DECENTRALIZED ACCELERATED NONCONVEX MINIMAX OPTIMIZATION VIA EXACT DIFFUSION

*Haoyuan Cai**, *Sulaiman A. Alghunaim*†, *Ali H. Sayed**

* École Polytechnique Fédérale de Lausanne, Switzerland
†Kuwait University, Kuwait

ABSTRACT

Exact Diffusion is a powerful algorithm for decentralized minimization problems over multi-agent networks, showing significant advantages over gradient-tracking methods, particularly in sparsely connected networks. Despite its success, this technique has remained largely unexplored in the context of decentralized minimax optimization, leaving it unclear whether its benefits extend to this setting. To address this gap, we propose a new algorithm that integrates Exact Diffusion with accelerated momentum for nonconvex-PL minimax problems. We provide a theoretical analysis showing that our method achieves a per-agent sample complexity of $\mathcal{O}\Big(\frac{\kappa^3}{K\varepsilon^3}+\frac{\kappa^2}{(1-\lambda)^2\varepsilon^2}\Big)$ for finding an ε-stationary point, thereby outperforming existing gradient-tracking-based minimax algorithms, where κ is the condition number, K is the number of agents, and λ is the mixing rate of the network. Numerical experiments support our theoretical findings.

***Index Terms*—** Decentralized minimax optimization, exact diffusion, STORM momentum, accelerated algorithm

1. INTRODUCTION

This work considers a network of $K \geq 1$ agents cooperating to solve the following decentralized minimax optimization problem:

$$\min_{x\in\mathbb{R}^{d_1}} \max_{y\in\mathbb{R}^{d_2}} J(x,y) = \frac{1}{K}\sum_{k=1}^{K} J_k(x,y), \tag{1}$$

$$\text{with} \quad J_k(x,y) = \mathbb{E}_{\boldsymbol{\xi}_k}[Q_k(x,y;\boldsymbol{\xi}_k)], \tag{2}$$

where the global cost (risk) function $J(x,y)$ is nonconvex in the variable $x \in \mathbb{R}^{d_1}$ and possibly nonconcave in the variable $y \in \mathbb{R}^{d_2}$, while satisfying a ν-Polyak-Łojasiewicz (PL) condition (see Assumption 1). We focus on the online scenario in which the size of the random samples $\{\boldsymbol{\xi}_k\}$ can be infinitely large and samples are continuously streaming in. Under this setting, the learning strategy can only leverage the information of the stochastic loss function $Q_k(x,y;\boldsymbol{\xi}_k)$ to approximate the true risk value. Note that formulation (1)—(2) captures a wide range of applications, including generative adversarial networks (GANs) [1], robust learning [2], and reinforcement learning [3], to name a few.

Decentralized minimax optimization has recently attracted significant attention for its scalability in solving large-scale games and minimax problems. Notably, several algorithms [4, 5, 6, 7, 8, 9, 10] have been proposed in this context during the past few years. Many of these works integrate the gradient tracking (GT) technique with variance-reducing strategies [11, 12, 13]. Nevertheless, several important distributed learning strategies remain unexplored. For instance, the work [14] showed that the performance bound of Exact Diffusion (ED) [15, 16] and EXTRA [17] strategies outperforms those based on the gradient tracking techniques under a sparsely connected network for minimization problems. These facts naturally raise the following question:

Q: Can we achieve better performance bounds for decentralized minimax optimization based on alternative distributed learning strategies, such as Exact Diffusion?

In this work, we provide an affirmative answer to this question. Specifically, we present a new decentralized minimax algorithm that integrates accelerated momentum with ED. Importantly, we do not simply rely on stochastic gradients to update the model variables. Prior results in [18] imply that variance reduction is essential in the stochastic minimax setting; otherwise, one must compute large-batch gradients at each iteration, which is memory-intensive. To address this issue, we employ the STORM momentum estimator [11] to reduce gradient noise, thereby improving performance.

