Mathematical and Computational Applications

MDPI

Article

Using Non-Lipschitz Signum-Based Functions for Distributed Optimization and Machine Learning: Trade-Off Between Convergence Rate and Optimality Gap

Mohammadreza Doostmohammadian [1,*], Amir Ahmad Ghods [1], Alireza Aghasi [2], Zulfiya R. Gabidullina [3] and Hamid R. Rabiee [4]

[1] Faculty of Mechanical Engineering, Semnan University, Semnan 35131-19111, Iran; amir-ghods@semnan.ac.ir
[2] Department of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA; alireza.aghasi@oregonstate.edu
[3] Institute of Computational Mathematics and Information Technologies, Kazan Federal University, Kazan 420008, Russia; zulfiya.gabidullina@kpfu.ru
[4] Computer Engineering Department, Sharif University of Technology, Tehran 15119-43943, Iran; rabiee@sharif.edu
* Correspondence: doost@semnan.ac.ir

Abstract

In recent years, the prevalence of large-scale datasets and the demand for sophisticated learning models have necessitated the development of efficient distributed machine learning (ML) solutions. Convergence speed is a critical factor influencing the practicality and effectiveness of these distributed frameworks. Recently, non-Lipschitz continuous optimization algorithms have been proposed to improve the slow convergence rate of the existing linear solutions. The use of signum-based functions was previously considered in consensus and control literature to reach fast convergence in the prescribed time and also to provide robust algorithms to noisy/outlier data. However, as shown in this work, these algorithms lead to an optimality gap and steady-state residual of the objective function in discrete-time setup. This motivates us to investigate the distributed optimization and ML algorithms in terms of trade-off between convergence rate and optimality gap. In this direction, we specifically consider the distributed regression problem and check its convergence rate by applying both linear and non-Lipschitz signum-based functions. We check our distributed regression approach by extensive simulations. Our results show that although adopting signum-based functions may give faster convergence, it results in large optimality gaps. The findings presented in this paper may contribute to and advance the ongoing discourse of similar distributed algorithms, e.g., for distributed constrained optimization and distributed estimation.

Keywords: linear regression; distributed optimization; network and graph theory; Lipschitz continuity

Academic Editor: Sandra Ferreira

Received: 28 August 2025
Revised: 30 September 2025
Accepted: 3 October 2025
Published: 4 October 2025

Citation: Doostmohammadian, M.; Ghods, A.A.; Aghasi, A.; Gabidullina, Z.R.; Rabiee, H.R. Using Non-Lipschitz Signum-Based Functions for Distributed Optimization and Machine Learning: Trade-Off Between Convergence Rate and Optimality Gap. *Math. Comput. Appl.* **2025**, *30*, 108. https://doi.org/10.3390/mca30050108

1. Introduction

Distributed algorithms for detection, estimation machine learning, and resource allocation have recently gained interest in signal processing, control, and optimization literature [1–4]. Such algorithms are known to have many benefits in terms of scalability, real-time and parallel data processing, and distributed learning over multi-agent networks, with specific applications in data mining [5,6]. To give more details, by distributing the computational load across multiple nodes or devices, these algorithms allow for the processing

of large datasets and complex tasks without overburdening a single central processing unit [7]. This is also motivated by recent cloud-based solutions for high-performance computing, which are well suited for computationally intensive tasks involved in distributed algorithms. This is especially beneficial in machine learning and simulation-based algorithms where large datasets and complex computations are common. Such distributed algorithms are further motivated by recent advances in Internet of Things (IoT) applications [8], cloud/edge computing [9], and cyber-physical systems (CPS) [10,11] with many processing devices interconnected over a network. In general, distributed optimization algorithms benefit from *robustness* to single-node failure, *parallelism* by assigning different parts of the optimization process to different computing nodes/agents, *scalability* (and resource utilization) by distributing the workload for large-scale optimization problems, and *data distribution* without the need for centralized storage. In this work, we investigate the use of non-Lipschitz consensus-based functions for distributed optimization in terms of convergence rate and steady-state optimization residual (optimality gap).

1.1. Literature Review

Signum-based functions can be robust to outliers in the data. In the presence of noisy or outlier data, algorithms based on the signum function may be more resilient compared to methods that rely on smooth functions [12,13]. In the consensus literature these algorithms are used to reach fast agreement over the multi-agent network in prescribed time [14], finite time [15,16], and fixed time [17,18]. For a similar reason, signum-based functions are used for distributed resource allocation [19], event-triggered systems [20], distributed gradient flow schemes [21], sliding mode control [22], parameter identification [23], distributed optimization [24], cooperative control [25,26], and distributed estimation [13]. A survey of finite-/fixed-time convergent algorithms can be found in [27].

In the context of distributed optimization, ML, and regression solutions, most existing literature provides linear algorithms [28–30], many of which are consensus-based solutions [31,32]. Recently distributed non-Lipschitz algorithms are proposed in the context of distributed optimization and learning, claiming improved convergence rate (stability in finite time, fixed time, or prescribed time) [33–41]. Although in continuous time these may work properly, the discrete-time dynamics (after discretization) results in steady-state oscillation around the optimal point, known as *chattering* phenomena. This is well known in nonlinear control applications, e.g., in sliding mode control [42]. For distributed optimization, this results in the final residual of the objective function and optimality gap. In this direction, the current study investigates the trade-off between the improved convergence rate and the optimization steady-state residual.

The use of non-Lipschitz function also includes non-Lipschitz activation functions in neural networks [43,44] and non-Lipschitz optimization in related fields such as deep learning [45,46] by designing algorithms that exploit structure, e.g., sign/threshold dynamics and proximal maps for nonconvex nonsmooth terms. In neural networks, while mainly Lipschitz activations (ReLU, ELU, GELU) are adopted, classical examples include the signum function and hard-threshold activations.

1.2. Contributions

This study investigates the distributed optimization algorithms with signum-based functions added to improve their convergence rate. We consider a gradient tracking (GT) distributed optimization algorithm which is based on consensus algorithm [47]. The idea is to improve the rate of convergence by adding non-Lipschitz sign-based functions, while checking the steady-state residual (the optimality gap). The adopted signum-based functions are sign preserving and odd; therefore, they do not violate the consensus-type

nature of the algorithm. Further, the GT-based dynamics ensures evolution of the states toward the optimal point. We specifically show that the optimal point is the invariant state under the proposed dynamics. As an ML application, we consider distributed linear regression over a randomly generated dataset. Our results show that, although sign-based nonlinearity may affect the convergence rate (and reach fixed-time, finite-time, or prescribed-time convergence), it may cause steady-state residual in the cost and certain optimality gap depending on the parameters of the signum-based function.

1.3. Paper Organization

Section 2 states the preliminaries. Section 3 frames the distributed linear regression as a distributed optimization problem. Section 4 presents the linear and signum-based GT solutions. Section 5 provides the simulation, and Section 6 concludes the paper.

2. Preliminaries

2.1. Notations

Let λ denote the eigenvalue. The abbreviations LHP and RHP correspond to left-half plane and right-half plane in the complex eigenspace. Let $\partial_t z = \frac{dz}{dt}$ be the derivative with respect to t. $\mathbf{1}_n^\top \coloneqq \underbrace{(1, \cdots, 1)}_{n}$ $\mathbf{0}_n^\top \coloneqq \underbrace{(0, \cdots, 0)}_{n}$, i.e., $\mathbf{1}_n$ and $\mathbf{0}_n$ are size n vectors of all 1s and 0s. The operator ';' implies column concatenation of vectors. ∇F is the gradient of F.

2.2. Algebraic Graph Theory

The distributed algorithm works over a connected undirected multi-agent network represented by an undirected graph topology $\mathcal{G}$ with real adjacency matrix $W = [w_{ij}] \in \mathbb{R}^{n\times n}$. The entry $w_{ij} > 0$ associated with the link $j \to i$ denotes the weighting factor, which defines the weight that agent i assigns to the information received from agent j. For a connected undirected $\mathcal{G}$, its associated matrix W is irreducible. Further, define the Laplacian matrix $\overline{W} = [\overline{w}_{ij}] \in \mathbb{R}^{n\times n}$ as $\overline{w}_{ij} = w_{ij}$ for $i \neq j$ and $\overline{w}_{ij} = -\sum_{i=1}^{n} w_{ij}$ for $i = j$. It is known that the connectivity of the graph is related to the rank of its Laplacian matrix. Given a connected undirected graph $\mathcal{G}$, its Laplacian $\overline{W}$ has only one zero eigenvalue and the rest are on LHP. The (left and right) eigenvectors $\mathbf{1}_n^\top$ and $\mathbf{1}_n$ are associated with these zero eigenvalues, i.e., $\mathbf{1}_n^\top \overline{W} = \mathbf{0}_n$ and $\overline{W}\mathbf{1}_n = \mathbf{0}_n$ [47].

