frontiers in Physiology

ORIGINAL RESEARCH
published: 21 September 2021
doi: 10.3389/fphys.2021.724044

NEO: NEuro-Inspired Optimization—A Fractional Time Series Approach

Sarthak Chatterjee[1]*, **Subhro Das**[2] **and Sérgio Pequito**[3]

[1] Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, United States, [2] MIT-IBM Watson AI Lab, IBM Research, Cambridge, MA, United States, [3] Delft Center for Systems and Control, Delft University of Technology, Delft, Netherlands

Solving optimization problems is a recurrent theme across different fields, including large-scale machine learning systems and deep learning. Often in practical applications, we encounter objective functions where the Hessian is ill-conditioned, which precludes us from using optimization algorithms utilizing second-order information. In this paper, we propose to use fractional time series analysis methods that have successfully been used to model neurophysiological processes in order to circumvent this issue. In particular, the long memory property of fractional time series exhibiting non-exponential power-law decay of trajectories seems to model behavior associated with the local curvature of the objective function at a given point. Specifically, we propose a NEuro-inspired Optimization (NEO) method that leverages this behavior, which contrasts with the short memory characteristics of currently used methods (e.g., gradient descent and heavy-ball). We provide evidence of the efficacy of the proposed method on a wide variety of settings implicitly found in practice.

Keywords: optimization, time series processes, iterative optimization algorithms, long memory time series, fractional calculus

OPEN ACCESS

Edited by:
Dietmar Plenz,
National Institute of Mental Health,
National Institutes of Health (NIH),
United States

Reviewed by:
Corina Stefania Drapaca,
The Pennsylvania State University (PSU), United States
Hussein M. Yahia,
Institut National de Recherche en Informatique et en Automatique (INRIA), France

****Correspondence:***
Sarthak Chatterjee
chatts3@rpi.edu

Specialty section:
This article was submitted to Fractal Physiology, a section of the journal Frontiers in Physiology

Received: *11 June 2021*
Accepted: *24 August 2021*
Published: *21 September 2021*

Citation:
Chatterjee S, Das S and Pequito S (2021) NEO: NEuro-Inspired Optimization—A Fractional Time Series Approach. Front. Physiol. 12:724044. doi: 10.3389/fphys.2021.724044

1. INTRODUCTION

Many problems in today's world can be modeled as optimization problems where we seek to find the solution to an unconstrained optimization problem with an objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (Bishop, 2006; Sra et al., 2012), which is a real-valued function of n real variables. For instance, in a learning problem, we aim to minimize a loss index that measures the performance of a neural network (NN) on a data set. Often, the loss index is composed of an error term and a regularization term that evaluates how well the NN fits the data set and discourages overfitting, respectively (Scholkopf and Smola, 2001). Besides controlling overfitting, the regularization term can also be designed to control the complexity of a NN, e.g., by reducing the number of non-zero weights $w \in \mathbb{R}^n$ (LeCun and Bengio, 1995).

Notwithstanding the above, most optimization problems do not possess numerically viable closed-form solutions (Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006). Furthermore, due to the ever-increasing dimensionality of data used to test a variety of optimization problems, iterative algorithms need to be employed to attain an approximate solution. At the core of the iterative algorithms, we can commonly find three key ingredients (Bertsekas, 1997; Nocedal and Wright, 2006): (*i*) a descent direction $d \in \mathbb{R}^n$, (*ii*) a learning rate (or, step) $\alpha \in \mathbb{R}$; and (*iii*) local spatial information across different variables (i.e., $w \in \mathbb{R}^n$). In a nutshell, the iterative algorithms can be written as

$$w_{k+1} = w_k + \alpha_k d_k, \qquad k = 0, 1, \ldots, \tag{1}$$

where the descent direction $d_k \in \mathbb{R}^n$ and the step $\alpha_k \in \mathbb{R}$ might change over time step k. Descent directions can be computed using one of the following options:

- Gradient descent, $d_k = -\nabla f(w_k)$, where ∇ denotes the first-order derivative (Nocedal and Wright, 2006);
- Stochastic gradient descent, $d_k = -\widetilde{\nabla} f(w_k)$, where sampled points and the approximate notion of the derivative are used to determine a possible descent direction and where $\widetilde{\nabla}$ represents the first-order derivative calculated from either a single point in the data set or an arbitrarily selected subset of the entire data set (an approach referred to in the literature as mini-batch, Saad, 2009);
- Newton's method, $d_k = -(Hf(w_k))^{-1}\nabla f(w_k)$, where $Hf(w_k)$ denotes the Hessian matrix at w_k, i.e., the second-order derivative (Boyd and Vandenberghe, 2004); and
- Quasi-Newton methods, $d_k = -B_k^{-1}\nabla f(w_k)$, where the B is an approximation of the Hessian matrix, e.g., using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Dennis and Schnabel, 1996).

Simply speaking, the second-order derivative captures (local) spatial properties of the function through its cross-derivatives and geometrically corresponds to a quadratic function approximation of the function that we are optimizing. Subsequently, first-order iterative methods (e.g., gradient descent and stochastic gradient descent) are slower than second-order iterative methods (e.g., Newton and quasi-Newton) but require less computational power and memory storage. Recent literature in this direction has been largely focused on the hybridization of these techniques in context-based scenarios (Schraudolph et al., 2007; Roux et al., 2012; Johnson and Zhang, 2013; Cevher et al., 2014; Defazio et al., 2014; Byrd et al., 2016; Moritz et al., 2016; Rodomanov and Kropotov, 2016; Zhang and Gu, 2016; Mokhtari et al., 2018; Paternain et al., 2019).

An alternative to speeding up the convergence while not resorting to second-order methods is to consider memory in the iterative process. Specifically, consider

$$w_{k+1} = w_k + \alpha_k d_k + m(w_{k-1}, \ldots, w_{k-T}), \quad k = 0, 1, \ldots, \tag{2}$$

where $m : \mathbb{R}^{n\times T} \rightarrow \mathbb{R}^n$ is a function of the previous instances of the parameters up to T time steps in the past. A particular case is the so-called heavy-ball method proposed by Polyak (Polyak, 1964) that builds on momentum-based physical intuition (that actually might not converge, Lessard et al., 2016) and later made more formal by Nesterov in today's celebrated accelerated convergence methods (Nesterov, 2013). Nesterov's proof techniques abandoned physical intuition and devised the method of estimate sequences to verify the correctness of these momentum-based methods. Nonetheless, researchers have struggled to understand the foundations and scope of the estimate sequence methodology since the proof techniques are often viewed as "algebraic tricks" that are only applicable to some classes of functions.

Consequently, a more complete arsenal of tools is needed to understand the convergence of first-order methods that consider memory. To address this issue, recent literature has leveraged insights and tools available in dynamical systems theory, as, in the limit, the iterative algorithm is a dynamical system in which evolution is described by ordinary differential equations (Su et al., 2014; Hardt et al., 2016; Lessard et al., 2016; Wilson et al., 2016; Hu and Lessard, 2017; Fazlyab et al., 2018; Zhang et al., 2019). These accelerated methods are also often driven by an exogenous signal (or control input) that regulates both the asymptotic convergence to the minimum and also the convergence rate.

