Ehsan Lari

Distributed Learning with Enhanced Efficiency, Robustness and Privacy

Thesis for the Degree of Philosophiae Doctor PhD

Trondheim, May 2025

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

ISBN (printed version)
ISBN (electronic version)
ISSN 1503-8181 (printed version)
ISSN 2703-8084 (online version)

Doctoral theses at NTNU, 2025:

Printed by NTNU Grafisk senter

بر چهرهٔ گل نسیم نوروز خوش است
در صحن چمن روی دل‌افروز خوش است
از دی که گذشت هر چه گویی خوش نیست
خوش باش و ز دی مگو که امروز خوش است

خیام نیشابوری

To my dear parents, Behrooz and Shahla.

Thanks for the lifetime supply of love, support, and forgivable loans. This thesis is a small token of appreciation for your unwavering belief in me and your constant encouragement.

Abstract

Distributed machine learning for Internet of things (IoT) and cyber-physical systems (CPS) applications face increasing demands for efficiency, privacy, and robustness. This thesis focuses on addressing these challenges through two primary approaches, i.e., fully-distributed learning algorithms and federated learning (FL) algorithms, which form the main contributions of the work.

This thesis proposes machine learning algorithms for distributed and federated learning in the context of IoT and CPS applications. Developing distributed algorithms for artificial intelligence is necessary as centralized data processing may be unfeasible due to computational and communication costs and privacy concerns. The thesis addresses challenges in fully-distributed learning and federated learning (FL) settings, focusing on resilience against attacks, robustness to communication noise, and privacy preservation.

The main contributions of the thesis can be grouped into the following categories:

- resilience of partial-sharing-based online federated learning against model-poisoning attacks
- noise-robust and resource-efficient federated learning
- privacy-preserving distributed nonnegative matrix factorization
- distributed maximum consensus with noisy communication links

In the context of federated learning, we analyze the resilience of the partial-sharing-based online FL (PSO-Fed) algorithm to model-poisoning attacks. We show that PSO-Fed outperforms other communication-efficient FL algorithms against model-poisoning attacks without introducing additional computational burdens on the clients. Theoretical analysis and simulations demonstrate PSO-Fed's convergence properties and robustness against attacks, as well as revealing an optimal stepsize in the presence of model-poisoning attacks.

To address communication noise in federated learning, we propose a novel noise-robust and resource-efficient algorithm called RERCE-Fed. This algorithm introduces key modifications to counteract the adverse effects of communication noise and improve performance through continued local updates. Theoretical analysis confirms convergence in both mean and mean-square senses, with numerical results validating RERCE-Fed's effectiveness.

In the context of privacy-preserving distributed learning, we develop a distributed nonnegative matrix factorization (PPDNMF) algorithm. This algorithm ensures secure information exchange between neighboring agents using the Paillier cryptosystem, protecting local data from internal and external eavesdroppers.

Finally, we introduce a noise-robust distributed maximum consensus (RD-MC) algorithm for estimating the maximum value within multi-agent ad-hoc networks with noisy communication links. RD-MC redefines the conventional maximum consensus problem as a distributed optimization problem, employing techniques to enhance robustness against noise.

Overall, this thesis lays a solid foundation for the development of secure, efficient, and privacy-preserving distributed and federated learning algorithms for IoT applications, offering critical solutions to key challenges in deploying smart and collaborative systems in emerging IoT domains.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the fulfillment of requirements for the degree of Doctor of Philosophy.

The doctoral work started in January 2020 at the Department of Electronic Systems, NTNU, Trondheim, Norway. The work has been supervised by Professor Stefan Werner and Dr. Reza Arablouei.

The doctoral work was funded by the Research Council of Norway.

Acknowledgements

First and foremost, to my family whom I owe a debt of gratitude that words can scarcely express. Their unwavering support and unbounded love have been the bedrock upon which I have built not just this thesis, but my entire life. In moments of doubt, their voices were my compass. In times of triumph, their joy amplified my own. This PhD journey, with all its challenges and revelations, has been illuminated by their presence. This thesis is dedicated to you.

I am profoundly grateful to my supervisor, Professor Stefan Werner. Stefan, your trust in me from the very beginning, when you offered me this opportunity despite my "complex" situation with Canada, means more than I can express. Your unwavering support, guidance, and belief in my abilities have been the cornerstone of my PhD journey. Your availability and encouragement have been invaluable, and I am truly thankful for your mentorship.

I also want to extend my heartfelt thanks to my co-supervisor, Dr. Reza Arablouei. Reza, your support and accessibility during these formative years have been crucial. More than that, you've had a transformative impact on my academic thinking. Your insights and guidance have shaped not just this thesis, but my entire approach to research. I am deeply appreciative of how you've influenced my academic growth.

I am grateful to the amazing people I found along the way whom I can proudly call my friends. They made living abroad so much easier despite the challenges it brought. Your constant support, companionship, and encouragement turned this difficult journey into a fulfilling experience.

I am deeply grateful to all my co-authors for their valuable contributions and the enriching discussions we've shared. A special, heartfelt thanks goes to my friend and colleague, Dr. Vinay Gogineni. Vinay, your generosity in sharing your expertise and insights has been invaluable. Your guidance has not only shaped this thesis significantly but has also been a constant source of inspiration. I am truly thankful

for your friendship and mentorship throughout this journey.

I thank the members of the Department of Electronic Systems, particularly the Signal Processing Group. Their contributions have created a supportive and intellectually stimulating environment that has facilitated my research work. A special thanks goes to Professor Snorre Aunet. The three years I spent as his teaching assistant were invaluable, not only in deepening my understanding of the subject matter but also in sharpening my skills as a teacher and I am profoundly grateful for the trust he placed in me.

Finally, to my beloved Narges, your unwavering support and encouragement have been the cornerstone of this journey. From our walks through the streets of Seoul to our adventures in Vancouver, you've been there every step of the way, even when we were apart. Despite the miles between us, your presence has been felt in every step of this process. Through late nights, moments of doubt, and countless video calls, your patience, understanding, and love have been my guiding light. Your belief in me, even when I doubted myself, gave me the strength to persevere. This thesis is as much a product of your long-distance support as it is of my work. Thank you for being my rock and my inspiration. I couldn't have done this without you, and I'm eternally grateful for your presence in my life.

This work was undertaken during an unprecedented time in modern history. As I began my work in January 2020, the world was on the verge of the COVID-19 pandemic, which emerged in full force less than two months later. This global crisis presented unique challenges to academic research, disrupting traditional work patterns and necessitating rapid adaptations. Despite these challenges, I learned how important it is to keep going even when things are tough. The pandemic showed me how vital science is for solving big problems. I am grateful for the support I got in navigating through these extraordinary circumstances.

Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations and Symbols

Abbreviations

ADMM	Alternating direction method of multipliers
BCD	Block-coordinate descent
CPS	Cyber-physical system
DL	Distributed learning
DMC	Distributed maximum consensus
DNMF	Distributed non-negative matrix factorization
FL	Federated learning
GD	Gradient descent
IID	Independent and identically distributed
IoT	Internet of Things
MC	Maximum consensus
MSD	Mean square deviation
MSE	Mean square error
NMF	Non-negative matrix factorization
Non-IID	Non-independent and identically distributed
Online-Fed	Online federated learning
P2P	Peer-to-peer communication
PP	Privacy-preserving
PS	Partial-sharing-based
PSO-Fed	Partial-sharing-based online federated learning
RFF	Random Fourier features
SGD	Stochastic gradient descent
SNR	Signal-to-noise ratio

Symbols

$\mathcal{D}_k$	Dataset of client k
$\mathcal{S}_n$	Set of clients at iteration n
$\mathcal{S}_B$	Set of Byzantine clients
$\mathcal{E}$	Mean square error
$\mathbf{w}_{k,n}$	Local model of client k at iteration n
$\widetilde{\mathbf{w}}_{k,n}$	Local model of client k at iteration n after noise perturbation
$\mathbf{w}_n$	Global model of server at iteration n
$\widetilde{\mathbf{w}}_n$	Global model of server at iteration n after noise perturbation
$\mathbf{w}^\star$	The optimal solution
K	Number of agents (clients) in the network
L	Size of model parameter vector
M	Number of shared elements of model parameter vector
$\mathbf{S}_{k,n}$	Selection matrix for local model of client k at iteration n
$\mathbf{I}$	Identity matrix
$\mathbf{O}$	Zero matrix
$\mathbf{1}$	Vector of ones
$\mathbf{0}$	Vector of zeros
$\mathcal{G}$	Graph
$\mathcal{N}$	Set of agents (clients)
$\mathcal{N}_k$	Set of neighboring agents (clients) of agent (client) k
$\mathcal{J}_k(\cdot)$	Local loss function of agent (client) k
$\mathcal{L}(\cdot)$	Augmented Lagrangian function
$\mathbb{E}[\cdot]$	Statistical expectation operator
$\mathcal{N}(\cdot,\cdot)$	Gaussian distribution
$\mathcal{U}(\cdot,\cdot)$	Uniform distribution
$\|\cdot\|_{\mathsf{F}}$	Frobenius norm of its matrix argument
$\|\cdot\|_2$	Euclidean norm of its vector argument
$\otimes$	Kronecker product
$\otimes_b$	Block Kronecker product
$(\cdot)^{\mathsf{T}}$	Transpose operator
$\mathrm{tr}(\cdot)$	Trace operator
$\mathrm{diag}\{\cdot\}$	Diagonalization operator
$\mathrm{bdiag}\{\cdot\}$	Block diagonalization operator
$\mathrm{col}\{\cdot\}$	Columnization operator
$\mathrm{bcol}\{\cdot\}$	Block columnization operator
$\mathrm{vec}\{\cdot\}$	Vectorization operator
$\mathrm{bvec}\{\cdot\}$	Block vectorization operator
$\mathrm{bvec}^{-1}\{\cdot\}$	Inverse block vectorization operator

Chapter 1

Introduction

1.1 Motivation and Scope

The proliferation of Internet of things (IoT) devices is reshaping our world into an interconnected network of smart objects. These devices, ranging from household appliances and wearable gadgets to industrial sensors and autonomous vehicles, are becoming increasingly common in both personal and commercial settings. This expansion is driven by reasons such as enhanced computing power, and widespread access to high-speed Internet. As more devices join this ocean of devices, they generate vast amounts of data, providing valuable insights into human behavior, environmental conditions, and industrial processes. The potential benefits of IoT technology are immense, spanning across various fields including healthcare, education, entertainment, social life, energy management, and smart cities [1–3]. However, this growth also presents significant challenges in energy efficiency, privacy, security, and data management [4, 5].

In a distributed IoT setting, devices are dispersed across various locations often operating with different communication and computational capabilities and limited energy resources. This gives new opportunities to gather and leverage data for a variety of applications. These devices can also learn from their collected data by extracting valuable information from the data. Distributed learning enables IoT systems to make data-driven decisions, generate insights, uncover patterns, detect anomalies, and make smart inferences that can be used to optimize various systems such as autonomous vehicles [6].

Distributed learning in IoT applications poses several technical challenges, e.g., communication efficiency, additive noise in the communication links, and chal-

lenges related to privacy preservation. Since IoT devices often operate with limited bandwidth and energy resources, communication efficiency is paramount for distributed learning applications. Transmitting large amounts of raw data from dispersed devices to their neighbors or a central server can quickly overwhelm the network. This necessitates the development of methods to transmit only the most relevant information. Additionally, the additive noise in the communication links causes unreliability in IoT systems. Hence, we require robust learning algorithms that are capable of mitigating the impact of noise-corrupted communication. Furthermore, the highly sensitive nature of data collected by IoT devices, ranging from personal health metrics to industrial process details, underscores the critical need for privacy-preserving distributed learning algorithms [7–10].

In addition to the challenges mentioned earlier, various types of adversaries can compromise system integrity, trustworthiness, and performance in distributed learning. The adversaries in a distributed setting can be categorized into external eavesdroppers and internal malicious agents. External eavesdroppers can intercept communications and internal adversaries can disrupt the network. Internal malicious agents can further be categorized into Byzantine agents and honest-but-curious agents. Byzantine agents may arbitrarily deviate from the rules of the network but honest-but-curious agents follow the rules but attempt to infer private information from the exchanged messages. Attacks in distributed networks can take multiple forms, including data-poisoning, model-poisoning attacks, eavesdropping, inference attacks and denial of service (DoS) attacks. This highlights the pressing need for algorithms that are capable of mitigating the impacts of adversarial attacks on a network [11–13].

This thesis aims to develop novel distributed and federated learning algorithms to solve real-world learning tasks that leverage signal processing techniques and encryption algorithms to address the critical challenges of resilience against adversarial attacks, communication efficiency, robustness against noise, and privacy preservation, along with minimizing computational load on devices.

1.2 Objectives

The objectives of this thesis can be summarized as follows:

O1 To analyze the effect of sharing only a fraction of the model parameter on the resilience against model-poisoning attacks and assess the performance of the network under the strain of such attack scenarios.

O2 To improve the communication-efficiency and noise-robustness of the network by developing distributed and federated algorithms that exhibit such properties without incurring additional computational and communication load on the network.

O3 To propose a strategy that fits well into the distributed setting and ensures the privacy of participating agents in the presence of external and internal adversaries.

1.3 List of Publications

The author of this thesis conducted the following research studies in accordance with the research objectives described in section 1.2. The results contained in this thesis are published in seven papers. They include five conferences and two journals. Below is the list of the publications:

P1 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "On the resilience of online federated learning to model poisoning attacks through partial sharing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2024, pp. 9201–9205.

P2 E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, "Resilience in Online Federated Learning: Mitigating Model-Poisoning Attacks via Partial Sharing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 388–400, 2025.

P3 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.

P4 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.

P5 E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, "Noise-Robust and Resource-Efficient ADMM-based Federated Learning for WLS Regression," submitted to *Elsevier Signal Processing*.

P6 E. Lari, R. Arablouei, and S. Werner, "Privacy-Preserving Distributed Nonnegative Matrix Factorization," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 1022–1026.

P7 E. Lari, R. Arablouei, N. K. D. Venkategowda, and S. Werner, "Distributed Maximum Consensus over Noisy Links," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 2247–2251.

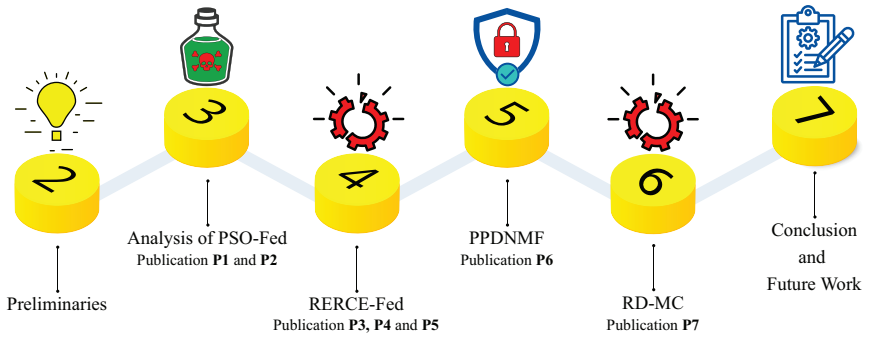

Figure 1.1: Thesis contributions and organization diagram.

1.4 Structure and Contributions

The rest of this thesis is organized as follows. Chapter 2 provides an overview of distributed and federated learning settings and some preliminary concepts relevant to solving distributed optimization problems within the remainder of this thesis. The main technical contributions of this thesis are presented across four chapters, with each chapter addressing one of the research objectives outlined in section 1.2 as follows:

- Chapter 3 analyzes the effect of partial sharing on the resilience of online FL (PSO-Fed) against model-poisoning attacks. It proposes an attack model that fits well into a federated setting and uncovers the existence of an optimal stepsize for PSO-Fed when a model-poisoning attack is present in the system.
- Chapter 4 proposes an ADMM-based FL algorithm to solve a weighted least square problem that is both noise-robust and communication-efficient. Furthermore, it allows the non-participating clients to continue their local updates to improve the overall performance.
- Chapter 5 develops a privacy-preserving distributed non-negative matrix factorization. It employs the Paillier cryptosystem to ensure the preservation of privacy and prevent sensitive information leakage.
- Chapter 6 designs a noise-robust distributed maximum consensus algorithm. It proposes to eliminate the dual variable and employ a moving average to mitigate the adverse effect of the additive noise in the communication links.

Finally, chapter 7 concludes the thesis and proposes future research directions. A diagram illustrating the thesis organization and contributions is given in Figure 1.1.

Chapter 2

From Distributed to Federated Learning over Networks

This chapter provides the background information required for the rest of the thesis. This chapter begins with an overview of the historical developments of learning algorithms, as detailed in section 2.1. Furthermore, section 2.1 motivates the transition from centralized learning frameworks to decentralized and distributed settings. In section 2.2, we state the fundamental principles of distributed learning, highlighting its abilities and challenges. In section 2.3, we introduce federated learning along with its benefits and challenges. In section 2.4, we introduce online federated learning, which is suitable for handling streaming data in a federated setup. In section 2.5, we introduce partial sharing of information as a method to lower the communication load in a distributed setup. We introduce the alternating direction method of multipliers, which is a first-order iterative algorithm to solve distributed optimization problems in section 2.6.

2.1 Background and Evolution

The evolution of learning algorithms has spanned several decades, marked by foundational breakthroughs and the emergence of new paradigms. Traditionally, machine learning relied on centralized frameworks, where all data was collected and processed on a server. This approach offered several benefits, such as efficient data processing and model training, and simple management and deployment. However, centralized learning presents significant challenges like data privacy and security, e.g., sensitive data must be transferred and stored centrally, which increases exposure to breaches and might violate privacy regulations. In addition, centralized learning suffers from communication load where large volumes of data

must be transferred to a central server, which can create bottlenecks and inefficiencies, especially with geographically distributed sources. Moreover, using a centralized setting might cause scalability limitations. In fact, scaling centralized systems requires increasingly powerful hardware, which can be costly and less flexible compared to distributed solutions [14].

To address these limitations, the field has shifted toward distributed and decentralized learning paradigms. Data and computation are spread across multiple nodes or devices. Each participant trains a model locally and sends updates (not raw data) to a central server for aggregation. This approach enhances scalability and computational efficiency, as workloads are shared [15].

A pioneering work in the field of distributed estimation was published in 1982 by Borkar and Varaiya [16]. In 1984, Tsitsiklis produced a significant piece of work for his PhD thesis focusing on decentralized and distributed decision-making and computation. He began by addressing the problem of whether a group of distributed but isolated decision-makers, each possessing different yet related information, can reach similar decisions without any communication. He then explored a scenario in which these decision-makers are allowed to communicate and share their decisions. In this setting, he demonstrated that they are guaranteed to reach a consensus. This conclusion was achieved by solving an optimization problem aimed at minimizing a global objective function [17]. Later in 1986, Tsitsiklis, Bertsekas, and Athans published another significant work that analyzed the convergence of various asynchronous distributed algorithms [18].

In 1989, Bertsekas and Tsitsiklis published a survey that introduced different types of distributed algorithms that can be solved via asynchronous updates. In their work, they showed that although a number of distributed algorithms fail to converge under an asynchronous setting, a fair number of algorithms still manage to converge under such a scenario [19]. In a separate work [20], Bertsekas and Tsitsiklis examined algorithms that are particularly amenable to parallelization. They also investigated the convergence properties of these algorithms, the rates of convergence, and the communication and synchronization challenges associated with their implementation.

Two decades later, the exploration of consensus and cooperation within distributed systems was undertaken in the 2000s by Olfati-Saber, Fax, and Murray, as published in their various works [21–25]. In [26], the authors studied the alignment problem that involved reaching a consensus in a distributed setting. These works laid the foundation for the theoretical framework for posing and solving consensus problems for distributed dynamic settings.

Another major and significant advancement in the field was marked by works on multi-agent adaptive networks by Sayed [27–29]. Multi-agent adaptive networks consist of collections of agents or nodes that interact and cooperate to address problems, such as distributed estimation, learning, and decision-making. These agents or nodes adapt their behaviors based on local data and information exchanged with their neighboring agents. Several distributed strategies can be employed to find solutions to these aforementioned problems. Three prominent strategies include incremental strategies [30–33], consensus strategies [34, 35], and diffusion strategies [36–38].

As an example, in [39], the authors studied distributed estimation algorithms based on diffusion protocols that enhance cooperation among adaptive nodes, where each node possesses local learning capabilities to generate estimates for a specific parameter, sharing information solely with neighboring nodes, thereby forming peer-to-peer protocols. This distributed and cooperative algorithm can respond to changes in the environment and demonstrates improved performance in transient and steady-state mean-square error compared to traditional non-cooperative methods.

The research on adaptive networks has significantly influenced the field of distributed learning. Specifically, it demonstrates that, under certain conditions, distributed strategies can achieve performance similar to that of centralized solutions, even in sparsely connected networks [40, 41]. Moreover, analytical studies have revealed how factors such as network topology, combination policies, and algorithm parameters affect the transient and steady-state performance behavior of adaptive networks, e.g., the convergence rate and steady-state error [40, 41].

In addition to various works on multi-agent adaptive networks, the topic of adaptive filters was studied in [42–46], which significantly influenced the field of distributed learning. One notable contribution is the energy conservation approach, which offers a robust analytical framework for both the understanding and design of adaptive filters [47]. This methodology effectively addresses the stochastic challenges that are characteristic of adaptive algorithms, facilitating a more straightforward analysis of their behavior and stability. We later utilize the energy conservation framework to analyze our algorithms.

In the following sections, we will present an overview of distributed and federated learning. This overview will establish a foundational basis for the subsequent chapters of this thesis.

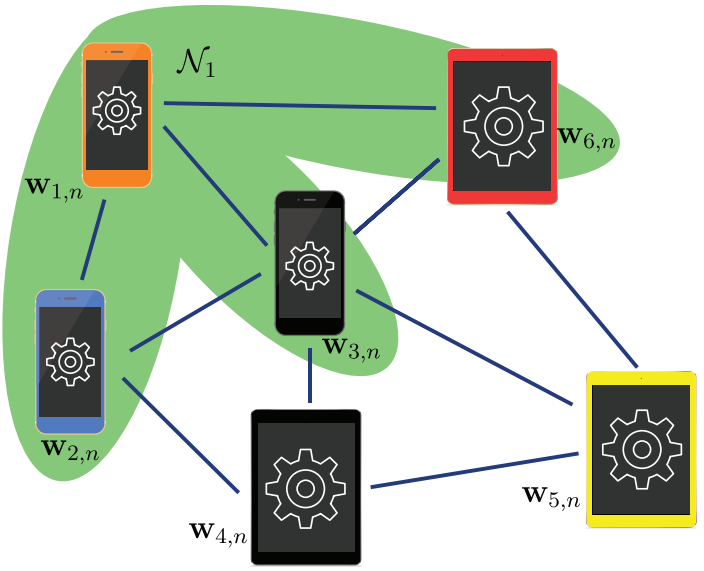

Figure 2.1: Conventional distributed learning structure.

2.2 Foundations of Distributed Learning

Distributed learning (DL) refers to a machine learning (ML) framework where the training process is distributed across multiple agents. It distinguishes itself from traditional centralized ML structures by allowing multiple agents to collaboratively train an ML model without the need for a central server or authority to orchestrate the process. In distributed learning, each participating agent k at iteration n maintains its own local dataset and performs local computations, sharing only its local trained model $\mathbf{w}_{k,n}$ with its neighboring agents $\ell \in \mathcal{N}_k$ and contributing to the global learning objective.

In contrast to centralized approaches where data is located in a single place, fully-distributed learning allows data to remain distributed across multiple agents, where these agents themselves can be diverse entities such as individual devices, different organizations, or even distributed among various geographical locations.

The learning process typically involves iterative rounds of local computations followed by peer-to-peer communication among neighboring agents. A conventional distributed learning structure is illustrated in Figure 2.1 and summarized in the form of a pseudo-code in Algorithm 1. We can observe from Figure 2.1 that peer-to-peer communication enables direct interaction with neighboring agents without the need for any intermediary server. This figure also depicts the set of neighbors

Algorithm 1: Conventional Distributed Learning.

Parameters: penalty parameter ρ, set of agents $\mathcal{N}$
Initialization: local models $\mathbf{w}_{k,0}$
For $n = 1, \ldots,$ *Until Convergence*
Local Update at Agent k:
    Receive $\mathbf{w}_{\ell,n}$ from the neighbors $\ell \in \mathcal{N}_k$.
    Update the local model.
    Send $\mathbf{w}_{k,n+1}$ to the neighbors $\ell \in \mathcal{N}_k$.
EndFor

of agent 1 that consists of $\mathcal{N}_1 \in \{2, 3, 6\}$. This communication model offers several advantages and exhibits some challenges and limitations. We delve into these topics in the following.

2.2.1 Advantages of Distributed Learning

DL structure exhibits several key advantages, including:

- Ability to handle large datasets: Distributed learning can process datasets that are too large for a single agent. The distribution of data is handled via partitioning a dataset across multiple agents [48]. In addition, data can be distributed in nature, e.g., data from distributed sensor networks.

- Improved processing speed through parallel computation: Depending of factors such as application and topology, distributing a dataset across multiple agents, can reduce the training time significantly [48].

- Fault tolerance and system reliability: A distributed system can continue functioning even if some agents or communication links between agents fail or become unavailable. This redundancy ensures that the learning process is not halted due to faulty agents [49].

2.2.2 Challenges and Limitations of Distributed Learning

DL faces some challenges and limitations, including [50]:

- Model update sharing among agents: A key part of the learning process in a distributed structure is the model exchange between agents. This can create an additional communication load on the agents.

- Privacy concerns: Although the goal of distributed learning is to omit the need to transfer raw data, it is not immune to privacy leakage concerns. The exchange of model updates can potentially reveal sensitive information about the underlying data through various inference attacks. To mitigate these risks, privacy-preserving techniques such as differential privacy [51] and homomorphic encryption [52] have been proposed in the literature. These methods aim to add noise to shared information or perform computations on encrypted data. However, implementing these techniques often involves a trade-off between privacy guarantees and model performance.

- Dealing with stragglers and asynchronous updates: Stragglers can be defined as agents that are significantly slower in completing their local computations or communicating updates [53, 54]. These types of agents can pose challenges in the form of slowing down the overall learning process. To address these challenges, several strategies have been developed. One approach is to ignore delayed updates from stragglers. Another solution is to perform asynchronous updates, where we allow the agents to perform updates at their own pace without waiting for slower participants. However, these methods introduce challenges in terms of algorithm convergence.

2.3 Federated Learning: A Shifting Frontier

Federated Learning (FL) represents a distinctive approach to distributed learning. The concept of FL was first introduced by Google researchers in 2016, notably in [55] by McMahan et al. This foundational study coined the term "Federated Learning" and described a system where numerous clients, such as mobile devices, collaboratively train a machine learning model under the coordination of a central server, all while keeping the training data decentralized and private on devices. The initial goal was to facilitate privacy-preserving machine learning, especially for applications like Google Keyboard (Gboard), where user data remains on the device, and only model updates are shared, rather than raw data [56]. FL is characterized by employing a central server to coordinate the collaborative learning process. Unlike fully-distributed learning, FL maintains a structure where a server orchestrates the training across multiple edge devices, called clients, within a network.

The FL literature frequently employs the FedAvg algorithm [55] as a standard benchmark. FedAvg begins with the server broadcasting its aggregated global model to a randomly chosen subset of clients $k \in \mathcal{S}_n$ during iteration n, typically over a wireless network. These clients then perform local training to refine their local models $\mathbf{w}_{k,n+1}$ before sending the updates back to the server. The

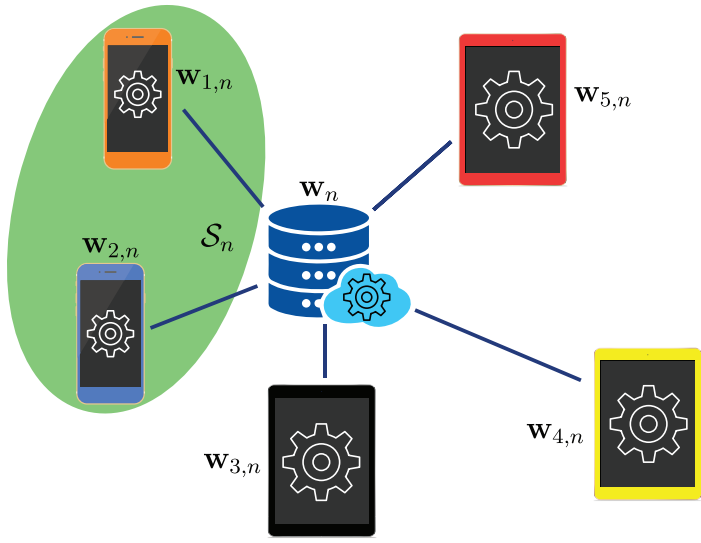

Figure 2.2: Conventional federated learning structure.

server aggregates these local models into a new global model, often by employing a weighted average, and repeats this process until a specific convergence criterion is met. The FedAvg algorithm is summarized in the form of a pseudo-code in Algorithm 2. A conventional FL setting is illustrated in Figure 2.2. We can observe from Figure 2.2 that the only connections in a federated network are between the clients and the central server. In addition, we can observe that in this figure, the set of clients that are communicating to the server is $\mathcal{S}_n \in \{1, 2\}$.

FL can be categorized into two types based on the participating entities and the scale of training: cross-silo FL and cross-device FL [57]. In cross-silo FL, a small number of trusted participants, typically organizations or institutions such as hospitals, banks, and research laboratories, engage in the training process. Each silo has access to a large local dataset and possesses strong computational and networking resources. The environment is stable, characterized by high client availability and reliable connections. This type of FL is particularly useful in scenarios where data privacy, governance, and regulatory compliance are essential, such as in healthcare and finance. On the other hand, cross-device FL involves a vast number of participants, including smartphones, IoT devices, and edge sensors. In this case, each device contributes a small amount of data and has limited computational power and intermittent connectivity. The environment is dynamic, with devices frequently joining and leaving the network. This approach emphasizes efficient communication, privacy (such as through differential privacy), and resilience to

Algorithm 2: Conventional Federated Learning.

Parameters: penalty parameter ρ, number of clients K, set of clients $\mathcal{S}$
Initialization: global model $\mathbf{w}_0$, local models $\mathbf{w}_{k,0}$
For $n = 1, \ldots,$ *Until Convergence*
    The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{w}_n$ to them.
Local Update at Client k:
    If $k \in \mathcal{S}_n$
        Receive $\mathbf{w}_n$ from the server.
        Update the local model.
        Send $\mathbf{w}_{k,n+1}$ to the server.
    EndIf
Aggregation at the Server:
    The server receives $\mathbf{w}_{k,n+1}$ from $k \in \mathcal{S}_n$ and aggregates them via
    $\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} \mathbf{w}_{k,n+1}$.
EndFor

unreliable devices [58].

2.3.1 Advantages of Federated Learning

FL exhibits several key advantages, such as [59, 60]:

- Enhanced privacy and security: FL keeps raw data on client devices, eliminating the need for centralized data storage and reducing the risk of data breaches. This facilitates security in sensitive sectors such as healthcare and finance. Additionally, differential privacy can be incorporated into the FL framework, adding noise to model updates before sharing them to provide guarantees against the extraction of individual data points.

- Reduced communication compared to DL: In FL, only model updates are transmitted between clients and the central server. Moreover, FL can also employ various communication-efficient techniques to further reduce the communication load.

- Leveraging computational power of clients: FL performs model training locally on clients and harnesses the computational power of these clients. In this way, FL reduces the computational load on the central server, which aggregating

updates from clients is its primary task.

2.3.2 Challenges of Federated Learning

In addition to challenged mentioned for DL, FL faces some other challenges, such as [60, 61]:

- Data heterogeneity: The heterogeneity in data arises from the non-independent and identically distributed (non-IID) nature of data or imbalances among client datasets used in training the global model. Diversity in data distribution across participants can significantly impact model convergence and performance.
- System heterogeneity: Device heterogeneity refers to the disparities in storage capacity, energy resources, computational power, and communication capabilities among participating clients. Variations in device characteristics can affect the efficiency of the FL process.
- Personalization: There is a need to balance the global model performance with local adaptations for different data distributions and various clients. In essence, there is a trade-off between a generally applicable model and one that performs well for individual clients.

2.4 Online Federated Learning

In an IoT network, where devices continuously generate or sense data, real-time processing and learning can be essential. However, conventional FL is typically batch-oriented, meaning that it relies on predefined data collections that are processed at certain intervals. While this approach works well for scenarios with static data, it struggles in dynamic environments where new data constantly arrives and the relevance of older data diminishes. Consequently, developing online algorithms is essential, as real-time data processing is often crucial in real-world dynamic settings. Therefore, Online FL (Online-Fed) addresses these limitations by enabling continuous learning from streaming data [62, 63].

In Online-Fed, clients receive data streams and perform local training at each iteration using only the most recent data. After updating their local models, clients share their models with the central server, which then aggregates the local models to form a new global model. This global model is then being broadcasted to all clients by the server and replaces their previous local models. To improve communication efficiency in an Online FL setup, the server is allowed to randomly select only a subset of the clients in every iteration. During a global iteration n, the set of randomly chosen client indices is denoted by $\mathcal{S}_n$. All clients can be selected

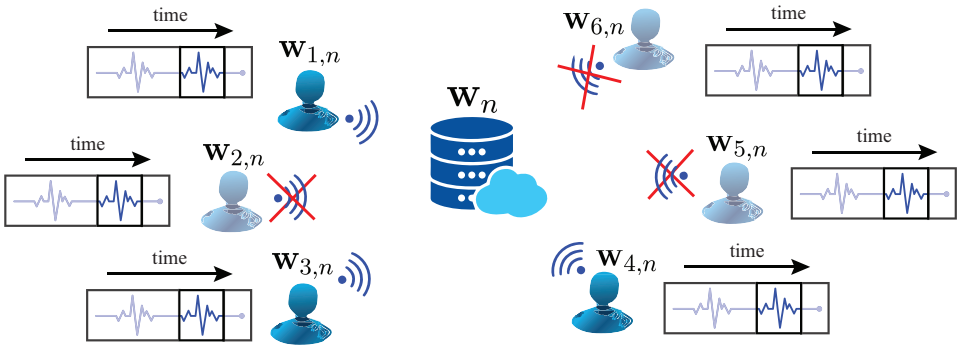

Figure 2.3: Online federated learning.

with the probability $p_c = \frac{\mathcal{C}}{K}$, where $\mathcal{C} = |\mathcal{S}_n|$ denotes the cardinality of the set $\mathcal{S}_n$. Subsequently, the server shares the global model $\mathbf{w}_n$ with the selected clients $k \in \mathcal{S}_n$. The selected clients then update their local models $\mathbf{w}_{k,n}$. The procedure of Online-Fed is illustrated in Figure 2.3. From Figure 2.3, it can be observed that during iteration n, the set of randomly selected clients is $\mathcal{S}_n = \{1, 3, 4\}$ and the rest of the clients are not communicating with the server. Furthermore, data continuously streams at each client over time. It is essential to note that local training utilizing the most recent data continues regardless of the server's selection of the corresponding client. The selection process solely influences the communication aspect of the learning procedure. This reflects in Figure 2.3 by the fact the clients $\{2, 5, 6\}$ still receive data.

2.5 Partial Sharing of Information

To further improve communication efficiency, one can share only a part of the model instead of sharing it entirely. This approach is particularly beneficial when the model is large and sharing the full model can be costly. However, the drawback of partial sharing is slower convergence and an increased error compared to the scenario where the full model is shared [64].

To share only a fraction of the model vector, in every global iteration n, an $L \times L$ diagonal selection matrix $\mathbf{S}_{k,n}$ specifies the model parameters that will be exchanged between clients and the server. The main diagonal of $\mathbf{S}_{k,n}$ has D ones and $L - D$ zeros. The positions of ones in $\mathbf{S}_{k,n}$ specify which parameters to be exchanged. The D model parameters can either be selected in a stochastic manner or sequentially. Figure 2.4 illustrates how partial sharing works. It is observed from Figure 2.4 that, in iteration n, only the first element of the model parameter vector of

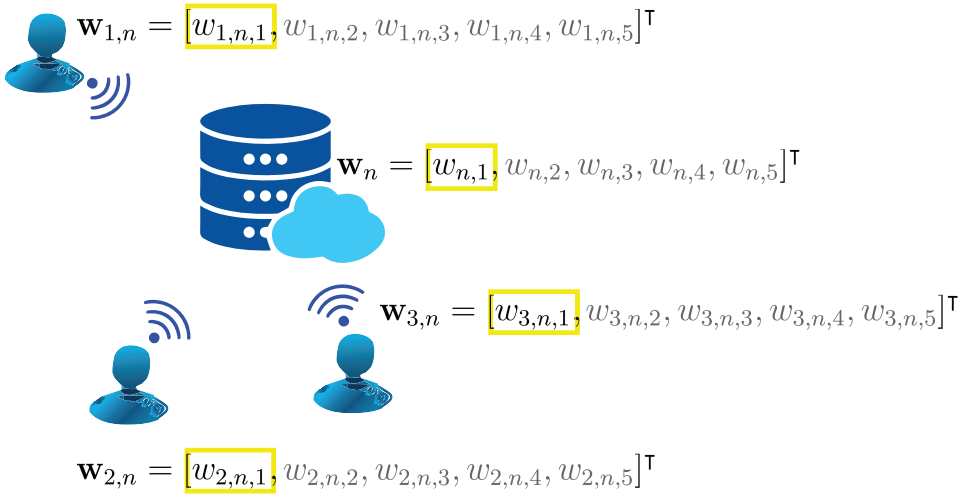

Figure 2.4: Partial sharing of information.

size $L = 5$ is shared. Hence, the selection matrix $\mathbf{S}_{k,n}$ can be written as

$$\mathbf{S}_{k,n} = \text{diag}\left\{[1, 0, 0, 0, 0]\right\}. \tag{2.1}$$

The rest of the vector comes from the clients or the server themselves. In other words, clients only share certain parts of their models, and the server does not have access to the unshared parameters. As a result, when aggregating parameters, the server substitutes the unshared parameters with the previously aggregated values. The same process applies to local aggregation. It is important to note that this approach leads to approximations that can impact performance, as analyzed for example in [64].