2.3. Background on Signum-Based Consensus

Consensus algorithms are widely used to coordinate (reach agreement) over multi-agent networks. The primary solution to solve consensus in a distributed way is to follow a linear dynamics, where the dynamics at node i is as follows [47]:

$$\dot{x}_i = -\eta_1 \sum_{j\in N_i} w_{ij}(x_i - x_j), \tag{1}$$

where $\eta_1 > 0$ as the step-rate and $W = [w_{ij}]$ as the stochastic adjacency weight matrix (a matrix is called stochastic if for every $i \in \{1, \ldots, n\}$ we have $\sum_{j=1}^{n} w_{ij} = \sum_{j=1}^{n} w_{ji} = 1$.), N_i denotes the neighboring set of node/agent i, and x_i, x_j denote the state values at nodes i, j. It is known that, under certain conditions, the solution of this dynamics asymptotically converges to the agreement/consensus state. On the other hand, finite-time consensus protocols [16,48] improve the convergence rate of the linear dynamics (1) in the region $|x_i - x_j| < 1$ by adding signum-based function as follows:

$$\dot{x}_i = -\eta_1 \sum_{j\in N_i} w_{ij} sgn^{v_1}(x_i - x_j), \tag{2}$$

where $0 < v_1 < 1$ and the signum-based function $sgn^{v_1}(x) : \mathbb{R} \to \mathbb{R}$ is

$$sgn^{v_1}(x) = x\,|x|^{v_1-1}, \tag{3}$$

recall that this function is non-Lipschitz at $x = 0$. As it is proved in finite-time consensus literature [16,48], this solution converges faster than linear dynamics (1) in the regions close to the equilibrium. This follows from the definition of signum-based function and the fact that $|sgn^v(x)|>|x|$ for all $|x| < 1$. Moreover, the non-Lipschitz continuity and infinite gradient at the agreement equilibrium allow to reach consensus in finite time. However, these solutions converge slower than linear dynamics (1) in the region farther from zero (or the agreement equilibrium) as $|sgn^v(x)|<|x|$ for all $|x|> 1$. Fixed-time consensus protocols [49–53] overcome this by adding a second term in the form $sgn^{v_2}(x)$ with $v_2 > 1$. For this function we have $|sgn^{v_2}(x)| > |x|$ for $|x|> 1$; this implies a faster convergence rate than the linear dynamics for states farther from the agreement equilibrium. By combining the two consensus dynamics, the solution has a fast convergence rate for all the regions, both close and far from the equilibrium. This convergence rate can be changed via the parameters v_1, v_2. The overall fixed-time consensus dynamics is in the following form:

$$\dot{x}_i = -\sum_{j\in N_i} w_{ij}\left(\eta_1 sgn^{v_1}(x_i - x_j) + \eta_2 sgn^{v_2}(x_i - x_j)\right), \tag{4}$$

with $0 < v_1 < 1$, $v_2 > 1$, η_2, $\eta_1 > 0$. The convergence rate of the dynamics (4) is faster than protocols (1) and (2). It should be mentioned that these dynamics are *non-Lipschitz* (a function $f(x) : x \in \mathbb{R}$ is called Lipschitz continuous if there exists a constant $\mathcal{K}$ such that for every two points x_1, x_2 we have, $|f(x_1) - f(x_2)| \leq \mathcal{K}|x_1 - x_2|$. Otherwise, the function is called non-Lipschitz.). Therefore, they may result in chattering around the equilibrium as stated in sliding-mode control literature [42].

3. The Framework: Distributed Regression Problem

Linear regression is a statistical method used to fit a linear line to a set of data points. Given a set of N data points $\chi_i \in \mathbb{R}^{m-1}$, $i = \{1, \ldots, n\}$, the model's prediction is $\beta^T\chi_i - \nu = y_i$, which gives the hyperplane that fits the data best. In the centralized regression, all the data points are sent to a central computation entity (the fusion center) which finds $[\nu;\ \beta]$ optimizing the following quadratic convex function:

$$\min_{[\nu;\beta]} \sum_{i=1}^{N} \left(\beta^T\chi_i - \nu - y_i\right)^2, \tag{5}$$

which is also known as the linear least square problem. Distributed linear regression (DLR) is an extension of linear regression that leverages distributed computing resources for handling large datasets. In traditional linear regression, all data are typically processed on a single machine (the fusion center), which can become impractical when dealing with massive datasets that may not fit into the memory of a single computer. Distributed linear regression distributes the computation across multiple machines or nodes in a computing cluster. For parallel processing of data, this approach allows one to make handling large-scale datasets more feasible (and more efficient). In DLR, the dataset is widespread over a network of n agents/machines, and each machine i has its own $\frac{N}{n} \leq N_i \leq N$ data points χ^i, where some of these data might be shared between two or more machines. The main idea is to solve the optimization problem (5) locally at each machine using its own data χ^i and information received from its neighboring machines. Note that every machine has access to partial data and thus the optimal values β_i and ν_i may differ for each machine i. Therefore, the machines share necessary information by reaching a consensus on β and ν. Then, the optimization problem changes to:

$$\begin{aligned} &\min_{\beta_1,\nu_1,\ldots,\beta_n,\nu_n} \sum_{i=1}^{n} f_i(\beta_i,\ \nu_i), \\ &\textit{subjectto } \beta_1 = \cdots = \beta_n,\ \ \nu_1 = \cdots = \nu_n \end{aligned} \tag{6}$$

$$f_i(\beta_i,\ \nu_i) = \sum_{j=1}^{N_i} \left(\beta_i^T \chi_j^i - \nu_i - y_j\right)^2, \tag{7}$$

This problem (6)–(7) represents a consensus-constrained distributed optimization framework. By denoting the optimization state variable as vector $x_i = \left[\beta_i^T; \nu_i\right]$ and vector x be the concatenation of all local state vectors x_i's, i.e., $x = [x_1; x_2; \ldots; x_n] \in \mathbb{R}^{mn}$. Then, this DLR problem (6)–(7) is framed as a distributed optimization formulation:

$$\begin{aligned} &\min_{x}\ F(x) = \sum_{i=1}^{n} f_i(x_i), \\ &\textit{subjectto } x_1 = x_2 = \cdots = x_n \end{aligned} \tag{8}$$

4. The Proposed Signum-Based Learning Dynamics

4.1. The Algorithm

First, we recall the existing linear gradient tracking dynamics to solve the DLR asymptotically. The following linear dynamics is proposed by the author in [7] to solve distributed optimization and support vector machine problems:

$$\dot{x}_i = -\sum_{j=1}^{n} w_{ij}(x_i - x_j) - \alpha y_i, \tag{9}$$

$$\dot{y}_i = -\sum_{j=1}^{n} a_{ij}(y_i - y_j) - \partial_t \nabla f_i(x_i), \tag{10}$$

with $x_i(t)$ and $y_i(t)$, respectively, denoting the state and the auxiliary variable at agent i at time t. The auxiliary variable $y_i(t)$ tracks and accumulates the sum of the local gradients. The matrices $A = \left[a_{ij}\right] \in \mathbb{R}^{n\times n}$ and $W = \left[w_{ij}\right] \in \mathbb{R}^{n\times n}$ are the adjacency weight matrices associated with the state x and auxiliary variable y. For convex objective functions, the convergence rate of the dynamics (9)–(10) toward the optimal point is $\mathcal{O}(exp(\lambda_2 t))$ with λ_2 as the smallest nonzero eigenvalue of the Laplacian matrix associated with matrices A and W (also known as the algebraic connectivity) [54]. Recalling the application of signum-based functions from Section 2.3, the convergence rate can be improved by using signum-based functions, and the accelerated version of the linear dynamics (9)–(10) is in the following form:

$$\dot{x}_i = -\sum_{j=1}^{n} w_{ij}\left(sgn^{v_1}(x_i - x_j) + sgn^{v_2}(x_i - x_j)\right) - \alpha y_i, \tag{11}$$

$$\dot{y}_i = -\sum_{j=1}^{n} a_{ij}\left(sgn^{v_1}(y_i - y_j) + sgn^{v_2}(y_i - y_j)\right) + sgn^{u_1}(\partial_t \nabla f_i(x_i)) + sgn^{u_2}(\partial_t \nabla f_i(x_i)), \tag{12}$$

with (note that for $u_1 = 1$, $u_2 = 1$, $v_1 = 1$, $v_2 = 1$, the nonlinear algorithm changes to the linear algorithm). $0 < u_1 < 1$, $u_2 > 1$, $0 < v_1 < 1$, $v_2 > 1$. The fast convergence of ML dynamics (11)–(12) includes one step of consensus on the states and one step of gradient tracking update. Note that the nonlinear signum function $sgn^{v_1}(\cdot)$ is odd, sign preserving, and monotonically increasing; therefore, the stability properties hold similar to the linear case [7]. These properties of $sgn^{v_1}(\cdot)$ function also ensure that:

$$\sum_{i=1}^{n} \dot{y}_i = \sum_{i=1}^{n} sgn^{u_1}(\partial_t \nabla f_i(x_i)) + sgn^{u_2}(\partial_t \nabla f_i(x_i)), \tag{13}$$

$$\sum_{i=1}^{n} \dot{x}_i = -\alpha \sum_{i=1}^{n} y_i, \tag{14}$$

these follow the stochastic property of the matrices W and A and the existing consensus-based distributed algorithms (see [47,55] as an example) saying that,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(sgn^{v_1}(x_i - x_j) + sgn^{v_2}(x_i - x_j)\right) = 0, \tag{15}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(sgn^{v_1}(y_i - y_j) + sgn^{v_2}(y_i - y_j)\right) = 0,$$

by initializing as $y(0) = 0$ it is straightforward to see that $\sum_{i=1}^{n} y_i$ tracks a nonlinear signum-based function of $-\sum_{i=1}^{n} \nabla f_i(x_i)$. This implies that the time derivative of $\sum_{i=1}^{n} x_i$ tracks a function of the accumulated gradient at all nodes over the network. This follows from the proposed nonlinear structure of (11)–(12) and can be extended to any form of odd sign-preserving model nonlinearities (e.g., quantization or saturation) while preserving the GT dynamics. This is in contrast to the existing linear alternating direction method of multipliers (ADMM) dynamics that does not allow one to consider model nonlinearity. In other words, the existing ADMM solutions [56–59] cannot directly address typical real-world model nonlinearity (such as saturation and quantization), while our proposed dynamics (11)–(12) can address such models. Our distributed solution is summarized in Algorithm 1. For the algorithm initialization, set the x states randomly such that $x(0) \notin \text{span}\{\mathbf{1}_n \otimes \varphi\}$ (with $\varphi \in \mathbb{R}^m$ and $\otimes$ as Kronecker network product) and set $y(0) = \mathbf{0}_{nm}$. From the strict convexity of the DLR objective function $F(x)$ one can see that at the optimal point $x = x^* = \mathbf{1}_n \otimes \overline{x}^*$(i.e., $x_i = \overline{x}^*$) the equilibrium uniquely holds as,

$$\sum_{i=1}^{n} \dot{x}_i = -\alpha\left(\mathbf{1}_n^\top \otimes I_m\right)\left(sgn^{u_1}(\nabla F(x^*)) + sgn^{u_2}(\nabla F(x^*))\right) = \mathbf{0}_m, \tag{16}$$

which follows from the oddness of the signum function. Similarly, from the proposed dynamics one can see that $\mathrm{x_i} = \mathrm{x_j} = \overline{x}^*$ and the gradient tracking term $y_i = \mathbf{0}_m$ at the optimal point; thus, we have $\dot{x}_i = \mathbf{0}_m$ and,

$$\begin{aligned} \dot{y}_i &= sgn^{u_1}(\partial_t \nabla f_i(\overline{x}^*)) + sgn^{u_2}(\partial_t \nabla f_i(\overline{x}^*)) \\ &= sgn^{u_1}\left(\nabla^2 f_i(\overline{x}^*)\dot{x}_i\right) + sgn^{u_2}\left(\nabla^2 f_i(\overline{x}^*)\dot{x}_i\right) = \mathbf{0}_m, \end{aligned} \tag{17}$$

the above imply that $[x^*; \mathbf{0}_{nm}]$ satisfying $\left(\mathbf{1}_n^\top \otimes I_m\right)\nabla F(x^*) = \mathbf{0}_m$ is invariant (and stable) equilibrium state of (11)–(12) for continuous time dynamics; thus, any randomly initialized solution of x_i with $y_i(0) = 0$ converges to the optimizer $\overline{x}^*$.

Table 1 summarizes the effect of the parameters u_1, u_2, v_1, v_2 on the convergence rate of the proposed signum-based dynamics (11)–(12).

Algorithm 1. GT-based distributed ML algorithm

Data: Undirected graph topology $\mathcal{G}$, adjacency matrices A, W, loss function f_i
Result: Optimal state x^*
Initialization: $t = 0$, $y_i(0) = 0$ at all nodes and states $x_i(0)$ randomly initialized;
While *termination criteria NOT hold;*
do
Node i receives x_j and y_j from neighbor nodes $j \in N_i$ over $\mathcal{G}$;
Node i calculates $\nabla f_i(x_i)$ (the gradient of local loss function $f_i(x_i)$);
Node i updates x_i and y_i via dynamics (11)-(12);
Node i shares updated x_i and y_i with its neighbor nodes $i \in N_j$ over $\mathcal{G}$;

Table 1. The change in the parameters of signum-based nonlinearity to reach faster convergence.

Parameter	Faster Convergence
$0 < v_1 < 1$	smaller $\to 0$
$v_2 > 1$	larger $\to \infty$
$0 < u_1 < 1$	smaller $\to 0$
$u_2 > 1$	larger $\to \infty$

The discretized version of the continuous time dynamics (11)–(12) can be represented as follows:

$$x_i(k+1) = x_i(k) - \eta \sum_{j=1}^{n} w_{ij}\big(sgn^{v_1}\big(x_i(k) - x_j(k)\big) + sgn^{v_2}\big(x_i(k) - x_j(k)\big)\big) - \alpha y_i(k), \quad (18)$$

$$\begin{aligned} y_i(k+1) &= y_i(k) - \eta \sum_{j=1}^{n} a_{ij}\big(sgn^{v_1}\big(y_i(k) - y_j(k)\big) + sgn^{v_2}\big(y_i(k) - y_j(k)\big)\big) \\ &+\eta sgn^{u_1}(\nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k))) + \eta sgn^{u_2}(\nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k))), \end{aligned} \quad (19)$$

with η as the discretization step rate and $k > 0$ as the discrete step time.

It is worth mentioning that applying the discretized version (18)–(19) may result in certain optimality gap because of the chattering phenomena. This refers to rapid and erratic oscillations in the system variables during the optimization process since the dynamics lacks Lipschitz continuity. This means that the gradients of the objective function can vary widely across different regions.

The discretization step size plays a key role in worsening or mitigating chattering phenomena. The step size determines how large or small the state updates we have at each iteration of the optimization algorithm are. If the step size is too large, the optimization algorithm may overshoot around the optimal solution, causing larger oscillations (and optimality gap). On the other hand, if the step size is too small, the algorithm may converge very slowly. Thus, there is a trade-off in terms of convergence rate and steady-state optimization residual.

Table 2 summarizes the effect of the parameters u_1, u_2, v_1, v_2 on the optimality gap of the discretized dynamics (18)–(19).

Table 2. The change in parameters of signum-based nonlinearity to reach lower optimality gap.

Parameter	Faster Convergence
$0 < v_1 < 1$	larger $\to 1$
$v_2 > 1$	smaller $\to 1$
$0 < u_1 < 1$	larger $\to 1$
$u_2 > 1$	smaller $\to 1$

To reduce the optimality gap introduced by non-Lipschitz signum-based update rules (18)–(19), we propose replacing a fixed step size with a diminishing step size sequence. Diminishing step sizes (e.g., $\eta_k = \frac{\eta_0}{k+1}$ or $\eta_k = \frac{\eta_0}{\sqrt{k+1}}\Big)$ balance the following two competing needs: early iterations require sufficiently large updates to exploit fast transient convergence by the signum dynamics, while later iterations require progressively smaller updates to attenuate persistent bias and oscillations caused by the non-Lipschitz terms. Formally, a diminishing sequence that is positive, nonincreasing, and satisfies $\sum_{k=0}^{\infty} \eta_k = \infty$, $\sum_{k=0}^{\infty} \eta_k^2 < \infty$ preserves asymptotic convergence. In our context, this yields to vanishing optimality gap as $k \to \infty$, while on the other hand it slows the convergence. Therefore, diminishing step sizes also provide a trade-off finite time convergence rate for improved asymptotic optimality in signum-based distributed algorithms. This is better demonstrated later by the simulations in Section 5.1.

4.2. Practical Implementations and Applications

Discretizing continuous time signum-based control laws for distributed optimization introduces several implementation challenges, mainly on stability and communication constraints. When the ideal continuous dynamics (11)–(12) is approximated with a discrete time update, the non-Lipschitz nature of signum-based terms causes sensitivity to sampling and numerical quantization. Step size selection in the discretized dynamics (18)–(19) trades off convergence rate against the achievable optimality gap, i.e., larger step sizes can accelerate transient progress but increases discretization error and steady-state bias introduced by the non-Lipschitz terms and any smoothing used to avoid chattering. In particular, for signum-like updates the discretization error does not necessarily vanish with time unless the step size is reduced; this implies that a fixed step size gives a persistent optimality gap proportional to the step magnitude. On the other hand, diminishing step sizes may reduce the gap asymptotically but give slow convergence and add more complexity to the coordination of distributed agents.