In this paper, we seek to leverage fractional-order calculus (Oldham and Spanier, 1974; Baleanu et al., 2011, 2012; Ortigueira, 2011) to develop a new iterative optimization algorithm. Fractional derivatives and fractional-order processes have been widely used to model phenomena having long-term memory in the context of neurophysiological data (Lundstrom et al., 2008; Xue et al., 2016). Additionally, autoregressive fractionally integrated moving average (ARFIMA) time series processes are successfully able to model and explain a wide variety of biological phenomena (Ionescu et al., 2017), particularly phenomena with long-term memory and power law dependence of trajectories (Miller et al., 2009), and have been used in contexts such as the prediction and forecasting of financial market data (Bukhari et al., 2020). We are inspired by the recent spate of successes that fractional-order based models have enjoyed in the context of neuronal data (Teka et al., 2014) as well as increasing evidence presented to explain the intricate relationships between the neural-like architectures used with great success in deep learning and their relationships with systems neuroscience (Richards et al., 2019).

Consequently, we propose a novel iterative method termed as NEO (NEuro-inspired Optimization). At each step, NEO models the local evolution (i.e., determines the ARFIMA model that best describes the local curvature) and takes the argument that attains the lowest predicted values as the next iteration point. We provide the proof of convergence of the proposed method and numerical evidence of the efficacy of NEO on a variety of problem settings implicitly found in practice. Additionally, we notice that NEO only requires the values of the objective at the points, without the need to compute derivatives, and without the explicit need to tune a step size. Furthermore, our simulations suggest that the major advantages of NEO lie in cases where the Hessian is ill-conditioned, which is particularly important in the context of several real-world problems, for instance, neural networks (Saarinen et al., 1993).

2. FRACTIONALLY INTEGRATED TIME SERIES PROCESSES

A class of stationary long-term processes z_t modeled as *autoregressive fractionally integrated moving average* (ARFIMA) processes are described by

$$\phi(B)(1-B)^d z_t = \theta(B)a_t, \tag{3}$$

for $d \in \mathbb{R}$, d being the *fractional differencing parameter*. Here, a_t is a white noise sequence having zero mean and bounded variance

σ_a^2, the polynomial equations

$$\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i = 0 \tag{4}$$

and

$$\theta(B) = 1 + \sum_{i=1}^{q} \theta_i B^i = 0 \tag{5}$$

have roots that are greater than unity in absolute value, and B is the *backward shift operator* with the property $B^m z_t = z_{t-m}$. The general form of the processes that can be represented using (3) are called ARFIMA(p, d, q) processes (Box et al., 2015).

For $d > -1$, we can employ the binomial expansion formula to explicitly expand the operator $(1 - B)^d$ as

$$(1 - B)^d = \sum_{j=0}^{\infty} \pi_j B^j, \tag{6}$$

with $\pi_0 = 1$, and $\pi_j = \frac{\Gamma(j-d)}{\Gamma(j+1)\Gamma(-d)}$, $j = 1, 2, \ldots$, with $\Gamma(\cdot)$ being the Gamma function defined by $\Gamma(x) = \int_0^\infty s^{x-1} e^{-s}\, ds$ for all complex numbers x with $\Re(x) > 0$. We note that the weighting coefficients π_j can be defined recursively in j. Further, even though the binomial expansion of the operator $(1 - B)^d$ consists of an infinite number of terms, in practice we will always consider an approximation that still preserves the dependency of parameters described.

More generally, ARFIMA processes can generalize ordinary autoregressive moving average (ARMA) models in the following way. Given a time series, we can carry out the following steps to obtain the parameters in (1):

- Apply fractional differencing on the original time series and note the order of the fractional difference d that makes the time series (close to) wide-sense stationary;
- Determine the ARMA parameters $p, q, \{\phi_i\}_{i=1}^p$, and $\{\theta_i\}_{i=1}^q$ using the (fractionally) differentiated time series;
- Perform a forecast for a requisite number of steps ahead in time with these ARMA terms; and;
- Fractionally integrate the forecasted ARMA data to obtain the forecast of the ARFIMA process. Note that fractional integration may be interpreted as fractional differentiation but with a fractional differencing parameter of $-d$.

3. THE NEURO-INSPIRED OPTIMIZATION (NEO) METHOD

In what follows, we outline the details of the NEO iterative optimization method that seeks to determine the solution to an unconstrained optimization problem. At each step, NEO models the local evolution (i.e., determines the ARFIMA model that best describes the local curvature) and takes the argument that attains the lowest predicted values as the next iteration point.

In order to develop the intuition about the NEO method, let us consider a pedagogical example – i.e., we seek to determine the solution to the following unidimensional unconstrained optimization problem

$$x^\star = \underset{x \in \mathbb{R}}{\operatorname{argmin}}\, x^2. \tag{7}$$

Furthermore, let us consider an initial point x_0, and let us generate a time series that considers a discretization step $h > 0$, the number of steps of memory $P \in \mathbb{N}$, and the corresponding functional values $f(x_0), f(x_0 - h), \ldots, f(x_0 - (P - 1)h)$. Next, we investigate the sample autocorrelation function (sACF) obtained from the aforementioned values. The sACF profile is shown in **Figure 1**. First, we notice that the sample autocorrelation function (sACF) obtained from the aforementioned values and depicted in **Figure 1** suggests slower than exponential algebraic decay and statistically significant (for a significance level of 5%) dependency on past lags, with a large area enclosed by the composite sACF curve and the horizontal axis. This suggests that the ARFIMA(p, d, q) processes described above can successfully predict the behavior of the functional values obtained. Furthermore, although the dependency on past lags is shown here for a quadratic function, other functions will exhibit similar behaviors, which are likely associated with properties related to the local curvature of the objective function at any given point. As such, we can use the determined (or learned) ARFIMA(p, d, q) model to predict the values that the function will take in a 'descending direction' to then consider the argument that attains the lowest predicted value. Then, we take the value of this argument as the initial point and proceed similarly to the above-mentioned point until a desirable stopping criterion is attained. It is worth mentioning that while using ARFIMA(p, d, q) processes, we do not explicitly need to know the function $f(\cdot)$, but instead, only the functional values.

NEO is described in its general form in **Algorithm 1** and a schematic overview is presented in **Figure 2** for the unidimensional unconstrained case, which can be applied to solve multiple dimensional problems as well by iteratively running it over each individual dimension. In what follows, the steps corresponding to the NEO method are described in more detail.

First, we consider the pre-specified values of p and q along with the Whittle estimation procedure (Whittle, 1961) (see details in the **Appendix** in the **Supplementary Material**) to find the fractional differencing parameter d and the autoregressive and moving average coefficients $\{\phi_i\}_{i=1}^p$ and $\{\theta_i\}_{i=1}^q$, respectively, from the functional values $f(x_k), f(x_k - h), \ldots, f(x_k - (P - 1)h)$, for $k = 0, 1, 2, \ldots$. From numerical experimentation, particularly the one presented in **Figure 1**, we can see from the sACF of the functional values that the latter are correlated over time, which is then directly associated with the curvature of the function in the optimization landscape.