Therefore, the global model update $\mathbf{w}_{n+1}$ can be written as

$$\begin{aligned}\mathbf{w}_{n+1} &= \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}\mathbf{w}_{k,n+1} \\ &= \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1} + \left(\mathbf{I} - \mathbf{S}_{k,n+1}\right)\mathbf{w}_n.\end{aligned} \tag{2.2}$$

As we can observe form (2.2), in the partial sharing process, the $\left(\mathbf{I} - \mathbf{S}_{k,n+1}\right)\mathbf{w}_{k,n}$ portion is substituted with $\left(\mathbf{I} - \mathbf{S}_{k,n+1}\right)\mathbf{w}_n$ for aggregation at the server. Similarly, to perform the local model update, the same procedure happens at the client's side.

2.6 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) originated in the mid-1970s when it was introduced by Glowinski, Marrocco, Gabay, and Mercier as an extension of the method of multipliers developed in the late 1960s [65, 66]. ADMM was created to overcome the limitations of earlier augmented Lagrangian methods, particularly in addressing large-scale and structured optimization problems where variables can be updated independently.

While the fundamental concepts behind ADMM trace back even further to the 1950s, its modern form gained popularity in the 1980s and 1990s as theoretical foundations and convergence properties were established. The algorithm remained relatively unexplored until the 2010s, when advancements in distributed computing and the emergence of large-scale data challenges in fields such as machine learning, signal processing, and image reconstruction showcased ADMM's strengths in breaking down complex problems and enabling parallel computation. Nowadays, ADMM is often employed to solve large-scale optimization problems in a distributed manner [48].

ADMM addresses optimization problems of the form

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s. t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} = \mathbf{0}, \end{aligned} \tag{2.3}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times m}$, $\mathbf{c} \in \mathbb{R}^p$, and f and g are convex functions [48].

To solve this problem, ADMM employs the augmented Lagrangian of (2.3) that can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{x},\mathbf{z},\mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^\intercal(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) \\ + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \end{aligned} \tag{2.4}$$

where $\mathbf{u}$ denotes the vector of dual variables or Lagrange multipliers, and $\rho > 0$ is the penalty parameter.

ADMM solves the optimization problem (2.3) iteratively by minimizing the augmented Lagrangian (2.4) for $\mathbf{x}$ and $\mathbf{z}$ and a dual variable update alternatively as

$$\mathbf{x}_{n+1} = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{z}_n, \mathbf{u}_n) \tag{2.5a}$$

$$\mathbf{z}_{n+1} = \min_{\mathbf{z}} \mathcal{L}(\mathbf{x}_{n+1}, \mathbf{z}, \mathbf{u}_n) \tag{2.5b}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \rho(\mathbf{A}\mathbf{x}_{n+1} + \mathbf{B}\mathbf{z}_{n+1} - \mathbf{c}), \tag{2.5c}$$

Table 2.1: Comparison of fully-distributed learning and federated learning

Characteristic	Fully-distributed learning	Federated learning
Architecture	Multiple agents without a central server	Central server coordinating with multiple clients
Data Distribution	Data remains on individual agents	Data remains on clients
Communication	Agents communicate only with their neighbors	Clients communicate only with the central server
Scalability	Scalable	Scalable
Privacy	Prone to internal and external adversarial attacks	Prone to internal and external adversarial attacks
Fault Tolerance	Robust to individual node failures	Single point of failure (central server)
Coordination	Might require consensus algorithms	Simpler coordination through central server
Model Aggregation	Local aggregation	Global aggregation at the central server
Application	Ad-hoc networks	Mobile devices

where the subscript n denotes the iteration number for ADMM.

One of the key advantages of ADMM is its ability to decompose large problems into subproblems that can be solved in a parallel and distributed manner. This makes ADMM a suitable candidate for solving distributed optimization problems in this thesis.

2.7 Summary

This chapter presented background information to help comprehend the remainder of the thesis and centralize the definition of concepts used in the following chapters. A comparison between different characteristics of fully-distributed and federated learning is presented in Table 2.1. In chapter 3, the effect of partial sharing on the resilience of online FL against model-poisoning attacks is analyzed. Chapter 4 develops a noise-robust and communication-efficient ADMM-based FL. In chapter 5, a privacy-preserving distributed non-negative matrix factorization is proposed. Finally, chapter 6 proposes a noise-robust distributed maximum consensus algorithm.

Chapter 3

Resilience of Partial Sharing Online FL to Model Poisoning Attacks

This chapter, which summarizes the results of publication **P1** and its extension **P2**, proposes a thorough analysis of the impact of partial sharing on the resilience of online federated learning algorithm (PSO-Fed) against model-poisoning attacks. The PSO-Fed algorithm aims to alleviate the communication burden by enabling clients to exchange only partial model estimates with the server during each update round. This approach not only reduces communication load but also fortifies the algorithm against model-poisoning attacks. To delve deeper into this phenomenon, our analysis focuses on evaluating the performance of the PSO-Fed algorithm in the presence of Byzantine clients, who may maliciously manipulate their local models by introducing noise before sharing them with the server. Our study demonstrates that PSO-Fed sustains convergence in both mean and mean-square senses even under the adversarial influence of model-poisoning attacks. Additionally, our work entails deriving the theoretical mean square error (MSE) of PSO-Fed, establishing its correlation with various parameters including stepsize, attack probability, number of Byzantine clients, client participation rate, partial-sharing ratio, and noise variance. Furthermore, we ascertain the existence of an optimal stepsize for PSO-Fed when encountering model-poisoning attacks. The outcomes of our comprehensive numerical experiments substantiate our theoretical affirmations and underscore the superior resilience of PSO-Fed in mitigating Byzantine attacks, surpassing other leading related algorithms.

3.1 Motivation

Byzantine clients in network environments are those that behave unpredictably or with malicious intent. They have the ability to disrupt normal operations by sending misleading information or not adhering to established protocols. This type of behavior presents significant risks to the FL process and could potentially result in disruption or reduced performance [67, 68]. The presence of Byzantine clients can compromise the integrity and reliability of the global model in FL. They might employ deceptive tactics, such as submitting false or inconsistent gradients or model weights during updates or even initiating denial-of-service (DoS) attacks, all of which have the potential to undermine the integrity and effectiveness of the FL process [49, 69, 70]. Various strategies have been suggested to address the impact of Byzantine clients in FL. These strategies include using resilient aggregate statistics, assigning trust scores to individual clients, and utilizing historical information to recover from Byzantine attacks [71–74].

Byzantine clients can launch various types of adversarial attacks such as data-poisoning, model-poisoning, and label-poisoning [75, 76]. Data-poisoning attacks involve the injection of malicious data into the training dataset [77, 78]. Evasion attacks involve manipulating input data to undermine the generalizability of the learned model while evading detection [79]. Backdoor attacks involve embedding hidden trigger mechanisms within the model that can cause targeted malicious behavior [80]. Inference-time attacks, like membership inference, seek to reveal if a particular data point was utilized during training, which raises concerns about privacy [81, 82].

Network infrastructure can also be the target of distributed denial-of-service (DoS) attacks and man-in-the-middle attacks, which can disrupt services or intercept communications [83]. In addition to these types of attacks, model-poisoning attacks where Byzantine clients manipulate model updates during training as well as defense strategies against model-poisoning attacks in FL are discussed in [13]. These defense strategies can be categorized into evaluation methods for local model updates and aggregation methods for the global model. Evaluation methods involve the server analyzing client submissions without accessing local client data, which creates difficulties for their practicality. In aggregation methods, the server modifies the local model updates using specific criteria or employs appropriate statistical techniques. Utilizing robust aggregation methods like median or trimmed mean can reduce the impact of malicious clients on the global model [84]. Using methods of aggregation assisted by blockchain can also assist in coping with attacks on the network, although this may result in increased communication load or computational complexity [85].

A communication-efficient FL algorithm based on SignSGD was recently introduced in [86] that exhibits resilience to Byzantine attacks. However, it assumes that all clients consistently participate in every FL iteration, which may not reflect the practical realities of FL. This is especially true in scenarios involving clients with limited resources, such as mobile or IoT devices, in terms of energy or computational capacity. As a result, there is a significant gap in current FL research to address the necessity for communication-efficient FL algorithms that are robust against model-poisoning attacks without imposing additional computational burden on clients. Filling this gap can make FL more feasible and practical for a wider range of devices, particularly those with constrained resources, and can improve the robustness and scalability of FL systems in real-world applications.

This chapter delves into the analysis of the performance of the partial-sharing-based online FL (PSO-Fed) algorithm [63, 87, 88] in the face of Byzantine attacks. Our investigation shows that employing partial sharing in online FL, with the main goal of improving communication efficiency, also provides the additional benefit of reducing the negative impact of model-poisoning attacks. Our rigorous theoretical analysis demonstrates that PSO-Fed converges in both mean and mean-square senses, even when subjected to model-poisoning attacks. Additionally, we calculate the theoretical steady-state mean square error (MSE) of PSO-Fed. Our analysis reveals an important discovery: the identification of an optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. Unlike gradient descent algorithms, where a smaller stepsize often leads to better performance, in PSO-Fed under model-poisoning attacks, a smaller stepsize does not necessarily result in improved estimation accuracy. This contrasts with the typical behavior of gradient-descent-based algorithms, where smaller stepsizes are frequently used to improve performance, albeit at the expense of slower convergence. We present a theoretical determination of the optimal stepsize. To validate our theoretical findings, we carry out extensive numerical experiments using synthetic non-IID data. The experimental results validate our theoretical predictions and demonstrate that PSO-Fed displays greater resilience to Byzantine attacks compared to other algorithms, such as Online-FL, all without incurring additional computational overhead on clients.

3.2 Proposed Method

We consider a federated network consisting of K clients and one server. At every time instance n, each client k has access to a data vector $\mathbf{x}_{k,n} \in \mathbb{R}^L$ and its corresponding response value $y_{k,n} \in \mathbb{R}$, which are related via the model

$$y_{k,n} = \mathbf{w}^\mathsf{T}\mathbf{x}_{k,n} + \nu_{k,n}, \tag{3.1}$$

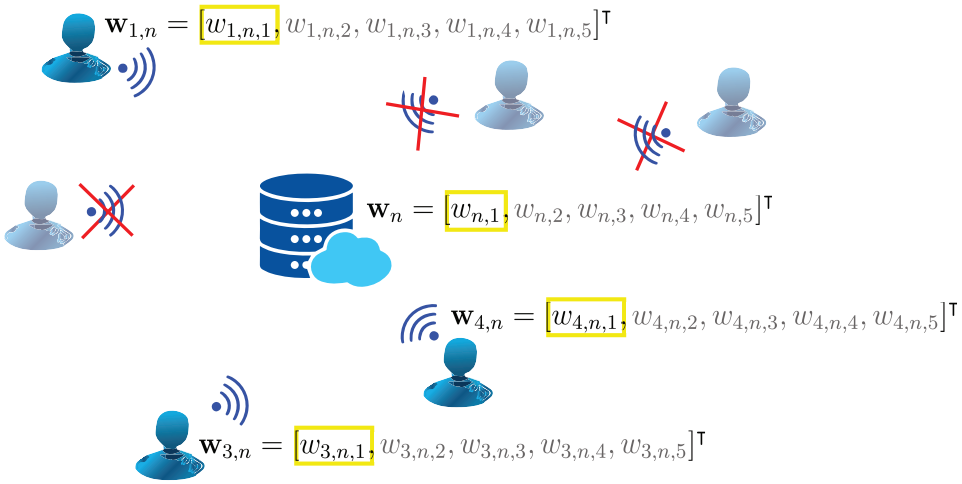

Figure 3.1: PSO-Fed algorithm.

where the model parameter vector $\mathbf{w} \in \mathbb{R}^L$ is collaboratively estimated using the locally stored client data, and $\nu_{k,n}$ is the observation noise. We define the global objective function for estimating $\mathbf{w}$ as

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}) \tag{3.2}$$

with the local objective function at client k being

$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E}\left[|y_{k,n} - \mathbf{w}^\intercal \mathbf{x}_{k,n}|^2\right]. \tag{3.3}$$

The goal is to find the optimal estimate of $\mathbf{w}$ by minimizing $\mathcal{J}(\mathbf{w})$, i.e., $\mathbf{w}^\star = \min_{\mathbf{w}} \mathcal{J}(\mathbf{w})$, albeit, in a distributed fashion via FL.

3.2.1 Partial-Sharing-Based Online FL (PSO-Fed)

In PSO-Fed, the server only sends a portion of the global model estimate to the clients as discussed in chapter 2.5 to reduce the communication load. In addition, the clients also transmit only a part of their local model estimates to the server. The model parameters exchanged between client k and the server at iteration n are specified using a diagonal selection matrix denoted by $\mathbf{S}_{k,n} \in \mathbb{R}^{L \times L}$ with M ones and $L - M$ zeros. The positions of the ones on the diagonal determine which model parameters are shared with the server at each iteration. They can be selected arbitrarily or in a round-robin fashion as in [64, 89] such that the model parameters are exchanged between each client and the server, on average, M times in every L iterations. Hence, the probability of each of the model parameters being shared

Algorithm 3: PSO-Fed.

Parameters: stepsize μ, number of clients K, set of clients $\mathcal{S}$
Initialization: global model $\mathbf{w}_0 = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \mathbf{0}$
For $n = 1, \ldots,$ *Until Convergence*
    The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the global model $\mathbf{S}_{k,n}\mathbf{w}_n$ to them.
Client Local Update:
    If $k \in \mathcal{S}_n$
        Update $\epsilon_{k,n} = y_{k,n} - [\mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n}]^\intercal\,\mathbf{x}_{k,n}$.
        Update $\mathbf{w}_{k,n+1} = \mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n} + \mu\mathbf{x}_{k,n}\epsilon_{k,n}$
        Send $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$ to the server.
    Else
        Update $\epsilon_{k,n} = y_{k,n} - \mathbf{w}_{k,n}^\intercal\mathbf{x}_{k,n}$.
        Update $\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu\mathbf{x}_{k,n}\epsilon_{k,n}$
    EndIf
Aggregation at the Server:
    The server receives $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$ from $k \in \mathcal{S}_n$ and aggregates them via
    $\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}[\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1})\,\mathbf{w}_n]$
EndFor

with the server in any iteration is denoted by $p_e = \frac{M}{L}$. Figure 3.1 illustrates how PSO-Fed operates. From Figure 3.1, it can be observed that during iteration n, the set of randomly selected clients is $\mathcal{S}_n = \{1, 3, 4\}$ and the rest of the clients are not communicating with the server. In addition, it is observed that only the first element of the model parameter vector of size $L = 5$ is shared.

Using the selection matrices $\mathbf{S}_{k,n}$, the recursive update equations of PSO-Fed, that iteratively minimize (3.2) through FL, are expressed as

$$\epsilon_{k,n} = y_{k,n} - \left[\mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n}\right]^\intercal\mathbf{x}_{k,n} \tag{3.4a}$$

$$\mathbf{w}_{k,n+1} = \mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n} + \mu\mathbf{x}_{k,n}\epsilon_{k,n} \tag{3.4b}$$

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}\left[\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1})\,\mathbf{w}_n\right], \tag{3.4c}$$

where $\mathbf{w}_{k,n}$ is the local model estimate at client k and iteration n, $\mathbf{w}_n$ is the global model estimate at iteration n, $\mathbf{I}_L$ is the $L \times L$ identity matrix, μ is the stepsize

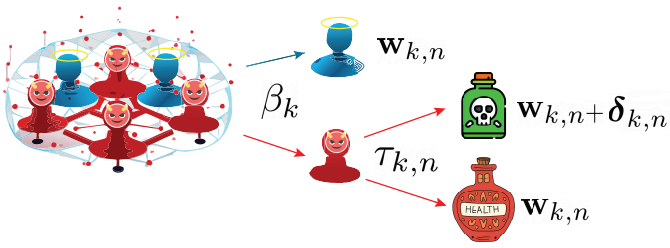

Figure 3.2: Model-poisoning attack model.

controlling the convergence rate and steady-state performance, $\mathcal{S}_n$ denotes the set of client selected at iteration n, and $|\mathcal{S}_n|$ denotes the number of the selected clients in each iteration. Algorithm 3 summarizes PSO-Fed [63].

3.2.2 Model-Poisoning Attack Model

We denote the group of potential Byzantine clients within the network as $\mathcal{S}_B$. To indicate whether a client exhibits Byzantine behavior, we utilize the indicator variable β_k, where $\beta_k = 1$ signifies that client k is part of the set $\mathcal{S}_B$ (i.e., is a Byzantine client), and $\beta_k = 0$ indicates otherwise. The total count of Byzantine clients in the network is represented by $|\mathcal{S}_B|$. These Byzantine clients aim to compromise the global model (e.g., diminish its accuracy) by intermittently transmitting corrupted local model estimates to the server. We assume that the Byzantine clients have access to accurate local model estimates. More specifically, we consider a situation in which, during each iteration, each Byzantine client intentionally alters its local model estimate by perturbing it before sending it to the server. This alteration is performed with a certain probability of attack, denoted as p_a. We represent the model update shared by each of the Byzantine clients via [12]

$$\mathbf{w}'_{k,n} = \begin{cases} \mathbf{w}_{k,n} + \boldsymbol{\delta}_{k,n} & \text{with probability } p_a \\ \mathbf{w}_{k,n} & \text{with probability } 1 - p_a, \end{cases} \tag{3.5}$$

where $\boldsymbol{\delta}_{k,n} \in \mathbb{R}^L$ denotes the perturbation signal associated with the attack which can be modeled as a zero-mean white Gaussian noise, i.e., $\boldsymbol{\delta}_{k,n} \sim \mathcal{N}(\mathbf{0}, \sigma_B^2 \mathbf{I}_L)$.

A Bernoulli random variable $\tau_{k,n}$ represents the probability of a Byzantine client compromising its local estimate. This variable equals 1 with a probability of p_a, which signifies the occurrence of an attack, and 0 otherwise. This representation is depicted in Figure 3.2. We can observe from Figure 3.2 that clients are categorized into Byzantine and non-Byzantine sets, differentiated by the parameter β_k. In addition, the probability of a Byzantine client executing an attack is represented by the random variable $\tau_{k,n}$ in Figure 3.2.

Therefore, considering that each Byzantine client sends a distorted local model estimate to the server with a probability of p_a when chosen by the server, in situations where Byzantine clients exist in the network, the global model update of PSO-Fed, as shown in (3.4c), can be rewritten as

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} \left[\mathbf{S}_{k,n+1} \mathbf{w}'_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1}) \, \mathbf{w}_n \right], \tag{3.6}$$

where

$$\mathbf{w}'_{k,n+1} = \mathbf{w}_{k,n+1} + \beta_k \tau_{k,n} \boldsymbol{\delta}_{k,n+1}. \tag{3.7}$$

3.3 Theoretical Results

In this section, we evaluate the robustness of PSO-Fed against Byzantine attacks by analyzing its theoretical mean and mean-square convergence and predicting its steady-state MSE and optimal stepsize.

To facilitate the analysis, we introduce some new quantities. We denote the extended optimal global model as $\mathbf{w}_e^\star = \mathbf{1}_{K+1} \otimes \mathbf{w}^\star$ and the extended global model estimate as $\mathbf{w}_{e,n} = \operatorname{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \ldots, \mathbf{w}_{K,n}\}$. We also define the following collective quantities

$$\mathbf{X}_n = \operatorname{bdiag}\{\mathbf{0}, \mathbf{x}_{1,n}, \ldots, \mathbf{x}_{k,n}\} \tag{3.8a}$$
$$\boldsymbol{\delta}_{e,n} = \operatorname{col}\{\mathbf{0}, \boldsymbol{\delta}_{1,n}, \ldots, \boldsymbol{\delta}_{k,n}\} \tag{3.8b}$$
$$\mathbf{B} = \operatorname{bdiag}\{\mathbf{O}_L, \beta_1 \mathbf{I}_L, \ldots, \beta_K \mathbf{I}_L\} \tag{3.8c}$$
$$\mathbf{T}_n = \operatorname{bdiag}\{\mathbf{O}_L, \tau_{1,n} \mathbf{I}_L, \ldots, \tau_{K,n} \mathbf{I}_L\} \tag{3.8d}$$
$$\boldsymbol{\nu}_{e,n} = \operatorname{col}\{0, \nu_{1,n}, \ldots, \nu_{k,n}\}, \tag{3.8e}$$

where the operators $\operatorname{col}\{\cdot\}$ and $\operatorname{bdiag}\{\cdot\}$ represent column-wise stacking and block diagonalization, respectively. Additionally, $\mathbf{1}_{K+1}$ denotes a column vector of ones with $K+1$ entries, and $\mathbf{O}_L$ denotes a square zero matrix of size L.

Subsequently, we define

$$\mathbf{y}_{e,n} = \operatorname{col}\{0, y_{1,n}, y_{2,n}, \ldots, y_{k,n}\} = \mathbf{X}_n^\intercal \mathbf{w}_e^\star + \boldsymbol{\nu}_{e,n} \tag{3.9a}$$
$$\boldsymbol{\epsilon}_{e,n} = \operatorname{col}\{0, \epsilon_{1,n}, \epsilon_{2,n}, \ldots, \epsilon_{k,n}\} = \mathbf{y}_{e,n} - \mathbf{X}_n^\intercal \boldsymbol{\mathcal{A}}_n \mathbf{w}_{e,n} \tag{3.9b}$$

and

$$\boldsymbol{\mathcal{A}}_n = \begin{bmatrix} \mathbf{I}_L & \mathbf{O}_L & \mathbf{O}_L & \ldots & \mathbf{O}_L \\ a_{1,n}\mathbf{S}_{1,n} & \mathbf{I}_L - a_{1,n}\mathbf{S}_{1,n} & \mathbf{O}_L & \ldots & \mathbf{O}_L \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k,n}\mathbf{S}_{k,n} & \mathbf{O}_L & \mathbf{O}_L & \ldots & \mathbf{I}_L - a_{k,n}\mathbf{S}_{k,n} \end{bmatrix}, \tag{3.10}$$

where $a_{k,n} = 1$ if $k \in \mathcal{S}_n$ and $a_{k,n} = 0$ otherwise. Hence, the global recursion equations of PSO-Fed can be expressed as

$$\mathbf{w}_{e,n+1} = \boldsymbol{\mathcal{B}}_{n+1} \left(\boldsymbol{\mathcal{A}}_n \mathbf{w}_{e,n} + \mu \mathbf{X}_n \boldsymbol{\epsilon}_{e,n} \right) + \boldsymbol{\mathcal{C}}_{n+1} \mathbf{B}\mathbf{T}_n \boldsymbol{\delta}_{e,n+1}, \tag{3.11}$$

where

$$\boldsymbol{\mathcal{B}}_{n+1} = \begin{bmatrix} \mathbf{I}_L - \sum_{k=1}^{K} \frac{a_{k,n}}{|\mathcal{S}_n|} \mathbf{S}_{k,n+1} & \frac{a_{1,n}}{|\mathcal{S}_n|} \mathbf{S}_{1,n+1} & \dots & \frac{a_{k,n}}{|\mathcal{S}_n|} \mathbf{S}_{K,n+1} \\ \mathbf{O}_L & \mathbf{I}_L & \dots & \mathbf{O}_L \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_L & \mathbf{O}_L & \dots & \mathbf{I}_L \end{bmatrix} \tag{3.12}$$

$$\boldsymbol{\mathcal{C}}_{n+1} = \begin{bmatrix} \mathbf{O}_L & \frac{a_{1,n}}{|\mathcal{S}_n|} \mathbf{S}_{1,n+1} & \dots & \frac{a_{k,n}}{|\mathcal{S}_n|} \mathbf{S}_{K,n+1} \\ \mathbf{O}_L & \mathbf{O}_L & \dots & \mathbf{O}_L \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_L & \mathbf{O}_L & \dots & \mathbf{O}_L \end{bmatrix}. \tag{3.13}$$

To make the analysis tractable, we adopt the following commonly used assumptions.

A1: The input vectors of each client, $\mathbf{x}_{k,n}$, originate from a wide-sense stationary multivariate random process characterized by the covariance matrix $\mathbf{R}_k = \mathbb{E}\left[\mathbf{x}_{k,n}\mathbf{x}_{k,n}^\intercal\right]$.

A2: The observation noise, $\nu_{k,n}$, and perturbation signal, $\boldsymbol{\delta}_{k,n}$, both follow identical and independent distributions. They are also independent of all other stochastic variables.

A3: The selection matrices, $\mathbf{S}_{k,n}$, of all clients and iterations are independent of each other.

3.3.1 Mean Convergence

We define the deviation (coefficient-error) vector as $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^\star - \mathbf{w}_{e,n}$. Since $\boldsymbol{\mathcal{B}}_{n+1}$ and $\boldsymbol{\mathcal{A}}_n$ are block right-stochastic, i.e., their block rows add up to identity matrix, we have $\boldsymbol{\mathcal{B}}_{n+1}\mathbf{w}_e^\star = \boldsymbol{\mathcal{A}}_n\mathbf{w}_e^\star = \mathbf{w}_e^\star$. Therefore, considering (3.11) and the definition of $\tilde{\mathbf{w}}_{e,n}$, we get

$$\begin{aligned} \tilde{\mathbf{w}}_{e,n+1} = {} & \boldsymbol{\mathcal{B}}_{n+1} \left(\mathbf{I} - \mu \mathbf{X}_n \mathbf{X}_n^\intercal \right) \boldsymbol{\mathcal{A}}_n \tilde{\mathbf{w}}_{e,n} - \mu \boldsymbol{\mathcal{B}}_{n+1} \mathbf{X}_n \boldsymbol{\nu}_{e,n} \\ & - \boldsymbol{\mathcal{C}}_{n+1} \mathbf{B}\mathbf{T}_n \boldsymbol{\delta}_{e,n+1}. \end{aligned} \tag{3.14}$$

Taking the expected value of both sides of (3.14) and under assumptions A1-A3, we obtain

$$\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] = \mathbb{E}[\boldsymbol{\mathcal{B}}_{n+1}] \left(\mathbf{I} - \mu \boldsymbol{\mathcal{R}} \right) \mathbb{E}[\boldsymbol{\mathcal{A}}_n] \mathbb{E}[\tilde{\mathbf{w}}_{e,n}], \tag{3.15}$$

where

$$\boldsymbol{\mathcal{R}} = \mathrm{bdiag}\{\mathbf{O}, \mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_K\}. \tag{3.16}$$

The mean convergence of PSO-Fed is guaranteed if $\| \left(\mathbf{I} - \mu\boldsymbol{\mathcal{R}}\right) \| < 1$ or equivalently $\|1 - \mu\lambda_i \left(\mathbf{R}_k\right) \| < 1 \; \forall k, i$. Here, $\| \cdot \|$ is any matrix norm and $\lambda_i \left(\cdot\right)$ denotes the ith eigenvalue of its matrix argument. As a result, PSO-Fed achieves convergence in the mean sense, even in the presence of Byzantine clients, provided that the stepsize μ is appropriately chosen to satisfy

$$0 < \mu < \frac{2}{\max_{i,k}\{\lambda_i \left(\mathbf{R}_k\right)\}}. \tag{3.17}$$

In other terms, PSO-Fed is unbiased, even in the presence of model-poisoning attacks.

3.3.2 Mean-Square Convergence

We denote the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ as $\|\tilde{\mathbf{w}}_{e,n}\|^2_{\boldsymbol{\Sigma}} = \tilde{\mathbf{w}}^{\intercal}_{e,n}\boldsymbol{\Sigma}\tilde{\mathbf{w}}_{e,n}$, where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix. Computing the weighted norm-square of both sides of (3.14) results in

$$\begin{aligned}\mathbb{E}[\|\tilde{\mathbf{w}}_{e,n+1}\|^2_{\boldsymbol{\Sigma}}] = \mathbb{E}[\|\tilde{\mathbf{w}}_{e,n}\|^2_{\boldsymbol{\Sigma}'}] + \mu^2\mathbb{E}[\boldsymbol{\nu}^{\intercal}_{e,n}\mathbf{Y}_n\boldsymbol{\nu}_{e,n}] \\ + \mathbb{E}[\boldsymbol{\delta}^{\intercal}_{e,n+1}\mathbf{U}_n\boldsymbol{\delta}_{e,n+1}],\end{aligned} \tag{3.18}$$

where the cross terms vanish under assumption A2 and

$$\boldsymbol{\Sigma}' = \mathbb{E}[\boldsymbol{\mathcal{A}}^{\intercal}_n\left(\mathbf{I} - \mu\mathbf{X}_n\mathbf{X}^{\intercal}_n\right)\boldsymbol{\mathcal{B}}^{\intercal}_{n+1}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{n+1}\left(\mathbf{I} - \mu\mathbf{X}_n\mathbf{X}^{\intercal}_n\right)\boldsymbol{\mathcal{A}}_n] \tag{3.19}$$

$$\mathbf{Y}_n = \mathbf{X}^{\intercal}_n\boldsymbol{\mathcal{B}}^{\intercal}_{n+1}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{n+1}\mathbf{X}_n \tag{3.20}$$

$$\mathbf{U}_n = \mathbf{T}^{\intercal}_n\mathbf{B}^{\intercal}\boldsymbol{\mathcal{C}}^{\intercal}_{n+1}\boldsymbol{\Sigma}\boldsymbol{\mathcal{C}}_{n+1}\mathbf{B}\mathbf{T}_n. \tag{3.21}$$

Subsequently, we define

$$\boldsymbol{\sigma'} = \mathrm{bvec}\{\boldsymbol{\Sigma}'\} = \boldsymbol{\mathcal{F}}^{\intercal}\boldsymbol{\sigma} \tag{3.22}$$

$$\boldsymbol{\sigma} = \mathrm{bvec}\{\boldsymbol{\Sigma}\} \tag{3.23}$$

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} - \mu\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} + \mu^2\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} \tag{3.24}$$

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}} = \mathbb{E}[\boldsymbol{\mathcal{B}}_{n+1} \otimes_b \boldsymbol{\mathcal{B}}_{n+1}] \tag{3.25}$$

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} = \mathbb{E}[\boldsymbol{\mathcal{A}}_n \otimes_b \boldsymbol{\mathcal{A}}_n] \tag{3.26}$$

$$\boldsymbol{\mathcal{K}} = (\mathbf{I} \otimes_b \boldsymbol{\mathcal{R}}) + (\boldsymbol{\mathcal{R}} \otimes_b \mathbf{I}) \tag{3.27}$$

$$\boldsymbol{\mathcal{H}} = \mathbb{E}[\mathbf{X}_n \mathbf{X}_n^\intercal \otimes_b \mathbf{X}_n \mathbf{X}_n^\intercal] \tag{3.28}$$

$$\boldsymbol{\mathcal{D}} = \begin{bmatrix} \boldsymbol{\mathcal{K}}/2 & -\boldsymbol{\mathcal{H}}/2 \\ \mathbf{I} & \mathbf{O} \end{bmatrix} \tag{3.29}$$

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}} = \mathbb{E}[\boldsymbol{\mathcal{C}}_{n+1} \otimes_b \boldsymbol{\mathcal{C}}_{n+1}] \tag{3.30}$$

$$\boldsymbol{\omega} = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}} \mathrm{bvec}\{\boldsymbol{\Omega}_{\boldsymbol{\delta}}\}, \tag{3.31}$$

$$\boldsymbol{\Omega}_{\boldsymbol{\delta}} = \mathrm{bdiag}\{0, \beta_1 \sigma_B^2 p_a \mathbf{I}_D, \beta_2 \sigma_B^2 p_a \mathbf{I}_D, \cdots, \beta_K \sigma_B^2 p_a \mathbf{I}_D\} \tag{3.32}$$

$$\boldsymbol{\phi} = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}} \boldsymbol{\phi}_{\boldsymbol{\nu}} \tag{3.33}$$

$$\boldsymbol{\phi}_{\boldsymbol{\nu}} = \mathrm{bvec}\{\mathbb{E}\left[\mathbf{X}_n \boldsymbol{\Theta}_{\boldsymbol{\nu}} \mathbf{X}_n^\intercal\right]\} \tag{3.34}$$

$$\boldsymbol{\Theta}_{\boldsymbol{\nu}} = \mathbb{E}\left[\boldsymbol{\nu}_{e,n} \boldsymbol{\nu}_{e,n}^\intercal\right] = \mathrm{diag}\{0, \sigma_{\nu_1}^2, \cdots, \sigma_{\nu_K}^2\}, \tag{3.35}$$

where $\otimes_b$ denotes the block Kronecker product (Tracy–Singh product [90]) and $\mathrm{bvec}\{\cdot\}$ denotes the block vectorization operation [91]. We evaluate $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}$, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}$, $\boldsymbol{\mathcal{H}}$, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}}$ and $\boldsymbol{\phi}_{\boldsymbol{\nu}}$ under A1-A3, in Appendices A-D in **P2**, respectively.

Proposition 3.1: PSO-Fed converges in the mean-square sense and has a bounded steady-state MSD, even in the presence of model-poisoning attacks given an appropriate choice of stepsize as

$$0 < \mu < \min\left\{ \frac{1}{\lambda_{\max}\left(\boldsymbol{\mathcal{K}}^{-1}\boldsymbol{\mathcal{H}}\right)}, \frac{1}{\max\{\lambda(\boldsymbol{\mathcal{D}}) \in \mathbb{R}, 0\}} \right\}, \tag{3.36}$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of its matrix argument. Note that we denote the upper bound of (3.36) as $\mu_{\max}$.

Proof. See **P2**.

3.3.3 Steady-State Mean Square Error

Proposition 3.2: The steady-state MSE of PSO-Fed (denoted by $\mathcal{E}$), while taking into account the impact of model-poisoning attacks can be computed as [92]

$$\begin{aligned} \mathcal{E} &= \frac{1}{K}\left[(\mu^2 \boldsymbol{\phi}^\intercal + \boldsymbol{\omega}^\intercal)\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1} \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal \mathrm{bvec}\{\boldsymbol{\mathcal{R}}\} + \mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)\right] \\ &= \underbrace{\frac{\mu^2}{K} \boldsymbol{\phi}^\intercal \left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1} \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal \mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}_{\mathcal{E}_{\boldsymbol{\phi}}} \\ &\quad + \underbrace{\frac{1}{K} \boldsymbol{\omega}^\intercal \left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1} \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal \mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}_{\mathcal{E}_{\boldsymbol{\omega}}} + \underbrace{\frac{1}{K} \mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)}_{\mathcal{E}_{\boldsymbol{\Theta}}}, \end{aligned} \tag{3.37}$$

Proof. See **P2**.

The partial sharing of model parameters, the scheduling of clients, the statistics of input data, and the value of stepsize all affect the first and second terms on the right-hand side of (3.37), specifically $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{E}_{\boldsymbol{\omega}}$. The observation noise impacts the first and third terms, $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{E}_{\boldsymbol{\Theta}}$, with the latter being solely attributable to this noise. Additionally, the influence of model-poisoning attacks from Byzantine clients on the steady-state MSE of PSO-Fed is restricted to the second term, $\mathcal{E}_{\boldsymbol{\omega}}$, which is induced by these attacks.

Remark 3.1: Reflecting on (3.31) and the computation of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}}$ in Appendix C of **P2**, it is evident that both partial sharing and client scheduling have a reducing impact on $\boldsymbol{\omega}$. As the probability of sharing each element of the model parameter vector decreases, denoted as $p_e = \frac{M}{L}$, the values of $\boldsymbol{\omega}$ decrease correspondingly. Consequently, (3.37) implies that partial sharing could provide online FL with increased resilience against model-poisoning attacks. Nonetheless, partial sharing also influences $\mathcal{E}_{\boldsymbol{\phi}}$, causing its value to rise as indicated in [63, Figure 1(a)]. This complicates the ability to easily ascertain the overall influence of partial sharing on the steady-state MSE of PSO-Fed in situations related to model-poisoning attacks, solely by examining (3.37). To achieve a more thorough understanding and substantiate these theoretical predictions, we conduct extensive simulations in section 3.4.

3.3.4 Optimal Stepsize

When there are no model-poisoning attacks and $\mathcal{E}_{\boldsymbol{\omega}} = 0$, the function $\mathcal{E}$ increases monotonically with the stepsize μ, provided that (3.36) holds true. However, in the presence of model-poisoning attacks, the presence of $\mathcal{E}_{\boldsymbol{\omega}}$ disrupts this monotonic behavior. It amplifies $\mathcal{E}$ for smaller values of μ and moves the point of minimal $\mathcal{E}$ from $\mu = 0$ to a larger value of $\mu > 0$. As a result, this introduces a non-trivial optimal stepsize that achieves the minimum $\mathcal{E}$ when confronted with model-poisoning attacks. We refer to this optimal stepsize as $\mu^\star$.

Theorem 3.1. The approximate optimal stepsize as

$$\mu^\star \approx \frac{\boldsymbol{\omega}^\intercal \mathbf{B}_{1,J} \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal \mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}{2(\boldsymbol{\phi}^\intercal \mathbf{B}_{0,J} + \boldsymbol{\omega}^\intercal \mathbf{B}_{2,J}) \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal \mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}. \tag{3.38}$$

Note that, when no model-poisoning attack occurs, i.e., $\boldsymbol{\omega} = 0$, the numerator on the RHS of (3.38) becomes zero, leading to $\mu^\star = 0$.