The proposed signum-based distributed optimization algorithm may offer practical advantages in decentralized control and multi-agent systems where fast and robust convergence among many agents is of interest. In large-scale distributed control systems signum-like coupling may provide finite time or very fast convergence that help the network quickly reach consensus or track a reference despite the disturbances [60]. The non-Lipschitz nature allows stronger corrective property near disagreement regions to reduce transient errors and also improves tolerance against impulsive noise/faults. However, the trade-off between convergence speed and steady-state residual must be managed: very aggressive signum terms as in Table 1 can drive the system rapidly but introduce a non-vanishing steady-state error or chattering.

In financial data analysis and distributed machine learning on market data, signum-like functions can be used to design decentralized and robust aggregation rules, e.g., in federated learning updates resilient to outliers and deep learning models resilient to low signal-to-noise ratios [61]. For instance, signum-based penalties or gradient modifications give higher weight to correcting large discrepancies among local models or estimates as in financial chaotic systems [62]. In fact, the same non-Lipschitz behavior that accelerates convergence may prevent reaching the absolute optimal parameter set or introduce oscillations around it; therefore, hybrid designs (tempered signum terms or decaying/diminishing gains) are proposed for financial data analysis.

5. Simulations

5.1. Academic Example

For MATLAB (R2022) simulation over a Core i5 Laptop, we consider a (randomly generated) dataset of $N = 100$ data points and a network of $n = 10$ agents each having access to 50% of the (randomly chosen) data points. The dataset is shown in Figure 1. Each agent performs local regression analysis on its own batch of data and shares the regressor parameters over an Erdos–Renyi (ER) random network. The linking probability of the connected ER network is 30%. The objective function to be optimized is in the form (6)–(7). We compare the convergence under the nonlinear dynamics with different signum-based models (18)–(19) (as the discrete version of (11)–(12)) with $\eta = 2 \times 10^{-6}$ and $\alpha = 4$.) Following Algorithm 1, agents update their regressor parameters based on the proposed dynamics and share their states x_i and y_i over the network. We consider four scenarios for comparison of the convergence rate and the optimality gap.

- Case (i): $u_1 = 1,\ u_2 = 1,\ 0 < v_1 = 0.5 < 1,\ v_2 = 1$;
- Case (ii): $u_1 = 1,\ u_2 = 1,\ 0 < v_1 = 0.5 < 1,\ v_2 = 1.5 > 1$;
- Case (iii): $0 < u_1 = 0.6 < 1,\ u_2 = 1,\ 0 < v_1 = 0.5 < 1,\ v_2 = 1.5 > 1$;

- Case (iv): $0 < u_1 = 0.6 < 1,\ u_2 = 1.4 > 1,\ 0 < v_1 = 0.5 < 1,\ v_2 = 1.5 > 1$;
- The time evolution of the cost functions is compared in Figure 2. As it can be seen from the figure, although as claimed in the literature the signum-based dynamics may result in finite time stability, it results in steady-state optimization residual (depending on the parameters u_1, u_2, v_1, v_2).
- The parameters of the regressor at different agents under different dynamics are shown in Figures 3–6. The regressor line parameters β_i, ν_i are calculated at every node/agent i. In these figures, different colors show the parameters associated with different computing nodes/agents. As it can be seen, applying sign function may result in inexact convergence and steady-state error, especially when adding it to the gradient tracking part of the dynamics (as shown in Figures 5 and 6).
- Next, to strengthen the generalizability of the results, we redo the simulations for large-scale example with $N = 1000$ data points and a network of $n = 100$ agents each having access to 35% of the (randomly chosen) data points. The large dataset is shown in Figure 7. We redo the simulation on local regression with the objective function (6)–(7) over an ER random network with 20% linking probability. For this simulation, we set different values for step size as $\eta = 1 \times 10^{-5}$ and gradient tracking rate as $\alpha = 2$. Note that large step sizes, although they may lead to faster convergence, may result in larger optimality gap of the signum-based dynamics. In case of very large step size the solution may diverge. We compare the optimality gap under four different signum-based models based on (18)–(19) as,
- Case (1): $u_1 = 1,\ u_2 = 1,\ 0 < v_1 = 0.3 < 1,\ v_2 = 1$;
- Case (2): $u_1 = 1,\ u_2 = 1,\ 0 < v_1 = 0.3 < 1,\ v_2 = 2 > 1$;
- Case (3): $0 < u_1 = 0.8 < 1,\ u_2 = 1,\ 0 < v_1 = 0.3 < 1,\ v_2 = 2 > 1$;
- Case (4): $0 < u_1 = 0.8 < 1,\ u_2 = 1.2 > 1,\ 0 < v_1 = 0.3 < 1,\ v_2 = 2 > 1$;
- The residual cost function over iteration k is compared in Figure 8. Despite the finite time convergence, the solution may result in steady-state residual or optimality gap depending on the parameters u_1, u_2, v_1, v_2. To better highlight this optimality gap on the regressor line parameters β_i, ν_i, these parameters are shown in Figures 9–12 for different cases under signum-based dynamics. It is clear that applying sign function may result in steady-state error, especially for $u_1 \neq 1,\ u_2 \neq 1$ where the gradient tracking is under signum-based nonlinearity (e.g., see Figures 11 and 12).
- Next, we repeat the simulation to compare the optimality gap under fixed and diminishing step sizes η. For fixed step size we consider $\eta = 5 \times 10^{-6}$ and for diminishing step size $\eta = \frac{5 \times 10^{-5}}{\sqrt{k+1}}$. The signum function parameters are set as $u_1 = 1,\ u_2 = 1,\ 0 < v_1 = 0.75 < 1,\ v_2 = 1.25 > 1$. As we see from Figure 13, the optimality gap is smaller for diminishing step size as compared with the fixed step size, while converging in slower rate. This is one remedy to decrease the optimality gap while it causes slower convergence rate.

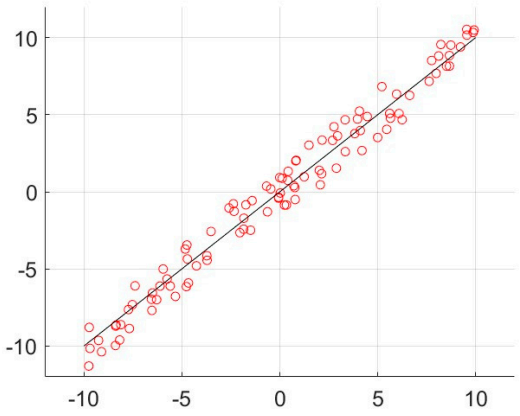


Figure 1. This figure shows the randomly generated dataset and its associated regressor line to be calculated by agents in a distributed way.

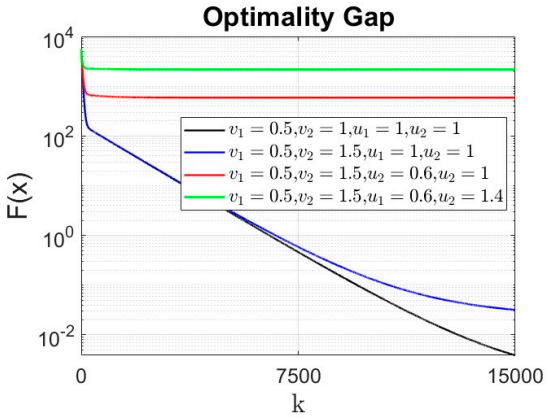
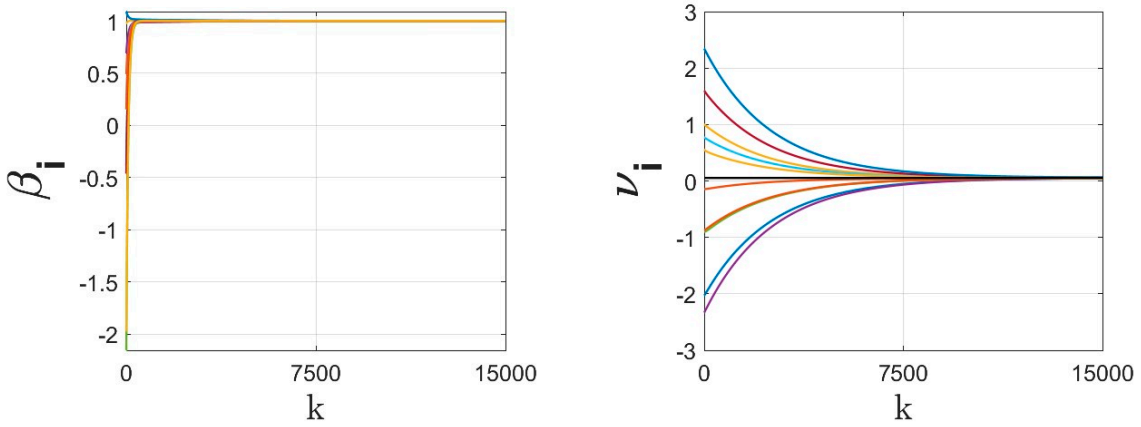
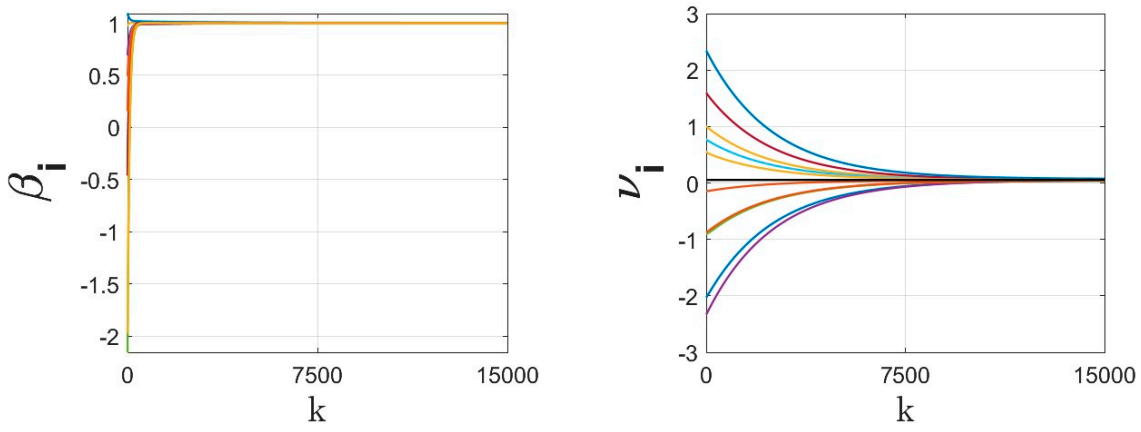