Next, we employ the above estimated parameters to perform an ARFIMA time series prediction, P' steps into the future, in order to obtain the time series $y_1, y_2, \ldots, y_{P'}$. As depicted in the **Figure 2**, we find that our ARFIMA time series predictions are limited in their predictive capabilities since they can only capture information about the local behavior of the function upto a certain finite number of time steps into the future. Since

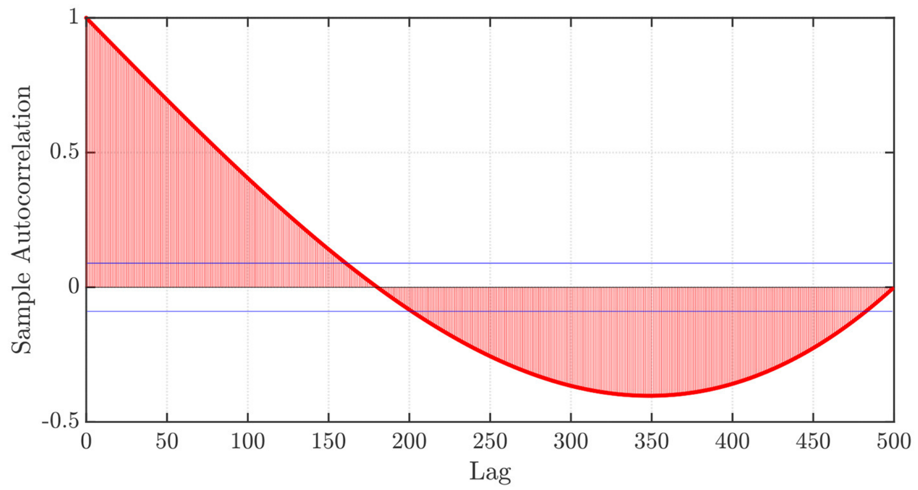
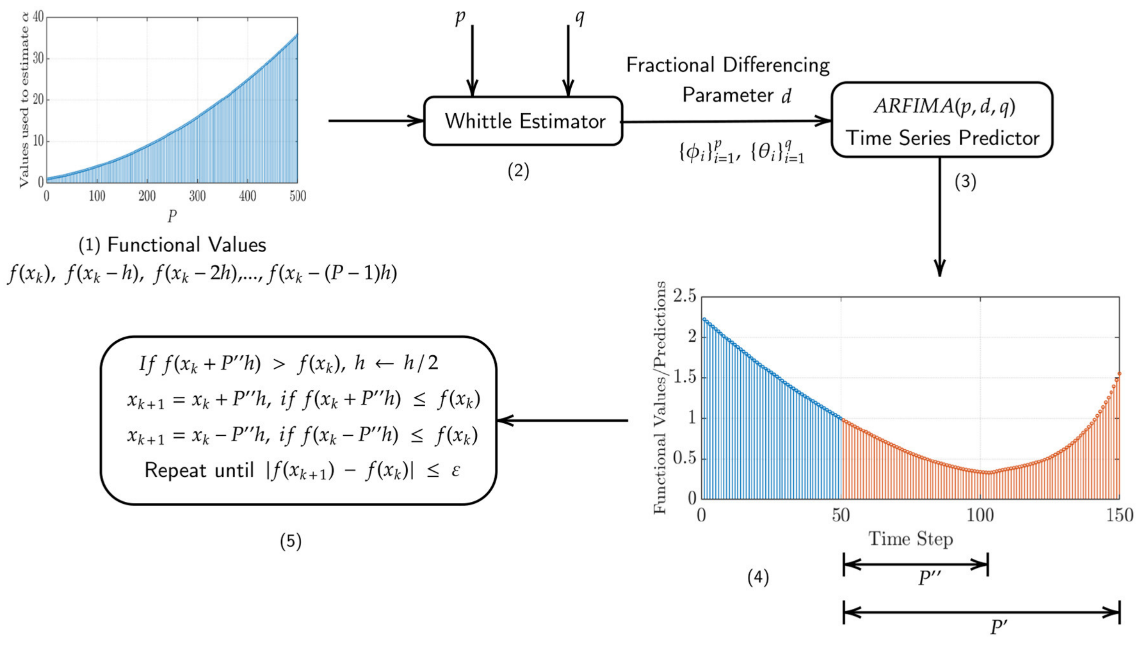


FIGURE 1 | sACF plot of the functional values $f(x_0), f(x_0 - h), \ldots, f(x_0 - (P-1)h)$, with $f(x) = x^2$, $x_0 = -1$, $P = 500$, and $h = 0.01$.



Values used to estimate α
P
(1) Functional Values
$f(x_k),\ f(x_k - h),\ f(x_k - 2h), \ldots, f(x_k - (P-1)h)$
p
q
Whittle Estimator
(2)
Fractional Differencing
Parameter d
$\{\phi_i\}_{i=1}^{p},\ \{\theta_i\}_{i=1}^{q}$
$ARFIMA(p, d, q)$
Time Series Predictor
(3)
Functional Values/Predictions
Time Step
(4)
P''
P'
If $f(x_k + P''h) > f(x_k)$, $h \leftarrow h/2$
$x_{k+1} = x_k + P''h$, if $f(x_k + P''h) \le f(x_k)$
$x_{k+1} = x_k - P''h$, if $f(x_k - P''h) \le f(x_k)$
Repeat until $|f(x_{k+1}) - f(x_k)| \le \varepsilon$
(5)

FIGURE 2 | Schematic representation of the NEO method. (1) The functional values $f(x_0), f(x_0 - h), \ldots, f(x_0 - (P-1)h)$, with pre-specified values of x_0, P, and h, used by the Whittle estimator. (2) The Whittle estimator, which takes the aforementioned values as input and outputs the fractional differencing parameter d and the autoregressive and moving average coefficients $\{\phi_i\}_{i=1}^{p}$ and $\{\theta_i\}_{i=1}^{q}$, respectively. (3) The ARFIMA time series predictor, which predicts P' steps into the future. (4) Illustration of the largest possible value of $P'' \le P'$, such that P'' satisfies $y_1 \ge y_2 \ge \ldots \ge y_{P''} \le y_{P''+1}$. (5) Update step for the iterate x_k.

many descent methods in the optimization literature require us to satisfy $f(x_{k+1}) \le f(x_k)$, we select the largest possible value of $P'' \le P'$, such that P'' satisfies $y_1 \ge y_2 \ge \ldots \ge y_{P''} \le y_{P''+1}$. If, at this stage, $f(x_k + P''h) > f(x_k)$, we update the discretization step h by $h/2$ until the condition $f(x_k + P''h) \le f(x_k)$ is satisfied. Once that is obtained, we update the current iterate as

$$x_{k+1} = \begin{cases} x_k + P''h, & \text{if } f(x_k + P''h) \le f(x_k) \\ x_k - P''h, & \text{if } f(x_k - P''h) \le f(x_k) \end{cases}. \tag{8}$$

The method terminates when $|f(x_{k+1}) - f(x_k)| \le \varepsilon$, where $\varepsilon \in \mathbb{R}^+$ is a specified tolerance.

4. SIMULATION RESULTS

In this section, we generalize the NEO method shown for unidimensional problems in **Algorithm 1** to two-dimensional problems.