Proof. See **P2**.

3.4 Numerical Results

To verify our theoretical findings, we conduct several numerical experiments. We also compare the performance of the PSO-Fed algorithm with that of the Online-Fed, SignSGD, CS-Fed, and QS-Fed algorithms within a federated network comprising $K \in \{50, 100\}$ clients and a model parameter vector of size $L = 5$. Each client possesses non-IID data vectors $\mathbf{x}_{k,n}$ and their associated response values $y_{k,n}$, which are interconnected as per (3.1) with model parameter vector $\mathbf{w} = \frac{1}{\sqrt{L}}[1, \cdots, 1]^\intercal \in \mathbb{R}^L$. Each entry of $\mathbf{x}_{k,n}$ for each client k is drawn from a zero-mean Gaussian distribution with variance ς_K^2, where ς_K^2 itself is sampled from a uniform distribution between 0.2 and 1.2, denoted as $\mathcal{U}(0.2, 1.2)$. In addition, the observation noise $\nu_{k,n}$ is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$ drawn from $\mathcal{U}(0.005, 0.025)$. In our experiments, only $M = 1$ entry of the model parameter vector is shared per iteration, unless specified differently.

We evaluate the performance of the considered algorithms on the server's side by employing a test dataset consisting of $N_t = 50$ instances $\{\breve{\mathbf{X}}, \breve{\mathbf{y}}\}$ and calculating the test MSE at the server as

$$\frac{1}{N_t} \|\breve{\mathbf{y}} - \breve{\mathbf{X}}^\intercal \mathbf{w}_n\|_2^2. \tag{3.39}$$

We also evaluate the performance of PSO-Fed on the client side using (3.4a) and calculating the network-wide average steady-state MSE as

$$\frac{1}{K} \sum_{k=1}^{K} \lim_{n\to\infty} \epsilon_{k,n}^2. \tag{3.40}$$

In our first experiment, we examine the effect of different numbers of Byzantine clients $|\mathcal{S}_B|$ on the steady-state test MSE for PSO-Fed, Online-Fed, SignSGD, CS-Fed, and QS-Fed. In this experiment, We have $K = 100$ clients. In every iteration n, the server randomly selects a set of $|\mathcal{S}_n| = 5$ clients to participate in FL, with each client having an equal probability of being selected. In addition, all Byzantine clients are characterized by model-poisoning noise variance of $\sigma_B^2 = 0.25$ and attack probability of $p_a = 1$. In each iteration, we allocate the same communication budget to all algorithms. Therefore, to allow all algorithms converge to their lowest steady-state MSD within 3000 iterations, we set $\mu = 0.15$ for PSO-Fed, Online-Fed, CS-Fed and QS-Fed, and $\mu = 0.08$ for SignSGD at all clients. According to (3.36), the maximum stepsize ensuring the mean-square convergence of PSO-Fed in this experiment is $\mu_{max} = 0.245$. The results displayed in Figure 3.3 indicate that PSO-Fed outperforms the existing algorithms across all considered number of Byzantine clients. Notably, Online-Fed and PSO-Fed perform similarly

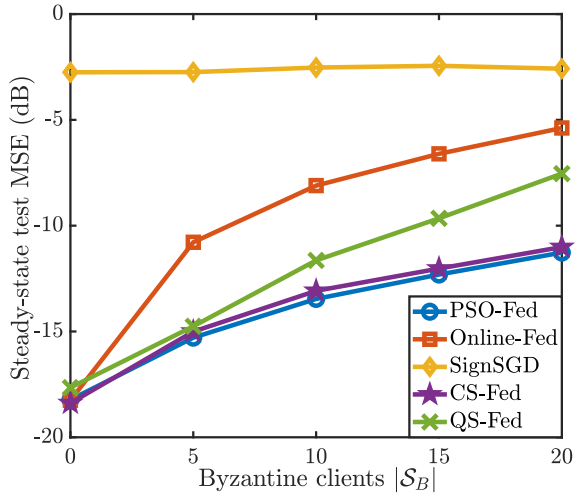

Figure 3.3: Steady-state test MSE for different algorithms with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 1$.

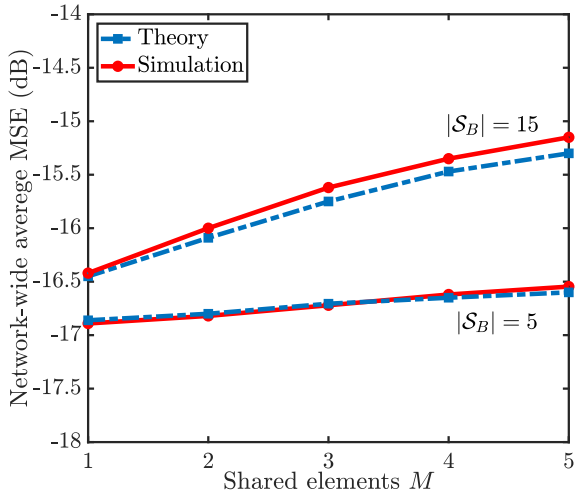

Figure 3.4: Steady-state test MSE of PSO-Fed for different numbers of shared elements M with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

when there is no Byzantine client or poisoning attack (i.e., $\sigma_B^2 = 0$). However, the performance of Online-Fed deteriorates more than PSO-Fed as the number of Byzantine clients increases. While CS-Fed performs closely to PSO-Fed in our experiment, it is important to note that the computational complexity of CS-Fed exceeds that of PSO-Fed.

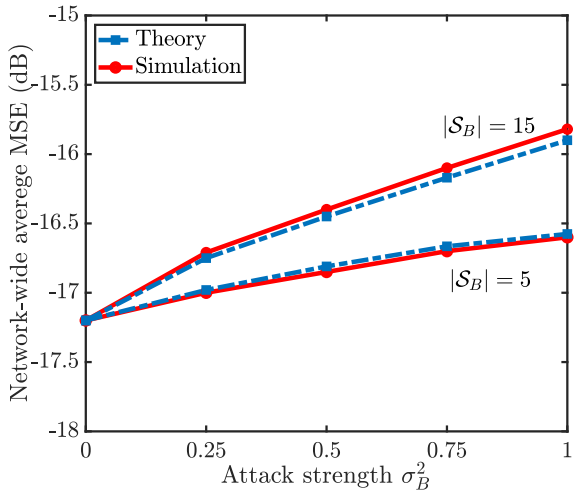

Figure 3.5: Network-wide average MSE of PSO-Fed for different values of attack strengths σ_B^2 and Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack probability $p_a = 0.2$.

In our second experiment, we examine the impact of the number of shared entries, M, on the steady-state test MSE of PSO-Fed. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$, and attack probability $p_a = 0.2$. The results are shown in Figure 3.5. We observe that increasing M leads to higher steady-state test MSE. In addition, PSO-Fed exhibits greater resilience to model-poisoning attacks compared to Online-Fed, without incurring any extra computational or communication burden on the clients. This experiment corroborates our theoretical findings discussed in section 3.3.3.

In our third experiment, we investigate the effect of varying the attack strength σ_B^2 on the network-wide average steady state MSE. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 \in \{0, 0.25, 0.5, 0.75, 1\}$, and attack probability $p_a = 0.2$. The server randomly selects 5 clients in each iteration. The results depicted in Figure3.4 align closely with our theoretical predictions. We notice an upward trend in the network-wide average steady-state MSE as the attack strength or the number of Byzantine clients increases. In addition, the presence of $|\mathcal{S}_B| = \alpha_1$ Byzantine clients with an attack strength of $\sigma_B^2 = \beta_1$ results in the same MSE as having $|\mathcal{S}_B| = \alpha_2$ Byzantine clients with an attack strength $\sigma_B^2 = \beta_2$, provided that the condition $\alpha_1\beta_1 = \alpha_2\beta_2$ is met.

In our fourth experiment, we explore how the attack probability p_a influences the steady-state test MSE of PSO-Fed, considering different numbers of shared entries

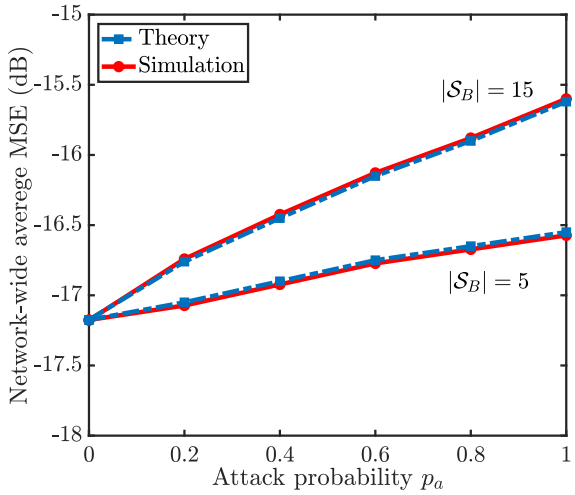

Figure 3.6: Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack strength $\sigma_B^2 = 0.25$.

$M \in \{1, 5\}$, with the number of Byzantine clients set to $|\mathcal{S}_B| = 5$ and attack strength to $\sigma_B^2 = 0.25$. The results, presented in Figure 3.7, align with observations from the previous experiment, showing that an increase in the attack probability leads to a higher MSE. Moreover, partial sharing (i.e., when $M < L$) enhances the resilience against model-poisoning attacks as elaborated upon in section 3.3.3.

In our fifth experiment, we investigate the impact of attack probability p_a on the steady-state test MSE of PSO-Fed, considering different numbers of Byzantine clients $|\mathcal{S}_B| \in \{5, 15\}$ and setting the attack strength as $\sigma_B^2 = 0.25$. The results shown in Figure 3.6 indicate that increasing the attack probability increases the steady-state MSE.

In our sixth experiment, we evaluate the performance of PSO-Fed using different stepsize values and Byzantine client numbers in the presence or absence of model-poisoning attacks. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, attack strength $\sigma_B^2 = 0.25$, attack probability $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results illustrated in Figure 3.8, show that with no model-poisoning attack ($|\mathcal{S}_B| = 0$), the larger the stepsize μ, the worse PSO-Fed performs, a trend consistent with gradient descent optimization methods. However, with model-poisoning attacks ($|\mathcal{S}_B| > 0$), the performance first improves with an increase in μ, then deteriorates when μ grows larger. This observation confirms the existence of an optimal stepsize $\mu^\star$ that ensures the best performance of PSO-Fed under model-poisoning attacks. By calculating the optimal stepsize using (3.38),

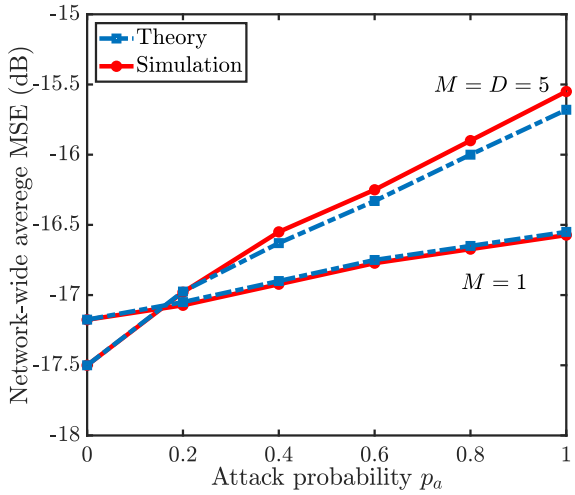

Figure 3.7: Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of shared elements $M \in \{1, 5\}$, number of Byzantine clients $|\mathcal{S}_B| = 5$ and attack strength $\sigma_B^2 = 0.25$.

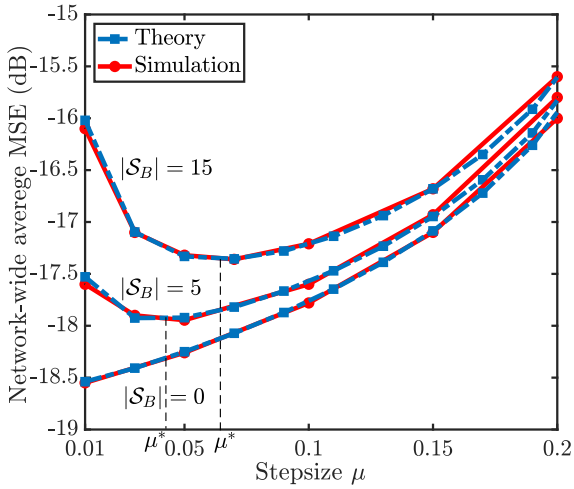

Figure 3.8: Network-wide average MSE of PSO-Fed for different values of stepsize μ, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 0.25$.

we determine $\mu^\star \approx 0.03$, which corresponds to the experimental results.

In our seventh experiment, we analyze the performance of PSO-Fed using different stepsizes in the presence or absence of model-poisoning attacks, considering different attack strengths σ_B^2. We simulate PSO-Fed with $K = 50$ cli-

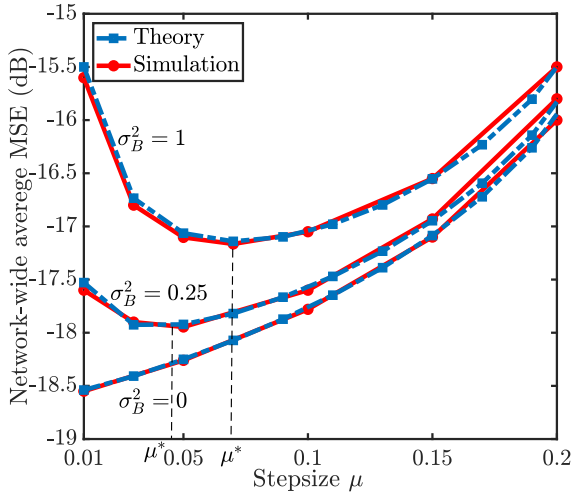

Figure 3.9: Network-wide average MSE of PSO-Fed for different values of stepsize μ, numbers of Byzantine clients $|\mathcal{S}_B| = 5$ and attack probability $p_a = 0.25$.

ents, $L = 5$, $|\mathcal{S}_B| = 5$ Byzantine clients, and attack probability of $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results, presented in Figure 3.9, echo the observations from the previous experiment: without a poisoning attack ($\sigma_B^2 = 0$), PSO-Fed's performance decreases as the stepsize μ increases. Conversely, with model-poisoning attacks ($\sigma_B^2 > 0$), performance initially improves with an increase in μ but begins to decline as μ is further increased. This pattern confirms the existence of an optimal stepsize $\mu^\star$ for scenarios with model-poisoning attacks. Using (3.38) to estimate the optimal stepsize, we find $\mu^\star \approx 0.03$, corresponding to the experiment's findings. This experiment also reinforces our conclusion following the third experiment that identical attack properties result in the same MSE.

In our eighth experiment, we investigate the impact of the commonly-adopted small stepsize approximation (SSA) on predicting the performance of PSO-Fed in the presence or absence of model-poisoning attacks. Specifically, we assume that higher orders of μ are negligible when computing $\boldsymbol{\mathcal{F}}$ via (3.24). We then compare the simulated network-wide average MSE of PSO-Fed with the corresponding theoretical predictions with or without SSA. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, attack probability $p_a = 0.25$, attack strength $\sigma_B^2 = 0.5$, and numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$. The server randomly selects 5 clients in each iteration. The results in Figure 3.10 reveal that, when μ is sufficiently small, SSA does not hinder the accurate prediction of steady-state MSE even without the knowledge of the 4th moment of the input vector, $\boldsymbol{\mathcal{H}}$. However, as the stepsize in-

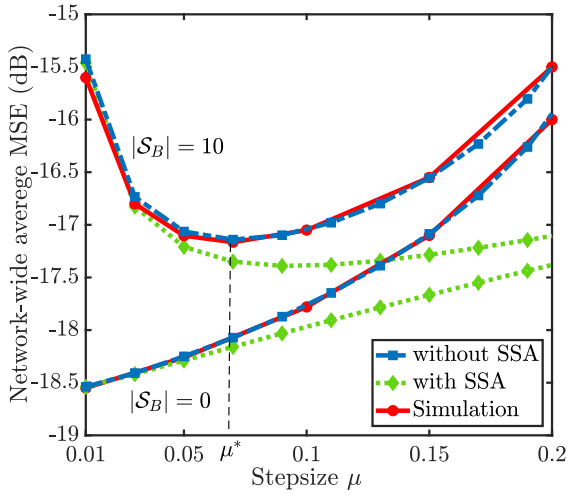

Figure 3.10: Effect of small stepsize approximation (SSA) on network-wide average MSE of PSO-Fed, numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.25$.

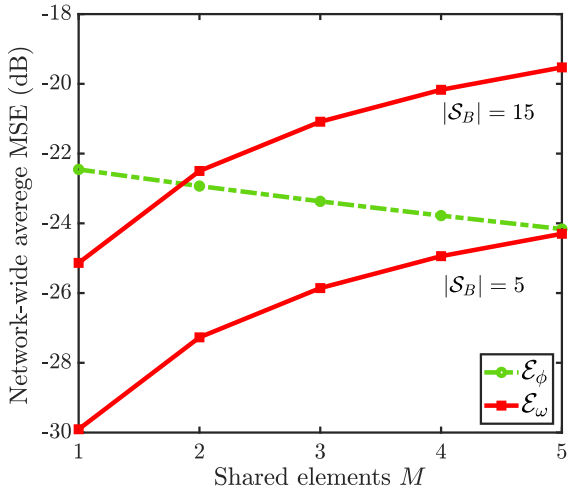

Figure 3.11: Effect of number of shared elements M on $\mathcal{E}_{\boldsymbol{\omega}}$ and $\mathcal{E}_{\boldsymbol{\phi}}$ in (3.37) for $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

creases, discrepancies arise in both scenarios, with and without model-poisoning attacks, diverging from the ideal case.

In our final experiment, we investigate the effect of partial sharing on different terms of MSE (3.37), namely, $\mathcal{E}_{\boldsymbol{\omega}}$ and $\mathcal{E}_{\boldsymbol{\phi}}$, in the presence of model-poisoning

attacks. We compute both terms using the same setup as in our second experiment. The results depicted in Figure 3.11 show that decreasing the number of shared parameters considerably lowers $\mathcal{E}_{\omega}$, significantly mitigating the impact of model-poisoning attacks. However, sharing fewer entries increases $\mathcal{E}_{\phi}$. Still, the benefits of partial sharing in reducing $\mathcal{E}_{\omega}$ are more substantial, leading to reduced MSE. Note that $\mathcal{E}_{\phi}$ remains unchanged across both considered cases, as it depends solely on scheduling and data noise. Finally, this experiment corroborates our theoretical findings and confirms our assertions regarding the effect of partial sharing on $\mathcal{E}_{\omega}$ and $\mathcal{E}_{\phi}$, as discussed at the end of section 3.3.3.

3.5 Summary

We conducted a theoretical analysis of the recently proposed PSO-Fed algorithm to examine its resilience to model-poisoning (Byzantine) attacks imparted by partial sharing. In our analysis, we considered a linear regression task with the local objective function of each client defined as an empirical risk. We showed that, even under Byzantine attacks, PSO-Fed converges in both mean and mean-square senses, given an appropriate choice of stepsize. More importantly, we showed that, in the presence of Byzantine clients, the steady-state MSE of PSO-Fed is significantly smaller than that of the Online-Fed algorithm, which does not feature partial sharing. Our theoretical analysis also uncovered the existence of a non-trivial optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. The simulation results corroborated our theoretical findings regarding PSO-Fed's effectiveness against Byzantine attacks as well as the accuracy of the theoretically predicted values of its steady-state MSE and optimal stepsize.

Chapter 4

Resource-Efficient FL Robust to Communication Errors

This chapter presents the results of publications **P3**, **P4** and their extension **P5**. It presents an FL algorithm that is robust to communication noise/error while reducing the communication burden of edge devices. We derive the proposed algorithm through solving the weighted least-squares (WLS) regression problem. We first cast the WLS regression problem as distributed convex optimization over a federated network that employs random scheduling for communication efficiency. We then use the alternating direction method of multipliers (ADMM) to solve the problem iteratively. To mitigate the adverse effects of cumulative communication noise/error, we introduce a new local model update at each participating edge device. Our theoretical analysis shows that the convergence of our algorithm is ensured when the server communicates with a random subset of the clients at each iteration. Our numerical results also demonstrate the superiority of our algorithm over the state of the art and corroborate our theoretical findings.

4.1 Motivation

FL depends on effective communication to train a machine learning model using decentralized data, aiming to create a more precise global model. When the communication channels between the server and the edge devices are unreliable, the server receives distorted local updates stemming from uplink noise, while each client gets a noisy version of the aggregated model from the server due to downlink noise [93–95]. Utilizing models that have been affected by communication noise or errors can degrade the accuracy of the learned model. Many studies on FL have mainly concentrated on uplink noise [96, 97]. The work in [98] explores

the effects of downlink noise on FL. These investigations indicate that the effectiveness of gradient-descent-based FL algorithms can decline when noise is present in the communication channels. In [94], a novel loss function is introduced for FL, employing the first-order derivative of the loss function as a regularizer to address the issue of additive noise in communication links. In [98], two strategies are suggested to enhance FL's resilience against downlink noise. The first strategy involves using a quantization technique while transmitting the global model update through digital links and implementing channel coding with a standard rate. The second strategy centers on an analog downlink transmission method, where the server sends an uncoded global model update.

The authors of [99, 100] suggest that by controlling the scale of the communication signal-to-noise ratio, the impact of noise can be mitigated, allowing the convergence rate of FedAvg with reliable communication links to be preserved. Nonetheless, they do not address any strategies to counteract the effects of the noise. In [101], the authors implement precoding and scaling during transmissions to reduce the detrimental impacts of noisy channels and to ensure the convergence of their algorithm. Many of these approaches to handle link noise in FL typically necessitate extra resources on the client side. This can be counterproductive, as clients in FL often function with limited resources regarding power/energy, memory, or computational ability.

FL algorithms that utilize the alternating direction method of multipliers (ADMM) tend to show a degree of resilience against additive communication noise [102]. Nevertheless, these methods necessitate that all clients take part in every round of FL, which can be unrealistic in practical situations where clients are limited in resources or consist of diverse edge devices. Thus, there is a significant need for FL algorithms that can withstand noise in communication channels while requiring minimal additional communication or computational efforts.

This chapter aims to develop a resource-efficient ADMM-based FL algorithm that is robust to communication noise/error while imposing no additional computational burden on the participating clients. We consider the presence of noise in both uplink and downlink communications. Considering the weighted least-squares (WLS) regression problem, we develop our proposed FL algorithm by iteratively solving an appropriately formulated distributed convex optimization problem via ADMM. To achieve communication efficiency, we employ random client scheduling. Furthermore, to prevent error accumulation from degrading the learning, we communicate a linear combination of the last two global model updates, as well as eliminating the dual model parameters at all participating edge devices. Through theoretical analysis, we show that the convergence of the proposed algorithm is ensured when the server chooses a random subset of the clients at each iteration,

even with noisy communication links. Our simulation results also attest to the effectiveness of our algorithm in comparison with the state of the art, as well as corroborating our theoretical findings.

It is noteworthy to mention that the emphasis on the WLS regression problem enables us to address challenges associated with random client participation in noisy FL settings. By concentrating on the WLS problem, we can guarantee mean convergence for both global and local models towards the optimal solution; an assurance that is often lacking in more generalized formulations. Furthermore, this focus allows a novel mean-square convergence analysis, as well as a closed-form expression for the mean-square error.

4.2 Proposed Method

We consider a federated network consisting of K clients and one server, where the clients communicate with the server over wireless channels. Each client k has access to a local dataset denoted by $\mathcal{D}_k = \{\mathbf{X}_k, \mathbf{y}_k\}$, which comprises a column response vector $\mathbf{y}_k$ with d_k entries, and a data matrix $\mathbf{X}_k$ of size $d_k \times L$. For each client k, a linear model is employed to relate the data matrix $\mathbf{X}_k$ to the response vector $\mathbf{y}_k$ as

$$\mathbf{y}_k = \mathbf{X}_k \boldsymbol{\omega} + \boldsymbol{\nu}_k, \tag{4.1}$$

where $\boldsymbol{\omega}$ denotes the global regression model parameter vector of size L, and $\boldsymbol{\nu}_k$ represents the observation noise or perturbation, which is a vector of size d_k with each entry assumed to be zero-mean Gaussian.

4.2.1 Federated Weighted Least-Squares Regression

Weighted least-squares (WLS) regression serves as a logical extension of least-squares regression, providing notable benefits in numerous signal processing applications, including power system state estimation [103], position estimation [104], and image noise reduction [105]. In WLS, various observations are given weights according to their reliability, enabling the model to more accurately represent the data when the quality of observations differs. This method can effectively reduce the influence of less reliable data, leading to more accurate and trustworthy models.

In federated WLS regression, the goal is to collaboratively estimate the global model parameter vector $\boldsymbol{\omega}$ by minimizing a global objective function across a federated network. This is expressed as a global WLS estimation problem in the FL

framework as

$$\begin{aligned} &\min_{\{\mathbf{w}_k\}} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}_k) \\ &\text{s.t. } \mathbf{w}_k = \mathbf{w},\ k \in \{1, 2, \cdots, K\}, \end{aligned} \tag{4.2}$$

where $\mathcal{J}_k(\mathbf{w}_k) = \|\mathbf{y}_k - \mathbf{X}_k\mathbf{w}_k\|^2_{\boldsymbol{\mathcal{W}}_k}$ is the local objective function for estimating $\boldsymbol{\omega}$ at client k, $\boldsymbol{\mathcal{W}}_k$ is the weight matrix specific to client k, $\mathbf{w}_k$ is the local model estimate at client k and $\mathbf{w}$ can be viewed as the global model estimate.

The optimal solution to (4.2), denoted by $\mathbf{w}^\star$, can be stated as

$$\mathbf{w}^\star = \left(\sum_{k=1}^{K} \mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{X}_k\right)^{-1} \left(\sum_{k=1}^{K} \mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{y}_k\right). \tag{4.3}$$

To solve (4.2) within an FL framework, we utilize the ADMM algorithm [48]. Therefore, we express the augmented Lagrangian function for (4.2) as

$$\sum_{k=1}^{K} \mathcal{L}_k(\mathbf{w}_k, \mathbf{w}, \mathbf{z}_k) = \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}_k) + \langle \mathbf{w}_k - \mathbf{w}, \mathbf{z}_k \rangle + \frac{\rho}{2}\|\mathbf{w}_k - \mathbf{w}\|_2^2, \tag{4.4}$$

where $\mathbf{z}_k$ is the Lagrange multiplier vector and and $\rho > 0$ is the penalty parameter. Consequently, we derive the recursive update equations at each client k and iteration number n as

$$\mathbf{z}_{k,n} = \mathbf{z}_{k,n-1} + \rho(\mathbf{w}_{k,n} - \mathbf{w}_k) \tag{4.5a}$$

$$\mathbf{w}_{k,n+1} = \hat{\mathbf{w}}_k - \mathbf{N}_k(\mathbf{z}_{k,n} - \rho\mathbf{w}_k), \tag{4.5b}$$

along with the recursive update equation at the server as

$$\mathbf{w}_{n+1} = \frac{1}{K}\sum_{k=1}^{K} \left(\mathbf{w}_{k,n+1} + \rho^{-1}\mathbf{z}_{k,n}\right), \tag{4.6}$$

where we define

$$\mathbf{N}_k = \left(2\mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{X}_k + \rho\mathbf{I}\right)^{-1} \tag{4.7}$$

$$\hat{\mathbf{w}}_k = 2\mathbf{N}_k \mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{y}_k. \tag{4.8}$$

In this solution, after performing local training, i.e., (4.5a) and (4.5b), each client shares its local estimate of $\mathbf{w}_{k,n+1} + \rho^{-1}\mathbf{z}_{k,n}$ with the server. The server then obtains the global estimate as in (4.6) and broadcasts it to all clients while the FL process continues.

4.2.2 Dual Variable Elimination

We posit that, in the recursions (4.5)-(4.6), it is necessary to transmit a combination of the primal and dual model parameter estimates to the server to enable the aggregation that produces the global model estimate. Nevertheless, the dual update information can be incorporated into the primal update by carefully choosing the initial estimates and introducing a new local primal update at the clients.

Accordingly, we can reformulate (4.5)-(4.6) as

$$\mathbf{w}_n = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}_{k,n} \tag{4.9a}$$

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k(2\mathbf{w}_n - \mathbf{w}_{n-1}) \tag{4.9b}$$

by initializing with $\mathbf{w}_{-1} = \mathbf{0}$, $\mathbf{z}_{k,-1} = \mathbf{0}$, and $\mathbf{w}_{k,0} = \hat{\mathbf{w}}_k$, hence eliminating the Lagrange multipliers $\mathbf{z}_{k,n}$. The recursion begins with clients sharing their $\hat{\mathbf{w}}_k$ with the server, which then aggregates them and broadcasts the resulting global model estimate to the clients. We further define $\mathbf{s}_n = 2\mathbf{w}_n - \mathbf{w}_{n-1}$ and modify (4.9b) as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\mathbf{s}_n. \tag{4.10}$$

One of the main benefits of (4.10) compared to (4.9b) is that the server carries out the linear combination of the latest two global model estimates needed for local model updates, instead of individual clients handling this task. This approach can greatly minimize the effects of communication noise since the combination takes place prior to being sent to the clients.

4.2.3 Communication Noise

Clients and the server often communicate via wireless channels, which are prone to noise in both uplink and downlink communications. In the downlink, clients receive noisy versions of the aggregated model updates from the server. Specifically, at iteration n, client k receives $\tilde{\mathbf{s}}_{k,n} = \mathbf{s}_n + \boldsymbol{\zeta}_{k,n}$ where $\boldsymbol{\zeta}_{k,n}$ represents the downlink noise affecting the transmission. In the uplink, the server receives a noisy version of each client's local model update, i.e., $\tilde{\mathbf{w}}_{k,n+1} = \mathbf{w}_{k,n+1} + \boldsymbol{\eta}_{k,n}$ where $\boldsymbol{\eta}_{k,n}$ denotes the uplink noise for client k at iteration n.

Considering the impact of communication noise and allowing client updates to occur before server aggregation, we obtain

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n} \tag{4.11a}$$

$$\mathbf{w}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\tilde{\mathbf{w}}_{k,n+1} \tag{4.11b}$$

4.3 Resource-efficient FL over Noisy Channels

In this section, we introduce RERCE-Fed, our proposed FL algorithm that is both resource-efficient and robust to communication errors.

4.3.1 Random Scheduling

In (4.11), there is an implicit assumption that all clients are involved in every global model update iteration. However, in FL, clients often have constraints in their communication and computational capabilities. Thus, requiring participation of all clients in every global update round can result in considerable drawbacks, such as extended convergence times or excessive resource consumption. To tackle this issue, we allow the server to randomly select a subset of clients, indicated by $\mathcal{S}_n$, to take part in model aggregation during each iteration n. We assume that the size of this subset, $\mathcal{C} = |\mathcal{S}_n|$, remains unchanged throughout the FL process.

4.3.2 RERCE-Fed

In our proposed algorithm, during each global iteration n, the selected clients $k \in \mathcal{S}_n$ receive $\tilde{\mathbf{s}}_{k,n}$ from the server and update their models. The server subsequently receives $\tilde{\mathbf{w}}_{k,n+1}$ from these clients and aggregates them. Afterward, it broadcasts the most recent global update to a new set of selected clients in the next iteration. Clients that are not selected by the server in a specific round keep their most recent local update until their next selection. Hence, the recursions of the proposed RERCE-Fed algorithm are expressed as [106, 107]

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - a_{k,n}\rho\mathbf{N}_k)\,\mathbf{w}_{k,n} + a_{k,n}\rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n} \tag{4.12a}$$

$$\mathbf{w}_{n+1} = \frac{1}{\mathcal{C}}\sum_{k=1}^{K} a_{k,n}\tilde{\mathbf{w}}_{k,n+1}, \tag{4.12b}$$

where $a_{k,n}$ is the indicator variable for random scheduling, with $a_{k,n} = 1$ when client k is selected by the server in iteration n (i.e., $k \in \mathcal{S}_n$) and $a_{k,n} = 0$ otherwise. We summarize RERCE-Fed in Algorithm 4.

4.3.3 RERCE-Fed with Continual Local Updates

In contrast to the traditional random scheduling method outlined in section 4.3.1, where clients that are not selected refrain from making local updates and their most recent model estimates are excluded from the global aggregation, we introduce a novel approach in which all clients, irrespective of their selection status, consistently revise their local model estimates during each iteration. This innovative strategy can improve overall performance without adding any extra communication overhead or significantly increasing the computational burden for either the clients or the server, as will be demonstrated later.

Algorithm 4: RERCE-Fed.

Parameters: penalty parameter ρ, number of clients K, set of clients $\mathcal{S}$
Initialization: global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \hat{\mathbf{w}}_k$
For $n = 1, \ldots,$ *Until Convergence*
    The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{s}_n$ to them.
Client Local Update:
    If $k \in \mathcal{S}_n$
        Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of $\mathbf{s}_n$, from the server.
        Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k)\mathbf{w}_{k,n} + \rho \mathbf{N}_k \tilde{\mathbf{s}}_{k,n}$
        Send $\mathbf{w}_{k,n+1}$ to the server.
    EndIf
Aggregation at the Server:
    The server receives $\tilde{\mathbf{w}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them via
    $\mathbf{w}_{n+1} = \frac{1}{C} \sum_{k \in \mathcal{S}_n} \tilde{\mathbf{w}}_{k,n+1}$
    $\mathbf{s}_{n+1} = 2\mathbf{w}_{n+1} - \mathbf{w}_n$
EndFor

To implement this method, clients retain the latest global model estimate they receive from the server, while the server maintains the most up-to-date local model estimates from all clients. As a result, clients are able to continually refine their local models using the most current global model estimate, and the server updates the global model by incorporating the latest local updates from every client, regardless of the random scheduling. When a client is chosen at iteration n, its most recent local model estimate is synchronized with the server, and the global model estimate provided by the server replaces the previous version held by the client.

Therefore, the recursions of the RERCE-Fed algorithm with continual local updates are given by [106, 108]

$$\begin{aligned} \mathbf{w}_{k,n+1} = &(\mathbf{I} - \rho \mathbf{N}_k)\mathbf{w}_{k,n} \\ &+ \rho \mathbf{N}_k \left[a_{k,n} \tilde{\mathbf{s}}_{k,n} + (1 - a_{k,n}) \tilde{\mathbf{s}}_{k,m} \right] \end{aligned} \tag{4.13a}$$

$$\mathbf{w}_{n+1} = \frac{1}{K} \sum_{k=1}^{K} \left[a_{k,n} \tilde{\mathbf{w}}_{k,n+1} + (1 - a_{k,n}) \tilde{\mathbf{w}}_{k,m} \right], \tag{4.13b}$$

Algorithm 5: RERCE-Fed with continual local updates.

Parameters: penalty parameters ρ, number of clients K, set of clients $\mathcal{S}$
Initialization: global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \hat{\mathbf{w}}_k$
For $n = 1, \ldots,$ *Until Convergence*
    The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{s}_n$ to them.
Client Local Update:
    If $k \in \mathcal{S}_n$
        Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of $\mathbf{s}_n$, from the server.
        Store the latest global model $\tilde{\mathbf{s}}_{k,m} = \tilde{\mathbf{s}}_{k,n}$.
        Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n}$
        Send $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$ to the server.
    Else
        Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,m}$
    EndIf
Aggregation at the Server:
    The server receives $\tilde{\mathbf{t}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them with $\tilde{\mathbf{t}}_{k,m}$, the stored local model estimates of the non-selected clients via
    $\mathbf{s}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left[a_{k,n}\tilde{\mathbf{t}}_{k,n+1} + (1 - a_{k,n})\tilde{\mathbf{t}}_{k,m}\right]$
EndFor

where $\tilde{\mathbf{s}}_{k,m}$ denotes the most recent global model estimate received from the server and stored in client k. This estimate is utilized when the client is not selected by the server. Additionally, $\tilde{\mathbf{w}}_{k,m}$ represents the most recent local model estimate associated with client k, which is stored at the server and utilized during iterations when the client is not chosen through random scheduling. Defining $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$, we can restate (4.13b) as

$$\mathbf{s}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left[a_{k,n}\tilde{\mathbf{t}}_{k,n+1} + (1 - a_{k,n})\tilde{\mathbf{t}}_{k,m}\right]. \tag{4.14}$$

We summarize RERCE-Fed with continual local updates in Algorithm 5.

4.4 Theoretical Results

In this section, we examine the theoretical performance of RERCE-Fed. We demonstrate that the iterates $\mathbf{w}_{k,n}$ converge in both mean and mean-square senses as $n \to \infty$, even when there are noisy communication links. Since $\mathbf{w}_n$ denotes the average of client estimates $\mathbf{w}_{k,n}$, its convergence follows accordingly.

We define the extended optimal global model as $\mathbf{w}_e^\star = 1_{2K} \otimes \mathbf{w}^\star$ and the vector containing the client local model estimates as

$$\mathbf{w}_{e,n} = \mathrm{col}\{\mathbf{w}_{1,n}, \cdots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \cdots, \mathbf{w}_{K,n-1}\},$$

where 1_{2K} is the $2K \times 1$ vector of all ones, $\otimes$ is the Kronecker product, and $\mathrm{col}\{\cdot\}$ denotes column-wise stacking.