This work makes the following contributions: 1) We propose a new algorithm for solving the stochastic decentralized nonconvex minimax optimization problem by integrating ED with accelerated momentum; 2) we establish new theoretical results for the proposed algorithm, showing that it outperforms GT–based methods [4, 5] and thereby provides a positive answer to the previous question; and 3) we tackle the analytical challenges arising from the integration of ED with accelerated momentum, noting that the analysis of ED alone is inherently more involved than that of GT strategies.

Notation. We use normal font, e.g., x, to denote deterministic quantities and use bold font, e.g., $\boldsymbol{x}$, to denote stochastic quantities. The calligraphic font, e.g., $\boldsymbol{\mathcal{X}}$, is used for network augmented quantities. The symbol $\mathbb{E}[\cdot]$ denotes the expecta-

tion operator and $\|\cdot\|$ denotes the Euclidean norm (ℓ_2-norm). The neighboring index set of agent k is denoted by $\mathcal{N}_k$. The notation $\mathrm{col}\{a,b,c\}$ denotes the column vector obtained by stacking a,b,c together. The symbol $\otimes$ represents the Kronecker product. In addition, $\mathbb{1}_K$ denotes the K-dimensional all-ones vector, and I_K denotes the $K\times K$ identity matrix.

2. ALGORITHM DEVELOPMENT

In this section, we introduce our proposed algorithm for solving problem (1) over a decentralized network. The algorithm design integrates the STORM momentum [11] with ED [15, 16]. For completeness, we briefly review these core techniques next.

2.1. Revisiting core techniques

We first define $\boldsymbol{m}^x_{k,i}\in\mathbb{R}^{d_1}$ as the momentum vector associated with the x-variable of agent k at iteration i. Using this notation, the STORM momentum recursively update $\boldsymbol{m}^x_{k,i}$ by leveraging the stochastic gradients evaluated at the current and most recent iterates with a new random sample $\boldsymbol{\xi}_{k,i}$, namely

$$\begin{aligned}\boldsymbol{m}^x_{k,i} = (1-\beta_x)[\boldsymbol{m}^x_{k,i-1} - \nabla_x Q_k(\boldsymbol{x}_{k,i-1},\boldsymbol{y}_{k,i-1};\boldsymbol{\xi}_{k,i})] \\ + \nabla_x Q(\boldsymbol{x}_{k,i},\boldsymbol{y}_{k,i};\boldsymbol{\xi}_{k,i}),\end{aligned} \tag{3}$$

where $\beta_x\in[0,1]$ is a smoothing factor. The momentum vector associated with the y-variable follows a similar update in parallel. After evaluation, the momentum vector is used locally to update the model, which is then diffused to neighbors for further aggregation. In this work, we focus on the ED strategy [15, 16], which does not require an explicit tracking step for the momentum vectors and involves only a single gossip step per communication round. In our setting, the model variable recursively updates itself according to the following rule:

$$\boldsymbol{x}_{k,i+1} = \sum_{\ell\in\mathcal{N}_k} a_{k\ell}(2\boldsymbol{x}_{\ell,i} - \boldsymbol{x}_{\ell,i-1} - \mu_x(\boldsymbol{m}^x_{\ell,i} - \boldsymbol{m}^x_{\ell,i-1})), \tag{4}$$

where $a_{k\ell}$ is a scaling weight of the information flowing from agent ℓ to agent k and μ_x is the stepsize parameter. It is important to note that the update for the y-variable involves flipping the sign of the momentum vector, reflecting its maximizing goal compared to the x-variable.

2.2. Algorithm description

The detailed implementation is summarized in **Algorithm** 1 at a node-level. The algorithm starts by initializing identical values for all local model variables, ensuring that the initial consensus error vanishes. At the initial iteration $i=0$, the algorithm computes a b_0-minibatch stochastic gradient and performs a simple adapt-then-combine (ATC) diffusion step to update models across the network. After that iteration, the algorithm recursively performs the steps outlined in lines 8–9 to update the model weights until the T communication round is completed.