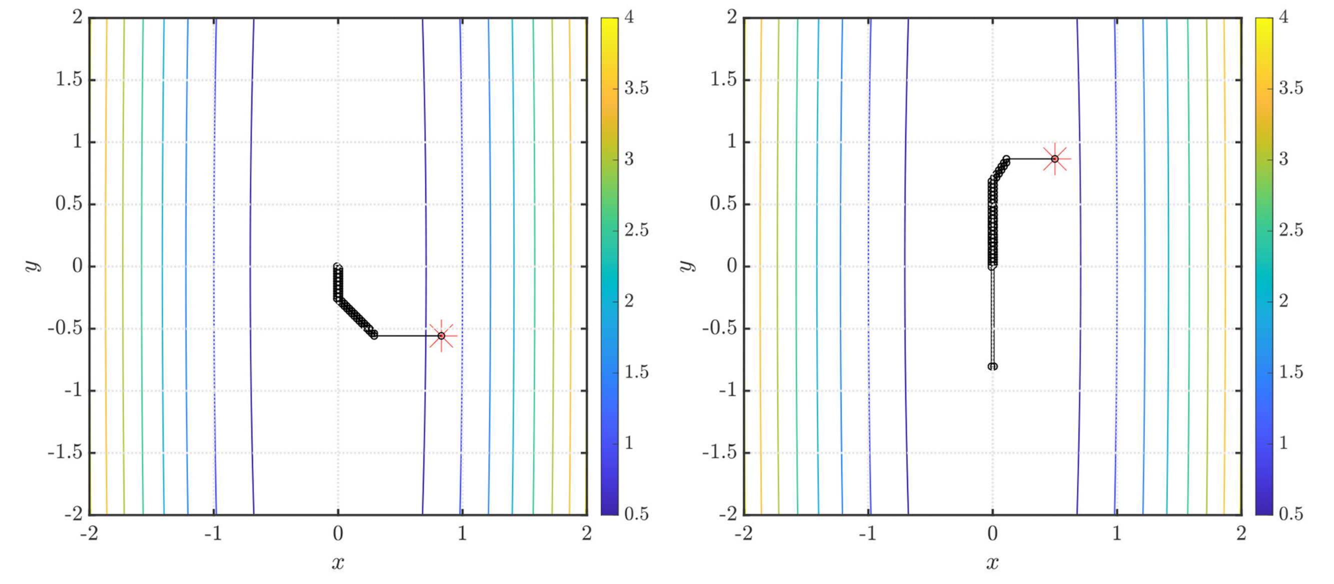


FIGURE 3 | (**Left**) Steps taken in the two-dimensional plane when the NEO method is used to find the global minimum of the function $f(x, y) = x^2 + 0.01y^2$. The initial point (denoted by the red asterisk) x_0 is arbitrarily selected on the unit circle. We use ARFIMA(4, d, 0) time series predictions. (**Right**) The initial point (denoted by the red asterisk) $x_0 = [1/2\ \sqrt{3}/2]^T$. We use ARFIMA(4, d, 0) time series predictions and use a constant value of d averaged out after the first 10 iterations of the method.

Algorithm 1: The NEO method for unidimensional unconstrained optimization problems.

```
Result: Finding the global minimum $x^\star$ of a unidimensional
        unconstrained optimization problem
        $x^\star = \text{argmin}_{x \in \mathbb{R}} f(x)$
Initialize $x_0, P, h, P', \varepsilon, p, q$;
while $|f(x_{k+1}) - f(x_k)| > \varepsilon$ do
    For each point $x_k$, $k = 0, 1, 2, \ldots$, obtain the functional
    values $f(x_k), f(x_k - h), \ldots, f(x_k - (P - 1)h)$;
    Use the functional values
    $f(x_k), f(x_k - h), \ldots, f(x_k - (P - 1)h)$, the autoregressive
    parameter $p$, and the moving average parameter $q$, in
    order to estimate the fractional differencing parameter $d$
    using the Whittle estimation method;
    Perform an ARFIMA($p, d, q$) time series prediction $P'$
    steps into the future, in order to obtain the time series
    values $y_1, y_2, \ldots, y_{P'}$;
    Obtain the largest possible $P'' \leq P'$ that satisfies
    $y_1 \geq y_2 \geq \ldots \geq y_{P''} \leq y_{P''+1}$;
    while $f(x_k + P''h) > f(x_k)$ do
        $h \leftarrow h/2$ ;
    end
    if $f(x_k + P''h) \leq f(x_k)$ then
        $x_{k+1} \leftarrow x_k + P''h$ ;
    else if $f(x_k - P''h) \leq f(x_k)$ then
        $x_{k+1} \leftarrow x_k - P''h$ ;
end
```

4.1. Results on Two-Dimensional Optimization Problems

In the first example, we show the performance of our method in finding the global minimum of the function $f(x, y) = x^2 + 0.01y^2$ in **Figure 3** (left). The starting point x_0, denoted by the red asterisk, is chosen to be an arbitrary point on the unit circle. Convergence is obtained in 56 iterations using ARFIMA(4, d, 0) time series predictions. We use $P = 100$ steps of memory and an initial grid discretization step of $h = 0.01$, settings we will preserve for the remainder of the paper.

We also consider the problem setting where we use our method in order to find the global minimum of the functions $f(x, y) = \mathbf{x}^T Q\mathbf{x}$, with $\mathbf{x} = [x\ y]^T$ and $Q = \text{diag}(1, \kappa)$, with $\kappa \in \{1, 0.1, 0.01, 0.001\}$. For each case, we use 12 different equispaced starting points initialized on the unit circle. To highlight the differences, we compare our approach against gradient descent implemented with inexact backtracking line search (Bertsekas, 1997; Nocedal and Wright, 2006), which we use to tune the step size of gradient descent. The results, in terms of iterations taken to convergence as well as running time averaged over the entire run, are provided in **Table 1**.

We see from **Table 1** that significant advantages are gained by using our approach for more ill-conditioned optimization problems, i.e., functions with a lower value of κ. Specifically, our method performs better where small changes in the inputs to the function cause significant changes to the functional value, owing to the function being very sensitive to its inputs. From the comparative study, we also note that with our approach, we have done away with the need to have a step size as an integral

TABLE 1 | Iterations taken to convergence to find the minimum of $f(x, y) = x^2 + \kappa y^2$ with 12 different starting points x_0 initialized on the unit circle, given by $x_0 = [\cos\theta \ \ \sin\theta]^T$, for values of κ in the set $\{1, 0.1, 0.01, 0.001\}$, for the NEO method vs. gradient descent (GD) implemented with inexact backtracking line search (for the same starting point in each row). ARFIMA$(4, d, 0)$ time series predictions are used in our method for each case.