Substituting (4.12b) into (4.12a), the global recursion of the proposed algorithm can be stated as

$$\mathbf{w}_{e,n+1} = \boldsymbol{\mathcal{A}}_n \mathbf{w}_{e,n} + \boldsymbol{\zeta}_n + \boldsymbol{\eta}_n, \tag{4.15}$$

where

$$\boldsymbol{\mathcal{A}}_n = \begin{bmatrix} \boldsymbol{\mathcal{A}}_{n,1} & \boldsymbol{\mathcal{A}}_{n,2} \\ \boldsymbol{\mathcal{A}}_{n,3} & \boldsymbol{\mathcal{A}}_{n,4} \end{bmatrix}, \tag{4.16}$$

and the extended noise vectors $\boldsymbol{\zeta}_n$ and $\boldsymbol{\eta}_n$ stack the vectors

$$a_{k,n}\rho \mathbf{N}_k \boldsymbol{\zeta}_{k,n} \tag{4.17a}$$

$$a_{k,n}\frac{\rho}{\mathcal{C}}\mathbf{N}_k \sum_{j=1}^{K} \left(2a_{j,n-1}\boldsymbol{\eta}_{j,n-1} - a_{j,n-2}\boldsymbol{\eta}_{j,n-2}\right) \tag{4.17b}$$

at their top halves, respectively, and zeros at their bottom halves. The value of $\boldsymbol{\mathcal{A}}_n \in \mathbb{R}^{2LK \times 2LK}$ depends on the iteration number n as the server selects a random number of clients at each iteration. Its sub-matrices of size $LK \times LK$ are block matrices whose $L \times L$ blocks are calculated as

$$[\boldsymbol{\mathcal{A}}_{n,1}]_{ii} = \mathbf{I} - a_{i,n}\rho \mathbf{N}_i + 2a_{i,n}a_{i,n-1}\frac{\rho}{\mathcal{C}}\mathbf{N}_i \tag{4.18a}$$

$$[\boldsymbol{\mathcal{A}}_{n,1}]_{ij} = 2a_{i,n}a_{j,n-1}\frac{\rho}{\mathcal{C}}\mathbf{N}_i, \; i \neq j \tag{4.18b}$$

$$[\boldsymbol{\mathcal{A}}_{n,2}]_{ij} = -a_{i,n}a_{j,n-2}\frac{\rho}{\mathcal{C}}\mathbf{N}_i. \tag{4.18c}$$

To make the analysis tractable, we adopt the following assumptions:

A1: The communication noise vectors of both uplink and downlink, $\boldsymbol{\eta}_{k,n}$ and $\boldsymbol{\zeta}_{k,n}$ $\forall k, n$, are independently and identically distributed. In addition, they are independent of each other and all other stochastic variables.

A2: The random scheduling variables, $a_{k,n}\ \forall k,n$, are independent and follow the same Bernoulli distribution with parameter $\bar{a}=\frac{\mathcal{C}}{K}$, i.e., $a_{k,n}=1$ with probability $\bar{a}$ and $a_{k,n}=0$ with probability $1-\bar{a}$.

4.4.1 Mean Convergence

Proposition 4.1: RERCE-Fed algorithm is unbiased and converges in the mean sense such that

$$\lim_{n\to\infty}\mathbb{E}[\mathbf{w}_{e,n}]=\mathbf{w}_e^\star. \tag{4.19}$$

Proof. See **P5**.

4.4.2 Mean-Square Convergence

We define the deviation vector as $\tilde{\mathbf{w}}_{e,n}=\mathbf{w}_{e,n}-\mathbf{w}_e^\star$. Since $\boldsymbol{\mathcal{A}}_n$ is block right-stochastic, i.e., its block rows add up to the identity matrix, we have $\boldsymbol{\mathcal{A}}_n\mathbf{w}_e^\star=\mathbf{w}_e^\star$. Therefore, by defining the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ as $\|\tilde{\mathbf{w}}_{e,n}\|_{\boldsymbol{\Sigma}}^2=\tilde{\mathbf{w}}_{e,n}^\intercal\boldsymbol{\Sigma}\tilde{\mathbf{w}}_{e,n}$, where $\boldsymbol{\Sigma}$ is an arbitrary positive semi-definite matrix, we obtain the variance relation as

$$\mathbb{E}[\|\tilde{\mathbf{w}}_{e,n+1}\|_{\boldsymbol{\Sigma}}^2]=\mathbb{E}[\|\tilde{\mathbf{w}}_{e,n}\|_{\boldsymbol{\Sigma}'}^2]+\mathbb{E}[\boldsymbol{\zeta}_n^\intercal\boldsymbol{\Sigma}\boldsymbol{\zeta}_n]+\mathbb{E}[\boldsymbol{\eta}_n^\intercal\boldsymbol{\Sigma}\boldsymbol{\eta}_n], \tag{4.20}$$

where the cross terms vanish under assumption A1.

Subsequently, we define

$$\boldsymbol{\Sigma}'=\mathbb{E}[\boldsymbol{\mathcal{A}}_n^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{A}}_n] \tag{4.21}$$
$$\boldsymbol{\sigma'}=\mathrm{bvec}\{\boldsymbol{\Sigma}'\}=\boldsymbol{\mathcal{Q}}\boldsymbol{\sigma} \tag{4.22}$$
$$\boldsymbol{\sigma}=\mathrm{bvec}\{\boldsymbol{\Sigma}\} \tag{4.23}$$
$$\boldsymbol{\mathcal{Q}}=\mathbb{E}[\boldsymbol{\mathcal{A}}_n^\intercal\otimes_b\boldsymbol{\mathcal{A}}_n^\intercal] \tag{4.24}$$
$$\mathbb{E}[\boldsymbol{\zeta}_n^\intercal\boldsymbol{\Sigma}\boldsymbol{\zeta}_n]=\boldsymbol{\phi}^\intercal\boldsymbol{\sigma} \tag{4.25}$$
$$\mathbb{E}[\boldsymbol{\eta}_n^\intercal\boldsymbol{\Sigma}\boldsymbol{\eta}_n]=\boldsymbol{\varphi}^\intercal\boldsymbol{\sigma} \tag{4.26}$$
$$\boldsymbol{\phi}=\mathrm{bvec}\{\mathbb{E}[\boldsymbol{\zeta}_n\boldsymbol{\zeta}_n^\intercal]\} \tag{4.27}$$
$$\boldsymbol{\varphi}=\mathrm{bvec}\{\mathbb{E}[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^\intercal]\} \tag{4.28}$$
$$\mathbb{E}[\boldsymbol{\zeta}_n\boldsymbol{\zeta}_n^\intercal]=\bar{a}\rho^2\mathrm{bdiag}\{\sigma_{\zeta_1}^2\mathbf{N}_1^2,\cdots,\sigma_{\zeta_K}^2\mathbf{N}_K^2,\underbrace{\mathbf{O},\cdots,\mathbf{O}}_{K}\} \tag{4.29}$$
$$\mathbb{E}[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^\intercal]=\frac{5\rho^2}{K^2}\sum_{k=1}^{K}\sigma_{\eta_k}^2\mathrm{bdiag}\{\mathbf{N}_1^2,\cdots,\mathbf{N}_K^2,\underbrace{\mathbf{O},\cdots,\mathbf{O}}_{K}\}, \tag{4.30}$$

where $\otimes_b$ denotes the block Kronecker product, $\mathrm{bvec}\{\cdot\}$ denotes the block vectorization operation, $\mathrm{bdiag}\{\cdot\}$ denotes the block diagonalization, $\sigma_{\zeta_k}^2$ denotes the

variance of the downlink noise for client k and $\sigma_{\eta_k}^2$ denotes the variance of the uplink noise for client k. Note that in (4.30), $\mathbb{E}[a_{i,n}a_{j,n}] \ll 1 \; \forall i \neq j$, and can be neglected. Note that we evaluate $\boldsymbol{\mathcal{Q}}$ in Appendix A in **P5**.

Utilizing the above results, we can write the global recursion for the weighted mean-squared error (MSE) of RERCE-Fed as

$$\mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right] = \mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\mathcal{Q}}^n\boldsymbol{\sigma}\}}\right] + \boldsymbol{\psi}^{\intercal}\sum_{i=0}^{n-1}\boldsymbol{\mathcal{Q}}^i\boldsymbol{\sigma}, \tag{4.31}$$

where $\boldsymbol{\psi} = \boldsymbol{\phi} + \boldsymbol{\varphi}$ and $\mathrm{bvec}^{-1}\{\cdot\}$ denotes the reverse operation of block vectorization.

Using the Jordan canonical form of $\boldsymbol{\mathcal{Q}}$, we have

$$\boldsymbol{\mathcal{Q}}^i = \boldsymbol{\mathcal{U}}\boldsymbol{\mathcal{J}}^i\boldsymbol{\mathcal{U}}^{-1} = \sum_{\ell=1}^{4L^2K^2}\mu_\ell^i\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\intercal}, \tag{4.32}$$

where μ_ℓ, $\boldsymbol{u}_\ell$ and $\boldsymbol{v}_\ell$ denote the ℓth eigenvalue of $\boldsymbol{\mathcal{Q}}$ and its corresponding right and left eigenvectors, respectively. In Appendix B in **P5**, we show that the spectral radius of $\boldsymbol{\mathcal{Q}}$ is one with the geometric and algebraic multiplicity of L^2, i.e., $\mu_\ell = 1 \; \forall \ell \in \{1, \ldots, L^2\}$.

Proposition 4.2: $\forall \ell \in \{1, \ldots, L^2\}$, we have

$$\boldsymbol{\psi}^{\intercal}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\intercal}\boldsymbol{\sigma} = 0. \tag{4.33}$$

Proof: See Appendix C in **P5**.

4.4.3 Steady-State Mean-Square Error

Setting $\boldsymbol{\sigma} = \mathrm{bvec}\{\mathbf{I}_{2KL}\}$, letting $n \to \infty$ on both sides of (4.31), and using Proposition 4.2, we compute the steady-state MSE of RERCE-Fed, denoted by $\mathcal{E}$, as

$$\mathcal{E} = \lim_{n\to\infty}\mathbb{E}\left[\tilde{\mathbf{w}}_{e,n}^{\intercal}\tilde{\mathbf{w}}_{e,n}\right] = \underbrace{\tilde{\mathbf{w}}_{e,1}^{\intercal}\boldsymbol{\Sigma}_\infty\tilde{\mathbf{w}}_{e,1}}_{\mathcal{E}_\nu} + \underbrace{\sum_{\ell=L^2+1}^{4L^2K^2}(1-\mu_\ell)^{-1}\boldsymbol{\psi}^{\intercal}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\intercal}\boldsymbol{\sigma}}_{\mathcal{E}_\psi},$$

$$\boldsymbol{\Sigma}_\infty = \mathrm{bvec}^{-1}\left\{\boldsymbol{\mathcal{Q}}^\infty\boldsymbol{\sigma}\right\} = \mathrm{bvec}^{-1}\left\{\sum_{\ell=1}^{L^2}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\intercal}\mathrm{bvec}\{\mathbf{I}_{2KL}\}\right\}. \tag{4.34}$$

Remark 4.1: The observation noise $\boldsymbol{\nu}_k$ in (4.1) and the initial client estimates $\mathbf{w}_{k,0}$ influence the first component on the right-hand side of (4.34), denoted as $\mathcal{E}_\nu$, leading to a noise floor. Furthermore, scheduling parameters such as the probability of client participation and the number of clients involved in each iteration, along with the penalty parameter, the total number of clients, the local dataset at each client, and the noise variance in uplink and downlink communications affect the second component on the right-hand side of (4.34), represented as $\mathcal{E}_\psi$. We will delve deeper into these factors through simulations in section 4.5.3.

4.5 Numerical Results

In this section, we conduct several numerical experiments to evaluate the performance of the proposed algorithm and validate our theoretical findings. We consider a federated network consisting of K clients. Each client has non-IID data $\{\mathbf{X}_k, \mathbf{y}_k\}$, where the entries of $\mathbf{X}_k$ are drawn from a normal distribution $\mathcal{N}(\mu_k, \sigma_k^2)$ with $\mu_k \in \mathcal{U}(-0.5, 0.5)$ and $\sigma_k^2 \in \mathcal{U}(0.5, 1.5)$. The local data size for each client is randomly selected from a uniform distribution, i.e., $d_k \in \mathcal{U}(50, 90)$. We set the weight matrices at each client k to the inverse covariance matrix of $\mathbf{y}_k$, i.e., $\boldsymbol{\mathcal{W}}_k = \mathbb{E}[(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])\,(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])^\intercal]^{-1}$.

The local data is related as per (4.1) given the model parameter vector $\boldsymbol{\omega}$ with its entries drawn from a normal distribution $\mathcal{N}(0, 1)$. The observation noise $\boldsymbol{\nu}_k$ for each client is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$. The additive noise in both the uplink and downlink channels is zero-mean IID white Gaussian with variances $\sigma_{\eta_k}^2$ and $\sigma_{\zeta_k}^2$, respectively. In our experiments, we set the penalty parameter as $\rho = 1$. In addition, the server randomly selects a subset of $\mathcal{C}$ clients with uniform probability in each iteration.

We evaluate the algorithm performance on the client side via the normalized MSE (NMSE) defined at iteration n as

$$\frac{1}{K}\sum_{k=1}^{K}\frac{\|\mathbf{w}_{k,n} - \mathbf{w}^\star\|_2^2}{\|\mathbf{w}^\star\|_2^2}. \tag{4.35}$$

We average the NMSE learning curves over 100 independent trials to obtain our simulation results.

We outline the findings of our numerical experiments in three subsections. In the first subsection, we assess the effectiveness of the RERCE-Fed algorithm in comparison to its predecessors. The second subsection focuses on evaluating RERCE-Fed with continual local updates and comparing it to RERCE-Fed without continual local updates. Finally, in the last subsection, we validate our theoretical results by illustrating mean convergence and comparing theoretical predictions with

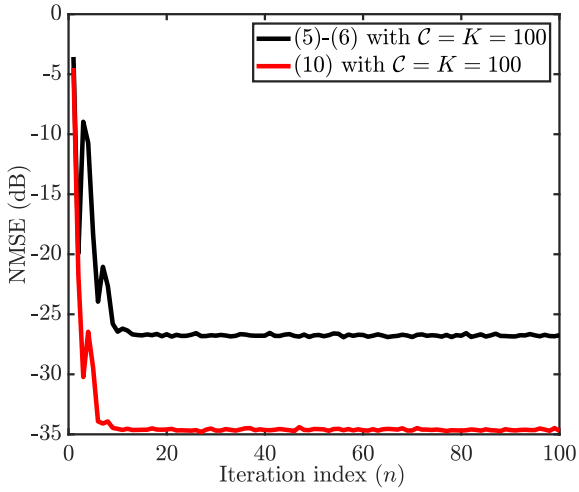

Figure 4.1: NMSE of (4.5)-(4.6) and (4.10) for $\mathcal{C} = K = 100$.

simulation results, confirming the accuracy of our theoretical expression for MSE of RERCE-Fed.

4.5.1 Performance of RERCE-Fed

In our first experiment, we simulate the algorithms described by (4.5)-(4.6) and (4.10) to solve the considered WLS problem. We run these simulations with $K = 100$ clients, model parameter vector size of $L = 128$, and link noise variance of $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 6.25 \times 10^{-4}$. All clients participate in the FL process, i.e., $\mathcal{C} = K = 100$. The corresponding learning curves are shown in Figure 4.1. We observe that (4.10) exhibits a 7dB improvement over (4.5)-(4.6) in the presence of noisy communication links when all clients are involved in each iteration of the FL process.

In our second experiment, we examine the performance of (4.5)-(4.6) and (4.10) under similar conditions as the first experiment, but with the server randomly selecting a subset of clients to participate in each iteration. We simulate (4.5)-(4.6) with $\mathcal{C} = 4$ and (4.10) with $\mathcal{C} \in \{4, 75, 90\}$. The corresponding learning curves are shown in Figure 4.2. Unlike the first experiment, Figure 4.2 illustrates that (4.10) fails to converge due to error accumulation, even when a majority of clients participate in every FL round, as observed for $\mathcal{C} \in \{75, 90\}$. Additionally, the performance of (4.5)-(4.6) degrades significantly when only a small subset of clients participate in each FL round. Consequently, both (4.5)-(4.6) and (4.10) exhibit an inability to cope with additive noise in communication links when the server selects only a subset of clients in each iteration.

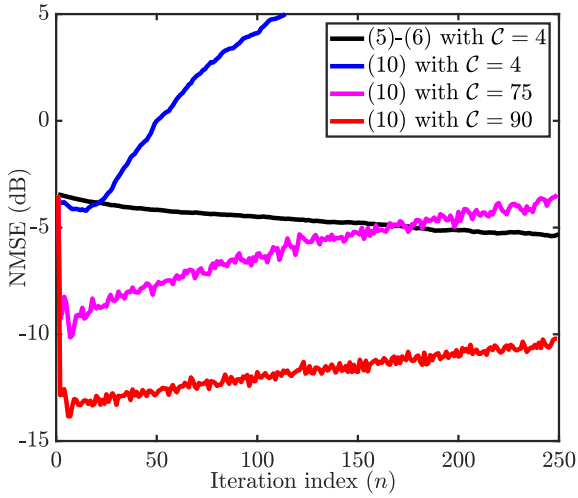

Figure 4.2: NMSE of (4.5)-(4.6) with $\mathcal{C} = 4$ and (4.10) with $\mathcal{C} \in \{4, 75, 90\}$.

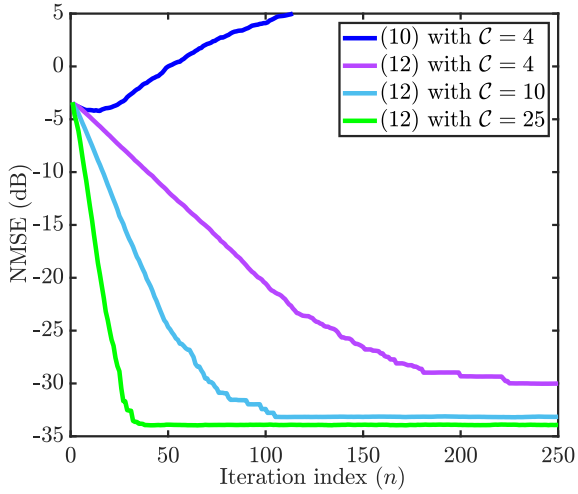

Figure 4.3: NMSE of (4.10) with $\mathcal{C} = 4$ and RERCE-Fed (4.12) for different numbers of participating clients $\mathcal{C} \in \{4, 10, 25\}$.

In our third experiment, we assess the performance of the proposed RERCE-Fed algorithm in the presence of link noise and random client scheduling by the server to enhance communication efficiency. We simulate RERCE-Fed with $K = 100$ clients, $L = 128$, link noise variance of $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 6.25 \times 10^{-4}$, and different numbers of participating clients $\mathcal{C} \in \{4, 10, 25\}$. The corresponding learning curves are depicted in Figure 4.3. As shown in Figure 4.3, RERCE-Fed exhibits robustness against communication noise, even when only a subset of clients par-

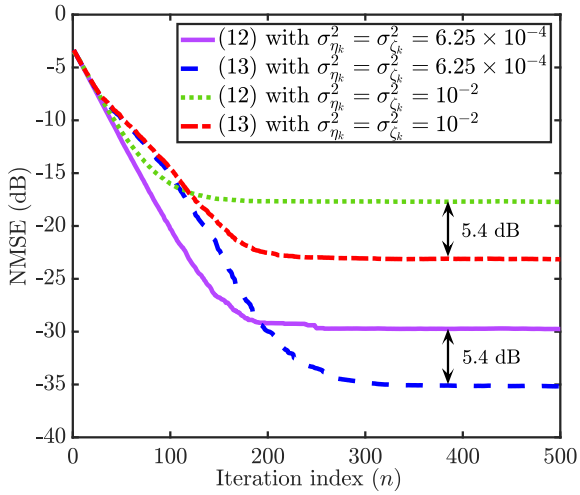

Figure 4.4: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 4$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

ticipate in each FL round, a contrast to the results observed with (4.5)-(4.6) and (4.10) in the second experiment. Another important observation from Figure 4.3 is the trade-off between the number of participating clients $\mathcal{C}$ and the performance and convergence rate of RERCE-Fed. Specifically, increasing the number of participating clients leads to faster convergence and lower NMSE. However, this benefit diminishes once $\mathcal{C} \geq 10$, as the performance of RERCE-Fed with $\mathcal{C} \geq 10$ approaches that of the scenario where all clients participate in each FL round. This implies that RERCE-Fed can achieve accurate model parameter estimation while making more efficient use of available communication resources, even in the presence of noisy communication links.

4.5.2 Performance of RERCE-Fed with Continual Local Updates

In our fourth experiment, we compare the performance of RERCE-Fed with and without continual local updates by plotting their corresponding learning curves given different numbers of participating clients and link noise variances. We present the results in Figures 4.4-4.6. The number of clients selected at each iteration is $\mathcal{C} \in \{4, 10, 25\}$, and the link noise variances are $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$. From these figures, we observe that RERCE-Fed with continual local updates consistently exhibits robustness against communication noise, even when only a small subset of clients participate in every FL round. In addition, RERCE-Fed with continual local updates significantly outperforms its counterpart

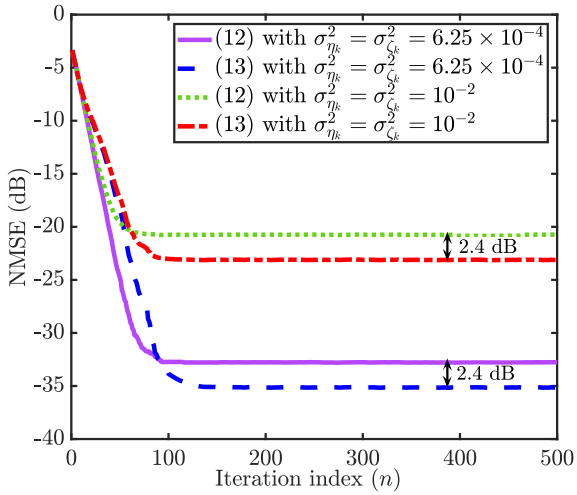

Figure 4.5: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 10$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

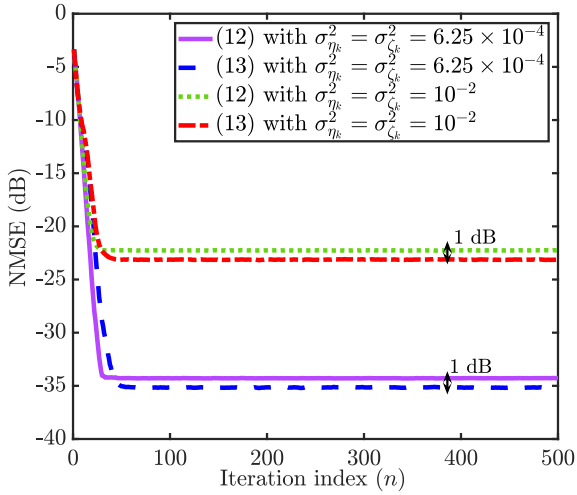

Figure 4.6: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 25$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

without continual local updates in all considered scenarios. These results indicate that allowing clients to continually update their local models, even when not selected by the server, leads to a substantial improvement in steady-state NMSE without adversely impacting the convergence rate. Additionally, as anticipated, increasing

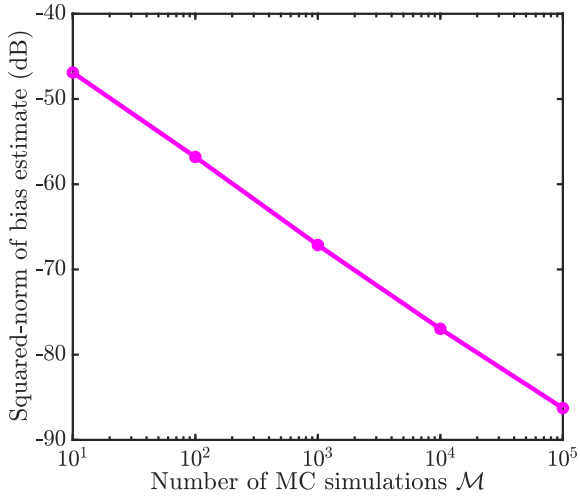

Figure 4.7: Squared-norm of bias estimate of RERCE-Fed (4.12) with $K = 6$, $L = 6$, $\mathcal{C} = 3$, and $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 10^{-4}$ for different numbers of MC runs $\mathcal{M} \in \{10, 10^2, 10^3, 10^4, 10^5\}$.

the link noise variance results in performance degradation.

4.5.3 Comparisons of Theory and Experiment

In our fifth experiment, we demonstrate the mean convergence of RERCE-Fed. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$, and link noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 10^{-4}$. In addition, the server randomly selects $\mathcal{C} = 3$ clients in each iteration. In Figure 4.7, we plot the squared ℓ_2-norm of the global model parameter bias estimated for different numbers of Monte Carlo (MC) runs, denoted by $\mathcal{M}$, specifically $\frac{1}{L}\|\frac{1}{\mathcal{M}}\sum_{i=1}^{\mathcal{M}} \mathbf{w}_n^{(i)} - \mathbf{w}^\star\|_2^2$, where $\mathbf{w}_n^{(i)}$ is the global model parameter estimate of the ith MC run. We can observe from Figure 4.7 that the squared-norm of the multivariate bias estimate decreases log-linearly with the number of MC runs, indicating unbiasedness.

In our sixth experiment, we validate the accuracy of our theoretical expression for the steady-state MSE of RERCE-Fed in (4.34) and explore the impact of varying uplink and downlink noise variances, $\sigma^2_{\eta_k}$ and $\sigma^2_{\zeta_k}$, on performance. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$ parameters, and different values for uplink and downlink noise variances. The server randomly selects $\mathcal{C} = 3$ clients in each iteration. We present the theoretical predictions of steady-sate MSE using (4.34) alongside the corresponding experimental values in Figures 4.8-4.9 as functions of $\sigma^2_{\eta_k}$, $\sigma^2_{\zeta_k}$, and $\mathcal{C}$. The results show a close alignment between theory and experiment. Furthermore, we observe an upward trend in the steady-state NMSE

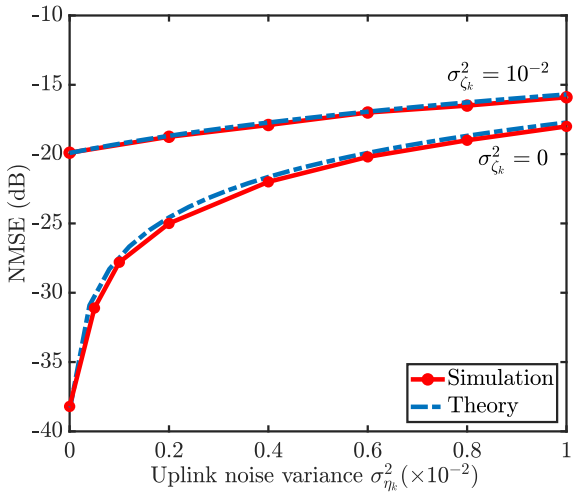

Figure 4.8: NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different uplink noise variances.

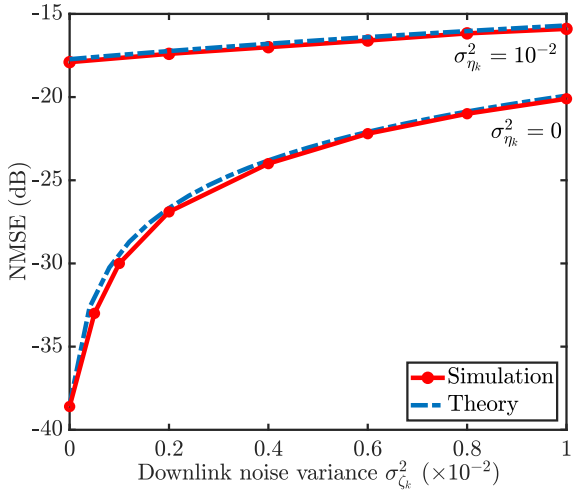

Figure 4.9: NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different downlink noise variances.

as either uplink or downlink noise variance increases.

The uplink and downlink noise variances exhibit distinct effects depending on the number of participating clients $\mathcal{C}$. While the error induced by uplink noise remains constant, the impact of downlink noise intensifies with an increasing number of participating clients. This observation is consistent with the intuition that averaging the model parameter estimates at the server can mitigate the adverse effect

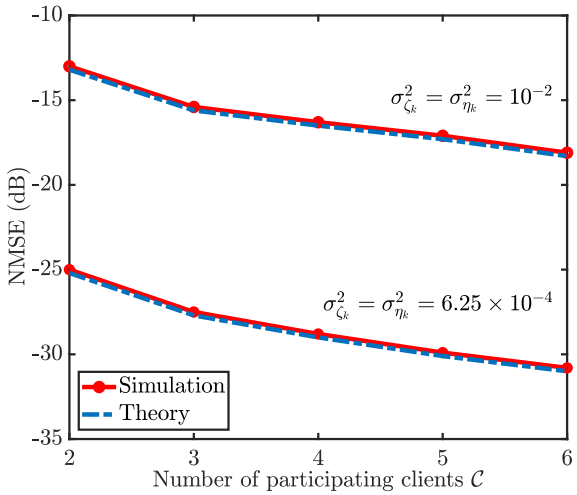

Figure 4.10: NMSE of RERCE-Fed (4.12) for different numbers of participating clients $\mathcal{C} \in \{2, 3, 4, 5, 6\}$ and different link noise variances.

of uplink noise on the performance of RERCE-Fed. However, changing $\mathcal{C}$ also affects $\boldsymbol{\mathcal{Q}}$. To obtain a better understanding, we investigate the impact of $\mathcal{C}$ on the performance of RERCE-Fed in our final experiment, where we consider $K = 6$, $L = 6$, $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$, and $\mathcal{C} \in \{2, 3, 4, 5, 6\}$. The results presented in Figure 4.10 illustrate that increasing the number of participating clients improves the performance of RERCE-Fed. However, this improvement comes at the cost of higher resource utilization on the client side, which can be limiting in real-world FL scenarios.

4.6 Summary

We proposed RERCE-Fed, an FL algorithm designed to effectively reduce communication load while maintaining robustness against additive communication noise or errors. By employing ADMM to solve the WLS problem, we introduced a new local model update at the clients. This innovative solution mitigates the effects of communication noise without imposing additional computational burden on clients. Furthermore, we improved the communication efficiency by randomly selecting a subset of clients to participate in each learning round. To further optimize performance, we enabled non-selected clients to continue with their local updates, resulting in a modified version of RERCE-Fed. Our theoretical analysis confirmed the convergence of RERCE-Fed in both mean and mean-square sense, even with random client scheduling and communication over noisy communication links. In addition, we derived a closed-form expression for the steady-state

MSE of the RERCE-Fed algorithm. Comprehensive numerical analysis substantiated our findings and confirmed the accuracy of our theoretical predictions.

Chapter 5

Privacy-Preserving Nonnegative Matrix Factorization

This chapter, which outlines the findings of publication **P6**, introduces a privacy-preserving algorithm designed for fully-distributed nonnegative matrix factorization (NMF). This algorithm aims to decompose a large data matrix into left and right matrix factors in a distributed manner while ensuring the privacy of each agent's local data. It enables collaborative estimation of the left matrix factor among agents and allows them to estimate their respective right factors without compromising raw data privacy. To achieve data privacy, the algorithm employs the Paillier cryptosystem, a probabilistic asymmetric algorithm for public-key cryptography that supports computations on encrypted data without decryption, to secure information exchanges between neighboring agents. Simulation results based on synthetic and real-world datasets highlight the efficacy of the proposed algorithm in achieving privacy-preserving distributed NMF over ad-hoc networks.

5.1 Motivation

Nonnegative Matrix Factorization (NMF) [109–113] is a powerful technique in linear algebra and machine learning. NMF is a specific case of constrained low-rank matrix approximation [114] and a linear dimensionality reduction (LDR) technique aimed at representing nonnegative data more compactly through nonnegative factors. Formally, given a matrix $\mathbf{Z}$ with dimensions $N \times M$, NMF aims to find two matrices $\mathbf{X}$ with dimensions $N \times K$ and $\mathbf{Y}$ with dimensions $K \times M$ such that $\mathbf{Z} \approx \mathbf{XY}$, where all elements in $\mathbf{Z}$, $\mathbf{X}$, and $\mathbf{Y}$ are nonnegative. The dimension K, typically chosen to be smaller than both N and M, represents the number of latent features or components in the factorization. This decomposition allows for a

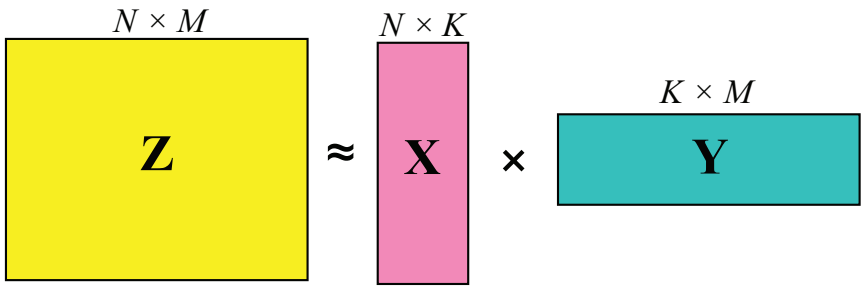

Figure 5.1: Illustration of nonnegative matrix factorization.

parts-based representation of the data, where each column of $\mathbf{X}$ can be interpreted as a basis vector and each column of $\mathbf{Y}$ as the corresponding encoding. The interpretation of NMF results is often intuitive and domain-specific. For instance, in image processing, the columns of $\mathbf{X}$ might represent distinct facial features such as eyes, nose, and mouth when applied to a dataset of face images, while the corresponding columns in $\mathbf{Y}$ would indicate the strength of each feature's presence in a given image. In text analysis, the matrix $\mathbf{X}$ might represent topics or themes, with the matrix $\mathbf{Y}$ indicating the relevance of each topic to individual documents. Figure 5.1 provides an illustration of NMF.

NMF was originally introduced as positive matrix factorization in [115] and has gained significant research interest, particularly after being popularized by [116]. It has found widespread applications in various fields such as signal and image processing, data mining and analytics, machine learning, and federated learning. Examples include air emission control [115], visual object recognition [117], video background-foreground separation [118], spectral unmixing [119], text mining [120], blind source separation [121], clustering [122], computational biology [123], music analysis [124], molecular pattern discovery [111], efficient implementation of deep neural networks [125], and detecting malware activities [126]. Its popularity stems from its utility in identifying and extracting meaningful features from data in addition to serving as a powerful LDR technique.

NMF has also proven to be a valuable tool in recommender systems and matrix completion tasks in recent years, e.g., the Netflix challenge [127]. Recommender systems often work with user-item interaction matrices, where rows represent users, columns represent items, and entries indicate preferences or ratings. NMF can decompose this matrix into user and item factor matrices, effectively capturing latent features that explain user preferences and item characteristics. This factorization allows for the prediction of missing entries, enabling person-

alized recommendations. For example, in a movie recommendation system, the matrix $\mathbf{X}$ might represent user preferences for different movie genres or styles, while the matrix $\mathbf{Y}$ could represent how strongly each movie aligns with these latent features. The product of these matrices can then be used to predict a user's potential rating for a movie they haven't seen yet [127]. In matrix completion tasks, where the goal is to fill in missing entries in a partially observed matrix, NMF can leverage the low-rank structure and nonnegativity constraints to provide meaningful estimates for the unknown values. This approach has been successfully applied in various domains, including collaborative filtering [128] and image inpainting [129].

This chapter presents a privacy-preserving distributed NMF (PPDNMF) algorithm for scenarios where data is spread among agents in an ad-hoc network, with each agent holding part of the data matrix's columns. Our goal is to perform NMF on the entire dataset in a secure, decentralized manner. Agents collaborate to estimate the left and right factors, sharing information only with immediate neighbors over secure links, while keeping their local data private. We use the block coordinate-descent (BCD) method and alternating direction method of multipliers (ADMM) to develop the algorithm and integrate the Paillier cryptosystem for privacy preservation. Performance evaluations through simulations with synthetic and real data show that our algorithm achieves results comparable to centralized methods.

5.2 Proposed Method

As mentioned earlier, the objective of NMF is to approximate a data matrix $\mathbf{Z} \in \mathbb{R}^{N\times M}$ consisting of nonnegative entries using the product of left and right factor matrices, both with nonnegative entries. That is, $\mathbf{Z} = \mathbf{XY}$ where $\mathbf{X} \in \mathbb{R}^{N\times L}$ and $\mathbf{Y} \in \mathbb{R}^{L\times M}$, typically with $L \leq \min(N, M)$. This approximation represents the N-dimensional datapoints (columns of the data matrix) within a L-dimensional linear subspace spanned by the columns of the left factor, whose coordinates are given by the columns of the right factor. The nonnegativity constraint on the factors induces sparsity, further enhancing the compactness of the representation. Moreover, in many applications, the factors' nonnegativity is essential to their physical plausibility and intuitive interpretability.

We utilize the least-squares criterion, which is appropriate when the perturbation in the data matrix $\mathbf{Z}$ can be modeled as a Gaussian process. Therefore, the NMF problem can be formulated as

$$\begin{aligned} &\min_{\mathbf{X},\mathbf{Y}} \frac{1}{2} \|\mathbf{Z} - \mathbf{XY}\|_{\mathsf{F}}^2 \\ &\text{s.t. } \mathbf{X} \geq 0, \mathbf{Y} \geq 0, \end{aligned} \tag{5.1}$$

where $\|\cdot\|_{\mathsf{F}}$ denotes the Frobenius norm.