Algorithm 1 Exact Diffusion Minimax Algorithm

1: **Initialize:** $\boldsymbol{x}_{1,0}=\cdots=\boldsymbol{x}_{K,0}, \boldsymbol{y}_{1,0}=\cdots=\boldsymbol{y}_{K,0}$, step size μ_x,μ_y, smoothing factor β_x,β_y, initial batch size b_0, communication round T
2: **for** $i=0,\ldots,T$ **do**
3: **for** each agent k in parallel **do**
4: **if** $i==0$ **then**
5: Collect a b_0-minibatch i.i.d sample $\mathcal{E}_{k,0}$ and compute

$$\boldsymbol{m}^x_{k,0} = \frac{1}{b_0}\sum_{\boldsymbol{\xi}_{k,0}\sim\mathcal{E}_{k,0}} \nabla_x Q_k(\boldsymbol{x}_{k,0},\boldsymbol{y}_{k,0};\boldsymbol{\xi}_{k,0}), \tag{5}$$

$$\boldsymbol{m}^y_{k,0} = \frac{1}{b_0}\sum_{\boldsymbol{\xi}_{k,0}\sim\mathcal{E}_{k,0}} \nabla_y Q_k(\boldsymbol{x}_{k,0},\boldsymbol{y}_{k,0};\boldsymbol{\xi}_{k,0}). \tag{6}$$

6: and let

$$\boldsymbol{x}_{k,1} = \sum_{\ell\in\mathcal{N}_k}(\boldsymbol{x}_{\ell,0} - \mu_x\boldsymbol{m}^x_{\ell,0}), \boldsymbol{y}_{k,1} = \sum_{\ell\in\mathcal{N}_k}(\boldsymbol{y}_{\ell,0} + \mu_y\boldsymbol{m}^y_{\ell,0}).$$

7: **else**
8: Compute

$$\begin{aligned}\boldsymbol{m}^x_{k,i} &= (1-\beta_x)[\boldsymbol{m}^x_{k,i-1} - \nabla_x Q_k(\boldsymbol{x}_{k,i-1},\boldsymbol{y}_{k,i-1};\boldsymbol{\xi}_{k,i})] \\ &\quad + \nabla_x Q(\boldsymbol{x}_{k,i},\boldsymbol{y}_{k,i};\boldsymbol{\xi}_{k,i}), \\ \boldsymbol{m}^y_{k,i} &= (1-\beta_y)[\boldsymbol{m}^y_{k,i-1} - \nabla_y Q_k(\boldsymbol{x}_{k,i-1},\boldsymbol{y}_{k,i-1};\boldsymbol{\xi}_{k,i})] \\ &\quad + \nabla_y Q(\boldsymbol{x}_{k,i},\boldsymbol{y}_{k,i};\boldsymbol{\xi}_{k,i}).\end{aligned}$$

9: and

$$\boldsymbol{x}_{k,i+1} = \sum_{\ell\in\mathcal{N}_k} a_{k\ell}(2\boldsymbol{x}_{\ell,i} - \boldsymbol{x}_{\ell,i-1} - \mu_x(\boldsymbol{m}^x_{\ell,i} - \boldsymbol{m}^x_{\ell,i-1})),$$

$$\boldsymbol{y}_{k,i+1} = \sum_{\ell\in\mathcal{N}_k} a_{k\ell}(2\boldsymbol{y}_{\ell,i} - \boldsymbol{y}_{\ell,i-1} + \mu_y(\boldsymbol{m}^y_{\ell,i} - \boldsymbol{m}^y_{\ell,i-1})).$$

10: **end if**
11: **end for**
12: $i=i+1$
13: **end for**

2.3. Network representation

The description presented in **Algorithm** 1 at the node-level is convenient for implementation. We introduce the following network-augmented variables to facilitate the theoretical analysis:

$$\boldsymbol{\mathcal{X}}_i = \mathrm{col}\{\boldsymbol{x}_{1,i},\ldots,\boldsymbol{x}_{K,i}\}\in\mathbb{R}^{Kd_1}, \tag{7}$$

$$\boldsymbol{\mathcal{M}}_{x,i} = \mathrm{col}\{\boldsymbol{m}^x_{1,i},\ldots,\boldsymbol{m}^x_{K,i}\}\in\mathbb{R}^{Kd_1}, \tag{8}$$

$$\boldsymbol{x}_{c,i} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}_{k,i} \in \mathbb{R}^{d_1} \quad \text{(network centroid)}, \tag{9}$$

$$\mathcal{W}_x = W \otimes I_{m_1} \in \mathbb{R}^{Kd_1}. \tag{10}$$

We also define $\boldsymbol{\mathcal{Y}}_i, \boldsymbol{\mathcal{M}}_{y,i}, \mathcal{W}_y \in \mathbb{R}^{Kd_2}, \boldsymbol{y}_{c,i} \in \mathbb{R}^{d_2}$ using a similar way.