	θ	Iterations for NEO	Time for NEO Sys. Id. (s)	Time for NEO (s)	Iterations for GD	Time for GD (s)
$\kappa = 1$	$\pi/6$	24	$5.3050e-01$	$6.6400e-05$	3	$3.2000e-03$
	$\pi/3$	24	$9.2580e-01$	$5.7020e-04$	3	$3.7000e-03$
	$\pi/2$	22	$7.0860e-01$	$3.0400e-05$	3	$4.2000e-03$
	$2\pi/3$	9	$6.3920e-01$	$6.3700e-05$	30	$9.6000e-03$
	$5\pi/6$	51	$6.5400e-01$	$7.2700e-05$	3	$3.9000e-03$
	π	50	$5.4640e-01$	$2.9600e-05$	3	$4.1000e-03$
	$7\pi/6$	68	$5.4910e-01$	$5.9800e-05$	3	$5.7000e-03$
	$4\pi/3$	68	$4.6320e-01$	$6.4600e-05$	3	$3.5000e-03$
	$3\pi/2$	50	$7.2680e-01$	$3.0700e-05$	3	$6.1000e-03$
	$5\pi/3$	51	$6.1120e-01$	$8.9000e-05$	3	$3.9000e-03$
	$11\pi/6$	42	$5.8540e-01$	$6.6900e-05$	3	$4.1000e-03$
	2π	5	$5.8140e-01$	$2.7930e-04$	3	$3.7000e-03$
$\kappa = 0.1$	$\pi/6$	39	$3.5680e-01$	$1.2690e-04$	24	$7.4000e-03$
	$\pi/3$	44	$5.5990e-01$	$8.6200e-05$	26	$7.7000e-03$
	$\pi/2$	21	$7.1380e-01$	$6.6500e-05$	25	$8.9000e-03$
	$2\pi/3$	56	$3.1770e-01$	$6.1500e-05$	26	$5.6000e-03$
	$5\pi/6$	84	$8.2920e-01$	$9.4500e-05$	24	$6.5000e-03$
	π	50	$4.5560e-01$	$4.4100e-05$	30	$6.0000e-03$
	$7\pi/6$	68	$4.0790e-01$	$6.3800e-05$	24	$7.1000e-03$
	$4\pi/3$	68	$4.9120e-01$	$9.6300e-05$	26	$5.7000e-03$
	$3\pi/2$	50	$3.9610e-01$	$4.0700e-05$	25	$6.1000e-03$
	$5\pi/3$	51	$3.7460e-01$	$1.2800e-04$	26	$6.7000e-03$
	$11\pi/6$	42	$4.9560e-01$	$8.6100e-05$	24	$7.9000e-03$
	2π	14	$4.3690e-01$	$6.3250e-04$	30	$6.5000e-03$
$\kappa = 0.01$	$\pi/6$	37	$4.1270e-01$	$6.4600e-05$	459	$2.0000e-03$
	$\pi/3$	37	$3.5440e-01$	$2.4000e-03$	486	$1.0400e-02$
	$\pi/2$	38	$3.4640e-01$	$3.4800e-05$	492	$7.7000e-03$
	$2\pi/3$	35	$2.9140e-01$	$8.2900e-05$	486	$1.0800e-02$
	$5\pi/6$	64	$2.4370e-01$	$1.1600e-04$	459	$7.8000e-03$
	π	50	$3.8860e-01$	$3.1900e-05$	60	$5.9000e-03$
	$7\pi/6$	64	$3.5810e-01$	$1.0760e-04$	459	$8.000e-03$
	$4\pi/3$	33	$4.2200e-01$	$1.5310e-04$	486	$6.3000e-03$
	$3\pi/2$	24	$4.9150e-01$	$3.0500e-05$	492	$6.5000e-03$
	$5\pi/3$	40	$8.0420e-01$	$6.0420e-04$	486	$8.7000e-03$
	$11\pi/6$	47	$5.2720e-01$	$6.2600e-05$	459	$8.000e-03$
	2π	2	$3.7580e-01$	$7.1330e-04$	60	$7.3000e-03$
$\kappa = 0.001$	$\pi/6$	41	$3.4880e-01$	$7.3300e-05$	3453	$1.4600e-02$
	$\pi/3$	44	$3.5140e-01$	$9.1600e-05$	3728	$1.8700e-02$
	$\pi/2$	42	$5.5600e-01$	$2.9800e-05$	3798	$1.5800e-02$
	$2\pi/3$	55	$3.5600e-01$	$6.5300e-05$	3728	$1.9800e-02$
	$5\pi/6$	67	$4.7020e-01$	$6.6900e-05$	3453	$1.3100e-02$
	π	50	$3.2680e-01$	$1.1030e-04$	60	$5.8000e-03$
	$7\pi/6$	58	$7.9010e-01$	$5.6100e-05$	3453	$1.5100e-02$
	$4\pi/3$	58	$2.9740e-01$	$8.5100e-05$	3728	$1.7800e-02$
	$3\pi/2$	37	$3.4430e-01$	$3.4900e-05$	3798	$1.3600e-02$
	$5\pi/3$	51	$4.2040e-01$	$7.6490e-04$	3728	$1.6800e-02$
	$11\pi/6$	43	$3.1390e-01$	$9.0800e-04$	3453	$1.1100e-02$
	2π	5	$3.6100e-01$	$2.9000e-03$	60	$9.2000e-03$

component of our method, thus reducing the complexities involved in its tuning. We also do not need to use any first-order or second-order gradient or Hessian information, with only the functional values sufficing. All implementations are done in MATLAB R2019b, running on an x64-based AMD Ryzen 5 2500U (with Radeon Vega Mobile Gfx) processor at 2.00 GHz, with 8 GB of RAM.

We note here that although a rich body of results is provided in this paper for two-dimensional quadratic problems, we argue that this important subclass of optimization problems actually provides a diverse setting that generalizes to more complicated scenarios. To see why this is true, consider the performance of the gradient descent method used to minimize the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = \frac{1}{2}x^{\mathrm{T}}Ax - b^{\mathrm{T}}x + c, \tag{9}$$

where $A \in \mathbb{R}^{n\times n}$ is positive definite and $b, c \in \mathbb{R}^n$. We assume that $LI \succeq A \succeq \ell I$, where the notation $A \succeq B$ signifies the partial order (Loewner order) defined by the convex cone of positive semi-definite matrices and that $A - B$ is positive semi-definite. The above problem can, of course, be analytically solved, and the unique global minimizer is $x^\star = A^{-1}b$. Further, $f(x)$ is convex (actually strictly convex) since the Hessian $Hf(x) = A \succ 0$, i.e., $Hf(x)$ is positive definite[1]. Consider the standard gradient descent method

$$x_{k+1} = x_k - \eta_k \nabla f(x_k), \tag{10}$$

for some starting point x_0 and some sequence of step sizes $\{\eta_k\}$. We can show that if we choose

$$\eta_k = \frac{2}{\ell + L} \; \forall\, k, \tag{11}$$

then

$$\|x_k - x^\star\| \leq \left(\frac{\kappa_{\mathrm{GD}} - 1}{\kappa_{\mathrm{GD}} + 1}\right)^k \|x_0 - x^\star\|, \tag{12}$$

where

$$\kappa_{\mathrm{GD}} = \frac{L}{\ell} = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \tag{13}$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote, respectively, the maximum and minimum singular values of the matrix A. In other words, we find that even for an optimization problem in high dimensions, the maximum and minimum singular values govern the linear convergence of the constant step-size policy of gradient descent, which means that for a matrix $A \in \mathbb{R}^{n\times n}$, the convergence rate is determined by the two directions corresponding to the semiaxes of the corresponding ellipsoid in n dimensions.

[1] An $n \times n$ symmetric real matrix A is said to be positive definite (respectively, positive semi-definite) if $x^TAx > 0$ (respectively, $x^TAx \geq 0$) for all non-zero x in $\mathbb{R}^n$.

We also note from the results in **Table 1** that for the NEO method proposed in the paper, the identification of the fractional differencing parameter d (using the Whittle estimation procedure outlined in the **Appendix** in the **Supplementary Material**) seems to be the key bottleneck step when it comes to the computation of running times for the method. In practice, however, this issue can be mitigated using a scheduling approach where we use the average of the values of the parameter d obtained after a few time steps and then use that value instead of re-computing d via system identification. This approach is tried for the function $f(x, y) = x^2 + 0.01y^2$ and the results are shown in **Figure 3** (right). Convergence is obtained in 70 iterations with a starting point $x_0 = [1/2\ \sqrt{3}/2]^T$ using ARFIMA$(4, d, 0)$ time series predictions. Additionally, we also present, in **Figure 4** (left, right), respectively, the values of the fractional differencing parameters along the x and y axis as estimated by the Whittle estimation procedure from the functional values for the case of the objective function and algorithm settings detailed in the case of **Figure 3** (left, right).

4.2. Results on the Rosenbrock Function

In this section, we demonstrate the working of the NEO method on the Rosenbrock function (Rosenbrock, 1960), which is a non-convex function used as a test problem for a wide variety of optimization scenarios and algorithms. The non-convexity of the function as well as its global minimum lying in a narrow, flat, parabolic valley makes the minimization of the Rosenbrock function a difficult problem to solve. The function is defined by

$$f(x, y) = (a - x)^2 + b(y - x^2)^2, \tag{14}$$

and has a global minimum at the point (a, a^2), where $f(x, y) = 0$. Hereafter, we consider a commonly used set of parameters, $a = 1$ and $b = 100$ (Shang and Qiu, 2006).