We consider the scenario where $\mathbf{Z}$ is distributed over a network with K agents such that we have $\mathbf{Z} = [\mathbf{Z}_1, \cdots, \mathbf{Z}_K]$ and consequently $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_K]$ with $\mathbf{Y}_k \in \mathbb{R}^{L \times M_k}$ and we have $\sum_{k=1}^{K} M_k = M$. Therefore, we rewrite (5.1) as

$$\begin{aligned} \min_{\mathbf{X}, \{\mathbf{Y}_k\}} \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{Z}_k - \mathbf{X}\mathbf{Y}_k\|_{\mathsf{F}}^2 \\ \text{s.t. } \mathbf{X} \geq 0, \mathbf{Y}_k \geq 0. \end{aligned} \tag{5.2}$$

In a fully distributed approach, every agent, indexed by k, aims to estimate $\mathbf{X}$ and its own $\mathbf{Y}_k$ using its local data $\mathbf{Z}_k$ and by exchanging information solely with its immediate neighbors through single-hop communication. To this end, we utilize the BCD algorithm and iteratively solve two optimization subproblems for $\mathbf{X}$ and $\mathbf{Y}$. That is, we repeat the following alternating minimizations until convergence is achieved:

$$\begin{aligned} \mathbf{X}_n = \min_{\mathbf{X}} \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{Z}_k - \mathbf{X}\mathbf{Y}_{k,n-1}\|_{\mathsf{F}}^2 \\ \text{s.t. } \mathbf{X} \geq 0, \end{aligned} \tag{5.3}$$

$$\begin{aligned} \mathbf{Y}_{k,n} = \min_{\mathbf{Y}_k} \frac{1}{2} \|\mathbf{Z}_k - \mathbf{X}_n\mathbf{Y}_k\|_{\mathsf{F}}^2, \forall k \in \{1, \ldots, K\} \\ \text{s.t. } \mathbf{Y}_k \geq 0. \end{aligned} \tag{5.4}$$

The subscript n denotes the estimate of its respective parameter at the nth BCD iteration.

The solution of (5.4) can be localized straightforwardly, provided that each agent has access to the estimate $\mathbf{X}_n$. To solve (5.3) in a fully distributed manner, we introduce the variable $\mathbf{X}_k$ at each agent k as a local copy of $\mathbf{X}$ and enforce it to be equal to those of the agents within the immediate neighborhood of agent k, thereby achieving consensus across the network. Thus, we reformulate (5.3) into the following equivalent form

$$\begin{aligned} \mathbf{X}_{k,n} = \min_{\mathbf{X}_k} \frac{1}{2} \|\mathbf{Z}_k - \mathbf{X}_k\mathbf{Y}_{k,n-1}\|_{\mathsf{F}}^2 + \imath(\mathbf{X}_k) \\ \text{s.t. } \mathbf{X}_k = \mathbf{X}_\ell \ \ \forall \ell \in \mathcal{N}_k, \ \forall k \in \{1, \ldots, K\} \end{aligned} \tag{5.5}$$

where $\imath(\cdot)$ denotes the indicator function accounting for the nonegativity constraint and $\mathcal{N}_k$ denotes the set of neighbors of agent k with cardinality $d_k = |\mathcal{N}_k|$.

Subsequently, we decompose and decouple the optimization problems at the agents by introducing the auxiliary variables $\mathbf{U}_k, \mathbf{S}_{k,\ell} \in \mathbb{R}^{N\times L}$ and rewriting the optimization in (5.5) as

$$\begin{aligned} \min_{\mathbf{X}_k,\mathbf{U}_k,\mathbf{S}_{k,\ell}} \quad & \frac{1}{2}\left\|\mathbf{Z}_k - \mathbf{U}_k\mathbf{Y}_{k,n-1}\right\|_{\mathsf{F}}^2 + \imath(\mathbf{X}_k) \\ \text{s. t.} \quad & \mathbf{U}_k = \mathbf{X}_k \\ & \mathbf{S}_{k,\ell} = \mathbf{U}_k \;\; \forall \ell \in \mathcal{N}_k, \; \forall k \in \{1,\ldots,K\} \\ & \mathbf{S}_{\ell,k} = \mathbf{S}_{k,\ell}. \end{aligned} \tag{5.6}$$

We can express the corresponding aggregate augmented Lagrangian function as

$$\begin{aligned} &\mathcal{L}\left(\{\mathbf{X}_k\},\{\mathbf{U}_k\},\{\mathbf{S}_{k,\ell}\},\{\mathbf{P}_k\},\{\mathbf{Q}_{k,\ell}\}\right) \\ &= \frac{1}{2}\sum_{k=1}^{K}\left\|\mathbf{Z}_k - \mathbf{U}_k\mathbf{Y}_{k,n-1}\right\|_{\mathsf{F}}^2 + \sum_{k=1}^{K}\imath(\mathbf{X}_k) \\ &+ \frac{\mu}{2}\sum_{k=1}^{K}\left\|\mathbf{X}_k - \mathbf{U}_k - \mathbf{P}_k\right\|_{\mathsf{F}}^2 \\ &+ \frac{1}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k}\rho_{k,\ell}\left\|\mathbf{U}_k - \mathbf{S}_{k,\ell} - \mathbf{Q}_{k,\ell}\right\|_{\mathsf{F}}^2, \end{aligned} \tag{5.7}$$

where μ and $\rho_{k,\ell}$ are penalty parameters and $\mathbf{P}_k, \mathbf{Q}_{k,\ell} \in \mathbb{R}^{N\times L}$ are scaled Lagrange multipliers. We maintain μ consistent across all agents and iterations. However, we allow each $\rho_{k,\ell}$, unique to the edge connecting agents k and ℓ, to vary over iterations [130]. Additionally, we consider $\rho_{k,\ell} = \rho_{\ell,k}\;\forall k,\ell$.

5.2.1 Estimating the Left Factor

Minimizing (5.7) using ADMM, which leads to the elimination of the auxiliary variables $\{\mathbf{S}_{k,\ell}\}$, yields the following iterations at each agent i as [131]

$$\mathbf{X}_{k,n,m} = \Pi_{\geq 0}\left\{\mathbf{U}_{k,m-1} + \mathbf{P}_{k,m-1}\right\} \tag{5.8}$$

$$\begin{aligned}\mathbf{U}_{k,m} = &\Big[\mathbf{Z}_k\mathbf{Y}_{k,n-1}^{\intercal} + \mu\left(\mathbf{X}_{k,n,m} - \mathbf{P}_{k,m-1}\right) \\ &+ \rho_{k,m}\left(\mathbf{U}_{k,m-1} + 2\mathbf{Q}_{k,m-1} - \mathbf{Q}_{k,m-2}\right)\Big] \\ &\times \left[\mathbf{Y}_{k,n-1}\mathbf{Y}_{k,n-1}^{\intercal} + \left(\mu + \rho_{k,m}\right)\mathbf{I}\right]^{-1}\end{aligned} \tag{5.9}$$

$$\mathbf{P}_{k,m} = \mathbf{P}_{k,m-1} - \left(\mathbf{X}_{k,n,m} - \mathbf{U}_{k,m}\right) \tag{5.10}$$

$$\mathbf{Q}_{k,m} = \mathbf{Q}_{k,m-1} + \sum_{\ell\in\mathcal{N}_k}\rho_{k,\ell,m}\left(\mathbf{U}_{\ell,m} - \mathbf{U}_{k,m}\right). \tag{5.11}$$

Here, the subscript m denotes the mth ADMM iteration index, $\Pi_{\geq 0}$ represents the projection onto the nonnegative orthant, and $(\cdot)^{\intercal}$ stands for matrix transpose. In addition, we define $\rho_{k,m} = \sum_{\ell\in\mathcal{N}_k}\rho_{k,\ell,m}$ and $\mathbf{Q}_{k,m} = \sum_{\ell\in\mathcal{N}_k}\rho_{k,\ell,m}\mathbf{Q}_{k,\ell,m}$. These ADMM iterations can be executed in a fully distributed manner, relying solely on locally available information and single-hop communications. Upon convergences of the algorithm, we utilize the latest estimates $\mathbf{X}_{k,n,m}$ for optimizing $\mathbf{Y}_k$ in the subsequent BCD iteration, i.e., $\mathbf{X}_{k,n} \leftarrow \mathbf{X}_{k,n,m}$. Note that we enforce the nonnegativity constraint and consensus simultaneously.

5.2.2 Estimating the Right Factor

Similarly, we can employ ADMM to iteratively solve (5.4) via [131]

$$\mathbf{Y}_{k,n,r} = \Pi_{\geq 0}\left\{\mathbf{V}_{k,r-1} + \mathbf{R}_{k,r-1}\right\} \tag{5.12}$$

$$\begin{aligned}\mathbf{V}_{k,r} = &\left(\mathbf{X}_{k,n}^{\intercal}\mathbf{X}_{k,n} + \eta\mathbf{I}\right)^{-1} \\ &\times \left[\mathbf{X}_{k,n}^{\intercal}\mathbf{Z}_k + \eta\left(\mathbf{Y}_{k,n,r} - \mathbf{R}_{k,r-1}\right)\right]\end{aligned} \tag{5.13}$$

$$\mathbf{R}_{k,r} = \mathbf{R}_{k,r-1} - \left(\mathbf{Y}_{k,n,r} - \mathbf{V}_{k,r}\right). \tag{5.14}$$

Here, the superscript r represents the rth ADMM iteration index and η is the penalty parameter. Once convergence is attained, we utilize the latest estimates $\mathbf{Y}_{k,n,r}$ to update $\mathbf{X}_k$ estimates in the subsequent BCD iteration, i.e., $\mathbf{Y}_{k,n} \leftarrow \mathbf{Y}_{k,n,r}$.

Note that, we employ warm start in both ADMM algorithms for estimating the left and right factors. At the onset of each BCD iteration, we initialize both ADMM inner iterations using the most recent estimates from the preceding iterations.

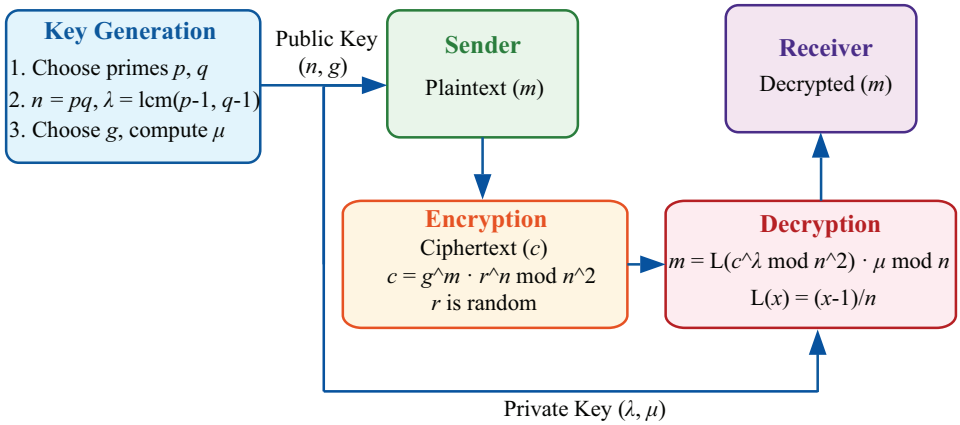

Figure 5.2: The Paillier cryptosystem flow diagram.

5.2.3 Convergence Analysis

Proposition 5.1: The ADMM iterations (5.8)-(5.11) and (5.12)-(5.14) converge using the approach proposed in [48, 132, 133].

Proof. See **P6.**

5.3 Privacy-Preserving Distributed NMF

In this section, we provide an overview of the Paillier cryptosystem, which we employ to enhance the privacy of the distributed NMF algorithm developed in section 5.2. Subsequently, we introduce our proposed privacy-preserving distributed NMF (PPDNMF) algorithm.

5.3.1 Paillier Cryptosystem

The Paillier cryptosystem, introduced by Pascal Paillier in 1999 [134], is a probabilistic asymmetric algorithm for public key cryptography. It is based on the mathematical problem of computing nth residue classes, which is believed to be computationally difficult. The system is notable for its homomorphic properties, particularly its additive homomorphism [130, 135, 136], which allows for certain operations to be performed on encrypted data without decrypting it first, i.e., $\mathcal{E}\left(m_3\left(m_1+m_2\right)\right)=\left(\mathcal{E}\left(m_1\right)\mathcal{E}\left(m_2\right)\right)^{m_3}$. The randomness in encryption ensures semantic security and makes it resistant to attacks. The Paillier cryptosystem's unique properties make it particularly useful in various applications, including e-voting systems and privacy-preserving data mining.

The Paillier cryptosystem operates as illustrated in Figure 5.2 as follows:

- Key Generation:

  1. Choose two large prime numbers p and q randomly.
  2. Confirm that $\gcd(pq, (p-1)(q-1)) = 1$, where $\gcd(\cdot,\cdot)$ is the greatest common divisor of its arguments.
  3. Compute $n = pq$ and $\lambda = \mathrm{lcm}(p-1, q-1)$, where $\mathrm{lcm}(\cdot,\cdot)$ is the least common multiple of its arguments.
  4. Select a random integer g, where $g \in \mathbb{Z}^*_{n^2}$, i.e., integers between 1 and n^2.
  5. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, where $L(x) = \frac{x-1}{n}$.
  6. The public key is (n, g) and the private key is (λ, μ).

- Encrypt a message m where $m < n$:

  1. Select a random r where $r < n$.
  2. Compute the ciphertext as $c = g^m \cdot r^n \bmod n^2$.

- Decrypt a ciphertext c as $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$.

5.3.2 Privacy Preservation

The only update equation among (5.8)-(5.14) that relies on information received from neighboring agents is (5.11). To protect the privacy of agents at this step, we adopt a similar approach to [130] and enable the agents to encrypt all messages communicated with their neighbors. To this end, we decompose each edge-specific penalty parameter as $\rho_{k,\ell,m} = g_m^{k\to\ell} g_m^{\ell\to k}$ where $g_m^{k\to\ell}$ and $g_m^{\ell\to k}$ are exclusively known to agents k and ℓ, respectively. In addition, we implement a secure data exchange procedure as outlined in lines 9-14 of Algorithm 6, which provides a summary of the proposed PPDNMF algorithm. In Figure 5.3, we summarize the information exchange part of the PPDNMF algorithm.

Consequently, note the following:

- Data exchanged between agents k and ℓ is encrypted, rendering it inaccessible to other agents or eavesdroppers, even if intercepted.
- The parameter $g_m^{k\to\ell}$ is unique to each edge and iteration. Therefore, an agent cannot infer the private information $\mathbf{U}_{\ell,m}$ of any of its neighbors by decrypting the messages it receives from them, as each neighbor ℓ uses its unique $g_m^{\ell\to k}$ in its encrypted message to agent k.

Algorithm 6: PPDNMF.

Initialization: $\mathbf{X}_{k,0} = \mathbf{U}_{k,0} = \mathbf{1}_{N\times L}$, $\mathbf{P}_{k,0} = \mathbf{Q}_{k,0} = \mathbf{Q}_{k,-1} = \mathbf{O}_{N\times L}$, $\mathbf{Y}_{k,0} = \mathbf{V}_{k,0} = \mathbf{R}_{k,0} = \mathbf{O}_{L\times M_k}$

For $n = 1, \ldots,$ *Until Convergence*

  For $m = 1, \ldots,$ *Until Convergence*

    $\mathbf{X}_{k,n,m} = \Pi_{\geq 0}\left\{\mathbf{U}_{k,m-1} + \mathbf{P}_{k,m-1}\right\}$

    $\mathbf{U}_{k,m} = \Big[\mathbf{Z}_k\mathbf{Y}_{k,n-1}^{\intercal} + \mu\left(\mathbf{X}_{k,n,m} - \mathbf{P}_{k,m-1}\right)$

      $+\rho_{k,m}\left(\mathbf{U}_{k,m-1} + 2\mathbf{Q}_{k,m-1} - \mathbf{Q}_{k,m-2}\right)\Big]$

      $\times\left[\mathbf{Y}_{k,n-1}\mathbf{Y}_{k,n-1}^{\intercal} + \left(\mu + \rho_{k,m}\right)\mathbf{I}\right]^{-1}$

    $\mathbf{P}_{k,m} = \mathbf{P}_{k,m-1} - \left(\mathbf{X}_{k,n,m} - \mathbf{U}_{k,m}\right)$

    Encrypt $-\mathbf{U}_{k,m}$ using the public key κ_{pk} as $\mathcal{E}_k\left(-\mathbf{U}_{k,m}\right)$

    Send $\mathcal{E}_k\left(-\mathbf{U}_{k,m}\right)$ and κ_{pk} to all neighbors in $\mathcal{N}_k$

    For $\ell \in \mathcal{N}_k$

      Encrypt $\mathbf{U}_{\ell,m}$ using κ_{pk} as $\mathcal{E}_k\left(\mathbf{U}_{\ell,m}\right)$

      Send $\left[\mathcal{E}_k\left(\mathbf{U}_{\ell,m}\right)\mathcal{E}_k\left(-\mathbf{U}_{k,m}\right)\right]^{g_m^{\ell\to k}}$ to agent k

    EndFor

    Decrypt messages received from neighbors using the private key κ_{sk}

    Multiply the received messages by $g_m^{k\to\ell}$

    $\mathbf{Q}_{k,m} = \mathbf{Q}_{k,m-1} + \sum_{\ell\in\mathcal{N}_k} g_m^{k\to\ell} g_m^{\ell\to k}\left(\mathbf{U}_{\ell,m} - \mathbf{U}_{k,m}\right)$.

  EndFor

  $\mathbf{X}_{k,n} \leftarrow \mathbf{X}_{k,n,m}$, $\mathbf{U}_{k,0} \leftarrow \mathbf{U}_{k,m}$, $\mathbf{P}_{k,0} \leftarrow \mathbf{P}_{k,m}$,

  $\mathbf{Q}_{k,0} \leftarrow \mathbf{Q}_{k,m}$, $\mathbf{Q}_{k,-1} \leftarrow \mathbf{Q}_{k,m-1}$

  For $r = 1, \ldots,$ *Until Convergence*

    $\mathbf{Y}_{k,n,r} = \Pi_{\geq 0}\left\{\mathbf{V}_{k,r-1} + \mathbf{R}_{k,r-1}\right\}$

    $\mathbf{V}_{k,r} = \left(\mathbf{X}_{k,n}^{\intercal}\mathbf{X}_{k,n} + \eta\mathbf{I}\right)^{-1}\left[\mathbf{X}_{k,n}^{\intercal}\mathbf{Z}_k + \eta\left(\mathbf{Y}_{k,n,r} - \mathbf{R}_{k,r-1}\right)\right]$

    $\mathbf{R}_{k,r} = \mathbf{R}_{k,r-1} - \left(\mathbf{Y}_{k,n,r} - \mathbf{V}_{k,r}\right)$

  EndFor

  $\mathbf{Y}_{k,n} \leftarrow \mathbf{Y}_{k,n,r}$, $\mathbf{V}_{k,0} \leftarrow \mathbf{V}_{k,r}$, $\mathbf{R}_{k,0} \leftarrow \mathbf{R}_{k,r}$

EndFor

- The Paillier cryptosystem is intended for encrypting scalar unsigned integers. To encrypt the entries of $\mathbf{U}_{k,m}$, which are typically floating-point values, we ini-

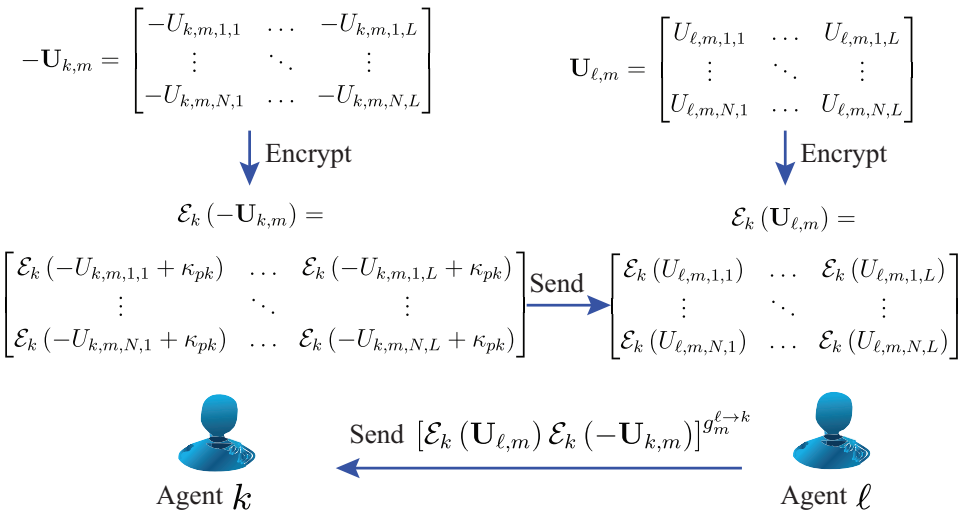

Figure 5.3: The information exchange between agents k and $\ell \in \mathcal{N}_k$ in PPDNMF.

tially quantize them. This involves multiplying each entry by a positive integer $N_{\max}$, which determines the quantization resolution, and then rounding the result to the nearest integer. To undo the quantization, we divide the decrypted values by $N_{\max}$.

- To handle the encryption of negative quantized values (note line 10 in Algorithm 6), we convert them to positive integers by adding the public key κ_{pk} to them [135].
- To guarantee convergence, we ensure that the parameters $g_m^{k\to\ell}$ increase monotonically over iterations without becoming unbounded [130]. Thus, we select each parameter uniformly from the interval $\left(g_{m-1}^{k\to\ell}, g_k\right]$, where g_k is a predefined positive constant, known only to agent k, and $g_0^{k\to\ell} = 0$.

5.4 Numerical Results

In this section, we conduct a series of numerical experiments to evaluate the performance of our PPDNMF algorithm. We consider a network consisting of $K = 10$ agents, interconnected arbitrarily, with each agent having three neighbors on average. We test our algorithm on two datasets, namely, a synthetic dataset and the MIT Center for Biological and Computational Learning (MIT-CBCL) face database [137], a widely utilized benchmark in the field of computer vision, comprises a diverse set of images capturing various poses, expressions, and lighting conditions, providing a robust foundation for evaluating the performance of face recognition algorithms.

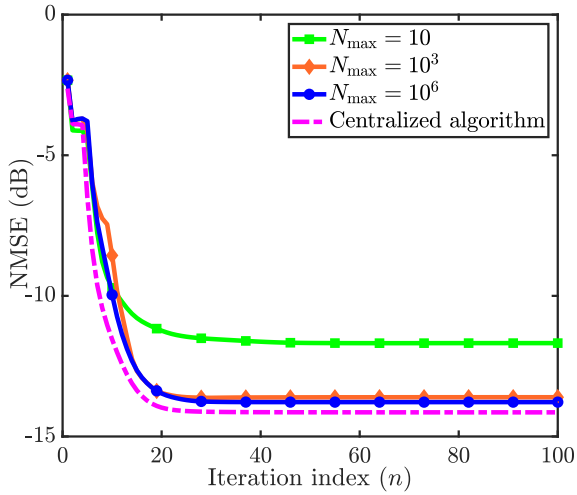

Figure 5.4: NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of $N_{\max}$ on synthetic data.

The agents collaboratively factorize a data matrix $\mathbf{Z} \in \mathbb{R}^{N\times M}$ to left and right factor matrices $\mathbf{X} \in \mathbb{R}^{N\times L}$ and $\mathbf{Y} \in \mathbb{R}^{L\times M}$, where $N \in \{30, 361\}, M \in \{200, 2429\}$, and $L \in \{5, 49\}$ in our two experiments. We set the number of BCD iterations to 100 and the number of ADMM iterations to 30. In our implementation of the Paillier cryptosystem, we use 128-bit public and private keys.

We evaluate the performance of PPDNMF in comparison with the centralized algorithm, i.e., where all data is available at a central hub. To quantify the performance, we utilize the normalized mean-square error (NMSE) at each BCD iteration, defined as

$$\frac{1}{K}\sum_{k=1}^{K}\frac{\|\mathbf{Z}_k - \mathbf{X}_{k,n}\mathbf{Y}_{k,n}\|_{\mathsf{F}}}{\|\mathbf{Z}_k\|_{\mathsf{F}}}. \tag{5.15}$$

In addition, we average the presented results over 100 independent trials.

In our first experiment utilizing synthetic data, we draw the entries of the nonnegative factor matrices $\mathbf{X} \in \mathbb{R}^{30\times 5}$ and $\mathbf{Y} \in \mathbb{R}^{5\times 200}$ independently from exponential distributions with parameter values 0.033 and 0.8, respectively. We calculate the data matrix as $\mathbf{Z} = \mathbf{XY} + \mathbf{\Gamma}$, where we draw the entries of $\mathbf{\Gamma}$ independently from a Gaussian distribution with zero mean and variance 3.6×10^{-4}, resulting in an SNR of approximately 20dB. We set $\mu = 0.1$, $\eta = 1$, and $g_i = 0.033$ for all agents. We consider $\mathbf{Z}$ to be distributed among the agents such that each agent has a varying number of columns between four and 40. We present the NMSE learning curves of

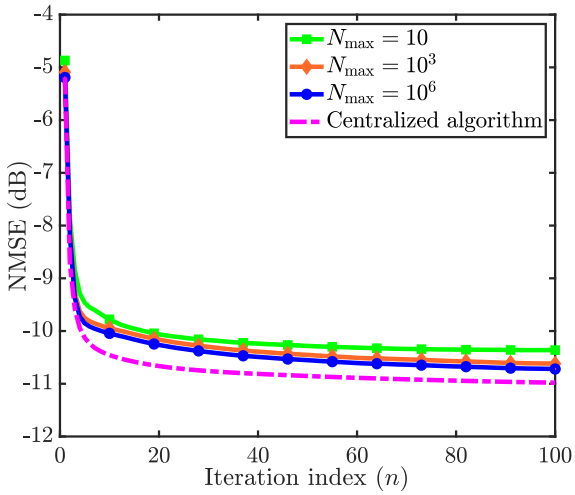

Figure 5.5: NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of $N_{\max}$ on MIT-CBCL database.

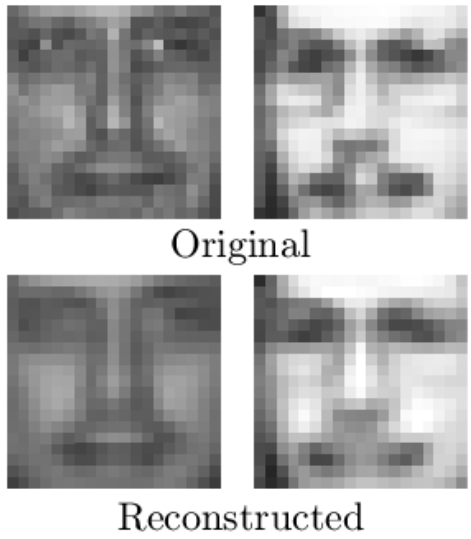

Figure 5.6: The original and reconstructed faces #1 and #2429 from the MIT-CBCL database.

PPDNMF for different values of $N_{\max}$ alongside that of the corresponding centralized algorithm in Figure 5.4. From Figure 5.4, it is evident that the proposed PPDNMF algorithm closely matches the performance of the centralized algorithm regarding both convergence rate and steady-state NMSE. Furthermore, it is clear

that increasing the value of $N_{\max}$ results in a lower steady-state NMSE.

Our second experiment utilizes the MIT-CBCL face database, which contains 2429 monochromatic face images in its training dataset. We allocate the corresponding data matrix among the agents so that each agent has between 224 and 245 columns. For this experiment, we configured $\mu = 2$, $\eta = 2$, and set $g_i = 0.05$ for all agents. The results presented in Figure 5.5 highlight the effectiveness of PPDNMF. Notably, PPDNMF exhibits robust performance even with $N_{\max} = 10$. Additionally, we compare the original faces #1 and #2429 with their reconstructed versions by PPDNMF, using $N_{\max} = 10^6$ in Figure 5.6. The reconstructed images closely match their original versions.

5.5 Summary

In this chapter, we introduced a novel privacy-preserving distributed nonnegative matrix factorization (PPDNMF) algorithm that employs the Paillier cryptosystem to enable secure collaboration among agents, thereby safeguarding their privacy and mitigating the risk of sensitive data leakage over ad-hoc networks. Our simulation results, based on both synthetic data and a real-world database, confirmed the efficacy of our proposed PPDNMF algorithm.

Chapter 6

Distributed Maximum Consensus over Noisy Links

This chapter presents the results of publication **P7**, which delves into the problem of estimating the maximum value within a multi-agent ad-hoc network in the presence of noise in the communication links. Our approach involves redefining the maximum consensus problem as a distributed optimization problem, allowing for a solution using ADMM. Unlike current algorithms that fail to converge when they work with noise-corrupted estimates, our noise-robust distributed maximum consensus (RD-MC) algorithm converges in a noisy setup. To achieve this robustness, we employ only a single set of noise-corrupted estimates and a moving averaging on the local estimates to mitigate the impact of link noise. Our extensive simulations demonstrate that RD-MC displays significantly improved resilience to communication link noise compared to existing maximum consensus algorithms.

6.1 Motivation

Consensus algorithms play a pivotal role in facilitating coordination and consensus formation among multiple agents within a distributed system [25, 138–145]. Numerous studies have delved into the challenge of attaining network-wide consensus on various values, such as average, minimum, and median, in a distributed manner. Achieving consensus in a multi-agent network requires local computations by agents and data exchange among neighboring agents [146–155]. The presence of noise in the communication links between agents can significantly impact the performance of consensus algorithms, introducing challenges in achieving accurate and reliable convergence. Navigating the effects of communication noise is a critical consideration in the design and implementation of consensus-based distrib-

uted systems, as it can lead to higher steady-state mean-square error and potential instability in the consensus process [156].

The distributed maximum consensus problem involves determining the largest value within a network in a decentralized manner. Extensive research has been dedicated to exploring this problem across various contexts [157–163]. The work in [157] presents a distributed algorithm for achieving maximum consensus, although it assumes noise-free communication links. Furthermore, the study by [158] establishes limits on the anticipated convergence time for achieving maximum consensus in asynchronous networks, without taking into account communication noise. The method employed in [159] tackles the maximum consensus problem by using the soft-max function as an approximation of the maximum function. Nevertheless, its effectiveness is constrained by a trade-off between estimation error and convergence speed. Although [160] introduces a distributed maximum consensus algorithm resilient to noise, the variance of its error increases proportionally with the network size.

This chapter presents a fully-distributed algorithm termed noise-robust distributed maximum consensus (RD-MC), aimed at effectively determining the maximum value within a multi-agent network, especially in situations where communication links are affected by noise. In RD-MC, we utilize the advantages of strategically planned parameter exchanges. We demonstrate the efficacy of RD-MC through comprehensive simulations and by contrasting its performance with that of existing algorithms.

6.2 Background

We consider a distributed network with K agents, modeled by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The set of vertices $\mathcal{V}$ corresponds to the agents and the edge set $\mathcal{E}$ represents the communication links between the agents. The adjacency matrix $\mathbf{A}$ describes the structure of the graph $\mathcal{G}$ with entries $a_{k\ell}$, where $a_{k\ell} = 1$ if and only if the agents k and ℓ are neighbors and $a_{k\ell} = 0$ otherwise. The set of neighbors for agent k is denoted by $\mathcal{N}_k$ with its cardinality denoted by $d_k = |\mathcal{N}_k|$. We assume that the set $\mathcal{N}_k$ does not include the agent k itself. We also assume only simple graphs, i.e., there are no self-loops or multiple edges.

The conventional maximum consensus algorithm relies on the agents communicating only with their immediate neighbors and can be written as

$$x_{k,n+1} = \max\left(x_{k,n}, \{x_{\ell,n}\}_{\ell\in\mathcal{N}_k}\right), \ \forall k \in \mathcal{V}, \tag{6.1}$$

where $x_{k,n}$ is the estimate of the kth agent at time instant n, and the initial value is $x_{k,0} = a_k \ \forall k \in \mathcal{V}$. The solution to (6.1) is denoted by $\max(\{a_k\}_{k\in\mathcal{V}}) =$

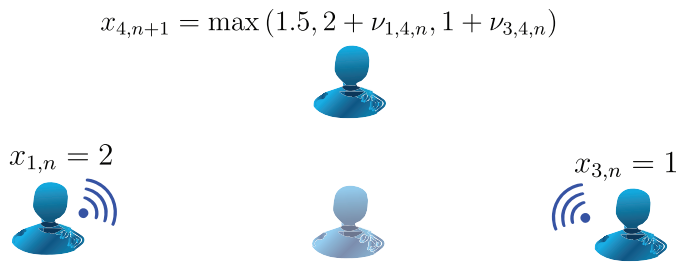

Figure 6.1: An illustration of the naive maximum consensus (naive-MC) algorithm.

$a^\star$. Given that the communications between agents are noiseless, (6.1) reaches consensus after a finite number of iterations denoted by N, i.e., $x_{k,n} = a^\star \; \forall n \geq N$ and $k \in \mathcal{V}$ [161].

We want to investigate the convergence of (6.1) when the communication links are noisy. The noise in the received message by agent k from agent ℓ at time instant n is denoted by $\nu_{\ell,k,n}$ and modeled as zero-mean additive white Gaussian noise with variance σ^2. The noise is assumed to be uncorrelated across different agents and time instants. Hence, upon accounting for additive link noise, (6.1) can be written as

$$x_{k,n+1} = \max\left(x_{k,n}, \{x_{\ell,n} + \nu_{\ell,k,n}\}_{\ell \in \mathcal{N}_k}\right), \; \forall k \in \mathcal{V}. \tag{6.2}$$

We refer to (6.2) as naive maximum consensus (naive-MC). We argue that even though (6.1) reaches the maximum value when we have noiseless communication links, (6.2) might not reach convergence because of potential deviation caused by noise in estimating the maximum value in each iteration. This potential deviation is illustrated in Figure 6.1. From Figure 6.1, we can observe that agent 4 has neighbors in the set $\mathcal{N}_4 \in \{1, 3\}$. It is evident that the states received from agents 1 and 3 are affected by link noise. One possible scenario is that this noise raises the true maximum state of 2 to a higher value. This deviation, accumulating over time, could result in substantial inaccuracies and even lead to divergence of (6.2).

To tackle this challenge, we reformulate the maximum consensus problem (6.1) as a distributed optimization problem that can be solved via ADMM. We show that our proposed RD-MC algorithm achieves convergence even when there is additive noise in communication links.

6.3 Proposed Method

6.3.1 Distributed Maximum Consensus Algorithm

A consensus-based reformulation of the maximum consensus problem (6.1) has been proposed in [161] as

$$\begin{aligned} a^\star = \arg\min_{x} \quad & x \\ \text{s.t.} \quad & x \geq a_k, \ \forall k \in \mathcal{V}. \end{aligned} \tag{6.3}$$

In (6.3), we reformulate the problem (6.1) in epigraph form as a linear program (LP). To solve (6.3) in a distributed manner, we assign each agent its own variable $\{x_k\}_{k=1}^K$. This approach imposes local epigraph constraints that enforce consensus among neighboring agents. Additionally, we utilize a separable objective that is equivalent to the original when all local variables are in agreement, i.e., we have consensus.

Hence, we can rewrite (6.3) as

$$\begin{aligned} \min_{\{x_k\}} \quad & \frac{1}{K}\sum_{k=1}^{K} x_k \\ \text{s.t.} \quad & x_k \geq a_k \quad \forall k \in \mathcal{V} \\ & x_k = x_\ell \quad \forall k \in \mathcal{V}, \ell \in \mathcal{N}_k. \end{aligned} \tag{6.4}$$

In (6.4), we enforce local consensus across each agent's neighborhood through equality constraints. It can be shown that (6.1) and (6.4) have an identical solution through the Karush-Kuhn-Tucker (KKT) optimality conditions [161], i.e. the solution of (6.4) is $x_k^\star = a^\star \ \forall k \in \mathcal{V}$.

In addition to employing a mathematical approach, such as KKT conditions, to establish this equivalence, it is beneficial to utilize our intuition regarding (6.4). In (6.4), the goal is to minimize the objective function while simultaneously ensuring that each local variable is greater than its initial state. Consequently, this reformulation is expected to converge to the maximum initial value.

To facilitate the reformulation of (6.4) for a distributed solution via utilizing ADMM, we incorporate the constraints into the objective function through the use of indicator functions. This approach transforms the problem (6.4) to a fully separable form of two distinct blocks, as elaborated in section 2.6.

Therefore, we can further express (6.4) as

$$\begin{aligned} \min_{\{x_k, y_k, q_k^\ell\}} \quad & \frac{1}{K}\sum_{k=1}^{K} x_k + \frac{1}{K}\sum_{k=1}^{K} \mathcal{I}_{a_k}(y_k) \\ \text{s.t.} \quad & x_k = y_k \quad \forall k \in \mathcal{V} \\ & x_k = q_k^\ell, x_\ell = q_k^\ell \quad \forall k \in \mathcal{V}, \ell \in \mathcal{N}_k, \end{aligned} \tag{6.5}$$

where the indicator function $\mathcal{I}_{a_k}(y_k)$, defined as $\mathcal{I}_a(y) = 0$, if $y \geq a$ and ∞ otherwise, imposes the inequality constraint to seek the maximum value, and the auxiliary variables $\mathcal{Q} = \{q_k^\ell\}_{k\in\mathcal{V},\ell\in\mathcal{N}_k}$ facilitate consensus within each agent's neighborhood and, consequently, across the network. The optimization problem (6.5) can be tackled using various methods, including those based on subgradients or ADMM. However, distributed subgradient methods applied to affine objective functions are known to converge slowly [164]. Therefore, we opt for ADMM to solve (6.5).