3. THEORETICAL RESULTS

In this section, we present the main results for the proposed algorithm. The established results rely on the following assumptions, which are standard in the context of distributed stochastic minimax optimization.

3.1. Assumptions

Assumption 1 (Cost function). *The cost function $J(x, y)$ is nonconvex in x and $-J(x, y)$ is ν-PL in y, i.e.,*

$$\|\nabla_y J(x, y)\|^2 \geq 2\nu(\max_y J(x, y) - J(x, y)), \tag{11}$$

where ν is a strictly positive constant. Furthermore, the maximum envelope function $P(x) \triangleq \max_y J(x, y)$ is lower bounded, i.e., $\inf_x P(x) > -\infty$.

Assumption 2 (Expected smoothness). *The gradient of the local loss function $Q_k(x, y)$ is L_f-smooth in expectation, i.e.,*

$$\begin{aligned}&\mathbb{E}\|\nabla_w Q(x_1, y_1; \boldsymbol{\xi}_k) - \nabla_w Q(x_2, y_2; \boldsymbol{\xi}_k)\|^2 \\ &\leq L_f^2(\|x_1 - x_2\|^2 + \|y_1 - y_2\|^2),\end{aligned} \tag{12}$$

where $w \in \{x, y\}$, $x_1, x_2 \in \mathbb{R}^{d_1}$ and $y_1, y_2 \in \mathbb{R}^{d_2}$ are any arguments from the problem domain.

Under the above two assumptions, we can leverage a key result for carrying out the theoretical analysis, i.e., the gradient of $P(x)$ is $L \triangleq L_f + \frac{\kappa L_f}{2}$ smooth [19], where the condition number is defined as $\kappa = \frac{L_f}{\nu}$. Moreover, our algorithm relies on the local stochastic gradients, which require us to make the following standard assumption.

Assumption 3 (Gradient noise process). *The local stochastic gradients are unbiased and have bounded variance for all k, i when taking the following conditional expectation,*

$$\mathbb{E}[\nabla_w Q_k(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i}; \boldsymbol{\xi}_{k,i}) \mid \boldsymbol{\mathcal{F}}_i] = \nabla_w J_k(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i}), \tag{13}$$

$$\mathbb{E}[\|\nabla_w Q_k(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i}; \boldsymbol{\xi}_{k,i}) - \nabla_w J_k(\boldsymbol{x}_{k,i}, \boldsymbol{y}_{k,i})\|^2 \mid \boldsymbol{\mathcal{F}}_i] \leq \sigma^2. \tag{14}$$

where $w \in \{x, y\}$ and $\boldsymbol{\mathcal{F}}_i = \{\boldsymbol{x}_{k,j}, \boldsymbol{y}_{k,j} \mid k = 1, \ldots, K, j = 0, \ldots, k\}$ is the filtration generated by the random process of the model variables, and σ^2 is a nonnegative constant. In addition, the random samples are independent from each other over different k and i.

Assumption 4. *The combination matrix $W = [a_{k\ell}] \in \mathbb{R}^{K\times K}$ is symmetric, primitive, doubly stochastic, and positive semi-definite.*

The above assumption is also made in the work [14], which ensures that the combination W has a unique eigenvalue 1 and all other eigenvalues $\lambda_2, \ldots, \lambda_K$ are strictly less than 1. Using this fact, we can define $\lambda \triangleq \max_{i=2,\ldots,K} \lambda_i$.

Under the above assumptions, we present the following main results.