In **Figure 5**, we show the performance of the NEO method in finding the global minimum of the Rosenbrock function. The starting point x_0, denoted by the red asterisk, is chosen to be the point $x_0 = [0.95\ 0.95]^T$. Convergence is obtained to the point $[0.99\ \ 0.97]^T$ in 4 iterations using ARFIMA$(4, d, 0)$ time series predictions. We use $P = 100$ steps of memory and an initial grid discretization step of $h = 0.01$. It is important to note that for the Rosenbrock function, the Hessian often possesses a large condition number, which leads to the poor performance of gradient-descent-like algorithms. Additionally, any optimization method generates a unique dynamics that will be sensitive to the chosen initial point; thus, warm starts are often desirable in the context of non-convex optimization.

4.3. Results on Non-linear Activation Functions

In this section, we evaluate the performance of the NEO method on two non-linear activation functions. Such activation functions are found in neural networks and act as non-linearities in the same. In the same way as integrated circuits receive a multitude of signals and then make a decision regarding whether the output signal will be on or off (as a function of the input),

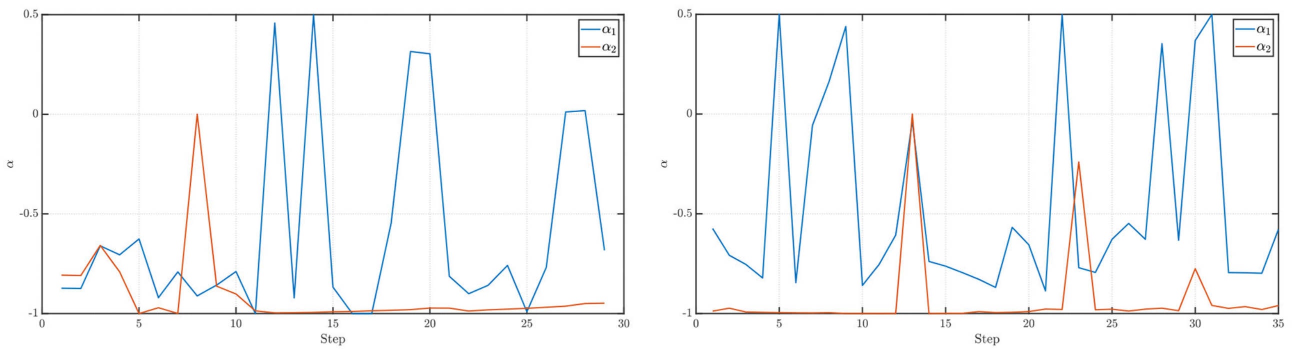


FIGURE 4 | (Left) Values of the fractional differencing parameter α_1 (along the x-axis) and α_2 (along the y-axis) as estimated by the Whittle estimation procedure for the example presented in **Figure 3** (left). **(Right)** Values of the fractional differencing parameter α_1 (along the x-axis) and α_2 (along the y-axis) as estimated by the Whittle estimation procedure for the example presented in **Figure 3** (right).

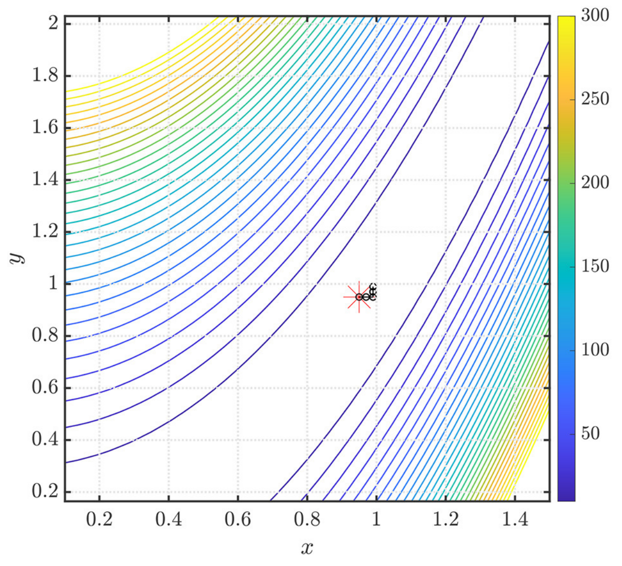


FIGURE 5 | Steps taken in the two-dimensional plane when the NEO method is used to find the global minimum of the Rosenbrock function. The initial point (denoted by the red asterisk) $x_0 = [0.95\ 0.95]^T$. We use ARFIMA(4, d, 0) time series predictions.

activation functions act as a proxy for this behavior in artificial neural networks. However, when neural networks are used, the particular form of the error or loss function used (which in turn contains these non-linear activation functions), often suffers from issues of ill-conditioning (Saarinen et al., 1993; Zhang et al., 1995; van der Smagt and Hirzinger, 2012) due to function optimization landscapes that are often ridden with flat areas and saddle points.

Additionally, if methods such as Newton's method or the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) are used to minimize the loss function, there is additional dependence on the conditioning of the Jacobian or the Hessian matrix evaluated at each of the iterates, and hence, there is considerable loss of performance if either of the Jacobian or the Hessian matrices are ill-conditioned or rank-deficient. In what follows we look into a simple yet meaningful example where non-linear activation functions found in neural networks are minimized using the NEO method.

Assume that we have the activation function

$$\varrho_{\text{GELU}}(s) = \frac{s}{2}\left(1 + \text{erf}\left(\frac{s}{\sqrt{2}}\right)\right), \tag{15}$$

where the *error function* $\text{erf}\, z = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,\text{d}t$ for any $z \in \mathbb{C}$. This activation function is known in the literature as a Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel, 2016). Accordingly, for a set of weights w_1 and w_2 and for a set of inputs x_1 and x_2, we assume that the GELU activation function acts on the weighted sum $w_1x_1 + w_2x_2$ in order to produce

$$\varrho_{\text{GELU}}(w_1x_1 + w_2x_2) = \frac{w_1x_1 + w_2x_2}{2}\left(1 + \text{erf}\left(\frac{w_1x_1 + w_2x_2}{\sqrt{2}}\right)\right). \tag{16}$$

We use the NEO method in order to minimize (16) and find the weights w_1 and w_2 when $x_1 = x_2 = 1$. Using $P = 100$ steps of memory, an initial grid discretization step of $h = 0.01$, $\varepsilon = 10^{-3}$, with $P' = 100$ steps ahead ARFIMA$(4, d, 0)$ time series predictions, we obtain convergence in 23 iterations with the optimal values $w^\star = [-0.5142\ -0.2188]^T$ and $\varrho^\star_{\text{GELU}} = -0.1699$.

Additionally, we also use the NEO method to minimize the activation function

$$\varrho_{\text{SiLU}}(s) = \frac{s}{1 + e^{-s}}, \tag{17}$$

with

$$\varrho_{\text{SiLU}}(w_1x_1 + w_2x_2) = \frac{w_1x_1 + w_2x_2}{1 + e^{-(w_1x_1 + w_2x_2)}}, \tag{18}$$

which is called the Sigmoid Linear Unit (SiLU) (Elfwing et al., 2018) or Swish-1 (Ramachandran et al., 2017). Once again, with $x_1 = x_2 = 1$, $P = 100$ steps of memory, an initial grid discretization step of $h = 0.01$, $\varepsilon = 10^{-3}$, with $P' = 100$ steps ahead ARFIMA$(4, d, 0)$