Let $\mathcal{L}(\{x_k, y_k\}_{k\in\mathcal{V}}, \mathcal{Q}, \mathcal{M})$ denote the augmented Lagrangian function associated with (6.5) as

$$\begin{aligned} \mathcal{L}(\{x_k, y_k\}_{k\in\mathcal{V}}, \mathcal{Q}, \mathcal{M}) = & \sum_{K=1}^{K} \left(\frac{x_k + \mathcal{I}_{a_k}(y_k)}{K} + u_k(x_k - y_k) + \frac{\rho_y}{2}(x_k - y_k)^2 \right) \\ & + \sum_{k=1}^{K} \sum_{\ell\in\mathcal{N}_k} \left(\mu_{k,\ell}(x_k - q_k^\ell) + \pi_{k,\ell}(x_\ell - q_k^\ell) \right) \\ & + \frac{\rho_z}{2} \sum_{k=1}^{K} \sum_{\ell\in\mathcal{N}_k} \left((x_k - q_k^\ell)^2 + (x_\ell - q_k^\ell)^2 \right), \end{aligned} \tag{6.6}$$

where $\mathcal{M} = \{u_k, \mu_{k,\ell}, \pi_{k,\ell}\}_{k\in\mathcal{V},\ell\in\mathcal{N}_k}$ represents the respective Lagrange multipliers. Minimizing $\mathcal{L}$, while applying the KKT conditions to (6.5) and defining $v_{k,n} = 2\sum_{\ell\in\mathcal{N}_k} \mu_{k,\ell,n}$, leads to the following iterative updates at the kth agent along with the elimination of $\{\pi_{k,\ell}\}_{k\in\mathcal{V},\ell\in\mathcal{N}_k}$ and $\mathcal{Q}$ [15, 161] as

$$x_{k,n+1} = n_k \left(\frac{-1}{K} + \rho_y \left[y_{k,n} - \bar{u}_{k,n} \right] - v_{k,n} + \rho_z \sum_{\ell\in\mathcal{N}_k} \left[x_{k,n} + \tilde{x}_{\ell,n} \right] \right), \tag{6.7a}$$

$$y_{k,n+1} = \max(x_{k,n+1} + \bar{u}_{k,n}, a_k), \tag{6.7b}$$

$$\bar{u}_{k,n+1} = \bar{u}_{k,n} + x_{k,n+1} - y_{k,n+1}, \tag{6.7c}$$

$$v_{k,n+1} = v_{k,n} + \rho_z \sum_{\ell\in\mathcal{N}_k} [x_{k,n+1} - \tilde{x}_{\ell,n+1}]. \tag{6.7d}$$

Here, n is the iteration index, $n_k = (\rho_y + 2\rho_z d_k)^{-1}$, $\rho_y > 0$ and $\rho_z > 0$ are the penalty parameters, and $\bar{u}_k = u_k/\rho_y$. In addition, all initial values $\{x_{k,0}, y_{k,0}, \bar{u}_{k,0}, v_{k,0}\}_{k\in\mathcal{V}}$ are set to zero. Note that, in (6.7a), agent k has access to $\tilde{x}_{\ell,n} = x_{\ell,n} + \nu_{\ell,k,n}$ rather than $x_{\ell,n}$. The iterations (6.7) can be implemented locally at each agent in a fully-distributed fashion, as the required information is available within each agent's neighborhood. We refer to this algorithm as distributed maximum consensus (D-MC).

6.3.2 Noise-Robust Distributed Maximum Consensus Algorithm

Our goal is to investigate the effect of noisy links in (6.4). Therefore, we describe two modifications to the D-MC algorithm (6.7). These modifications are aimed at enhancing the robustness of distributed maximum consensus to communication noise and lead to the proposed RD-MC algorithm.

Using the initial values $v_{k,0} = 0\ \forall k \in \mathcal{V}$, we obtain

$$v_{k,n} = \rho_z \sum_{m=1}^{n} \sum_{\ell\in\mathcal{N}_k} [x_{k,m} - \tilde{x}_{\ell,m}] \tag{6.8}$$

from (6.7d). Substituting (6.8) into (6.7a) while using the initial values $x_{k,0} = 0$ and $x_{k,1} = -K^{-1} n_k\ \forall k \in \mathcal{V}$, we can eliminate $v_{k,n}$ and modify (6.7a) as [165]

$$\begin{aligned} x_{k,n+1} &= (1 - \rho_y n_k) x_{k,n} - \rho_z d_k n_k x_{k,n-1} \\ &\quad + n_k \left(\rho_y z_{k,n} + \rho_z \sum_{\ell\in\mathcal{N}_k} \tilde{s}_{\ell,n} \right), \end{aligned} \tag{6.9a}$$

$$\bar{x}_{k,n+1} = \sum_{m=0}^{\mathcal{C}-1} \alpha_m x_{k,n-m+1}, \tag{6.9b}$$

$$z_{k,n+1} = 2y_{k,n+1} - y_{k,n}, \tag{6.9c}$$

$$s_{k,n+1} = 2\bar{x}_{k,n+1} - x_{k,n}. \tag{6.9d}$$

Note that, in (6.9b), to enhance robustness against spurious noise, we compute the convex combination of $\mathcal{C}$ past local estimates, utilizing the weights α_ℓ that sum to one.

In this alternative formulation, instead of $x_{k,n}$, agents exchange $s_{k,n}$, which is a smoothed version of $x_{k,n}$. However, due to communication noise, they receive noisy versions from their neighbors, i.e., agents $\ell \in \mathcal{N}_k$ receive $\tilde{s}_{k,n} = s_{k,n} + \nu_{k,\ell,n}$ from agent k. The recursions (6.9) alongside (6.7b) and (6.7c) constitute the proposed noise-robust distributed maximum consensus (RD-MC) algorithm, summarized in Algorithm 7.

Algorithm 7: RD-MC.

Parameters: penalty parameters ρ_z and ρ_y

Initialization: $x_{k,0} = 0$, $x_{k,1} = -K^{-1}n_k$, $\bar{u}_{k,1} = 0$,
$z_{k,1} = 0$, $s_{k,1} = -2K^{-1}n_k$, $\forall i \in \mathcal{V}$

For $n = 1, \ldots, N$ *until convergence*

Receive $\tilde{s}_{\ell,n}$ from neighbors $\ell \in \mathcal{N}_k$

$x_{k,n+1} = (1 - \rho_y n_k)x_{k,n} - \rho_z d_k n_k x_{k,n-1} + n_k \left(\rho_y z_{k,n} + \rho_z \sum_{\ell \in \mathcal{N}_k} \tilde{s}_{\ell,n}\right)$

$\bar{x}_{k,n+1} = \sum_{m=0}^{\mathcal{C}-1} \alpha_m x_{k,n-m+1},\ \sum_m \alpha_m = 1$

$y_{k,n+1} = \max(x_{k,n+1} + \bar{u}_{k,n}, a_k)$

$\bar{u}_{k,n+1} = \bar{u}_{k,n} + x_{k,n+1} - y_{k,n+1}$

$z_{k,n+1} = 2y_{k,n+1} - y_{k,n}$

Send $s_{k,n+1} = 2\bar{x}_{k,n+1} - x_{k,n}$ to neighbors $\ell \in \mathcal{N}_k$.

EndFor

We mitigate the effect of noisy links in RD-MC through two key modifications to D-MC. First, the introduction of $s_{k,n}$, a linear combination of $\bar{x}_{k,n}$ and $x_{k,n-1}$, offers a strategic advantage in alleviating the adverse effects of communication noise. By exchanging $s_{k,n}$ instead of $x_{k,n}$ over noisy links, we enhance robustness. Notably, while the aggregation of two sets of noisy estimates received from neighbors at consecutive iterations [i.e., $\tilde{x}_{\ell,n}$ in (6.7a) and $\tilde{x}_{\ell,n+1}$ in (6.7d)] renders D-MC vulnerable to noise accumulation, RD-MC's reliance on a single set of noisy estimates [i.e., $\tilde{s}_{k,n}$ in (6.9a)] enhances its resilience to link noise. Second, we further enhance robustness to link noise by applying a weighted averaging of $x_{k,n+1}$ over a sliding window of size $\mathcal{C}$ as in (6.9b).

6.4 Numerical Results

We conduct a series of experiments to assess the performance of the proposed RD-MC algorithm. We examine a network of $K = 20$ agents, as illustrated in Figure 6.2. The initial values (estimates) for the agents are drawn independently from a standard normal distribution, specifically, $a_k \sim \mathcal{N}(0, 1)\ \forall k \in \mathcal{V}$, and we define $a^\star = \max(\{a_k\}_{k \in \mathcal{V}})$. Furthermore, we set the penalty parameters to $\rho_z = \rho_y = 1$ and assign the weights in (6.9b) to $\alpha_\ell = 1/\mathcal{C}$ across all our experiments. The results are obtained by averaging over 1000 independent instances of communication noise. To represent the noise in the communication links, we utilize a truncated zero-mean normal distribution, limiting the noise to within $\pm 3\sigma$ to keep it within a reasonable range.

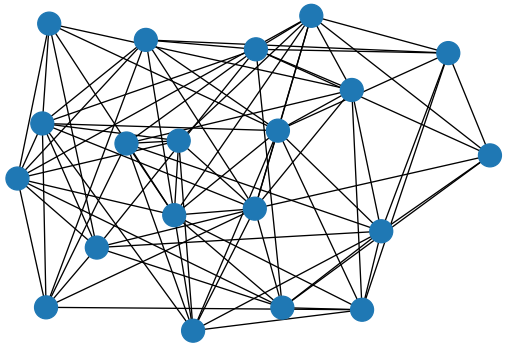

Figure 6.2: The considered network with an arbitrary topology and $K = 20$ agents.

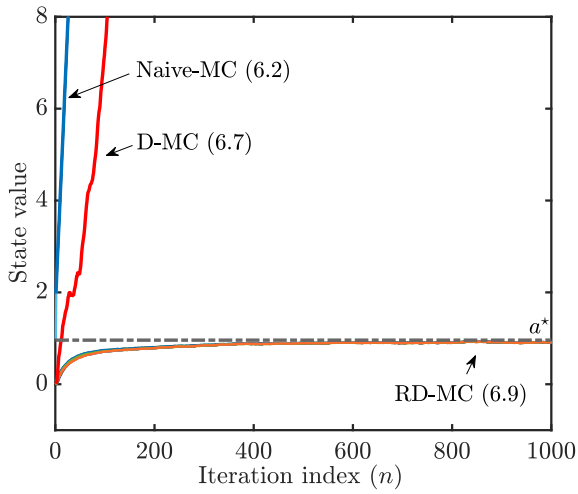

Figure 6.3: The impact of noise on the performance of Naive-MC (6.2), D-MC algorithm (6.7) and RD-MC algorithm (6.9) with window size $\mathcal{C} = 3$ and noise variance $\sigma^2 = 0.1$.

In our first experiment, we investigate the effects of noise on the performance of RD-MC, D-MC, and the naive solution (6.2), referred to as naive-MC, using a noise variance of $\sigma^2 = 0.1$ and a window size of $\mathcal{C} = 3$. Figure 6.3 depicts the evolution of the estimates of all agents using the considered algorithms over 1000 iterations. It is evident that RD-MC converges to the maximum value with a bounded error, while the other two algorithms diverge.

In our second experiment, we study the effect of noise variance on the network-

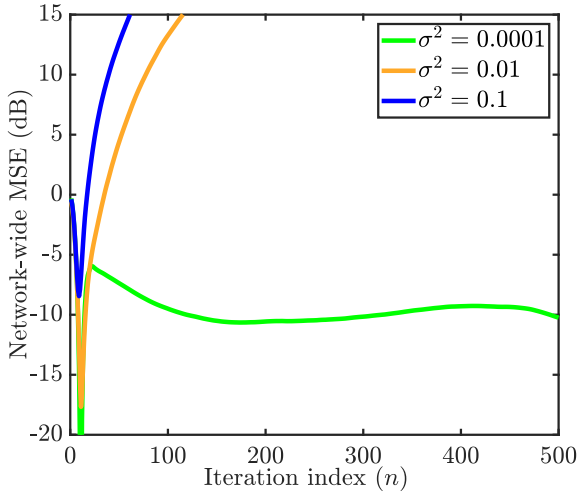

Figure 6.4: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 1$ and different noise variances $\sigma^2 \in \{0.0001, 0.01, 0.1\}$.

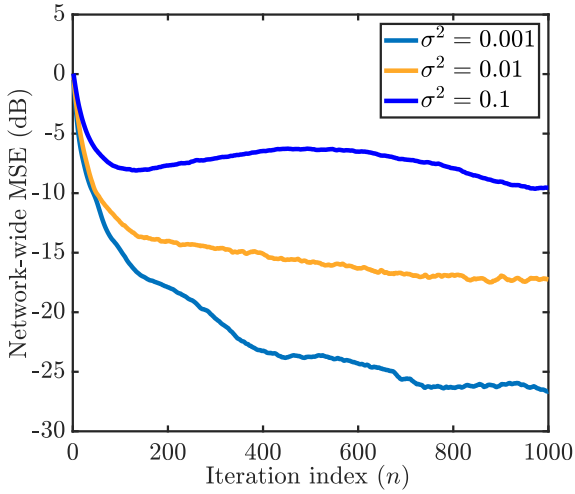

Figure 6.5: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 2$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

wide mean square error (MSE) of RD-MC calculated as

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[(x_{k,n} - a^{\star})^2\right]. \tag{6.10}$$

We conduct simulations of RD-MC using different window sizes $\mathcal{C} \in \{1, 2, 3\}$

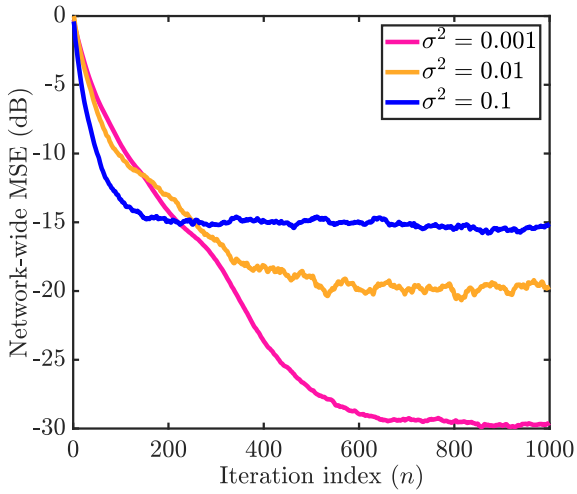

Figure 6.6: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

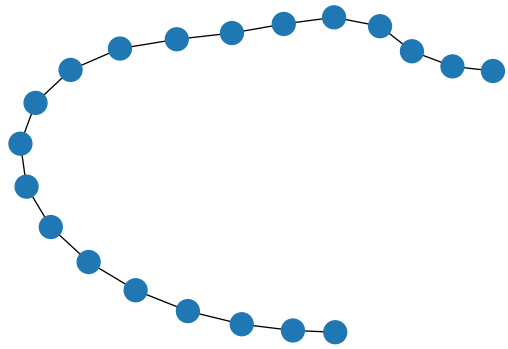

Figure 6.7: The considered network with linear topology and $K = 20$ agents.

and noise variances σ^2, and present the results in Figs. 6.4-6.6. We observe that increasing σ^2 results in higher steady-state network-wide error across all experiments. Nevertheless, the selection of window size has a significant impact on RD-MC's ability to reduce communication noise. While RD-MC struggles to converge with $\mathcal{C} = 1$, it maintains convergence with $\mathcal{C} \geq 2$ and increasing $\mathcal{C}$ enhances its robustness to noise. Figs. 6.3 and 6.6 demonstrate that RD-MC with $\mathcal{C} = 3$ shows a markedly improved resistance to link noise in comparison to D-MC, all while not adding any extra computational or communication burden.

In our final experiment, we evaluate how sensitive RD-MC's performance is to net-

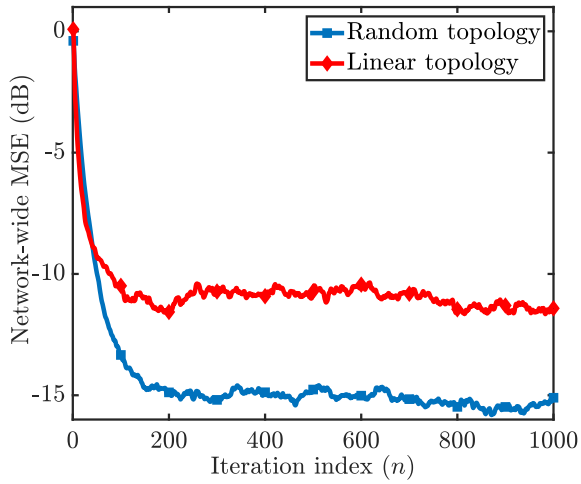

Figure 6.8: The impact of network connectivity on the network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ in the presence of link noise with variance $\sigma^2 = 0.1$.

work topology in the presence of link noise. We simulate RD-MC with a window size of $\mathcal{C} = 3$ across two networks: the network shown in 6.2 and another with a linear topology illustrated in Figure 6.7. The noise variance is set to $\sigma^2 = 0.1$. We present the results in Figure 6.8. It is evident that, in the case of a linear network topology, the network-wide steady-state MSE of RD-MC is greater compared to that of a network with a more complex topology and higher average degree. Nevertheless, RD-MC continues to exhibit good performance in the linear network topology, despite its lower connectivity.

6.5 Summary

In this chapter, a noise-robust distributed algorithm termed RD-MC has been proposed to mitigate the adverse effects of communication noise on the convergence of conventional distributed maximum consensus algorithms. This algorithm incorporates two modifications compared to the conventional distributed maximum consensus algorithm. Unlike existing algorithms tailored for ad-hoc networks, the proposed algorithm demonstrates robustness against additive communication noise. Extensive simulation results have underscored the effectiveness of RD-MC in diverse scenarios.

Chapter 7

Conclusion and Future Work

This thesis developed solutions for various application scenarios within distributed and federated learning settings. We can summarize the thesis and provide future directions for continuing this work as follows:

7.1 Summary

In chapter 2, we provided an overview of distributed and federated learning and their abilities and challenges. In addition, we introduced online federated learning, partial diffusion of information, and the alternating direction method of multipliers.

In chapter 3, we conducted a theoretical analysis of PSO-Fed to examine its resilience to model-poisoning (Byzantine) attack. We showed that PSO-Fed converges in both mean and mean-square senses, given an appropriate choice of stepsize even under Byzantine attacks. More importantly, we showed that, in the presence of Byzantine clients, the steady-state MSE of PSO-Fed is significantly smaller than that of the Online-Fed algorithm, which does not feature partial sharing. Our theoretical analysis also uncovered the existence of a non-trivial optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. The simulation results corroborated our theoretical findings regarding PSO-Fed's effectiveness against Byzantine attacks as well as the accuracy of the theoretically predicted values of its steady-state MSE and optimal stepsize.

In chapter 4, we developed a resource-efficient FL algorithm that has improved robustness against noise/error in communication links. To motivate the developed algorithm, we considered a weighted least-squares regression problem. To achieve robustness, we proposed to combine the last two global model updates and send

them together alongside eliminating the dual model update performed at each participating edge device. The proposed algorithm, called RERCE-Fed, ensures that clients receive a less corrupted global model update from the server even when the server uses random scheduling to achieve communication efficiency. We proved the convergence of RERCE-Fed in the mean sense at the presence of link noise. We also verified the desirable performance of RERCE-Fed via simulations, particularly, its robustness against additive communication link noise in comparison to existing related algorithms.

In chapter 5, we introduced a novel privacy-preserving distributed nonnegative matrix factorization (PPDNMF) algorithm that employs the Paillier cryptosystem to enable secure collaboration among agents, thereby safeguarding their privacy and mitigating the risk of sensitive data leakage over ad-hoc networks. Our simulation results, based on both synthetic and real data, confirmed the efficacy of the proposed algorithm. In future work, we plan to conduct a comprehensive theoretical privacy analysis of the proposed algorithm, exploring its resilience across various attack scenarios.

In chapter 6, we developed a distributed algorithm, called noise-robust distributed maximum consensus (RD-MC), to tackle the challenge of identifying the maximum value within an ad-hoc multi-agent network utilizing noisy communication channels. Unlike existing algorithms designed for ad-hoc networks, RD-MC exhibits robustness against additive communication noise. Our extensive simulation results demonstrated the effectiveness of RD-MC in different scenarios.

7.2 Future Directions

Building upon the proposed algorithms and findings of this thesis, several promising research directions emerge. These directions are motivated by the need for more robust, scalable, and privacy-preserving FL frameworks in dynamic and adversarial environments. For future research, a list of works can be done, such as:

1. Resilience Analysis from a Bayesian Perspective: A promising extension of the PSO-Fed algorithm involves analyzing its resilience to adversarial model-poisoning attacks using a Bayesian framework. This probabilistic approach enables modeling the local and global model updates as well as the attack signals as independent random processes. Such a formulation would allow the study of PSO-Fed's robustness under distributional shifts and stochastic perturbations, thereby offering more in-depth resilience analysis. However, Bayesian methods often come with computational overhead and convergence challenges, especially in a decentralized setting, which must be addressed to make this practical.

2. Privacy, Fairness, and Personalization in Distributed and Federated Learning: Another vital direction is to deepen the privacy guarantees of algorithms like PPDNMF through a rigorous theoretical privacy analysis, possibly incorporating differential privacy or local privacy mechanisms. Moreover, as FL moves toward real-world adoption, fairness and personalization are increasingly important. Ensuring that models do not disproportionately benefit certain user groups over others, while allowing tailored performance to individual user preferences or data distributions, is a key challenge. This requires balancing global model utility with local user adaptation, which could be addressed through personalized models.

3. Towards Privacy-Preserving Distributed Recommender Systems: Leveraging PPDNMF for building distributed recommender systems that maintain data locality presents a viable alternative to traditional centralized systems. Such a system would inherently protect user privacy while potentially preserving or even enhancing recommendation quality. The main challenge here lies in maintaining global performance with only local and incomplete data.

4. Application of PSO-Fed and RERCE-Fed to Emerging Domains: The implementation of PSO-Fed and RERCE-Fed could involve the utilization of advanced deep neural networks with real-world datasets and applications that feature a substantial number of complex features and parameters. Exploring the concept of partial sharing may prove effective in training large distributed models while enhancing their resilience against potential model-poisoning attacks.

5. Hardware Implementation and Empirical Validation: Finally, the proposed algorithms can be implemented on hardware to validate their effectiveness and gain deeper insights into their advantages and potential limitations.

Bibliography

[1] P. Sethi and S. R. Sarangi, "Internet of things: Architectures, protocols, and applications," *J. elec. comput. eng.*, vol. 2017, no. 1, p. 9 324 035, 2017.

[2] P. Bellini, P. Nesi and G. Pantaleo, "IoT-enabled smart cities: A review of concepts, frameworks and key technologies," *Appl. Sci.*, vol. 12, no. 3, p. 1607, 2022.

[3] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–41, 2022.

[4] J. Sheth and B. Dezfouli, "Enhancing the energy-efficiency and timeliness of IoT communication in wifi networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9085–9097, 2019.

[5] L. Tawalbeh, F. Muheidat, M. Tawalbeh and M. Quwaider, "IoT privacy and security: Challenges and solutions," *Appl. Sci.*, vol. 10, no. 12, p. 4102, 2020.

[6] B. V. Philip, T. Alpcan, J. Jin and M. Palaniswami, "Distributed real-time IoT for autonomous vehicles," *IEEE Trans. Ind. Inform.*, vol. 15, no. 2, pp. 1131–1140, 2018.

[7] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim and M. Debbah, "Communication-efficient and distributed learning over wireless networks: Principles and applications," *Proc. IEEE*, vol. 109, no. 5, pp. 796–819, 2021.

[8] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen and W. Ding, "Towards efficient communications in federated learning: A contemporary survey," *J. Frankl. Inst.*, vol. 360, no. 12, pp. 8669–8703, 2023.

[9] H. Wang and Y. Chi, "Communication-efficient federated optimization over semi-decentralized networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 147–160, 2025.

[10] M. Rapp, R. Khalili, K. Pfeiffer and J. Henkel, "Distreal: Distributed resource-aware learning in heterogeneous systems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 8062–8071.

[11] A. Moradi, N. K. Venkategowda, S. P. Talebi and S. Werner, "Privacy-preserving distributed kalman filtering," *IEEE Trans. Signal Process.*, vol. 70, pp. 3074–3089, 2022.

[12] B. Kailkhura, S. Brahma and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 145–158, Mar. 2017. DOI: 10.1109/TSIPN.2016.2607119.

[13] Z. Wang, Q. Kang, X. Zhang and Q. Hu, "Defense strategies toward model poisoning attacks in federated learning: A survey," in *Proc. IEEE Wireless Comm. Net. Conf.*, 2022, pp. 548–553. DOI: 10.1109/WCNC51071.2022.9771619.

[14] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, 2021. DOI: 10.1109/JIOT.2020.3030072.

[15] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas and H. Zhu, *Decentralized learning for wireless communications and networking*. Springer, 2017, pp. 461–497.

[16] V. Borkar and P. Varaiya, "Asymptotic agreement in distributed estimation," *IEEE Trans. Automatic Control*, vol. 27, no. 3, pp. 650–655, 1982. DOI: 10.1109/TAC.1982.1102982.

[17] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[18] J. Tsitsiklis, D. Bertsekas and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986. DOI: 10.1109/TAC.1986.1104412.

[19] D. Bertsekas and J. Tsitsiklis, "A survey of some aspects of parallel and distributed iterative algorithms," 1989.

[20] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation*. Upper Saddle River: Prentice-Hall, 1989.

[21] J. A. Fax, *Optimal and cooperative control of vehicle formations*. California Institute of Technology, 2002.

[22] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[23] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proc. Am. Control Conf.*, IEEE, vol. 2, 2003, pp. 951–956.

[24] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[25] R. Olfati-Saber, J. A. Fax and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007. DOI: 10.1109/JPROC.2006.887293.

[26] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003. DOI: 10.1109/TAC.2003.812781.

[27] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, 2014. DOI: 10.1109/JPROC.2014.2306253.

[28] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[29] A. H. Sayed, F. Cattivelli, S. Haykin, K. R. Liu *et al.*, "Distributed adaptive learning mechanisms," in *Handbook on array processing and sensor networks*, Wiley Online Library, 2009, pp. 695–722.

[30] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.

[31] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[32] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4064–4077, 2007. DOI: 10.1109/TSP.2007.896034.

[33] F. S. Cattivelli and A. H. Sayed, "Analysis of spatial and incremental lms processing for distributed estimation," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1465–1480, 2011. DOI: 10.1109/TSP.2010.2100386.

[34] M. H. DeGroot, "Reaching a consensus," *J. Am. Stat. Assoc.*, vol. 69, no. 345, pp. 118–121, 1974.

[35] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, 2005. DOI: 10.1109/TAC.2005.846556.

[36] F. S. Cattivelli, C. G. Lopes and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, 2008. DOI: 10.1109/TSP.2007.913164.

[37] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012. DOI: 10.1109/TSP.2012.2198470.

[38] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, 2013. DOI: 10.1109/MSP.2012.2231991.

[39] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, 2008. DOI: 10.1109/TSP.2008.917383.

[40] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—part i: Transient analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, 2015. DOI: 10.1109/TIT.2015.2427360.

[41] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—part ii: Performance analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3518–3548, 2015. DOI: 10.1109/TIT.2015.2427352.

[42] S. Haykin, *Adaptive filter theory*. Prentice-Hall, NJ, 1996.

[43] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Prentice-Hall, NJ, 1985.

[44] P. S. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Nature, 2019.

[45] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2008.

[46] V. H. Nascimento and M. T. M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing: Volume 1*, Elsevier, 2014, pp. 619–761.

[47] A. H. Sayed, T. Y. Al-Naffouri and V. H. Nascimento, "Energy conservation in adaptive filtering," in *Nonlinear Signal and Image Processing*, CRC Press, 2003, pp. 23–58.

[48] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.

[49] Y. Chen, L. Su and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[50] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan and H. V. Poor, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, 2021. DOI: 10.1109/JSAC.2021.3118346.

[51] C. Dwork, F. McSherry, K. Nissim and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptography*, Springer, 2006, pp. 265–284.

[52] Q. Lou and L. Jiang, "She: A fast and accurate deep neural network for encrypted data," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[53] J. Park, D.-J. Han, M. Choi and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 840–851, 2021.

[54] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari and R. Pedarsani, "Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, 2022. DOI: 10.1109/JSAIT.2022.3205475.

[55] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[56] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[57] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[58] C. Huang, J. Huang and X. Liu, "Cross-silo federated learning: Challenges and opportunities," *arXiv preprint arXiv:2206.12949*, 2022.

[59] C. Ren, H. Yu, H. Peng, X. Tang, B. Zhao, L. Yi, A. Z. Tan, Y. Gao, A. Li, X. Li, Z. Li and Q. Yang, "Advances and open challenges in federated foundation models," *IEEE Commun. Surv. Tutor.*, pp. 1–41, 2025. DOI: 10.1109/COMST.2025.3552524.

[60] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari and A. S. Avestimehr, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 24–29, 2022. DOI: 10.1109/IOTM.004.2100182.

[61] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020. DOI: 10.1109/MSP.2020.2975749.

[62] A. Kuh, "Real time kernel learning for sensor networks using principles of federated learning," in *Proc. IEEE Int. Conf. Asia-Pacific Signal Info. Process. Assoc.*, Dec. 2021, pp. 2089–2093.

[63] V. C. Gogineni, S. Werner, Y.-F. Huang and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4531–4544, Mar. 2023. DOI: 10.1109/JIOT.2022.3218484.

[64] R. Arablouei, S. Werner, Y.-F. Huang and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 472–484, Jan. 2014. DOI: 10.1109/TSP.2013.2292035.

[65] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Comput. Math. Appl.*, vol. 2, no. 1, pp. 17–40, 1976.

[66] R. Glowinski and A. Marroco, "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires," *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, vol. 9, no. R2, pp. 41–76, 1975.

[67] E. M. E. Mhamdi, R. Guerraoui and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3521–3530.

[68] X. Cao and N. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2022, pp. 3395–3403.

[69] J. Bernstein, J. Zhao, K. Azizzadenesheli and A. Anandkumar, "SignSGD with majority vote is communication efficient and fault tolerant," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[70] M. Fang, X. Cao, J. Jia and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *USENIX Security Symp.*, 2020, pp. 1605–1622.

[71] X. Cao, M. Fang, J. Liu and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[72] X. Cao, J. Jia, Z. Zhang and N. Z. Gong, "FedRecover: Recovering from poisoning attacks in federated learning using historical information," in *Proc. IEEE Symp. Security Privacy*, Jul. 2023, pp. 326–343.

[73] P. Blanchard, E. M. El Mhamdi, R. Guerraoui and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[74] K. Pillutla, S. M. Kakade and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022. DOI: 10.1109/TSP.2022.3153135.

[75] G. Xia, J. Chen, C. Yu and J. Ma, "Poisoning attacks in federated learning: A survey," *IEEE Access*, vol. 11, pp. 10 708–10 722, 2023. DOI: 10.1109/ACCESS.2023.3238823.

[76] F. Hu, W. Zhou, K. Liao, H. Li and D. Tong, "Toward federated learning models resistant to adversarial attacks," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16 917–16 930, 2023. DOI: 10.1109/JIOT.2023.3272334.

[77] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11 365–11 375, 2022. DOI: 10.1109/JIOT.2021.3128646.

[78] Y. Chen, X. Zhu, X. Gong, X. Yi and S. Li, "Data poisoning attacks in internet-of-vehicle networks: Taxonomy, state-of-the-art, and future directions," *IEEE Trans. Industr. Inform.*, vol. 19, no. 1, pp. 20–28, 2023. DOI: 10.1109/TII.2022.3198481.

[79] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Mach. Learn. Knowl. Discovery Databases*, Springer, 2013, pp. 387–402.

[80] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang and X. Zhang, "Trojaning attack on neural networks," in *Proc. Net. Dist. Syst. Security Symp.*, Internet Soc, 2018.

[81] R. Shokri, M. Stronati, C. Song and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 3–18. DOI: 10.1109/SP.2017.41.

[82] L. Liu, Y. Wang, G. Liu, K. Peng and C. Wang, "Membership inference attacks against machine learning models via prediction sensitivity," *IEEE Trans. Dependable Secure Computing*, vol. 20, no. 3, pp. 2341–2347, 2023. DOI: 10.1109/TDSC.2022.3180828.

[83] C. Douligeris and D. N. Serpanos, *Network security: current status and future directions*. John Wiley & Sons, 2007.

[84] D. Yin, Y. Chen, R. Kannan and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[85] Z. Liu, K. Zheng, L. Hou, H. Yang and K. Yang, "A novel blockchain-assisted aggregation scheme for federated learning in IoT networks," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17 544–17 556, 2023. DOI: 10.1109/JIOT.2023.3277463.

[86] R. Jin, Y. Liu, Y. Huang, X. He, T. Wu and H. Dai, "Sign-based gradient descent with heterogeneous data: Convergence and byzantine resilience," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, 2024. DOI: 10.1109/TNNLS.2023.3345367.

[87] V. C. Gogineni, S. Werner, Y.-F. Huang and A. Kuh, "Communication-efficient online federated learning framework for nonlinear regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232. DOI: 10.1109/ICASSP43922.2022.9746228.

[88] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, "On the resilience of online federated learning to model poisoning attacks through partial sharing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2024, pp. 9201–9205.

[89] R. Arablouei, K. Doğançay, S. Werner and Y.-F. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014. DOI: 10.1109/TSP.2014.2327005.

[90] D. S. Tracy and R. P. Singh, "A new matrix product and its applications in partitioned matrix differentiation," *Statistica Neerlandica*, vol. 26, no. 4, pp. 143–157, 1972.

[91] R. H. Koning, H. Neudecker and T. Wansbeek, "Block kronecker products and the vecb operator," *Linear algebra and its applications*, vol. 149, pp. 165–184, 1991.

[92] E. Lari, R. Arablouei, V. C. Gogineni and S. Werner, "Resilience in online federated learning: Mitigating model-poisoning attacks via partial sharing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 388–400, 2025. DOI: 10.1109/TSIPN.2025.3559444.

[93] S. Zhou and G. Y. Li, "Communication-efficient ADMM-based federated learning," *arXiv preprint arXiv:2110.15318*, Jan. 2021.

[94] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Mar. 2020. DOI: 10.1109/TCOMM.2020.2979149.

[95] S. Zheng, C. Shen and X. Chen, "Design and analysis of uplink and downlink communications for federated learning," *" IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2150–2167, Jul. 2021. DOI: 10.1109/JSAC.2020.3041388.

[96] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020. DOI: 10.1109/TWC.2020.2974748.

[97] H. Guo, A. Liu and V. K. N. Lau, "Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 197–210, Jan. 2021. DOI: 10.1109/JIOT.2020.3002925.

[98] M. M. Amiri, D. Gündüz, S. R. Kulkarni and H. V. Poor, "Convergence of federated learning over a noisy downlink," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1422–1437, Mar. 2022. DOI: 10.1109/TWC.2021.3103874.

[99] X. Wei and C. Shen, "Federated learning over noisy channels," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500833.

[100] X. Wei and C. Shen, "Federated learning over noisy channels: Convergence analysis and design examples," *IEEE Trans. Cognitive Commun. and Networking*, vol. 8, no. 2, pp. 1253–1268, Jun. 2022. DOI: 10.1109/TCCN.2022.3140788.

[101] T. Sery, N. Shlezinger, K. Cohen and Y. C. Eldar, "COTAF: Convergent over-the-air federated learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322580.

[102] I. D. Schizas, A. Ribeiro and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008. DOI: 10.1109/TSP.2007.906734.

[103] A. Abur and A. G. Exposito, *Power system state estimation: theory and implementation*. CRC press, 2004.

[104] L. Li, J. Zhong and M. Zhao, "Doppler-aided gnss position estimation with weighted least squares," *IEEE Trans. Veh. Technol.*, vol. 60, no. 8, pp. 3615–3624, 2011. DOI: 10.1109/TVT.2011.2163738.

[105] Z. Farbman, R. Fattal, D. Lischinski and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–10, 2008.

[106] E. Lari, R. Arablouei, V. C. Gogineni and S. Werner, *Noise-robust and resource-efficient admm-based federated learning for wls regression*, 2024. [Online]. Available: https://arxiv.org/abs/2409.13451.

[107] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.

[108] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.

[109] N. Gillis, "The why and how of nonnegative matrix factorization," *Connections*, vol. 12, no. 2, 2014.

[110] A. Cichocki, R. Zdunek, A. H. Phan and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[111] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Comput. Stat. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.

[112] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, 2013. DOI: 10.1109/TKDE.2012.51.

[113] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, T. Leen, T. Dietterich and V. Tresp, Eds., vol. 13, MIT Press, 2000.

[114] M. Udell, C. Horn, R. Zadeh, S. Boyd *et al.*, "Generalized low rank models," *Found. Trends Mach. Learn.*, vol. 9, no. 1, pp. 1–118, 2016.

[115] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[116] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[117] M. W. Spratling and P. Dayan, "Learning image components for object recognition.," *J. Mach. Learn. Res.*, vol. 7, no. 5, 2006.

[118] A. Kumar and V. Sindhwani, "Near-separable non-negative matrix factorization with ℓ_1 and Bregman loss functions," in *Proc. SIAM Int. Conf. Data Min.*, 2015, pp. 343–351.