3.2. Main results

Theorem 1. *Let Assumptions 1–4 hold. By running **Algorithm** 1 with sufficiently small step sizes $\mu_x \leq \mu_y/(C\kappa^2)$ and smoothing factors $\beta_x = \beta_y = \beta$, where $\kappa > 1$ and $C > 1$ is a constant. For sufficiently large T, it holds that:*

$$\begin{aligned}&\frac{1}{T}\sum_{i=0}^{T-1}(\mathbb{E}\|\nabla_x J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\|^2 + \mathbb{E}\|\nabla_y J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\|^2) \\ &\leq \mathcal{O}\Bigg(\underbrace{\frac{1}{T\mu_x} + \frac{\kappa^2}{\mu_y T}}_{\text{initial error bound}} + \underbrace{\frac{\kappa^2\beta\sigma^2}{K} + \frac{\kappa^2\sigma^2}{b_0 K\beta T}}_{\text{stochastic error bound}} \\ &\quad + \underbrace{\frac{\kappa^2\zeta_0^2\mu_y^2}{K\beta T(1-\lambda)^2} + \frac{\kappa^2\lambda^2\mu_y^2\beta\sigma^2}{K(1-\lambda)^3}}_{\text{network induced error}}\Bigg).\end{aligned} \tag{15}$$

where

$$\begin{aligned}\zeta_0^2 \triangleq\ &\frac{1}{K}\mathbb{E}\big\|(\mathcal{W}_x - \tfrac{1}{K}\mathbb{1}_K\mathbb{1}_K^\top \otimes \mathrm{I}_{d_1})\boldsymbol{\mathcal{M}}_{x,0}\big\|^2 \\ &+ \frac{1}{K}\mathbb{E}\big\|(\mathcal{W}_y - \tfrac{1}{K}\mathbb{1}_K\mathbb{1}_K^\top \otimes \mathrm{I}_{d_2})\boldsymbol{\mathcal{M}}_{y,0}\big\|^2.\end{aligned} \tag{16}$$

Proof. Due to space limitations, proof details are omitted and can be found in an extended version of this work [20, 21]. □

Corollary 1. *Suppose the conditions stated in Theorem 1 hold and let*

$$\mu_x = \mathcal{O}\Big(\frac{K^{2/3}}{T^{1/3}\kappa^2}\Big), \quad \mu_y = \mathcal{O}\Big(\frac{K^{2/3}}{T^{1/3}}\Big), \tag{17}$$

$$\beta_x = \beta_y = \mathcal{O}\Big(\frac{K^{1/3}}{T^{2/3}}\Big), \quad b_0 = \mathcal{O}\Big(\frac{T^{1/3}}{K^{2/3}}\Big). \tag{18}$$

*Then, for sufficiently large T, the convergence rate of **Algorithm** 1 is given by*

$$\begin{aligned}&\frac{1}{T}\sum_{i=0}^{T-1}(\mathbb{E}\|\nabla_x J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{y,i})\|^2 + \mathbb{E}\|\nabla_y J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\|^2) \\ &\leq \mathcal{O}\left(\frac{\kappa^2}{(TK)^{2/3}} + \frac{\kappa^2}{T(1-\lambda)^2} + \frac{\kappa^2 K^{2/3}\sigma^2}{T^{4/3}(1-\lambda)^3}\right).\end{aligned} \tag{19}$$

Furthermore, the per-agent sample complexity is given by

$$\mathcal{O}\left(\frac{\kappa^3\epsilon^{-3}}{K} + \frac{\kappa^2\epsilon^{-2}}{(1-\lambda)^2} + \frac{\kappa^{1.5}K^{0.5}\epsilon^{-1.5}}{(1-\lambda)^{9/4}}\right). \tag{20}$$

and the transient time in achieving linear speedup is given by

$$T = \max\left\{\mathcal{O}\left(\frac{K^2}{(1-\lambda)^{4.5}}\right), \mathcal{O}\left(\frac{K^2}{(1-\lambda)^6}\right)\right\}. \quad (21)$$

It is worth noting that the dominant term of our convergence rate, i.e., $\mathcal{O}\big(\kappa^3\epsilon^{-3}/K\big)$, is independent of the network spectral gap $(1-\lambda)$. This stands in sharp contrast to the gradient tracking algorithm analyzed in [4], whose sample complexity is $\mathcal{O}\big(\kappa^3\epsilon^{-3}/(K(1-\lambda)^2)\big)$. Therefore, our sample complexity significantly outperforms theirs under sparsely connected networks.