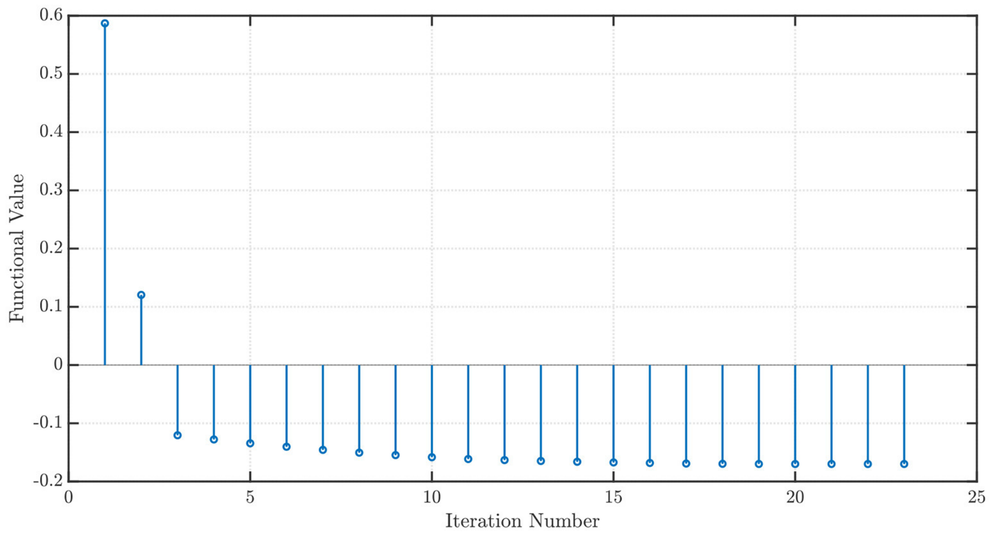


FIGURE 6 | Convergence profile of the NEO method minimizing the Gaussian Error Linear Unit (GELU) activation function.

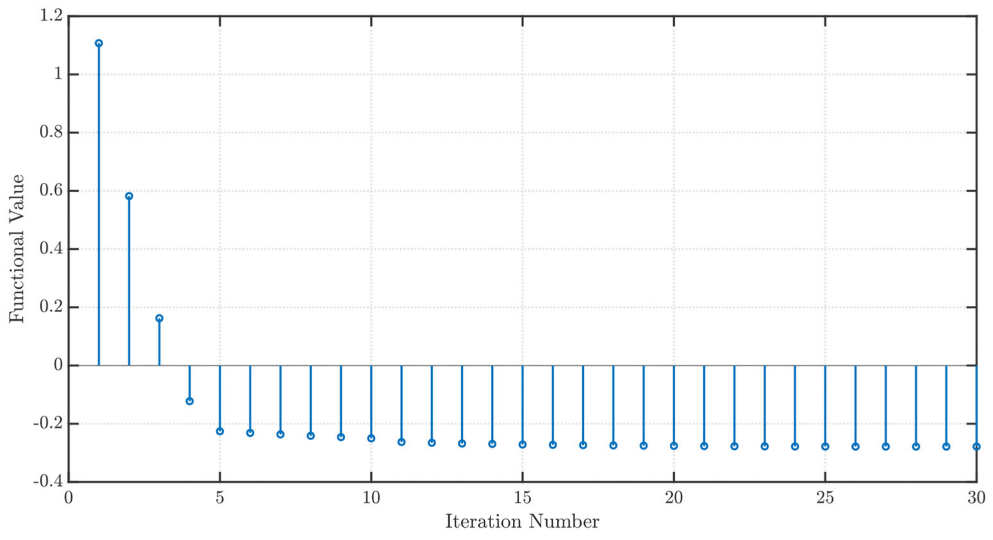


FIGURE 7 | Convergence profile of the NEO method minimizing the Sigmoid Linear Unit (SiLU) activation function.

time series predictions, we obtain convergence in 30 iterations with the optimal values $w^{\star} = [-2.1316 \quad 0.8362]^T$ and $\varrho^{\star}_{\text{SiLU}} = -0.2784$. The convergence profiles when the NEO method is used to minimize the GELU and the SiLU activation functions are shown in **Figures 6**, **7**, respectively.

5. DISCUSSION AND CONCLUSIONS

In this paper, we introduced a NEuro-inspired Optimization (NEO) method, that is motivated by the neurophysiological modeling capabilities possessed by fractional calculus-based ARFIMA time series models to determine an approximation of the argument that minimizes a given unconstrained optimization problem. Our proposed method does not require the computation of gradient or Hessian information, or explicitly tuning a step size, an issue that, in spite of receiving widespread coverage in the optimization literature, still proves to be a bottleneck in the design of such algorithms.

Future work will entail the automated selection of the autoregressive and moving average orders from the functional values at every time step [based on the Akaike Information Criterion (AIC) (Akaike, 1974) or the Bayesian Information Criterion (BIC) (Schwarz, 1978)] and the automation of determining the fractional-order coefficient, which proved to be the most computationally intensive step in our approach, in a wider range of optimization benchmarks.

It is also interesting to note here the connections between descent-like methods such as `NEO` with proximal algorithms that are used to solve non-differentiable optimization problems (Parikh and Boyd, 2014). The latter aims to consider a regularizer with an additional quadratic term to be optimized, which adds smoothness to the optimization problem to be explored. That being said, it is similar to `NEO`, which seeks to explore memory and local changes in the functional values to smooth the predictions. Additionally, it can be shown that under some mild technical assumptions, one can use averaged proximal operators and algorithms in order to convert minimization problems into fixed-point iterations, much like `NEO` looks at minimization problems from the iterative step descent point-of-view. It would, therefore, be interesting to establish the relationship between proximal algorithms and the `NEO` method based on the aforementioned relationships. Lastly, we believe that the validation of our approach in the context of a variety of real-world applications would also be interesting to look into.

DATA AVAILABILITY STATEMENT

The original contributions generated for the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

SC, SD, and SP performed the research. SC was responsible for the execution of the numerical experiments and wrote the manuscript with revisions by SD and SP. All authors contributed to the article and approved the submitted version.

FUNDING

The authors gratefully acknowledge the support by the National Science Foundation under Grant Number CMMI 1936578.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphys.2021.724044/full#supplementary-material

REFERENCES

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Automat. Control* 19, 716–723. doi: 10.1109/TAC.1974.1100705

Baleanu, D., Diethelm, K., Scalas, E., and Trujillo, J. J. (2012). *Fractional Calculus: Models and Numerical Methods*, Vol. 3. Singapore: World Scientific. doi: 10.1142/8180

Baleanu, D., Machado, J. A. T., and Luo, A. C. (2011). *Fractional Dynamics and Control*. New York, NY: Springer Science & Business Media. doi: 10.1007/978-1-4614-0457-6

Bertsekas, D. P. (1997). Nonlinear programming. *J. Operat. Res. Soc*. 48, 334–334. doi: 10.1057/palgrave.jors.2600425

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. Hoboken, NJ: John Wiley & Sons.

Boyd, S., and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge, UK: Cambridge University Press. doi: 10.1017/CBO9780511804441

Bukhari, A. H., Raja, M. A. Z., Sulaiman, M., Islam, S., Shoaib, M., and Kumam, P. (2020). Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting. *IEEE Access* 8, 71326–71338. doi: 10.1109/ACCESS.2020.2985763

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. (2016). A stochastic quasi-Newton method for large-scale optimization. *SIAM J. Optimizat*. 26, 1008–1031. doi: 10.1137/140954362

Cevher, V., Becker, S., and Schmidt, M. (2014). Convex optimization for big data: scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Process. Mag*. 31, 32–43. doi: 10.1109/MSP.2014.2329397

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). "SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, eds Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Montreal, QC: Curran). 1646–1654.

Dennis, J. E. Jr., and Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia, PA: SIAM. doi: 10.1137/1.9781611971200

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw*. 107, 3–11. doi: 10.1016/j.neunet.2017.12.012

Fazlyab, M., Ribeiro, A., Morari, M., and Preciado, V. M. (2018). Analysis of optimization algorithms via integral quadratic constraints: nonstrongly convex problems. *SIAM J. Optimizat*. 28, 2654–2689. doi: 10.1137/17M1136845

Hardt, M., Recht, B., and Singer, Y. (2016). "Train faster, generalize better: stability of stochastic gradient descent," in *Proceedings of the International Conference on Machine Learning* (PMLR), New York, NY, 1225–1234.