Figure 2. This figure compares the time evolution of DLR objective residual under different signum-based dynamics. Evidently, the discrete time dynamics under signum-based function leads to steady-state residual.



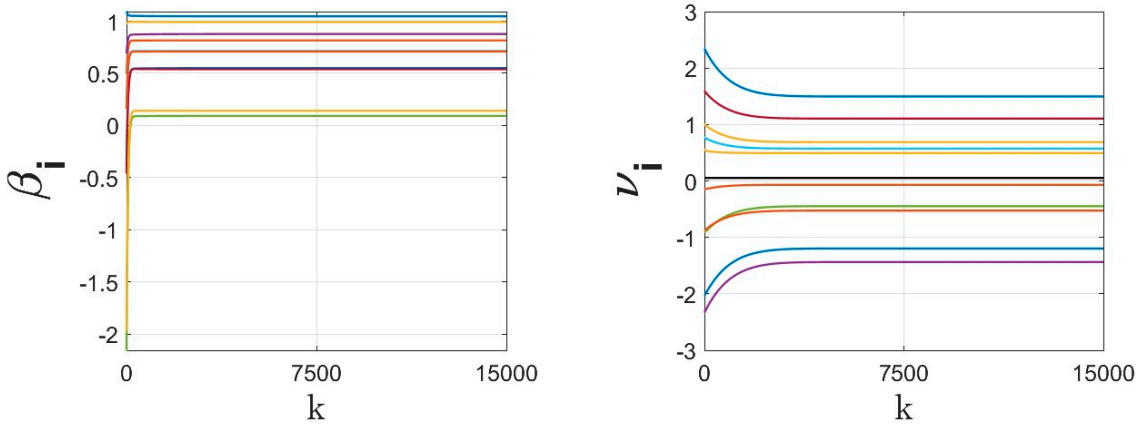
(**a**) (**b**)

Figure 3. This figure shows the time evolution of the regressor parameters under Case (i).



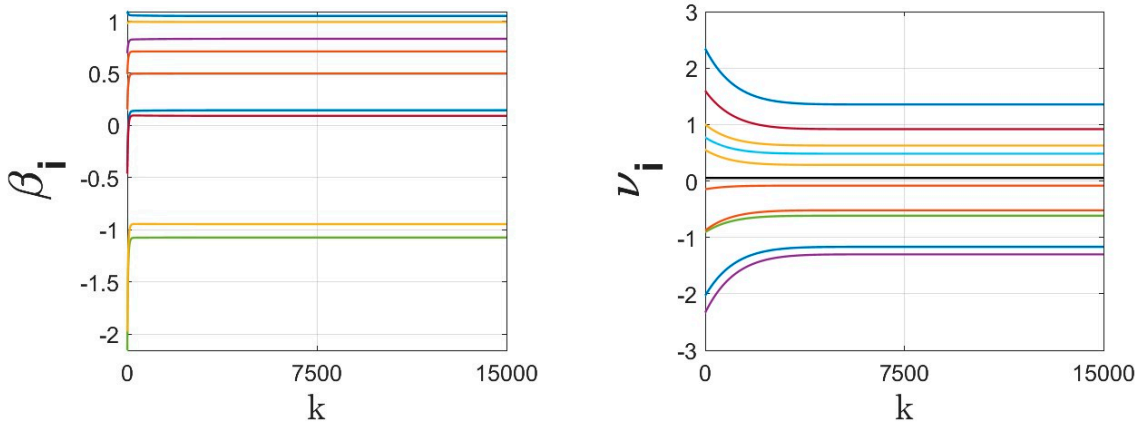
(**a**) (**b**)

Figure 4. This figure shows the time evolution of the regressor parameters under Case (ii).



(**a**) (**b**)

Figure 5. This figure shows the time evolution of the regressor parameters under Case (iii).



(**a**) (**b**)

Figure 6. This figure shows the time evolution of the regressor parameters under Case (iv).

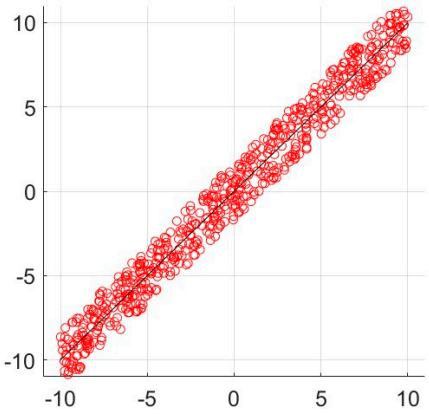


Figure 7. This figure shows the randomly generated data points with the associated regressor line for the large-scale distributed optimization simulation.

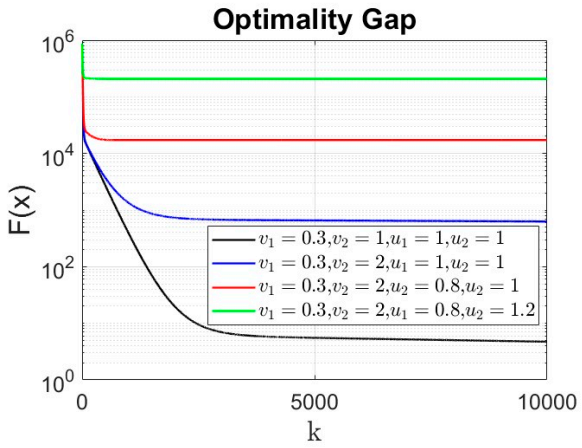
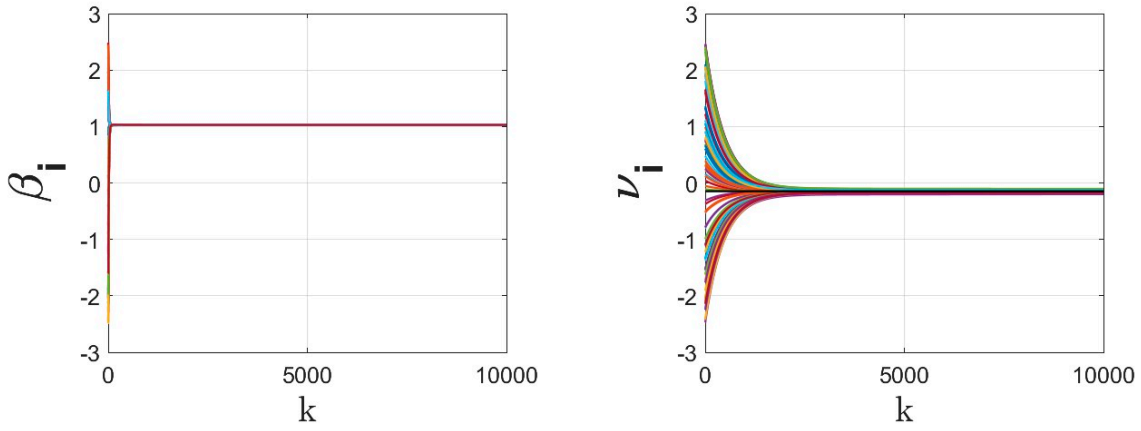
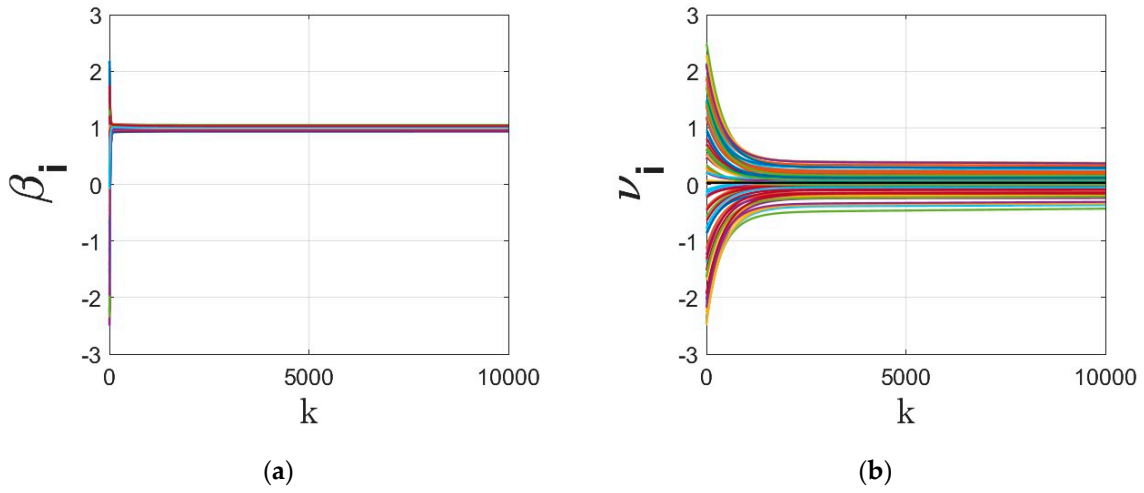