4. SIMULATION RESULTS

We consider a numerical example to illustrate the performance of the proposed algorithm under a sparsely connected network. The cost in this example is given by

$$\min_{x\in\mathbb{R}^{d_1}} \max_{y\in\mathbb{R}^{d_2}} J(x,y) = \frac{1}{K}\sum_{k=1}^{K} J_k(x,y), \text{where} \quad (22)$$

$$J_k(x,y) = \mathbb{E}_{\boldsymbol{a}_k^\top, \boldsymbol{e}_k}\left[\frac{1}{2}(\boldsymbol{a}_k^\top x)^2 + y^\top(B_k x + \boldsymbol{e}_k) - \frac{\nu}{2}\|y\|^2\right]. \quad (23)$$

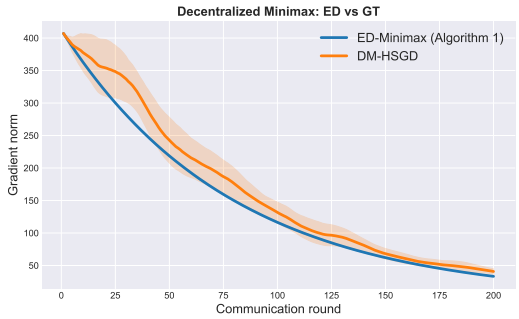
and $\boldsymbol{a}_k \in \mathbb{R}^{d_1}$ is the local random sample, $\boldsymbol{e}_k \in \mathbb{R}^{d_2}$ is the random error vector, $B_k \in \mathbb{R}^{d_2\times d_1}$ is a deterministic matrix that varies across different agents, and ν is the strong concavity factor.

We consider a ring network topology consisting of $K = 50$ agents. The problem setups are given as follows: we consider $\nu = 10$; we generate the random coupling matrix B_k with i.i.d. entries $[B]_{ij} \sim \mathcal{N}(0, 0.001)$. The entries of the local samples $\boldsymbol{a}_k$ is independently sampled from the normal distribution $\mathcal{N}(1.0, 10.0)$; The entries of the error vector $\boldsymbol{e}_k$ is independently sampled from the distribution $\mathcal{N}(0.0, 10.0)$. Each agent owns a randomly generated dataset with size $N_k = 2000$. The dimensions of the vectors x and y is set to $d_1 = 30$ and $d_2 = 30$.

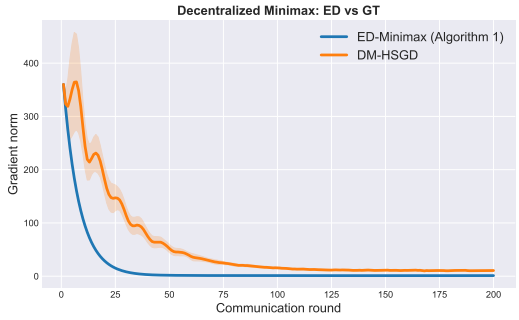
We compare **Algorithm** 1 against the GT-based method DM-HSGD [4, 5], which used the same gradient estimator as ours. For both algorithms, we consider the same hyperparameter setting: the step sizes are set as $\mu_x = 0.001$(or 0.0001) and $\mu_y = 0.01$(or 0.001), the smoothing factors are set as $\beta_x = \beta_y = 0.1$. We then plot the gradient norm $\|\nabla_x J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\| + \|\nabla_y J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\|$ produced by the two algorithms over different communication rounds. From simulation results, we observe that **Algorithm** 1 converges faster than DM-HSGD while achieving lower state-state errors.