Hendrycks, D., and Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Hu, B., and Lessard, L. (2017). "Control interpretations for first-order optimization methods," in *Proceedings of the 2017 American Control Conference* (IEEE), Seattle, WA, 3114–3119. doi: 10.23919/ACC.2017.7963426

Ionescu, C., Lopes, A., Copot, D., Machado, J. T., and Bates, J. H. (2017). The role of fractional calculus in modeling biological phenomena: a review. *Commun. Nonlinear Sci. Num. Simul*. 51, 141–159. doi: 10.1016/j.cnsns.2017.04.001

Johnson, R., and Zhang, T. (2013). "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, eds C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Lake Tahoe, NV: Curran). 315–323.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw*. 3361:1995.

Lessard, L., Recht, B., and Packard, A. (2016). Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM J. Optimizat*. 26, 57–95. doi: 10.1137/15M1009597

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math*. 2, 164–168. doi: 10.1090/qam/10666

Lundstrom, B. N., Higgs, M. H., Spain, W. J., and Fairhall, A. L. (2008). Fractional differentiation by neocortical pyramidal neurons. *Nat. Neurosci*. 11:1335. doi: 10.1038/nn.2212

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indus. Appl. Math*. 11, 431–441. doi: 10.1137/0111030

Miller, K. J., Sorensen, L. B., Ojemann, J. G., and Den Nijs, M. (2009). Power-law scaling in the brain surface electric potential. *PLoS Comput. Biol*. 5:e1000609. doi: 10.1371/journal.pcbi.1000609

Mokhtari, A., Eisen, M., and Ribeiro, A. (2018). IQN: an incremental quasi-Newton method with local superlinear convergence rate. *SIAM J. Optimizat.* 28, 1670–1698. doi: 10.1137/17M1122943

Moritz, P., Nishihara, R., and Jordan, M. (2016). "A linearly-convergent stochastic L-BFGS algorithm," in *Artificial Intelligence and Statistics*, eds A. Gretton, C. C. Robert (Cadiz: Proceedings of Machine Learning Research (PMLR)). 249–258.

Nesterov, Y. (2013). *Introductory Lectures on Convex Optimization: A Basic Course*, Vol. 87, Boston, MA: Springer Science & Business Media.

Nocedal, J., and Wright, S. (2006). *Numerical Optimization*. New York, NY: Springer Science & Business Media.

Oldham, K. B., and Spanier, J. (1974). *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*. Mineola, NY: Elsevier.

Ortigueira, M. D. (2011). *Fractional Calculus for Scientists and Engineers*, Vol. 84, Dordrecht, The Netherlands: Springer Science & Business Media. doi: 10.1007/978-94-007-0747-4

Parikh, N., and Boyd, S. (2014). Proximal algorithms. *Found. Trends Optimizat.* 1, 127–239. doi: 10.1561/9781601987174

Paternain, S., Mokhtari, A., and Ribeiro, A. (2019). A Newton-based method for nonconvex optimization with fast evasion of saddle points. *SIAM J. Optimizat.* 29, 343–368. doi: 10.1137/17M1150116

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* 4, 1–17. doi: 10.1016/0041-5553(64)90137-5

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., et al. (2019). A deep learning framework for neuroscience. *Nat. Neurosci.* 22, 1761–1770. doi: 10.1038/s41593-019-0520-2

Rodomanov, A., and Kropotov, D. (2016). "A superlinearly-convergent proximal Newton-type method for the optimization of finite sums," in *Proceedings of the International Conference on Machine Learning*, New York, NY, 2597–2605.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *Comput. J.* 3, 175–184. doi: 10.1093/comjnl/3.3.175

Roux, N. L., Schmidt, M., and Bach, F. R. (2012). "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Advances in Neural Information Processing Systems*, Roux et al., 2012 - eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Lake Tahoe, NV: Curran). 2663–2671.

Saad, D. (2009). *On-Line Learning in Neural Networks*. Cambridge, UK: Cambridge University Press.

Saarinen, S., Bramley, R., and Cybenko, G. (1993). Ill-conditioning in neural network training problems. *SIAM J. Sci. Comput.* 14, 693–714. doi: 10.1137/0914044

Scholkopf, B., and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT press.

Schraudolph, N. N., Yu, J., and Günter, S. (2007). "A stochastic quasi-Newton method for online convex optimization," in *Artificial Intelligence and Statistics*, eds M. Meila and X. Shen (San Juan, PR: Proceedings of Machine Learning Research (PMLR)) 436–443.

Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Stat.* 6, 461–464. doi: 10.1214/aos/1176344136

Shang, Y.-W., and Qiu, Y.-H. (2006). A note on the extended Rosenbrock function. *Evol. Comput.* 14, 119–126. doi: 10.1162/evco.2006.14.1.119

Sra, S., Nowozin, S., and Wright, S. J. (2012). *Optimization for Machine Learning*. Cambridge, MA, USA: MIT Press. doi: 10.7551/mitpress/8996.001.0001

Su, W., Boyd, S., and Candes, E. (2014). "A differential equation for modeling Nesterov's accelerated gradient method: theory and insights," in *Advances in Neural Information Processing Systems*, eds Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Montreal, QC: Curran). 2510–2518.

Teka, W., Marinov, T. M., and Santamaria, F. (2014). Neuronal spike timing adaptation described with a fractional leaky integrate-and-fire model. *PLoS Comput. Biol.* 10:e1003526. doi: 10.1371/journal.pcbi.1003526

van der Smagt, P., and Hirzinger, G. (2012). "Solving the ill-conditioning in neural network learning," in *Neural Networks: Tricks of the Trade* (Springer), eds G. Montavon, G. B. Orr and K. R. Müller (Berlin: Heidelberg), 191–203. doi: 10.1007/978-3-642-35289-8_13

Whittle, P. (1961). Gaussian estimation in stationary time series. *Bull. Int. Stat. Instit.* 39, 105–129.

Wilson, A. C., Recht, B., and Jordan, M. I. (2016). A Lyapunov analysis of momentum methods in optimization. *Journal of Machine Learning Research*, 22, 1–34.

Wu, S. (1992). Convergence properties of descent methods for unconstrained minimization. *Optimization* 26, 229–237. doi: 10.1080/02331939208843854

Xue, Y., Pequito, S., Coelho, J. R., Bogdan, P., and Pappas, G. J. (2016). "Minimum number of sensors to ensure observability of physiological systems: a case study," in *Proceedings of the 2016 54th Annual Allerton Conference on Communication, Control, and Computing* (IEEE), Monticello, IL, USA, 1181–1188. doi: 10.1109/ALLERTON.2016.7852369

Zhang, A., and Gu, Q. (2016). "Accelerated stochastic block coordinate descent with optimal sampling," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM), San Francisco, CA, USA, 2035–2044. doi: 10.1145/2939672.2939819

Zhang, J., Uribe, C. A., Mokhtari, A., and Jadbabaie, A. (2019). "Achieving acceleration in distributed optimization via direct discretization of the heavy-ball ODE," in *Proceedings of the 2019 American Control Conference* (IEEE), Philadelphia, PA, USA, 3408–3413. doi: 10.23919/ACC.2019.8814686

Zhang, Y., Zhang, Y., and Ye, W. (1995). "Local-sparse connection multilayer networks," in *Proceedings of the 1995 International Conference on Neural Networks* (IEEE), Perth, Australia, 1254–1257.