Figure 8. This figure compares the DLR optimality gap under different signum-based dynamics where signum-based solution may result in some optimality gap.



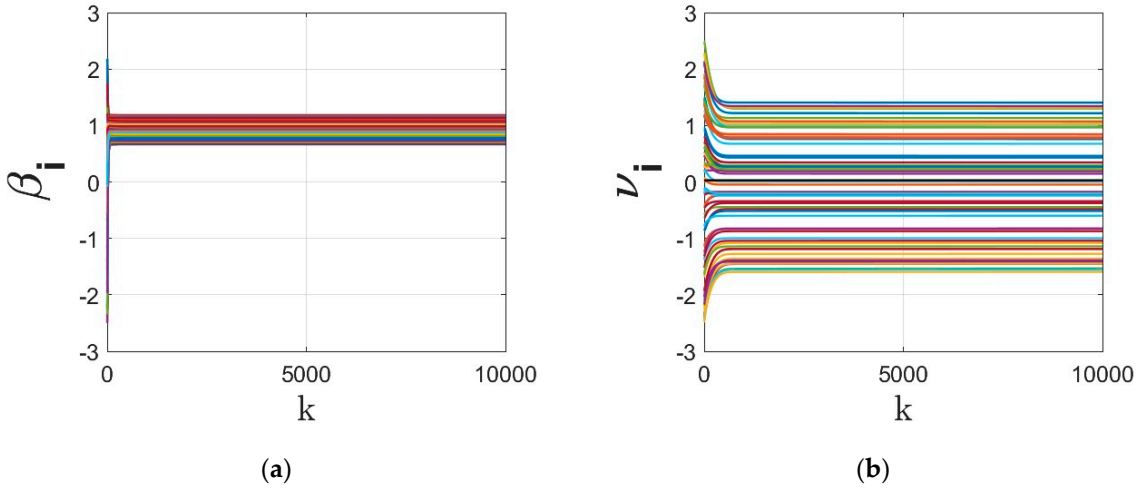
(**a**) (**b**)

Figure 9. This figure shows the time evolution of the regressor parameters under Case (1).



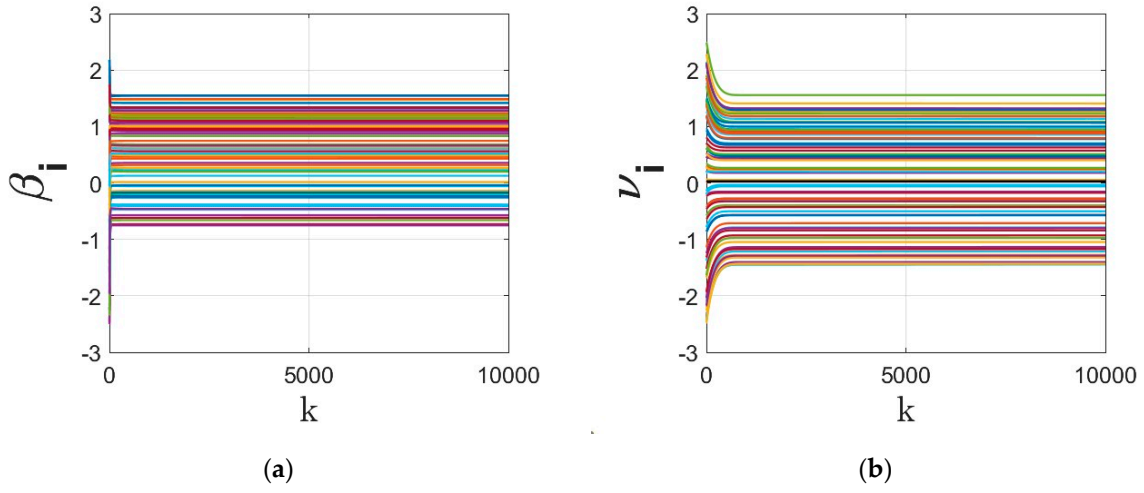
(**a**) (**b**)

Figure 10. This figure shows the time evolution of the regressor parameters under Case (2).



(**a**) (**b**)

Figure 11. This figure shows the time evolution of the regressor parameters under Case (3).



(**a**) (**b**)

Figure 12. This figure shows the time evolution of the regressor parameters under Case (4).

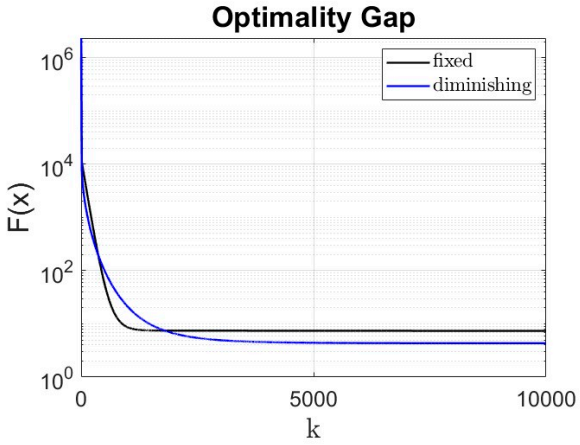


Figure 13. This figure compares the DLR optimality gap under fixed step size and diminishing step size.

5.2. Real Dataset Example

- For the next simulation, we consider the MNIST dataset and the optimization data from [63]. We randomly select $N = 12,000$ labelled images from this dataset and classify these images via logistic regression with a convex regularizer over an exponential network of $n = 16$ computing nodes. For this problem, the global cost function is

$$\min_{b,c} F(b,c) = \frac{1}{n}\sum_{i=1}^{n} f_i, \tag{20}$$

with each computing node i taking a batch of $m_i = 750$ sample images. Then, every node i locally minimizes the following objective function:

$$f_i(b,c) = \frac{1}{m_i}\sum_{j=1}^{m_i} \ln\Big(1 + exp\Big(-\Big(b^\top x_{i,j} + c\Big)y_{i,j}\Big)\Big) + \frac{\lambda}{2}||b||_2^2, \tag{21}$$

where b, c denote the parameters of the separating hyperplane for classification. The optimization residual (or the optimality gap) of the signum-based Algorithm 1 with $u_1 = 1, u_2 = 1,\ 0 < v_1 = 0.9 < 1, v_2 = 1.1 > 1$ is compared with some existing algorithms in the literature. The following algorithms are considered for comparison: GP [64], SGP [65], S-ADDOPT [66]. The comparison results are shown in Figure 14. As it is clear from the figure, by choosing v_1, and v_2 moderately close to 1, Algorithm 1 reaches fast convergence with sufficiently low optimality gap.

