

In this paper, the convergence problem of fractional order gradient descent method has been tentatively investigated. It turns out that general fractional gradient method cannot converge to the actual extreme point. By exploiting the natural properties of fractional derivative, three individual solutions are proposed in detail, including the fixed memory step, the higher order truncation, and the variable fractional order. Both theoretical analysis and simulation study indicate that all the designed methods can achieve the true convergence quickly. It is believed that this work is beneficial for solving the pertinent optimization problems with fractional order methods. The following issues would be the topic of the future researches.

i) Design and analyze new convergence design solutions.
ii) Extended the results to fractional Lipschitz condition.
iii) Consider the nonsmooth or nonconvex target function.

REFERENCES

[1] J. S. Zeng and W. T. Yin, “On nonconvex decentralized gradient descent,” IEEE Transactions on Signal Processing, vol. 66, no. 11, pp. 2834–2848, 2018.
[2] H. Q. Li, S. Liu, Y. C. Soh, and L. H. Xie, “Event-triggered communication and data rate constraint for distributed optimization of multi-agent systems,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 11, pp. 1908–1919, 2018.
[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, 2015.
[4] Z. J. Li, H. Z. Xiao, C. G. Yang, and Y. W. Zhao, “Model predictive control of nonholonomic chained systems using general projection neural networks optimization,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 45, no. 10, pp. 1313–1321, 2015.
[5] A. Beck and M. Teboulle, “Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems,” IEEE Transactions on Image Processing, vol. 18, no. 11, pp. 2419–2434, 2009.
[6] Y. F. Pu, J. L. Zhou, and X. Yuan, “Fractional differential masks: A fractional differential-based approach for multiscale texture enhancement,” IEEE Transactions on Image Processing, vol. 19, no. 2, pp. 491–511, 2010.
[7] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge: Cambridge University Press, 2004.
[8] P. Liu, Z. G. Zeng, and J. Wang, “Multiple Mittag–Leffler stability of fractional-order recurrent neural networks,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 8, pp. 2279–2288, 2017.
[9] W. D. Yin, Y. H. Wei, T. Y. Liu, and Y. Wang, “A novel orthogonalized fractional order filtered-s normalized least mean squares algorithm for feedforward vibration rejection,” Mechanical Systems and Signal Processing, vol. 119, pp. 138–154, 2019.
[10] Y. Tan, Z. Q. He, and B. Y. Tian, “A novel generalization of modified LMS algorithm to fractional order,” IEEE Signal Processing Letters, vol. 22, no. 9, pp. 1244–1248, 2015.
[11] Y. F. Pu, J. L. Zhou, Y. Zhang, N. Zhang, G. Huang, and P. Siarry, “Fractional extreme value adaptive training method: fractional steepest descent approach,” IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 4, pp. 653–662, 2015.
[12] C. C. Tseng and S. L. Lee, “Designs of fractional derivative constrained 1-D and 2-D FIR filters in the complex domain,” Signal Processing, vol. 95, pp. 111–125, 2014.
[13] S. M. Shah, R. Samar, N. M. Khan, and M. A. Z. Raja, “Design of fractional-order variants of complex LMS and NLMS algorithms for adaptive channel equalization,” Nonlinear Dynamics, vol. 88, no. 2, pp. 839–858, 2017.
[14] Y. Q. Chen, Q. Gao, Y. H. Wei, and Y. Wang, “Study on fractional order gradient methods,” Applied Mathematics and Computation, vol. 314, pp. 310–321, 2017.
[15] S. S. Cheng, Y. H. Wei, D. Sheng, Y. Q. Chen, and Y. Wang, “Identification for Hammerstein nonlinear ARMAX systems based on multi-innovation fractional order stochastic gradient ,” Signal Processing, vol. 142, pp. 1–10, 2017.
[16] S. Zubair, N. I. Chaudhary, Z. A. Khan, and W. Wang, “Momentum fractional LMS for power signal parameter estimation,” Signal Processing, vol. 142, pp. 441–449, 2018.
[17] J. Wang, Y. Q. Wen, Y. D. Gou, Z. Y. Ye, and H. Chen, “Fractional-order gradient descent learning of BP neural networks with Caputo derivative,” Neural Networks, vol. 89, pp. 19–30, 2017.
[18] Z. A. Khan, N. I. Chaudhary, and S. Zubair, “Fractional stochastic gradient descent for recommender systems,” Electronic Markets, pp. 1–11, 2018.
[19] I. Podlubny, Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications. San Diego: Academic Press, 1998.
[20] Y. H. Wei, Y. Q. Chen, S. S. Cheng, and Y. Wang, “A note on short memory principle of fractional calculus,” Fractional Calculus and Applied Analysis, vol. 20, no. 6, pp. 1382–1404, 2017.
[21] H. Rosenbrock, “An automatic method for finding the greatest or least value of a function,” The Computer Journal, vol. 3, no. 3, pp. 175–184, 1960.