5. CONCLUSIONS

In this work, we presented a new algorithm for a decentralized nonconvex minimax algorithm. The proposed diffusion minimax algorithm is devised by integrating the accelerated momentum and ED learning strategies. The established convergence rate improves over GT momentum-based methods. We further validated the performance of the proposed algorithm through numerical simulation. The simulation results showed that our proposed algorithm outperformed the existing GT-based algorithm under a sparse network topology.



(a) Simulation results for $\mu_x = 0.0001, \mu_y = 0.001$.



(b) Simulation results for $\mu_x = 0.001, \mu_y = 0.01$.

Fig. 1: Simulation results of the gradient norm $\|\nabla_x J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\| + \|\nabla_y J(\boldsymbol{x}_{c,i}, \boldsymbol{y}_{c,i})\|$ over different communication rounds.

6. REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2014, vol. 27.

[2] Y. Yan, Y. Xu, Q. Lin, L. Zhang, and T. Yang, "Stochastic primal-dual algorithms with faster convergence than $\mathcal{O}(1/\sqrt{T})$ for problems without bilinear structure," *arXiv:1904.10112*, 2019.

[3] S. Li and et al., "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 4213–4220.

[4] W. Xian, F. Huang, Y. Zhang, and H. Huang, "A faster decentralized algorithm for nonconvex minimax problems," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021, vol. 34, pp. 25865–25877.

[5] F. Huang and S. Chen, "Near-optimal decentralized

momentum method for nonconvex-pl minimax problems," *arXiv:2304.10902*, 2023.

[6] X. Zhang, G. Mancino-Ball, N. S. Aybat, and Y. Xu, "Jointly improving the sample and communication complexities in decentralized stochastic minimax optimization," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, pp. 20865–20873.

[7] H. Cai, S. A. Alghunaim, and A. H. Sayed, "Diffusion stochastic optimization for min-max problems," *IEEE Trans. Signal Process.*, vol. 73, pp. 259–274, 2025.

[8] L. Chen, H. Ye, and L. Luo, "An efficient stochastic algorithm for decentralized nonconvex-strongly-concave minimax optimization," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*. 2024, pp. 1990–1998, PMLR.

[9] H. Cai, S. A. Alghunaim, and A. H. Sayed, "Communication-efficient algorithms for distributed nonconvex minimax optimization problems," *arXiv:2507.21901*, 2025.

[10] Y. Huang, J. Xu, J. Chen, and K. H. Johansson, "An optimistic gradient tracking method for distributed minimax optimization," *arXiv:2508.21431*, 2025.

[11] A. Cutkosky and F. Orabona, "Momentum-based variance reduction in non-convex sgd," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, vol. 32.

[12] H. Cai, S. A. Alghunaim, and A. H. Sayed, "Accelerated stochastic min-max optimization based on bias-corrected momentum," *arXiv:2406.13041*, 2024.

[13] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "Sarah: A novel method for machine learning problems using stochastic recursive gradient," in *Proc. Int. Conf. Mach. Learn. (ICML)*. 2017, pp. 2613–2621, PMLR.

[14] S. A. Alghunaim and K. Yuan, "Unified and refined convergence analysis for non-convex decentralized learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 3264–3279, 2022.

[15] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part I: Algorithm development," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 708–723, 2018.

[16] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part II: Convergence analysis," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 724–739, 2018.

[17] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[18] T. Lin, C. Jin, and M. I. Jordan, "On gradient descent ascent for nonconvex-concave minimax problems," in *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 2020, pp. 6083–6093.

[19] M. Nouiehed, M. Sanjabi, T. Huang, J. D. Lee, and M. Razaviyayn, "Solving a class of non-convex min-max games using iterative first order methods," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, vol. 32.

[20] H. Cai, S. A. Alghunaim, and A. H. Sayed, "Dama: A unified accelerated approach for decentralized nonconvex minimax optimization-part I: Algorithm development and results," *arXiv:2512.13920*, 2025.

[21] H. Cai, S. A. Alghunaim, and A. H. Sayed, "Dama: A unified accelerated approach for decentralized nonconvex minimax optimization-part II: Convergence and performance analyses," *arXiv:2512.13923*, 2025.