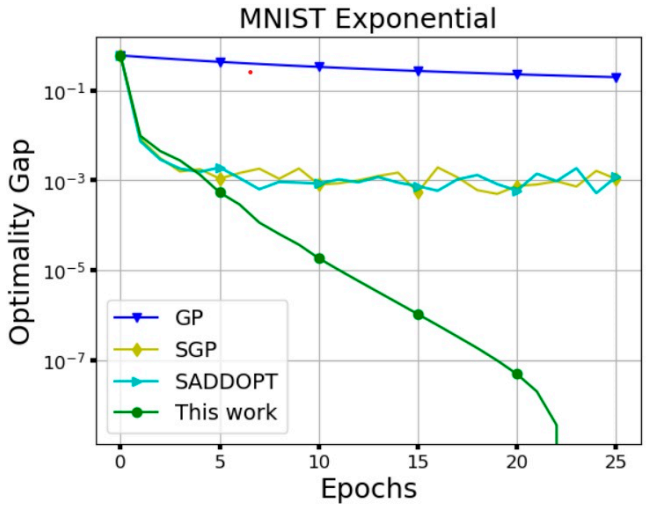


Figure 14. This figure compares the performance of the signum-based algorithm with some existing algorithms in the literature.

6. Conclusions

6.1. Concluding Remarks

Our findings in this paper suggest that although non-Lipschitz signum-based functions show interesting behavior in terms of convergence time/rate, they suffer from certain optimality gap in discrete time applications. The use of such functions results in losing the exact optimality while, on the other hand, reaching rapid convergence rates. In other words, such solutions are practical in scenarios where the exact convergence can be given up to reach faster convergence. This trade-off presents a valuable consideration to balance between the need for quick convergence with the required optimality of the solutions in distributed optimization and machine learning applications.

6.2. Future Directions

The results can be extended to other applications, to study the trade-off between convergence rate and optimality gap in distributed optimization and learning algorithms via alternating direction method of multipliers (D-ADMM) [67] and in the distributed control of the integrated energy system [68]. As we move forward, studying the integration of non-Lipschitz signum-based functions into distributed/decentralized frameworks is an interesting method for other multi-agent applications. This study provides practical insights for decision-making algorithms in real-world applications, for example, distributed techniques for estimation, detection, and resource allocation.

Author Contributions: Conceptualization, A.A., Z.R.G., H.R.R., and M.D.; methodology, A.A., Z.R.G., H.R.R., and M.D.; software, M.D.; validation, M.D.; formal analysis, M.D.; investigation, A.A., Z.R.G., H.R.R., and M.D.; resources, M.D.; data curation, M.D.; writing—original draft preparation, A.A.G., Z.R.G., and M.D.; writing—review and editing, A.A.G., A.A., Z.R.G., H.R.R., and M.D.; visualization, M.D.; supervision, M.D.; project administration, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Center for International Scientific Studies and Collaborations (CISSC), Ministry of Science, Research, and Technology of Iran. The Grant Number is 1403/3586.

Data Availability Statement: The MNIST dataset is available at the following link: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (accessed on 30 September 2025). The raw data on MATLAB simulations supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Necoara, I.; Nedelcu, V.; Dumitrache, I. Parallel and distributed optimization methods for estimation and control in networks. *J. Process Control* **2011**, *21*, 756–766. [CrossRef]
2. Verma, A.; Butenko, S. A distributed approximation algorithm for the bottleneck connected dominating set problem. *Optim. Lett.* **2012**, *6*, 1583–1595. [CrossRef]
3. Veremyev, A.; Boginski, V.; Pasiliao, E.L. Potential energy principles in networked systems and their connections to optimization problems on graphs. *Optim. Lett.* **2015**, *9*, 585–600. [CrossRef]
4. Qureshi, M.I.; Rikos, A.I.; Charalambous, T.; Learning, U.A. Learning and Optimization in Wireless Sensor Networks. In *Wireless Sensor Networks in Smart Environments: Enabling Digitalization from Fundamentals to Advanced Solutions*; Wiley: Hoboken, NJ, USA, 2025; Volume 10, pp. 35–64.
5. Gabidullina, Z.R. Design of the best linear classifier for box-constrained data sets. In *Mesh Methods for Boundary-Value Problems and Applications, Proceedings of the 13th International Conference, Kazan, Russia, 20–25 October 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 109–124.
6. Alzubi, J.; Nayyar, A.; Kumar, A. Machine learning from theory to algorithms: An overview. *J. Phys. Conf. Ser.* **2018**, *1142*, 012012.

7. Doostmohammadian, M.; Aghasi, A.; Charalambous, T.; Khan, U.A. Distributed support vector machines over dynamic balanced directed networks. *IEEE Control Syst. Lett.* **2021**, *6*, 758–763. [CrossRef]
8. Cui, L.; Yang, S.; Chen, F.; Ming, Z.; Lu, N.; Qin, J. A survey on application of machine learning for internet of things. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1399–1417. [CrossRef]
9. Fourati, H.; Maaloul, R.; Chaari, L. A survey of 5g network systems: Challenges and machine learning approaches. *Int. J. Mach. Learn. Cybern.* **2021**, *12*, 385–431. [CrossRef]
10. Xie, Z.; Wu, Z. Event-triggered consensus control for dc microgrids based on MKELM and state observer against false data injection attacks. *Int. J. Mach. Learn. Cybern.* **2024**, *15*, 775–793. [CrossRef]
11. Doostmohammadian, M.; Rabiee, H.R.; Khan, U.A. Cyber-social systems: Modeling, inference, and optimal design. *IEEE Syst. J.* **2019**, *14*, 73–83. [CrossRef]
12. Stankovic, S.S.; Beko, M.L.; Stanković, M.S. Nonlinear robustified stochastic consensus seeking. *Syst. Control Lett.* **2020**, *139*, 104667. [CrossRef]
13. Jakovetic, D.; Vukovic, M.; Bajovic, D.; Sahu, A.K.; Kar, S. Distributed recursive estimation under heavy-tail communication noise. *SIAM J. Control Optim.* **2023**, *61*, 1582–1609. [CrossRef]
14. Dai, L.; Chen, X.; Guo, L.; Zhang, J.; Chen, J. Prescribed-time group consensus for multiagent system based on a distributed observer approach. *Int. J. Control Autom. Syst.* **2022**, *20*, 3129–3137. [CrossRef]
15. Zhang, B.; Mo, S.; Zhou, H.; Qin, T.; Zhong, Y. Finite-time consensus tracking control for speed sensorless multi-motor systems. *Appl. Sci.* **2022**, *12*, 5518. [CrossRef]
16. Doostmohammadian, M. Single-bit consensus with finite-time convergence: Theory and applications. *IEEE Trans. Aerosp. Electron. Syst.* **2020**, *56*, 3332–3338. [CrossRef]
17. Ge, C.; Ma, L.; Xu, S. Distributed fixed-time leader-following consensus for multi-agent systems: An event-triggered mechanism. *Actuators* **2024**, *13*, 40. [CrossRef]
18. Yang, J.; Li, R.; Gan, Q.; Huang, X. Zero-sum-game-based fixed-time event-triggered optimal consensus control of multi-agent systems under FDI attacks. *Mathematics* **2025**, *13*, 543. [CrossRef]
19. Doostmohammadian, M.; Aghasi, A.; Pirani, M.; Nekouei, E.; Khan, U.A.; Charalambous, T. Fast-convergent anytime-feasible dynamics for distributed allocation of resources over switching sparse networks with quantized communication links. In Proceedings of the IEEE European Control Conference, London, UK, 12–15 July 2022; pp. 84–89.
20. Doostmohammadian, M.; Meskin, N. Finite-time stability under denial of service. *IEEE Syst. J.* **2020**, *15*, 1048–1055. [CrossRef]
21. Budhraja, P.; Baranwal, M.; Garg, K.; Hota, A. Breaking the convergence barrier: Optimization via fixed-time convergent flows. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022; Volume 36, pp. 6115–6122.
22. Mishra, J.; Yu, X. On fixed-time convergent sliding mode control design and applications. In *Emerging Trends in Sliding Mode Control: Theory and Application*; Springer: Singapore, 2021; pp. 203–237.
23. Rı, H.; Efimov, D.; Moreno, J.A.; Perruquetti, W.; Rueda-Escobedo, J.G. Time-varying parameter identification algorithms: Finite and fixed-time convergence. *IEEE Trans. Autom. Control* **2017**, *62*, 3671–3678.
24. Malaspina, G.; Jakovetic, D.; Krejić, N. Linear convergence rate analysis of a class of exact first-order distributed methods for weight-balanced time-varying networks and uncoordinated step sizes. *Optim. Lett.* **2023**, *18*, 825–846. [CrossRef]
25. Zuo, Z.; Han, Q.; Ning, B. *Fixed-Time Cooperative Control of Multi-Agent Systems*; Springer: Berlin/Heidelberg, Germany, 2019.
26. Acho, L.; Buenestado, P.; Pujol, G. A finite-time control design for the discrete-time chaotic logistic equations. *Actuators* **2024**, *13*, 295. [CrossRef]
27. Basin, M. Finite-and fixed-time convergent algorithms: Design and convergence time estimation. *Annu. Rev. Control* **2019**, *48*, 209–221. [CrossRef]
28. Xi, C.; Khan, U.A. Distributed subgradient projection algorithm over directed graphs. *IEEE Trans. Autom. Control* **2016**, *62*, 3986–3992. [CrossRef]
29. Szabo, Z.; Sriperumbudur, B.K.; Póczos, B.; Gretton, A. Learning theory for distribution regression. *J. Mach. Learn. Res.* **2016**, *17*, 5272–5311.
30. Xin, R.; Khan, U.; Kar, S. A hybrid variance-reduced method for decentralized stochastic non-convex optimization. In Proceedings of the International Conference on Machine Learning, PMLR, 2021, Virtual, 18–24 July 2021; pp. 11459–11469.
31. Du, B.; Zhou, J.; Sun, D. Improving the convergence of distributed gradient descent via inexact average consensus. *J. Optim. Theory Appl.* **2020**, *185*, 504–521. [CrossRef]
32. Simonetto, A.; Jamali-Rad, H. Primal recovery from consensus-based dual decomposition for distributed convex optimization. *J. Optim. Theory Appl.* **2016**, *168*, 172–197. [CrossRef]
33. Liu, Z.; Jahanshahi, H.; Volos, C.; Bekiros, S.; He, S.; Alassafi, M.O.; Ahmad, A.M. Distributed consensus tracking control of chaotic multi-agent supply chain network: A new fault-tolerant, finite-time, and chatter-free approach. *Entropy* **2021**, *24*, 33. [CrossRef]

34. Yu, Z.; Yu, S.; Jiang, H.; Mei, X. Distributed fixed-time optimization for multi-agent systems over a directed network. *Nonlinear Dyn.* **2021**, *103*, 775–789. [CrossRef]
35. Shi, X.; Wen, G.; Yu, X. Finite-time convergent algorithms for time-varying distributed optimization. *IEEE Control Syst. Lett.* **2023**, *7*, 3223–3228. [CrossRef]
36. Tang, W.; Daoutidis, P. Fast and stable nonconvex constrained distributed optimization: The ellada algorithm. *Optim. Eng.* **2022**, *23*, 259–301. [CrossRef]
37. Li, S.; Nian, X.; Deng, Z.; Chen, Z. Predefined-time distributed optimization of general linear multi-agent systems. *Inf. Sci.* **2022**, *584*, 111–125. [CrossRef]
38. Gong, X.; Cui, Y.; Shen, J.; Xiong, J.; Huang, T. Distributed optimization in prescribed-time: Theory and experiment. *IEEE Trans. Netw. Sci. Eng.* **2021**, *9*, 564–576. [CrossRef]
39. Deng, C.; Ge, M.G.; Liu, Z.; Wu, Y. Prescribed-time stabilization and optimization of cps-based microgrids with event-triggered interactions. *Int. J. Dyn. Control* **2023**, *12*, 2522–2534. [CrossRef]
40. Wen, X.; Qin, S. A projection-based continuous-time algorithm for distributed optimization over multi-agent systems. *Complex Intell. Syst.* **2022**, *8*, 719–729. [CrossRef]
41. Li, Q.; Wang, M.; Sun, H.; Qin, S. An adaptive finite-time neurodynamic approach to distributed consensus-based optimization problem. *Neural Comput. Appl.* **2023**, *35*, 20841–20853. [CrossRef]
42. Slotine, J.J.; Li, W. *Applied Nonlinear Control*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1991.
43. Wu, H.; Tao, F.; Qin, L.; Shi, R.; He, L. Robust exponential stability for interval neural networks with delays and non-lipschitz activation functions. *Nonlinear Dyn.* **2011**, *66*, 479–487. [CrossRef]
44. Yu, H.; Wu, H. Global robust exponential stability for hopfield neural networks with non-lipschitz activation functions. *J. Math. Sci.* **2012**, *187*, 511–523. [CrossRef]
45. Balcan, M.; Blum, A.; Sharma, D.; Zhang, H. An analysis of robustness of non-lipschitz networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–43.
46. Li, W.; Bian, W.; Xue, X. Projected neural network for a class of non-lipschitz optimization problems with linear constraints. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3361–3373. [CrossRef] [PubMed]
47. Olfati-Saber, R.; Murray, R.M. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control* **2004**, *49*, 1520–1533. [CrossRef]
48. Wang, F.; Zhang, Y.; Zhang, L.; Zhang, J.; Huang, Y. Finite-time consensus of stochastic nonlinear multi-agent systems. *Int. J. Fuzzy Syst.* **2020**, *22*, 77–88. [CrossRef]
49. Zhang, B.; Jia, Y. Fixed-time consensus protocols for multi-agent systems with linear and nonlinear state measurements. *Nonlinear Dyn.* **2015**, *82*, 1683–1690. [CrossRef]
50. Liu, J.; Yu, Y.; Wang, Q.; Sun, C. Fixed-time event-triggered consensus control for multi-agent systems with nonlinear uncertainties. *Neurocomputing* **2017**, *260*, 497–504. [CrossRef]
51. Zuo, Z.; Tian, B.; Defoort, M.; Ding, Z. Fixed-time consensus tracking for multiagent systems with high-order integrator dynamics. *IEEE Trans. Autom. Control* **2017**, *63*, 563–570. [CrossRef]
52. Yan, K.; Han, T.; Xiao, B.; Yan, H. Distributed fixed-time and prescribed-time average consensus for multi-agent systems with energy constraints. *Inf. Sci.* **2023**, *647*, 119471. [CrossRef]
53. Yu, Y.; Liu, C.; Li, Y.; Li, H. Practical fixed-time distributed average-tracking with input delay based on event-triggered method. *Int. J. Control Autom. Syst.* **2023**, *21*, 845–853. [CrossRef]
54. Doostmohammadian, M.; Kharazmi, S.; Rabiee, H.R. How clustering affects the convergence of decentralized optimization over networks: A monte-carlo-based approach. *Soc. Netw. Anal. Min.* **2024**, *14*, 135. [CrossRef]
55. Doostmohammadian, M.; Aghasi, A. Accelerated distributed allocation. *IEEE Signal Process. Lett.* **2024**. [CrossRef]
56. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends® Mach. Learn.* **2011**, *3*, 1–122.
57. Song, C.; Yoon, S.; Pavlovic, V. Fast ADMM algorithm for distributed optimization with adaptive penalty. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
58. Lin, Z.; Li, H.; Fang, C. ADMM for distributed optimization. In *Alternating Direction Method of Multipliers for Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 207–240.
59. Ma, M.; Nikolakopoulos, A.N.; Giannakis, G.B. Hybrid admm: A unifying and fast approach to decentralized optimization. *EURASIP J. Adv. Signal Process.* **2018**, *2018*, 1–17. [CrossRef]
60. Gautam, M.; Pati, A.; Mishra, S.; Appasani, B.; Kabalci, E.; Bizon, N.; Thounthong, P. A comprehensive review of the evolution of networked control system technology and its future potentials. *Sustainability* **2021**, *13*, 2962. [CrossRef]
61. Zhang, L.; Hua, L. Major issues in high-frequency financial data analysis: A survey of solutions. *Mathematics* **2025**, *13*, 347. [CrossRef]

62. Wei, Y.; Xie, C.; Qing, X.; Xu, Y. Control of a new financial risk contagion dynamic model based on finite-time disturbance. *Entropy* **2024**, *26*, 999. [CrossRef] [PubMed]
63. Qureshi, M.I.; Khan, U.A. Stochastic first-order methods over distributed data. In Proceedings of the IEEE 12th Sensor Array and Multichannel Signal Processing Workshop (SAM), Trondheim, Norway, 20–23 June 2022; pp. 405–409.
64. Nedic, A.; Olshevsky, A. Distributed optimization over time-varying directed graphs. *IEEE Trans. Autom. Control* **2014**, *60*, 601–615. [CrossRef]
65. Spiridonoff, A.; Olshevsky, A.; Paschalidis, I. Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions. *J. Mach. Learn. Res.* **2020**, *21*, 1–47.
66. Qureshi, M.I.; Xin, R.; Kar, S.; Khan, U.A. S-ADDOPT: Decentralized stochastic first-order optimization over directed graphs. *IEEE Control Syst. Lett.* **2020**, *5*, 953–958. [CrossRef]
67. Noah, Y.; Shlezinger, N. Distributed learn-to-optimize: Limited communications optimization over networks via deep unfolded distributed ADMM. *IEEE Trans. Mob. Comput.* **2025**, *24*, 3012–3024. [CrossRef]
68. Zhang, N.; Sun, Q.; Yang, L.; Li, Y. Event-triggered distributed hybrid control scheme for the integrated energy system. *IEEE Trans. Ind. Inform.* **2022**, *18*, 835–846. [CrossRef]