[119] W.-K. Ma, J. M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi and C.-Y. Chi, "A signal processing perspective on hyperspectral unmixing: Insights from remote sensing," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 67–81, 2013.

[120] D. Godfrey, C. Johns, C. Meyer, S. Race and C. Sadek, "A case study in text mining: Interpreting twitter data from world cup tweets," *arXiv preprint arXiv:1408.5427*, 2014.

[121] T.-H. Chan, W.-K. Ma, C.-Y. Chi and Y. Wang, "A convex analysis framework for blind separation of non-negative sources," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 5120–5134, 2008.

[122] A. C. Türkmen, "A review of nonnegative matrix factorization methods for clustering," *arXiv preprint arXiv:1507.03194*, 2015.

[123] K. Devarajan, "Nonnegative matrix factorization: An analytical and interpretive tool in computational biology," *PLoS Comput. Biol.*, vol. 4, no. 7, pp. 1–12, 2008.

[124] C. Févotte, N. Bertin and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Comput.*, vol. 21, no. 3, pp. 793–830, 2009.

[125] S. Bhattacharya and N. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2016, pp. 176–189.

[126] Y.-W. Chang, H.-Y. Chen, C. Han, T. Morikawa, T. Takahashi and T.-N. Lin, "FINISH: Efficient and scalable NMF-based federated learning for detecting malware activities," *IEEE Trans. Emerg. Top. Comput.*, vol. 11, no. 4, pp. 934–949, 2023. DOI: 10.1109/TETC.2023.3292924.

[127] Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems," *Comput.*, vol. 42, no. 8, pp. 30–37, 2009. DOI: 10.1109/MC.2009.263.

[128] P. Melville and V. Sindhwani, "Recommender systems.," *Encycl. Mach. Learn.*, vol. 1, pp. 829–838, 2010.

[129] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 127–144, 2014. DOI: 10.1109/MSP.2013.2273004.

[130] C. Zhang, M. Ahmad and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 565–580, 2019.

[131] E. Lari, R. Arablouei and S. Werner, "Privacy-preserving distributed non-negative matrix factorization," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 1022–1026.

[132] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *J. Sci. Comput.*, vol. 66, pp. 889–916, 2016.

[133] Y. Wang, W. Yin and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, pp. 29–63, 2019.

[134] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 1999, pp. 223–238.

[135] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *Proc. IEEE Conf. Decis. Control*, 2015, pp. 6836–6843. DOI: 10.1109/CDC.2015.7403296.

[136] M. Ruan, H. Gao and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Trans. Automat. Control*, vol. 64, pp. 4035–4049, 2019. DOI: 10.1109/TAC.2019.2890887.

[137] R. Fischer, J. Skelley and B. Heisele, *The MIT-CBCL facial expression database*. [Online]. Available: http://cbcl.mit.edu/software-datasets/FaceData2.html.

[138] Y. Zhang, Z. Peng, G. Wen, J. Wang and T. Huang, "Privacy preserving-based resilient consensus for multiagent systems via state decomposition," *IEEE Trans. Control. Netw. Syst.*, vol. 10, no. 3, pp. 1172–1183, 2023. DOI: 10.1109/TCNS.2022.3182234.

[139] L. Xiao, S. Boyd and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.

[140] J. Zhang, J. Lu, J. Liang and K. Shi, "Privacy-preserving average consensus in multiagent systems via partial information transmission," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, pp. 2781–2791, 2023. DOI: 10.1109/TSMC.2022.3220578.

[141] X. Chen, L. Huang, K. Ding, S. Dey and L. Shi, "Privacy-preserving push-sum average consensus via state decomposition," *IEEE Trans. Automat. Contr.*, vol. 68, no. 12, pp. 7974–7981, 2023. DOI: 10.1109/TAC.2023.3256479.

[142] A.-R. Lagos, H. E. Psillakis and A. K. Gkesoulis, "Almost-sure finite-time stochastic min-max consensus," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 9, pp. 3509–3513, 2023. DOI: 10.1109/TCSII.2023.3265567.

[143] Y. Zhang and S. Li, "Distributed biased min-consensus with applications to shortest path planning," *IEEE Trans. Automat. Contr.*, vol. 62, no. 10, pp. 5429–5436, Oct. 2017. DOI: 10.1109/TAC.2017.2694547.

[144] J. Hu, Q. Sun, M. Zhai and B. Wang, "Privacy-preserving consensus strategy for secondary control in microgrids against multilink false data injection attacks," *IEEE Trans. Ind. Inform.*, vol. 19, no. 10, pp. 10 334–10 343, 2023. DOI: 10.1109/TII.2023.3240878.

[145] B. C. Tedeschini, S. Savazzi and M. Nicoli, "Weighted average consensus algorithms in distributed and federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 2, pp. 1369–1382, 2025. DOI: 10.1109/TNSE.2025.3528982.

[146] H. Rezaee and F. Abdollahi, "Average consensus over high-order multiagent systems," *IEEE Trans. Automat. Contr.*, vol. 60, no. 11, pp. 3047–3052, Nov. 2015. DOI: 10.1109/TAC.2015.2408576.

[147] V. Khatana and M. V. Salapaka, "Noise resilient distributed average consensus over directed graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 9, pp. 770–785, 2023. DOI: 10.1109/TSIPN.2023.3324583.

[148] G. Oliva, R. Setola and C. N. Hadjicostis, "Distributed finite-time average-consensus with limited computational and storage capability," *IEEE Trans. Control. Netw. Syst.*, vol. 4, no. 2, pp. 380–391, Jun. 2017. DOI: 10.1109/TCNS.2016.2524983.

[149] W. Chen, L. Liu and G.-P. Liu, "Privacy-preserving distributed economic dispatch of microgrids: A dynamic quantization-based consensus scheme with homomorphic encryption," *IEEE Trans. Smart Grid*, vol. 14, no. 1, pp. 701–713, 2023. DOI: 10.1109/TSG.2022.3189665.

[150] D. Deplano, N. Bastianello, M. Franceschelli and K. H. Johansson, "A unified approach to solve the dynamic consensus on the average, maximum, and median values with linear convergence," in *Proc. IEEE Conf. Decis. Control*, 2023, pp. 6442–6448.

[151] L. Rong, Y. Kan, X. Xie, G.-P. Jiang and S. Xu, "Edge-preserving consensus via non-recursive filters: A parallel system design," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 1, pp. 181–185, 2023. DOI: 10.1109/TCSII.2022.3204942.

[152] L. Gao, Y. Zhou, X. Chen, R. Cai, G. Chen and C. Li, "Privacy-preserving dynamic average consensus via random number perturbation," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 4, pp. 1490–1494, 2023. DOI: 10.1109/TCSII.2022.3219929.

[153] E. Montijano, J. I. Montijano, C. Sagüés and S. Martıénez, "Robust discrete time dynamic average consensus," *Automatica*, vol. 50, no. 12, pp. 3131–3138, 2014.

[154] M. Franceschelli, A. Giua and A. Pisano, "Finite-time consensus on the median value with robustness properties," *IEEE Trans. Automat. Contr.*, vol. 62, no. 4, pp. 1652–1667, 2017. DOI: 10.1109/TAC.2016.2590602.

[155] S. Yu, Y. Chen and S. Kar, "Dynamic median consensus over random networks," in *Proc. IEEE Conf. Decis. Control*, 2021, pp. 5695–5702. DOI: 10.1109/CDC45484.2021.9683179.

[156] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 355–369, Jan. 2009. DOI: 10.1109/TSP.2008.2007111.

[157] M. Abdelrahim, J. M. Hendrickx and W. Heemels, "Max-consensus in open multi-agent systems with gossip interactions," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 4753–4758. DOI: 10.1109/CDC.2017.8264362.

[158] A. Nowzari and M. G. Rabbat, "Improved bounds for max consensus in wireless networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 2, pp. 305–319, Jun. 2019.

[159] S. Zhang, C. Tepedelenlioğlu, M. K. Banavar and A. Spanias, "Max consensus in sensor networks: Non-linear bounded transmission and additive noise," *IEEE Sens. J.*, vol. 16, no. 24, pp. 9089–9098, Dec. 2016. DOI: 10.1109/JSEN.2016.2612642.

[160] G. Muniraju, C. Tepedelenlioglu and A. Spanias, "Analysis and design of robust max consensus for wireless sensor networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 779–791, Dec. 2019.

[161] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed maximum consensus," *IEEE Signal Process. Lett.*, vol. 27, pp. 1839–1843, 2020. DOI: 10.1109/LSP.2020.3029706.

[162] M. Lippi, A. Furchì, A. Marino and A. Gasparri, "An adaptive distributed protocol for finite-time infimum or supremum dynamic consensus," *IEEE Control Syst. Lett.*, vol. 7, pp. 401–406, 2023. DOI: 10.1109/LCSYS.2022.3188941.

[163] D. Deplano, M. Franceschelli and A. Giua, "Dynamic min and max consensus and size estimation of anonymous multiagent networks," *IEEE Trans. Automat. Contr.*, vol. 68, no. 1, pp. 202–213, 2023. DOI: 10.1109/TAC.2021.3135452.

[164] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, Jan. 2009. DOI: 10.1109/TAC.2008.2009515.

[165] E. Lari, R. Arablouei, N. K. Venkategowda and S. Werner, "Distributed maximum consensus over noisy links," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 2247–2251.

Appendix A

Publication 1

P1 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "On the resilience of online federated learning to model poisoning attacks through partial sharing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2024, pp. 9201–9205.

ON THE RESILIENCE OF ONLINE FEDERATED LEARNING TO MODEL POISONING ATTACKS THROUGH PARTIAL SHARING

Ehsan Lari[1], Vinay Chakravarthi Gogineni[2], Reza Arablouei[3], and Stefan Werner[1]

[1]Department of Electronic Systems, Norwegian University of Science and Technology, Norway
[2]The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark
[3]CSIRO's Data61, Pullenvale QLD 4069, Australia

ABSTRACT

We investigate the robustness of the recently introduced partial-sharing online federated learning (PSO-Fed) algorithm against model-poisoning attacks. To this end, we analyze the performance of the PSO-Fed algorithm in the presence of Byzantine clients, who may clandestinely corrupt their local models with additive noise before sharing them with the server. PSO-Fed can operate on streaming data and reduce the communication load by allowing each client to exchange parts of its model with the server. Our analysis, considering a linear regression task, reveals that the convergence of PSO-Fed can be ensured in the mean sense, even when confronted with model-poisoning attacks. Our extensive numerical results support our claim and demonstrate that PSO-Fed can mitigate Byzantine attacks more effectively compared with its state-of-the-art competitors. Our simulation results also reveal that, when model-poisoning attacks are present, there exists a non-trivial optimal stepsize for PSO-Fed that minimizes its steady-state mean-square error.

1. INTRODUCTION

Federated learning (FL) [1–5] is a distributed learning paradigm that allows a network of devices, such as smartphones, media streamers, and Internet of things nodes, to collaboratively learn a global shared model without explicitly sharing their local raw data with the server or other devices. Two key features of FL are its ability to handle heterogeneous data, such as non-independent and identically distributed (non-IID) or unbalanced local data, as well as to run on heterogeneous devices that may have limited resources [6–9].

Communication load in FL refers to the amount of data exchanged between the central server and the participating devices during the training process. It can impact the efficiency and scalability of FL in real-world applications as high data transmission can increase resource utilization and cost [10, 11]. Several algorithms have been proposed to decrease the communication load in FL. Sign stochastic gradient descent (SignSGD) [12] uses sign-based gradient compressors to lower the computational and communication costs of FL. Compressive sensing FL (CS-Fed) [13] improves communication efficiency using 1-bit compressive sensing in analog aggregation transmissions. Quantile sketch FL (QS-Fed) [14] achieves enhanced communication efficiency by compressing model updates using a so-called count sketch. In Online-Fed [15], the server selects a random subset of the clients in every iteration to reduce the communication load. However, none of the mentioned works consider any adversary in the network that may aim to impair the learning process through, for example, tampering with the message exchanges or intentionally providing corrupted or incorrect information. considering the presence of malicious actors in the network is an important aspect of developing robust and reliable FL algorithms.

In any network, Byzantine clients are clients that exhibit erratic or malicious behavior. These clients may send false information or fail to follow the protocols, which can lead to disruption or performance degradation in FL [16, 17]. Byzantine clients can undermine the integrity or reliability of the global model learned via FL by, e.g., sending incorrect/inconsistent gradients or model weights during updates or launching a denial-of-service attack to disrupt the FL process [18, 19]. Several methods have been proposed to address Byzantine clients in FL. They include utilizing robust aggregate statistics, assigning a trust score to each client, and using historical information to recover from model-poisoning attacks [20–22]. However, these methods require additional communication resources or increase the complexity of local model updates at the clients. This contradicts the essence of FL, which aims to leverage clients with constrained energy or computational resources.

In this paper, we examine the performance of a communication-efficient FL algorithm, namely, partial-sharing online FL (PSO-Fed) [10, 23], in the face of Byzantine attacks. We show that improving communication efficiency through partial sharing in online FL can have the additional benefit of mitigating the adverse effects of model-poisoning attacks. We analyze the performance of PSO-Fed under Byzantine attacks, considering a linear regression task. We study the mean convergence of PSO-Fed and conduct numerical experiments using synthetic non-IID data. The results indicate that PSO-Fed is more resilient to Byzantine attacks compared to other state-of-the-art algorithms, including Online-Fed, without incurring any extra computations. In addition, the simulation results uncover the existence of a non-trivial optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. Interestingly, we also observe that a smaller stepsize does not necessarily improve the estimation accuracy of PSO-Fed, which is in contrast to other gradient-descent-based algorithms that often utilize small stepsizes to enhance performance, albeit at the expense of slower convergence rate.

2. PRELIMINARIES

In this section, we first provide an overview of the Online-Fed algorithm [15] for linear regression. We then briefly describe the PSO-Fed algorithm [23], which is a communication-efficient variant of Online-Fed. Afterward, we outline the model-poisoning Byzantine attack model within the context of FL and probe its impact on PSO-Fed.

2.1. System Model

We consider a federated network consisting of K clients that can communicate with a central server. At every time instance n, each client k has access to a streaming data pair, $\mathbf{x}_{k,n} \in \mathbb{R}^D$ and $y_{k,n} \in \mathbb{R}$, which

This work was supported by the Research Council of Norway. *The first two authors contributed equally to this work. (Corresponding author: Ehsan Lari.)*

are related via the model

$$y_{k,n} = \mathbf{w}^{\intercal}\mathbf{x}_{k,n} + \nu_{k,n}, \tag{1}$$

where the model parameter vector $\mathbf{w} \in \mathbb{R}^D$ is to be estimated collaboratively from the locally stored client data and $\nu_{k,n}$ is the observation noise. We define the global objective function for estimating $\mathbf{w}$ as

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}) \tag{2}$$

with the local objective function at client k being

$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E}\left[|y_{k,n} - \mathbf{w}^{\intercal}\mathbf{x}_{k,n}|^2\right]. \tag{3}$$

The goal is to find the optimal estimate $\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} \mathcal{J}(\mathbf{w})$ in a distributed fashion via FL.

2.2. PSO-Fed

At each iteration of Online-Fed [15], the server randomly selects a subset of the clients. The selected clients then perform a stochastic gradient-descent step to solve their individual local optimization problems and subsequently share their updated local estimates with the server. In PSO-Fed, to further alleviate the communication burden, the server only sends a segment of the global model estimate to the clients. Likewise, the clients transmit segments of their local model estimates to the server. The model parameters exchanged at iteration n between client k and the server are specified using a diagonal selection matrix, denoted by $\mathbf{S}_{k,n} \in \mathbb{R}^{D\times D}$, with M ones. The positions of these ones on the diagonal dictate which model parameters are communicated with the server during each iteration. These positions can be selected arbitrarily or follow a round-robin pattern as described in [24, 25] such that, on average, the model parameters are exchanged between each client and the server M times in every D iterations. Therefore, the likelihood of any model parameter being shared with the server in any given iteration is $p_e = \frac{M}{D}$.

Using the selection matrices $\mathbf{S}_{k,n}$, the recursion equations of PSO-Fed minimizing (2) can be expressed as [10]:

$$\epsilon_{k,n} = y_{k,n} - [\mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_D - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n}]^{\intercal}\,\mathbf{x}_{k,n} \tag{4a}$$

$$\mathbf{w}_{k,n+1} = \mathbf{S}_{k,n}\mathbf{w}_n + (\mathbf{I}_D - \mathbf{S}_{k,n})\,\mathbf{w}_{k,n} + \mu\,\mathbf{x}_{k,n}\,\epsilon_{k,n} \tag{4b}$$

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}\left[\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1} + (\mathbf{I}_D - \mathbf{S}_{k,n+1})\,\mathbf{w}_n\right], \tag{4c}$$

where $\mathbf{w}_{k,n}$ is the local model estimate at client k and iteration n, $\mathbf{w}_n$ is the global model estimate at iteration n, $\mathbf{I}_D$ is the $D \times D$ identity matrix, μ is the stepsize controlling the convergence rate and steady-state performance, $\mathcal{S}_n$ denotes the set of clients selected at iteration n, and $|\mathcal{S}_n|$ denotes the cardinality of $\mathcal{S}_n$.

2.3. Model-Poisoning Attack

Let us denote the set of potential Byzantine clients in the network as $\mathcal{S}_B$. Let us also define the indicator variable β_k such that $\beta_k = 1$ if client $k \in \mathcal{S}_B$ and $\beta_k = 0$ otherwise. We denote the count of Byzantine clients as $|\mathcal{S}_B|$. These clients seek to compromise the accuracy of the global model by intermittently sending false or corrupted local model estimates to the server while possessing the true local model estimates. Specifically, we consider the scenario where, at each iteration, every Byzantine client corrupts its local model estimate by adding noise before sending it to the server with an attack probability of p_a. Therefore, we express the model estimate shared by any Byzantine client as $\mathbf{w}_{k,n+1} + \tau_{k,n}\boldsymbol{\delta}_{k,n}$. Here, $\tau_{k,n}$ is a Bernoulli random variable that takes the value of 1 with a probability of p_a and 0 otherwise. In addition, $\boldsymbol{\delta}_{k,n} \in \mathbb{R}^D$ represents the perturbation signal associated with the poisoning attack, which is typically modeled as zero-mean white Gaussian noise, i.e, $\boldsymbol{\delta}_{k,n} \sim \mathcal{N}(\mathbf{0}, \sigma_B^2\mathbf{I}_D)$ [26].

Therefore, to account for the impact of Byzantine clients, we consider that the clients share $\mathbf{w}_{k,n+1} + \beta_k\tau_{k,n}\boldsymbol{\delta}_{k,n}$ with the server, rather than $\mathbf{w}_{k,n+1}$. Consequently, we reformulate the global model update in PSO-Fed, (4c), as

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|}\sum_{k\in\mathcal{S}_n}\left[\mathbf{S}_{k,n+1}(\mathbf{w}_{k,n+1} + \beta_k\tau_{k,n}\boldsymbol{\delta}_{k,n}) + (\mathbf{I}_D - \mathbf{S}_{k,n+1})\,\mathbf{w}_n\right]. \tag{5}$$

3. ANALYSIS

In this section, we analyze the theoretical mean convergence of PSO-Fed in the presence of Byzantine clients to evaluate its robustness to Byzantine attacks. We first define the extended optimal global model $\mathbf{w}_e^{\star} = \mathbf{1}_{K+1} \otimes \mathbf{w}^{\star}$, the extended global model estimate $\mathbf{w}_{e,n} = \mathrm{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \ldots, \mathbf{w}_{K,n}\}$, and the collective quantities $\mathbf{X}_n = \mathrm{bdiag}\{\mathbf{0}, \mathbf{x}_{1,n}, \ldots, \mathbf{x}_{K,n}\}$, $\boldsymbol{\delta}_{e,n} = \mathrm{col}\{\mathbf{0}, \boldsymbol{\delta}_{1,n}, \ldots, \boldsymbol{\delta}_{K,n}\}$, $\mathbf{B} = \mathrm{bdiag}\{0, \beta_1\mathbf{I}_D, \ldots, \beta_K\mathbf{I}_D\}$, $\mathbf{T}_n = \mathrm{bdiag}\{0, \tau_{1,n}\mathbf{I}_D, \ldots, \tau_{K,n}\mathbf{I}_D\}$, and $\boldsymbol{\nu}_{e,n} = \mathrm{col}\{0, \nu_{1,n}, \ldots, \nu_{K,n}\}$. The operators $\mathrm{col}\{\cdot\}$, $\mathrm{diag}\{\cdot\}$, and $\mathrm{bdiag}\{\cdot\}$ denote column-wise stacking, diagonalization, and block diagonalization, respectively, and $\mathbf{1}_{K+1}$ is a column vector with $K+1$ entries, all of which are equal to one. Given these definitions, we have

$$\mathbf{y}_{e,n} = \mathrm{col}\{0, y_{1,n}, y_{2,n}, \ldots, y_{K,n}\} = \mathbf{X}_n^{\intercal}\mathbf{w}_e^{\star} + \boldsymbol{\nu}_{e,n} \tag{6a}$$

$$\boldsymbol{\epsilon}_{e,n} = \mathrm{col}\{0, \epsilon_{1,n}, \epsilon_{2,n}, \ldots, \epsilon_{K,n}\} = \mathbf{y}_{e,n} - \mathbf{X}_n^{\intercal}\boldsymbol{\mathcal{A}}_n\mathbf{w}_{e,n} \tag{6b}$$

$$\boldsymbol{\mathcal{A}}_n = \begin{bmatrix} \mathbf{I}_D & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ a_{1,n}\mathbf{S}_{1,n} & \mathbf{I}_D - a_{1,n}\mathbf{S}_{1,n} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{K,n}\mathbf{S}_{K,n} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_D - a_{K,n}\mathbf{S}_{K,n} \end{bmatrix}, \tag{7}$$

where $a_{k,n} = 1$ if $k \in \mathcal{S}_n$ and $a_{k,n} = 0$ otherwise. Hence, the global recursion equations of PSO-Fed can be stated as

$$\mathbf{w}_{e,n+1} = \boldsymbol{\mathcal{B}}_{n+1}\left(\boldsymbol{\mathcal{A}}_n\mathbf{w}_{e,n} + \mu\mathbf{X}_n\boldsymbol{\epsilon}_{e,n}\right) + \boldsymbol{\mathcal{C}}_{n+1}\mathbf{B}\mathbf{T}_n\boldsymbol{\delta}_{e,n} \tag{8}$$

$$\boldsymbol{\mathcal{B}}_{n+1} = \begin{bmatrix} \mathbf{I}_D - \sum_{k=1}^{K}\frac{a_{k,n}}{|\mathcal{S}_n|}\mathbf{S}_{k,n+1} & \frac{a_{1,n}}{|\mathcal{S}_n|}\mathbf{S}_{1,n+1} & \ldots & \frac{a_{K,n}}{|\mathcal{S}_n|}\mathbf{S}_{K,n+1} \\ \mathbf{0} & \mathbf{I}_D & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_D \end{bmatrix} \tag{9}$$

$$\boldsymbol{\mathcal{C}}_{n+1} = \begin{bmatrix} \mathbf{0} & \frac{a_{1,n}}{|\mathcal{S}_n|}\mathbf{S}_{1,n+1} & \ldots & \frac{a_{K,n}}{|\mathcal{S}_n|}\mathbf{S}_{K,n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix}. \tag{10}$$

To facilitate our analysis, we make the following assumptions.

A1: The sequence of local input vectors of each client, $\mathbf{x}_{k,n}$, is a wide-sense stationary multivariate random process with the covariance matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{x}_{k,n}\mathbf{x}_{k,n}^{\intercal}]$.

A2: The observation noise, $\nu_{k,n}$, and the attack signal, $\boldsymbol{\delta}_{k,n}$, are IID and independent of each other as well as all other considered stochastic variables.

A3: The selection matrices, $\mathbf{S}_{k,n}$, at all clients and iterations are independent of each other.

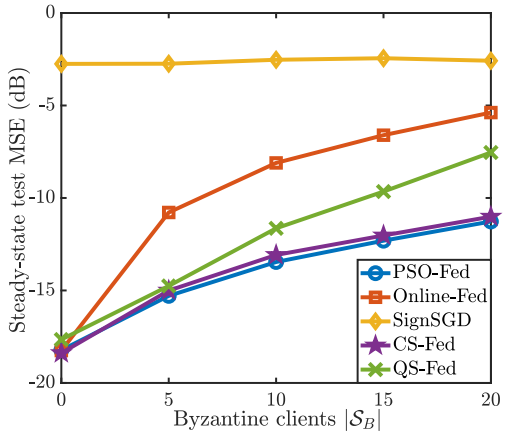

Fig. 1. Steady-state test MSE for different algorithms with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 1$.

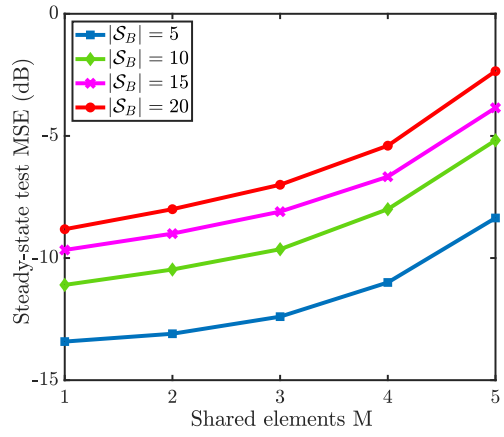

Fig. 2. Steady-state test MSE of PSO-Fed for different values of shared elements M with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 1$.

3.1. Mean Convergence

Let us define the deviation (estimation error) vector as $\widetilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^\star - \mathbf{w}_{e,n}$. Since $\boldsymbol{\mathcal{B}}_{n+1}$ and $\boldsymbol{\mathcal{A}}_n$ are right-stochastic, i.e., their rows add up to 1, we have $\boldsymbol{\mathcal{B}}_{n+1}\mathbf{w}_e^\star = \boldsymbol{\mathcal{A}}_n\mathbf{w}_e^\star = \mathbf{w}_e^\star$. Therefore, we can rewrite (8) in terms of $\widetilde{\mathbf{w}}_{e,n}$ as

$$\widetilde{\mathbf{w}}_{e,n+1} = \boldsymbol{\mathcal{B}}_{n+1}\big(\mathbf{I} - \mu\mathbf{X}_n\mathbf{X}_n^\intercal\big)\boldsymbol{\mathcal{A}}_n\widetilde{\mathbf{w}}_{e,n} - \mu\boldsymbol{\mathcal{B}}_{n+1}\mathbf{X}_n\boldsymbol{\nu}_{e,n} - \boldsymbol{\mathcal{C}}_{n+1}\mathbf{B}\mathbf{T}_n\boldsymbol{\delta}_{e,n}. \tag{11}$$

Taking the expected value of both sides of (11) and applying the assumptions A1-A3, we have

$$\mathbb{E}[\widetilde{\mathbf{w}}_{e,n+1}] = \mathbb{E}[\boldsymbol{\mathcal{B}}_{n+1}]\big(\mathbf{I} - \mu\boldsymbol{\mathcal{R}}\big)\mathbb{E}[\boldsymbol{\mathcal{A}}_n]\mathbb{E}[\widetilde{\mathbf{w}}_{e,n}], \tag{12}$$

where $\boldsymbol{\mathcal{R}} = \text{bdiag}\{\mathbf{0}, \mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_K\}$. Hence, the mean convergence of PSO-Fed is guaranteed if $\| \left(\mathbf{I} - \mu\boldsymbol{\mathcal{R}}\right) \| < 1$ or equivalently $\|1 - \mu\lambda_i\left(\mathbf{R}_k\right)\| < 1 \; \forall k, i$, where $\|\cdot\|$ denotes any matrix norm and $\lambda_i\left(\cdot\right)$ stands for the ith eigenvalue. In other words, PSO-Fed converges in the mean sense, even at the presence of Byzantine clients, provided that the stepsize μ is set such that $0 < \mu < 2/\max_{i,k}\{\lambda_i\left(\mathbf{R}_k\right)\}$.

4. SIMULATION RESULTS

In this section, we conduct several numerical experiments to examine the effect of model-poisoning attacks on the performance of PSO-Fed in comparison with the related existing algorithms, which can also operate on streaming data, namely, Online-Fed, SignSGD, CS-fed, and QS-Fed. We consider a federated network of $K = 100$ clients. At time instant n, each client k has access to a non-IID input vector $\mathbf{x}_{k,n} \in \mathbb{R}^D$ and its corresponding response value $y_{k,n}$ that relate to each other via (1) given $\mathbf{w}^\intercal = \frac{1}{\sqrt{D}}[1, \cdots, 1]^\intercal \in \mathbb{R}^D$.

Each entry of $\mathbf{x}_{k,n}$ follows a zero-mean Gaussian distribution with variance ς_k^2 sampled from a uniform distribution with lower bound 0.2 and upper bound 1.2, i.e., $\mathcal{U}(0.2, 1.2)$. In addition, the observation noise $\nu_{k,n}$ is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$ drawn from $\mathcal{U}(0.005, 0.025)$. We set the size of the model parameter vector to $D = 5$ and share only $M = 1$ of its entries during each iteration unless otherwise specified.

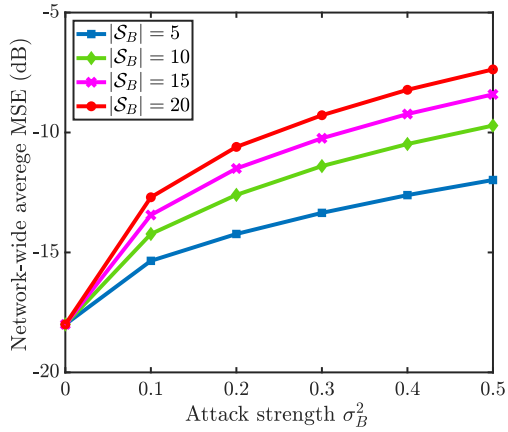

Fig. 3. Network-wide average MSE of PSO-Fed for different values of attack strengths σ_B^2 and Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack probability $p_a = 1$.

We assume that, in each iteration, all algorithms have the same communication budget. Therefore, to ensure that all algorithms can achieve their optimal performance within 5000 iterations, we set $\mu = 0.1$ for PSO-Fed, Online-Fed, CS-fed, and QS-fed, and $\mu = 0.08$ for SignSGD, across all clients. In every iteration n, the server selects a set of $|\mathcal{S}_n| = 5$ clients randomly to participate in FL, with each client having an equal probability of being selected. We evaluate the performance of the considered algorithms at the server's side using a test dataset consisting of $N_t = 50$ examples, $\{\check{\mathbf{X}}, \check{\mathbf{y}}\}$, and calculating the test mean square error (MSE) at iteration n as $\frac{1}{N_t}\|\check{\mathbf{y}} - \check{\mathbf{X}}^\intercal\mathbf{w}_n\|_2^2$. We also evaluate the performance at the clients' side using (4a) and calculating the network-wide average steady-state MSE as $\frac{1}{K}\sum_{k=1}^K \lim_{n\to\infty}\epsilon_{k,n}^2$.

In our first experiment, we investigate the effect of varying the

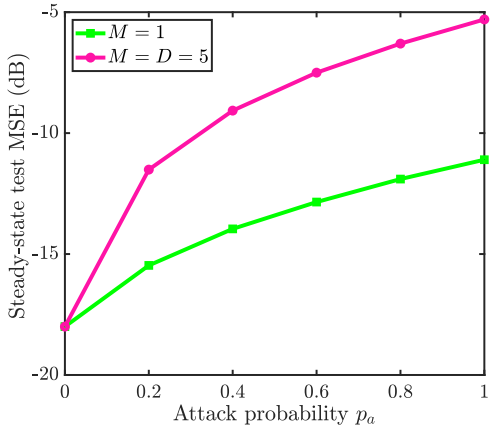

Fig. 4. Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of shared elements M, numbers of Byzantine clients $|\mathcal{S}_B| = 10$ and attack strength $\sigma_B^2 = 0.5$.

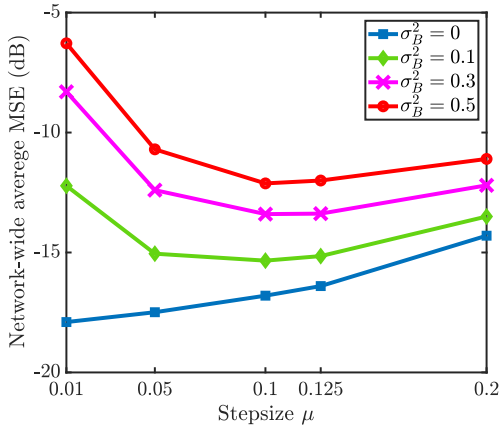

Fig. 5. Network-wide average MSE of PSO-Fed for different values of stepsize μ, numbers of Byzantine clients $|\mathcal{S}_B| = 5$ and attack probability $p_a = 1$.

number of Byzantine clients $|\mathcal{S}_B|$ on the steady-state test MSE for PSO-Fed, Online-Fed, SignSGD, CS-fed, and QS-Fed. In this experiment, we use different numbers of Byzantine clients, all characterized by $\sigma_B^2 = 0.25$ and an attack probability of $p_a = 1$. We present the results in Fig. 1. We observe that PSO-Fed outperforms the existing algorithms across all considered numbers of Byzantine clients. The Online-Fed algorithm performs similar to PSO-Fed when there is no Byzantine client or poisoning attack (i.e., $\sigma_B^2 = 0$). However, its performance deteriorates as the number of Byzantine clients increases.

In our second experiment, we examine the impact of the number of shared entries, M, on the steady-state test MSE of PSO-Fed. We simulate PSO-Fed with $|\mathcal{S}_B| \in \{5, 10, 15, 20\}$ Byzantine clients, the attack strength $\sigma_B^2 = 0.5$, and the attack probability $p_a = 1$. The results are shown in Fig. 2. We observe that increasing M leads to higher steady-state test MSE. In essence, PSO-Fed demonstrates greater resilience to model-poisoning attacks compared to Online-Fed, without incurring any additional computational and communication overhead.

In our third experiment, we study the effect of attack strength σ_B^2 on the network-wide average MSE for different numbers of Byzantine clients $|\mathcal{S}_B|$. We show the results in Fig. 3. It is evident that increasing the attack strength or the number of Byzantine clients leads to higher MSE. Another important observation from Fig. 3 is that when there are $|\mathcal{S}_B| = \alpha_1$ Byzantine clients with attack strength $\sigma_B^2 = \beta_1$, we obtain the same MSE as the case when there are $|\mathcal{S}_B| = \alpha_2$ Byzantine clients with attack strength $\sigma_B^2 = \beta_2$ as long as $\alpha_1\beta_1 = \alpha_2\beta_2$. Consequently, a more covert strategy for orchestrating an attack can involve employing a greater number of Byzantine clients with lower attack strength to achieve the same level of performance degradation.

In our fourth experiment, we investigate the impact of attack probability p_a on the steady-state test MSE of PSO-Fed for different values of M. We simulate PSO-Fed while having $|\mathcal{S}_B| = 10$ Byzantine clients with attack strength $\sigma_B^2 = 0.5$ and different attack probabilities. The results are shown in Fig. 4. We observe that increasing the attack probability leads to a higher MSE. However, partial sharing (i.e., when $M < D$) enhances resilience against model-poisoning attacks.

In our final experiment, we assess the performance of PSO-Fed with different stepsizes μ in the presence and absence of model-poisoning attacks. To this end, we simulate PSO-Fed when there are $|\mathcal{S}_B| = 5$ Byzantine clients with attack probability of $p_a = 1$. The results are presented in Fig. 5. Notably, when there is no Byzantine client or poisoning attack (i.e., $\sigma_B^2 = 0$), the performance of PSO-Fed deteriorates with increasing stepsize μ. This is consistent with our understanding of algorithms employing gradient descent for optimization. However, in the presence of any model-poisoning attack (i.e., $\sigma_B^2 > 0$), we observe an intriguing trend. The performance initially improves as the stepsize μ increases, but is starts to degrade as μ further increases. This suggests that there is an optimal value for the stepsize ($\mu_{opt} \approx 0.1$) that can provide the best performance for PSO-Fed when there is any model-poisoning attack. Note that, in this experiment, the largest stepsize that still guarantees convergence in the mean is $\mu_{max} \approx 0.25$.

5. CONCLUSION AND FUTURE WORK

We revisited the recently proposed communication-efficient PSO-Fed algorithm and presented a theoretical analysis demonstrating its resilience to model-poisoning (Byzantine) attacks that is imparted by its partial-sharing property. We conducted the analysis considering a linear regression task, where the local objective function at each client is an empirical risk. We showed that, even under Byzantine attacks, PSO-Fed converges in the mean sense, provided that an appropriate stepsize is chosen. We also demonstrated that, in the presence of Byzantine clients, PSO-Fed achieves a significantly lower steady-state MSE compared to the Online-Fed algorithm, which lacks partial sharing. Our comprehensive simulation results showcased PSO-Fed's effectiveness in dealing with Byzantine attacks. They also revealed the non-trivial nature of determining the stepsize that leads to the lowest MSE in PSO-Fed when Byzantine clients are present. In future work, we will theoretically analyze the mean-square performance of PSO-Fed when facing model-poisoning attacks. We aim to formulate the theoretical MSE of PSO-Fed as a function of various parameters, including stepsize, attack probability, number of Byzantine clients, number of participating clients in each iteration, number of shared entries, and variance of observation noise. We will also investigate the underlying reasons for the existence of a non-trivial optimal stepsize. Furthermore, we will explore scenarios where random Fourier features can be utilized to expand our study to encompass generalized linear regression and estimation of nonlinear functions.

6. REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[2] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Adv. Neural Inf. Process. Syst.*, vol. 30, Jan. 2017.

[3] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[4] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated learning with non-IID data in wireless networks," *" IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[5] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, 2021.

[6] Q. Yang, Y. L., T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Feb. 2019.

[7] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 2031–2063, Apr. 2020.

[8] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor, "Federated learning: A signal processing perspective," *IEEE Signal Process. Mag.*, vol. 39, no. 3, pp. 14–41, May 2022.

[9] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.

[10] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4531–4544, Mar. 2023.

[11] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.

[12] R. Jin, Y. Huang, X. He, H. Dai, and T. Wu, "Stochastic-sign SGD for federated learning with theoretical guarantees," *arXiv preprint arXiv:2002.10940*, Feb. 2020.

[13] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "1-bit compressive sensing for efficient federated learning over the air," *IEEE Trans. Wireless Commun.*, Oct. 2022.

[14] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. Int. Conf. Machine Learning*, Jul. 2020.

[15] A. Kuh, "Real time kernel learning for sensor networks using principles of federated learning," in *Proc. IEEE Int. Conf. Asia-Pacific Signal Info. Process. Assoc.*, Dec. 2021, pp. 2089–2093.

[16] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proc. Int. Conf. Machine Learning*, Jul. 2018, pp. 3521–3530.

[17] X. Cao and N. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2022, pp. 3395–3403.

[18] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[19] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *USENIX Security Symp.*, 2020, pp. 1605–1622.

[20] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Machine Learning*, 2018, pp. 5650–5659.

[21] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[22] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "FedRecover: Recovering from poisoning attacks in federated learning using historical information," in *Proc. IEEE Symp. Security Privacy*, Jul. 2023, pp. 326–343.

[23] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning framework for nonlinear regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232.

[24] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 472–484, Jan. 2014.

[25] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014.

[26] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 145–158, Mar. 2017.

Appendix B

Publication 2

P2 E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, "Resilience in Online Federated Learning: Mitigating Model-Poisoning Attacks via Partial Sharing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 388–400, 2025.

Resilience in Online Federated Learning: Mitigating Model-Poisoning Attacks via Partial Sharing

Ehsan Lari, *Graduate Student Member, IEEE*, Reza Arablouei, Vinay Chakravarthi Gogineni, *Senior Member, IEEE*, and Stefan Werner, *Fellow, IEEE*

***Abstract*—Federated learning (FL) allows training machine learning models on distributed data without compromising privacy. However, FL is vulnerable to model-poisoning attacks where malicious clients tamper with their local models to manipulate the global model. In this work, we investigate the resilience of the partial-sharing online FL (PSO-Fed) algorithm against such attacks. PSO-Fed reduces communication overhead by allowing clients to share only a fraction of their model updates with the server. We demonstrate that this partial sharing mechanism has the added advantage of enhancing PSO-Fed's robustness to model-poisoning attacks. Through theoretical analysis, we show that PSO-Fed maintains convergence even under Byzantine attacks, where malicious clients inject noise into their updates. Furthermore, we derive a formula for PSO-Fed's mean square error, considering factors like stepsize, attack probability, and the number of malicious clients. Interestingly, we find a non-trivial optimal stepsize that maximizes PSO-Fed's resistance to these attacks. Extensive numerical experiments confirm our theoretical findings and showcase PSO-Fed's superior performance against model-poisoning attacks compared to other leading FL algorithms.**

***Index Terms*—Online federated learning, Byzantine clients, partial-sharing, linear regression, model-poisoning.**

I. Introduction

FEDERATED learning (FL) [2], [3] is a distributed learning paradigm that enables a network of devices, such as smartphones, internet of things (IoT) nodes, and media streamers, to jointly learn a global model without the need to directly disclose their local raw data to the server or other devices. FL is particularly beneficial when data is distributed across numerous devices, and conducting centralized training by transferring the data to the server is unfeasible or impractical [4], [5]. FL addresses two major challenges: the ability to handle heterogeneous data, such as non-independent and identically distributed (non-IID) or unbalanced data, and the ability to run on heterogeneous devices that may have limited resources [6]–[9]. While FL enhances privacy, it also introduces vulnerabilities stemming from its reliance on unverified client updates and its susceptibility to adversarial manipulation. Malicious clients can exploit these weaknesses to degrade model performance or compromise its integrity. Developing robust mechanisms to detect and mitigate such adversarial behaviors is essential to address these challenges [10].

The communication load in FL refers to the volume of data transmitted between the central server and participating devices during training. It can affect the efficiency and scalability of FL in real-world applications as excessive data transmission may escalate resource usage and the cost associated with training [11]–[13]. Several algorithms have been proposed to alleviate the communication load in FL. Among them, sign stochastic gradient descent (SignSGD) [14], [15] utilizes sign-based gradient compressors to lower the communication and computational costs of FL. Compressive sensing FL (CS-Fed) [16] reduces the communication load using 1-bit compressive sensing in analog aggregation transmissions. Quantile sketch FL (QS-Fed) [17] improves communication efficiency by compressing model updates using a count sketch. Online-Fed [18] enhances communication efficiency by engaging only a randomly selected subset of clients to communicate with the server in each iteration. However, the aforementioned works do not consider the potential presence of adversaries within the network who may seek to undermine the learning process. This can be through tampering with message exchanges or intentionally providing corrupted or incorrect information. Factoring in the possibility of malicious clients in the network is an important aspect of developing reliable and robust FL algorithms.

In network environments, Byzantine clients present a critical challenge due to their unpredictable or malicious behavior, potentially disrupting training processes. Such clients may introduce misleading gradients or manipulated model updates [19]–[23], leading to performance degradation and compromised global model integrity. Particularly concerning are model-poisoning attacks, where adversarial clients deliberately manipulate their updates during training [24]. Addressing these threats has become a central focus in FL research, driving the

Manuscript received 16 August 2024; revised 15 January 2025 and 27 March 2025; accepted 5 April 2025. This work was supported by the Research Council of Norway. Part of this work has been presented to the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, South Korea, Apr. 2024 [DOI: 10.1109/ICASSP48485.2024.10447497] [1]. The associate editor coordinating the review of this article and approving it for publication was Prof. Gang Wang. *(Corresponding author: Ehsan Lari.)*

Ehsan Lari and Stefan Werner are with the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: {ehsan.lari, stefan.werner}@ntnu.no). Stefan Werner is also with the Department of Information and Communications Engineering, Aalto University, 00076, Finland.

Reza Arablouei is with the Commonwealth Scientific and Industrial Research Organisation, Pullenvale, QLD 4069, Australia (e-mail: reza.arablouei@csiro.au).

Vinay Chakravarthi Gogineni is with the Applied AI and Data Science Unit, The Maersk Mc-Kinney Moller Institute, Faculty of Engineering, University of Southern Denmark, Odense 5230, Denmark (e-mail: vigo@mmmi.sdu.dk).

Digital Object Identifier XXXXXXXXXXXXXXX

development of various Byzantine-robust strategies.

A significant portion of this research centers on robust aggregation techniques [25]–[30], designed to mitigate the influence of adversarial contributions. Techniques such as median and trimmed mean aggregations have demonstrated effectiveness in reducing vulnerability to malicious updates [25]. Blockchain-assisted approaches [31] offer enhanced robustness, albeit at the cost of increased communication and computational overhead. Additionally, anomaly detection techniques [32] and reputation-based filtering [33] aim to improve resilience by identifying and excluding suspicious updates. However, these methods often rely on specific assumptions regarding adversarial behavior, potentially limiting their practical applicability in diverse real-world FL scenarios.

SignSGD-based algorithms [34] provide an alternative robust solution, demonstrating resilience to adversarial updates. Nevertheless, these methods typically assume full client participation, an unrealistic assumption for practical FL scenarios involving intermittent client availability. This reveals a critical gap: the need for robust FL algorithms that balance computational efficiency and resilience without requiring full participation or imposing extra computational demands on clients. Addressing this gap is central to scalable and secure FL deployments.

In this paper, we study the impact of partial parameter sharing in online FL (Online-Fed) [1], [11], [12], [18] on the resilience against Byzantine attacks by malicious clients. Our primary motivation stems from a key observation: partial sharing, initially introduced to enhance communication efficiency in online FL, offers a surprising additional benefit by mitigating the negative effects of model-poisoning attacks. Remarkably, this advantage is achieved without imposing any additional computational burden on participating clients. In summary, our main contributions in this work are:

- We introduce an intermittent model-poisoning attack model along with random scheduling of the clients within the FL framework.
- We derive a non-trivial optimal stepsize for PSO-Fed under poisoning attacks, contrasting with existing algorithms for which smaller stepsizes are always favorable.
- We demonstrate PSO-Fed's superior performance in terms of MSE and robustness to model-poisoning attacks without any additional computational overhead, highlighting practical advantages over existing FL approaches.

We organize the remainder of the paper as follows. In section II, we provide an overview of the system model, PSO-Fed algorithm, and model-poisoning attacks. In section III, we evaluate the robustness of PSO-Fed against Byzantine attacks by analyzing its theoretical mean and mean-square convergence. In this section, we also calculate PSO-Fed's theoretical steady-state MSE and optimal stepsize. In section IV, we verify our theoretical findings through a series of numerical experiments. In these experiments, we evaluate the performance of PSO-Fed under model-poisoning attacks through both theoretical predictions and numerical simulations. In section V, we provide a more detailed explanation of our contributions and discuss the potential benefits and impacts of our work. Finally, in section VI, we present some concluding remarks.

Mathematical Notations: We denote scalars by lowercase letters, column vectors by bold lowercase letters, and matrices by bold uppercase letters. The superscripts $(\cdot)^\intercal$ and $(\cdot)^{-1}$ denote the transpose and inverse operations, respectively, while $\|\cdot\|$ signifies the Euclidean norm. Additionally, $\mathbf{1}_K$ denotes a column vector with K entries, all set to one, and $\mathbf{I}_K$ is the $K \times K$ identity matrix. The operators $\mathrm{col}\{\cdot\}$ and $\mathrm{bdiag}\{\cdot\}$ represent column-wise stacking and block diagonalization, respectively. Furthermore, $\otimes_b$ denotes the block Kronecker product [35], $\mathrm{bvec}\{\cdot\}$ refers to the block vectorization operation, and $\mathrm{bvec}^{-1}\{\cdot\}$ is its inverse. Lastly, $\mathrm{tr}(\cdot)$ denotes the trace of a matrix.

II. Problem Formulation

In this section, we first describe the considered system model. We then briefly outline the PSO-Fed algorithm [12], which is a communication-efficient variant of the Online-Fed algorithm [18]. Afterward, we define the model-poisoning Byzantine attack within the context of FL and probe its impact on the model update equations of PSO-Fed.

A. System Model

We consider a network consisting of K clients and a central server. At every time instance t, each client k has access to a data vector $\mathbf{x}_{k,t} \in \mathbb{R}^D$ and its corresponding response value $y_{k,t} \in \mathbb{R}$, which are related via the model

$$y_{k,t} = \mathbf{w}^\intercal \mathbf{x}_{k,t} + \nu_{k,t}. \tag{1}$$

The model parameter vector $\mathbf{w} \in \mathbb{R}^D$ is collaboratively estimated using the locally stored client data, and the observation noise $\nu_{k,t}$ is modeled as a zero-mean white Gaussian noise, i.e., $\nu_{k,t} \sim \mathcal{N}(0, \sigma_{\nu_k}^2)$.

We define the global objective function for estimating $\mathbf{w}$ as

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}), \tag{2}$$

where the local objective function at client k is

$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E}\left[|y_{k,t} - \mathbf{w}^\intercal \mathbf{x}_{k,t}|^2\right], \tag{3}$$

with the expectation taken with respect to the randomness introduced by the observation noise. The goal is to find the optimal estimate of $\mathbf{w}$ by minimizing $\mathcal{J}(\mathbf{w})$, i.e., $\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{J}(\mathbf{w})$, albeit, in a distributed fashion via FL.

B. PSO-Fed

In PSO-Fed, to reduce the communication load, the server only sends a portion of the global model estimate to the clients. Similarly, the clients transmit parts of their local model estimates to the server. The model parameters exchanged between client k and the server at iteration t are specified using a diagonal selection matrix with M ones, which is denoted by

$\mathbf{S}_{k,t} \in \mathbb{R}^{D\times D}$. The positions of the ones on the diagonal determine which model parameters are shared with the server at each iteration. They can be selected arbitrarily or in a round-robin fashion as in [36], [37] such that the model parameters are exchanged between each client and the server, on average, M times in every D iterations. Therefore, the probability of any model parameter being shared with the server in any iteration is $p_e = M/D$.

Using the selection matrices $\mathbf{S}_{k,t}$, the recursive update equations of PSO-Fed, that iteratively minimize (2) through FL, are expressed as [11]

$$\epsilon_{k,t} = y_{k,t} - [\mathbf{S}_{k,t}\mathbf{w}_t + (\mathbf{I}_D - \mathbf{S}_{k,t})\,\mathbf{w}_{k,t}]^\intercal\,\mathbf{x}_{k,t} \tag{4a}$$

$$\mathbf{w}_{k,t+1} = \mathbf{S}_{k,t}\mathbf{w}_t + (\mathbf{I}_D - \mathbf{S}_{k,t})\,\mathbf{w}_{k,t} + \mu\mathbf{x}_{k,t}\epsilon_{k,t} \tag{4b}$$

$$\mathbf{w}_{t+1} = \frac{1}{|\mathcal{S}_t|}\sum_{k\in\mathcal{S}_t}\left[\mathbf{S}_{k,t+1}\mathbf{w}_{k,t+1} + (\mathbf{I}_D - \mathbf{S}_{k,t+1})\,\mathbf{w}_t\right], \tag{4c}$$

where $\mathbf{w}_{k,t}$ is the local model estimate at client k and iteration t, $\mathbf{w}_t$ is the global model estimate at iteration t, $\mathbf{I}_D$ is the $D\times D$ identity matrix, μ is the stepsize controlling the convergence rate and steady-state performance, $\mathcal{S}_t$ denotes the set of client selected at iteration t, and $|\mathcal{S}_t|$ denotes the number of the selected clients in each iteration.

C. Model-Poisoning Attacks

We denote the set of potential Byzantine clients within the network as $\mathcal{S}_B$. To represent the presence or absence of Byzantine behavior in a client, we use the indicator variable β_k, where $\beta_k = 1$ indicates that client k belongs to the set $\mathcal{S}_B$ (i.e., is a Byzantine client), and $\beta_k = 0$ otherwise. The total number of Byzantine clients in the network is denoted by $|\mathcal{S}_B|$. These Byzantine clients seek to poison the global model (i.e., undermine its accuracy) by sporadically sending corrupt local model estimates to the server. We assume that the Byzantine clients possess the accurate local model estimates. More specifically, we consider a scenario where, in each iteration, every Byzantine client deliberately corrupts its local model estimate by perturbing it, before sending it to the server. This corruption is conducted with a certain probability of attack, denoted by p_a. Consequently, we represent the model update shared by any Byzantine client as [38]

$$\mathbf{w}'_{k,t} = \begin{cases} \mathbf{w}_{k,t} + \boldsymbol{\delta}_{k,t} & \text{with probability } p_a \\ \mathbf{w}_{k,t} & \text{with probability } 1-p_a, \end{cases} \tag{5}$$

where $\boldsymbol{\delta}_{k,t} \in \mathbb{R}^D$ denotes the perturbation signal associated with the attack. Typically, this perturbation is zero-mean white Gaussian noise, characterized as $\boldsymbol{\delta}_{k,t} \sim \mathcal{N}(\mathbf{0}, \sigma_B^2\mathbf{I}_D)$ [38].

The probability of a Byzantine client corrupting its local estimate can be modeled using a Bernoulli random variable $\tau_{k,t}$. This variable takes the value of 1 with probability p_a indicating the occurrence of an attack, and 0 otherwise. Hence, given that each Byzantine client transmits a corrupted local model estimate to the server with a probability of p_a when selected by the server, in scenarios where Byzantine clients are present in the network, the global model update of PSO-Fed, (4c), can be rewritten as

$$\mathbf{w}_{t+1} = \frac{1}{|\mathcal{S}_t|}\sum_{k\in\mathcal{S}_t}\left[\mathbf{S}_{k,t+1}\mathbf{w}'_{k,t+1} + (\mathbf{I}_D - \mathbf{S}_{k,t+1})\,\mathbf{w}_t\right]$$

where

$$\mathbf{w}'_{k,t+1} = \mathbf{w}_{k,t+1} + \beta_k\tau_{k,t}\boldsymbol{\delta}_{k,t+1}. \tag{6}$$

III. Performance Analysis

In this section, we evaluate the robustness of PSO-Fed against Byzantine attacks by analyzing its theoretical mean and mean-square convergence and predicting its steady-state MSE and optimal stepsize.

To facilitate the analysis, we introduce some new quantities. We denote the extended optimal global model as $\mathbf{w}_e^\star = \mathbf{1}_{K+1}\otimes\mathbf{w}^\star$ and the extended global model estimate as $\mathbf{w}_{e,t} = \mathrm{col}\{\mathbf{w}_t, \mathbf{w}_{1,t}, \ldots, \mathbf{w}_{K,t}\}$. We also define the following collective quantities

$$\begin{aligned}
\mathbf{X}_t &= \mathrm{bdiag}\{\mathbf{0}, \mathbf{x}_{1,t}, \ldots, \mathbf{x}_{K,t}\} \\
\boldsymbol{\delta}_{e,t} &= \mathrm{col}\{\mathbf{0}, \boldsymbol{\delta}_{1,t}, \ldots, \boldsymbol{\delta}_{K,t}\} \\
\mathbf{B} &= \mathrm{bdiag}\{\mathbf{0}, \beta_1\mathbf{I}_D, \ldots, \beta_K\mathbf{I}_D\} \\
\mathbf{T}_t &= \mathrm{bdiag}\{\mathbf{0}, \tau_{1,t}\mathbf{I}_D, \ldots, \tau_{K,t}\mathbf{I}_D\} \\
\boldsymbol{\nu}_{e,t} &= \mathrm{col}\{0, \nu_{1,t}, \ldots, \nu_{K,t}\},
\end{aligned}$$

where the operators $\mathrm{col}\{\cdot\}$ and $\mathrm{bdiag}\{\cdot\}$ represent column-wise stacking and block diagonalization, respectively. Additionally, $\mathbf{1}_{K+1}$ denotes a column vector with $K+1$ entries, all set to one. Subsequently, we define

$$\mathbf{y}_{e,t} = \mathrm{col}\{0, y_{1,t}, y_{2,t}, \ldots, y_{K,t}\} = \mathbf{X}_t^\intercal\mathbf{w}_e^\star + \boldsymbol{\nu}_{e,t} \tag{7a}$$

$$\boldsymbol{\epsilon}_{e,t} = \mathrm{col}\{0, \epsilon_{1,t}, \epsilon_{2,t}, \ldots, \epsilon_{K,t}\} = \mathbf{y}_{e,t} - \mathbf{X}_t^\intercal\boldsymbol{\mathcal{A}}_t\mathbf{w}_{e,t} \tag{7b}$$

and

$$\boldsymbol{\mathcal{A}}_t = \begin{bmatrix} \mathbf{I}_D & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ a_{1,t}\mathbf{S}_{1,t} & \mathbf{I}_D - a_{1,t}\mathbf{S}_{1,t} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{K,t}\mathbf{S}_{K,t} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_D - a_{K,t}\mathbf{S}_{K,t} \end{bmatrix}, \tag{8}$$

where $a_{k,t} = 1$ if $k \in \mathcal{S}_t$ and $a_{k,t} = 0$ otherwise. Hence, the global recursion equations of PSO-Fed can be expressed as

$$\mathbf{w}_{e,t+1} = \boldsymbol{\mathcal{B}}_{t+1}\left(\boldsymbol{\mathcal{A}}_t\mathbf{w}_{e,t} + \mu\mathbf{X}_t\boldsymbol{\epsilon}_{e,t}\right) + \boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1} \tag{9}$$

where

$$\boldsymbol{\mathcal{B}}_{t+1} = \begin{bmatrix} \mathbf{I}_D - \sum_{k=1}^K \frac{a_{k,t}}{|\mathcal{S}_t|}\mathbf{S}_{k,t+1} & \frac{a_{1,t}}{|\mathcal{S}_t|}\mathbf{S}_{1,t+1} & \ldots & \frac{a_{K,t}}{|\mathcal{S}_t|}\mathbf{S}_{K,t+1} \\ \mathbf{0} & \mathbf{I}_D & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_D \end{bmatrix} \tag{10}$$

and

$$\boldsymbol{\mathcal{C}}_{t+1} = \begin{bmatrix} \mathbf{0} & \frac{a_{1,t}}{|\mathcal{S}_t|}\mathbf{S}_{1,t+1} & \dots & \frac{a_{K,t}}{|\mathcal{S}_t|}\mathbf{S}_{K,t+1} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}. \tag{11}$$

To make the analysis tractable, we adopt the following commonly used assumptions.

A1: The input vectors $\mathbf{x}_{k,t}$ at each client k and time step (iteration) t are sampled from a wide-sense stationary (WSS) multivariate random process, characterized by the covariance matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal]$.

A2: The observation noise $\nu_{k,t}$ and perturbation signal $\boldsymbol{\delta}_{k,t}$ are IID across both clients and iterations. Additionally, $\nu_{k,t}$ and $\boldsymbol{\delta}_{k,t}$ are mutually independent and independent of all other stochastic variables, including $\mathbf{x}_{k,t}$ and $\mathbf{S}_{k,t}$.

A3: The selection matrices $\mathbf{S}_{k,t}$ are mutually independent across clients and iterations.

A. Mean Convergence

Let us define the deviation (coefficient-error) vector as $\widetilde{\mathbf{w}}_{e,t} = \mathbf{w}_e^\star - \mathbf{w}_{e,t}$. Since $\boldsymbol{\mathcal{B}}_{t+1}$ and $\boldsymbol{\mathcal{A}}_t$ are block right-stochastic, i.e., their block rows add up to identity matrix, we have $\boldsymbol{\mathcal{B}}_{t+1}\mathbf{w}_e^\star = \boldsymbol{\mathcal{A}}_t\mathbf{w}_e^\star = \mathbf{w}_e^\star$. Therefore, considering (9) and the definition of $\widetilde{\mathbf{w}}_{e,t}$, we get

$$\begin{aligned}\widetilde{\mathbf{w}}_{e,t+1} = \boldsymbol{\mathcal{B}}_{t+1}\big(\mathbf{I} - \mu\mathbf{X}_t\mathbf{X}_t^\intercal\big)\boldsymbol{\mathcal{A}}_t\widetilde{\mathbf{w}}_{e,t} - \mu\boldsymbol{\mathcal{B}}_{t+1}\mathbf{X}_t\boldsymbol{\nu}_{e,t} \\ - \boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}.\end{aligned} \tag{12}$$

Taking the expected value of both sides of (12) and under assumptions A1-A3, we obtain

$$\mathbb{E}[\widetilde{\mathbf{w}}_{e,t+1}] = \mathbb{E}[\boldsymbol{\mathcal{B}}_{t+1}]\big(\mathbf{I} - \mu\boldsymbol{\mathcal{R}}\big)\mathbb{E}[\boldsymbol{\mathcal{A}}_t]\mathbb{E}[\widetilde{\mathbf{w}}_{e,t}], \tag{13}$$

where

$$\boldsymbol{\mathcal{R}} = \mathrm{bdiag}\{\mathbf{0}, \mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_K\}.$$

The mean convergence of PSO-Fed is guaranteed if $\|\left(\mathbf{I} - \mu\boldsymbol{\mathcal{R}}\right)\| < 1$ or equivalently $\|1 - \mu\lambda_i\left(\mathbf{R}_k\right)\| < 1$ $\forall k, i$. Here, $\|\cdot\|$ is any matrix norm and $\lambda_i\left(\cdot\right)$ denotes the ith eigenvalue of its matrix argument. Consequently, PSO-Fed converges in the mean sense, even amidst the presence of Byzantine clients, as long as the stepsize μ is appropriately chosen to satisfy

$$0 < \mu < \frac{2}{\max_{i,k}\{\lambda_i\left(\mathbf{R}_k\right)\}}. \tag{14}$$

In other terms, PSO-Fed is unbiased, even under model-poisoning attacks.

B. Mean-Square Convergence

Let us denote the weighted norm-square of $\widetilde{\mathbf{w}}_{e,t}$ as $\|\widetilde{\mathbf{w}}_{e,t}\|_{\boldsymbol{\Sigma}}^2 = \widetilde{\mathbf{w}}_{e,t}^\intercal\boldsymbol{\Sigma}\widetilde{\mathbf{w}}_{e,t}$, where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix. Computing the weighted norm-square of both sides of (12) results in

$$\begin{aligned}\mathbb{E}[\|\widetilde{\mathbf{w}}_{e,t+1}\|_{\boldsymbol{\Sigma}}^2] = \mathbb{E}[\|\widetilde{\mathbf{w}}_{e,t}\|_{\boldsymbol{\Sigma}'}^2] + \mu^2\mathbb{E}[\boldsymbol{\nu}_{e,t}^\intercal\mathbf{Y}_t\boldsymbol{\nu}_{e,t}] \\ + \mathbb{E}[\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{U}_t\boldsymbol{\delta}_{e,t+1}],\end{aligned} \tag{15}$$

where the cross terms vanish under assumption A2 and

$$\begin{aligned}\boldsymbol{\Sigma}' &= \mathbb{E}[\boldsymbol{\mathcal{A}}_t^\intercal\big(\mathbf{I} - \mu\mathbf{X}_t\mathbf{X}_t^\intercal\big)\boldsymbol{\mathcal{B}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{t+1}\big(\mathbf{I} - \mu\mathbf{X}_t\mathbf{X}_t^\intercal\big)\boldsymbol{\mathcal{A}}_t] \\ \mathbf{Y}_t &= \mathbf{X}_t^\intercal\boldsymbol{\mathcal{B}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{t+1}\mathbf{X}_t \\ \mathbf{U}_t &= \mathbf{T}_t^\intercal\mathbf{B}^\intercal\boldsymbol{\mathcal{C}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t.\end{aligned} \tag{16}$$

Subsequently, we define

$$\boldsymbol{\sigma}' = \mathrm{bvec}\{\boldsymbol{\Sigma}'\} = \boldsymbol{\mathcal{F}}^\intercal\boldsymbol{\sigma} \tag{17}$$

$$\boldsymbol{\sigma} = \mathrm{bvec}\{\boldsymbol{\Sigma}\} \tag{18}$$

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} - \mu\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} + \mu^2\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} \tag{19}$$

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}} = \mathbb{E}[\boldsymbol{\mathcal{B}}_{t+1} \otimes_b \boldsymbol{\mathcal{B}}_{t+1}] \tag{20}$$

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}} = \mathbb{E}[\boldsymbol{\mathcal{A}}_t \otimes_b \boldsymbol{\mathcal{A}}_t] \tag{21}$$

$$\boldsymbol{\mathcal{K}} = (\mathbf{I} \otimes_b \boldsymbol{\mathcal{R}}) + (\boldsymbol{\mathcal{R}} \otimes_b \mathbf{I}) \tag{22}$$

$$\boldsymbol{\mathcal{H}} = \mathbb{E}[\mathbf{X}_t\mathbf{X}_t^\intercal \otimes_b \mathbf{X}_t\mathbf{X}_t^\intercal], \tag{23}$$

where $\otimes_b$ denotes the block Kronecker product and $\mathrm{bvec}\{\cdot\}$ the block vectorization operation [35]. We evaluate $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}$, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}$, and $\boldsymbol{\mathcal{H}}$, under A1-A3, in Appendixes A and B.

The third term on the right-hand side (RHS) of (15) quantifies the impact of model-poisoning attacks. We calculate this term as

$$\begin{aligned}&\mathbb{E}[\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{U}_t\boldsymbol{\delta}_{e,t+1}] \\ &= \mathbb{E}[\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal\boldsymbol{\mathcal{C}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}] \\ &= \mathbb{E}[\mathrm{tr}(\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal\boldsymbol{\mathcal{C}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1})] \\ &= \mathrm{tr}(\mathbb{E}[\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal\boldsymbol{\mathcal{C}}_{t+1}^\intercal\boldsymbol{\Sigma}\boldsymbol{\mathcal{C}}_{t+1}\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}]) \\ &= \mathrm{tr}\big(\mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1}\mathbb{E}[\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal]\boldsymbol{\mathcal{C}}_{t+1}^\intercal]\boldsymbol{\Sigma}\big),\end{aligned} \tag{24}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace. Under A2, we have

$$\begin{aligned}&\mathrm{tr}\big(\mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1}\mathbb{E}[\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal]\boldsymbol{\mathcal{C}}_{t+1}^\intercal]\boldsymbol{\Sigma}\big) \\ &\quad = \mathrm{tr}\big(\mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1}\boldsymbol{\Omega}_{\boldsymbol{\delta}}\boldsymbol{\mathcal{C}}_{t+1}^\intercal]\boldsymbol{\Sigma}\big),\end{aligned} \tag{25}$$

with

$$\begin{aligned}\boldsymbol{\Omega}_{\boldsymbol{\delta}} &= \mathbb{E}[\mathbf{B}\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t^\intercal\mathbf{B}^\intercal] \\ &= \mathbf{B}\mathbb{E}[\mathbf{T}_t\boldsymbol{\delta}_{e,t+1}\boldsymbol{\delta}_{e,t+1}^\intercal\mathbf{T}_t]\mathbf{B} \\ &= \mathrm{bdiag}\{\mathbf{0}, \beta_1\sigma_B^2 p_a\mathbf{I}_D, \beta_2\sigma_B^2 p_a\mathbf{I}_D, \cdots, \beta_K\sigma_B^2 p_a\mathbf{I}_D\}.\end{aligned} \tag{26}$$

Using the properties of block Kronecker product, we have $\mathrm{tr}\big(\mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1}\boldsymbol{\Omega}_{\boldsymbol{\delta}}\boldsymbol{\mathcal{C}}_{t+1}^\intercal]\boldsymbol{\Sigma}\big) = \boldsymbol{\omega}^\intercal\boldsymbol{\sigma}$, where

$$\begin{aligned}\boldsymbol{\omega} &= \mathrm{bvec}\{\mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1}\boldsymbol{\Omega}_{\boldsymbol{\delta}}\boldsymbol{\mathcal{C}}_{t+1}^\intercal]\} \\ &= \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}}\mathrm{bvec}\{\boldsymbol{\Omega}_{\boldsymbol{\delta}}\}\end{aligned} \tag{27}$$

and

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}} = \mathbb{E}[\boldsymbol{\mathcal{C}}_{t+1} \otimes_b \boldsymbol{\mathcal{C}}_{t+1}].$$

We evaluate $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}}$ in Appendix C. We also define

$$\begin{aligned}\boldsymbol{\phi} &= \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\boldsymbol{\phi}_{\boldsymbol{\nu}}, \\ \boldsymbol{\phi}_{\boldsymbol{\nu}} &= \mathrm{bvec}\{\mathbb{E}\left[\mathbf{X}_t\boldsymbol{\Theta}_{\boldsymbol{\nu}}\mathbf{X}_t^\intercal\right]\} \text{ (Appendix D)}, \\ \boldsymbol{\Theta}_{\boldsymbol{\nu}} &= \mathbb{E}\left[\boldsymbol{\nu}_{e,t}\boldsymbol{\nu}_{e,t}^\intercal\right] = \mathrm{diag}\{0, \sigma_{\nu_1}^2, \cdots, \sigma_{\nu_K}^2\},\end{aligned} \tag{28}$$

and $\mathrm{bvec}^{-1}\{\cdot\}$ as the reverse operation of block vectorization. Consequently, we can write the global recursion equation for

the weighted mean square deviation (MSD) of PSO-Fed under model-poisoning attacks as

$$\mathbb{E}\left[\|\widetilde{\mathbf{w}}_{e,t+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right] = \mathbb{E}\left[\|\widetilde{\mathbf{w}}_{e,t}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\mathcal{F}}^\intercal\boldsymbol{\sigma}\}}\right] + \mu^2\boldsymbol{\phi}^\intercal\boldsymbol{\sigma} + \boldsymbol{\omega}^\intercal\boldsymbol{\sigma}. \tag{29}$$

As $t \to \infty$, (29) converges provided that the spectral radius of $\boldsymbol{\mathcal{F}}$ is smaller than one, i.e., $\rho(\boldsymbol{\mathcal{F}}^\intercal) = \rho(\boldsymbol{\mathcal{F}}) < 1$. Using the properties of the block maximum-norm, denoted by $\|\cdot\|_{b,\infty}$, and knowing $\|\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\|_{b,\infty} = \|\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}\|_{b,\infty} = 1$, we have

$$\begin{aligned}\rho(\boldsymbol{\mathcal{F}}) &\leq \|\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}\left(\mathbf{I} - \mu\boldsymbol{\mathcal{K}} + \mu^2\boldsymbol{\mathcal{H}}\right)\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}\|_{b,\infty} \\ &\leq \|\mathbf{I} - \mu\boldsymbol{\mathcal{K}} + \mu^2\boldsymbol{\mathcal{H}}\|_{b,\infty}.\end{aligned} \tag{30}$$

Considering [39, Theorem 2] and defining

$$\boldsymbol{\mathcal{D}} = \begin{bmatrix}\boldsymbol{\mathcal{K}}/2 & -\boldsymbol{\mathcal{H}}/2 \\ \mathbf{I} & \mathbf{0}\end{bmatrix},$$

the spectral radius of $\boldsymbol{\mathcal{F}}$ is smaller than one when

$$0 < \mu < \min\left\{\frac{1}{\lambda_{\max}\left(\boldsymbol{\mathcal{K}}^{-1}\boldsymbol{\mathcal{H}}\right)}, \frac{1}{\max\{\lambda(\boldsymbol{\mathcal{D}}) \in \mathbb{R}, 0\}}\right\}, \tag{31}$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of its matrix argument. We denote the upper bound of (31) as $\mu_{\max}$. Thus, when (31) holds, PSO-Fed converges in the mean-square sense and has a bounded steady-state MSD, even when facing Byzantine attacks.

C. Mean Square Error

Utilizing the calculations of the three terms on the RHS of (15) and unfolding the iterations of the global recursion in (29), we obtain

$$\begin{aligned}\mathbb{E}\left[\|\widetilde{\mathbf{w}}_{e,t+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right] &= \mathbb{E}\left[\|\widetilde{\mathbf{w}}_{e,0}\|^2_{\mathrm{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^\intercal)^{t+1}\boldsymbol{\sigma}\}}\right] \\ &+ \mu^2\boldsymbol{\phi}^\intercal\sum_{j=0}^{t}\left(\boldsymbol{\mathcal{F}}^\intercal\right)^j\boldsymbol{\sigma} + \boldsymbol{\omega}^\intercal\sum_{j=0}^{t}\left(\boldsymbol{\mathcal{F}}^\intercal\right)^j\boldsymbol{\sigma}.\end{aligned} \tag{32}$$

Given an appropriate choice of μ as in (31), letting $t \to \infty$ on both sides of (32) yields

$$\lim_{t\to\infty}\mathbb{E}[\|\widetilde{\mathbf{w}}_{e,t}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}] = \left(\mu^2\boldsymbol{\phi}^\intercal + \boldsymbol{\omega}^\intercal\right)\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1}\boldsymbol{\sigma}. \tag{33}$$

To calculate the steady-state MSE of PSO-Fed (denoted by $\mathcal{E}$), while taking into account the impact of model-poisoning attacks, we set

$$\begin{aligned}\boldsymbol{\sigma} &= \mathrm{bvec}\{\mathbb{E}\left[\boldsymbol{\mathcal{A}}_t^\intercal\mathbf{X}_t\mathbf{X}_t^\intercal\boldsymbol{\mathcal{A}}_t\right]\} \\ &= \mathbb{E}\left[\mathrm{bvec}\{\boldsymbol{\mathcal{A}}_t^\intercal\mathbf{X}_t\mathbf{X}_t^\intercal\boldsymbol{\mathcal{A}}_t\}\right] \\ &= \mathbb{E}\left[\left(\boldsymbol{\mathcal{A}}_t \otimes_b \boldsymbol{\mathcal{A}}_t\right)^\intercal\mathrm{bvec}\{\mathbf{X}_t\mathbf{X}_t^\intercal\}\right] \\ &= \mathbb{E}\left[\left(\boldsymbol{\mathcal{A}}_t \otimes_b \boldsymbol{\mathcal{A}}_t\right)^\intercal\right]\mathrm{bvec}\{\mathbb{E}\left[\mathbf{X}_t\mathbf{X}_t^\intercal\right]\} \\ &= \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}.\end{aligned} \tag{34}$$

Therefore, considering (7), we have

$$\begin{aligned}\mathcal{E} &= \frac{1}{K}\lim_{t\to\infty}\mathbb{E}[\boldsymbol{\epsilon}_{e,t}^\intercal\boldsymbol{\epsilon}_{e,t}] \\ &= \frac{1}{K}\lim_{t\to\infty}\left(\mathbb{E}\left[\widetilde{\mathbf{w}}_{e,t}^\intercal\boldsymbol{\mathcal{A}}_t^\intercal\mathbf{X}_t\mathbf{X}_t^\intercal\boldsymbol{\mathcal{A}}_t\widetilde{\mathbf{w}}_{e,t}\right] + \mathbb{E}\left[\boldsymbol{\nu}_{e,t}^\intercal\boldsymbol{\nu}_{e,t}\right]\right) \\ &= \frac{1}{K}\left[\left(\mu^2\boldsymbol{\phi}^\intercal + \boldsymbol{\omega}^\intercal\right)\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\} + \mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)\right] \\ &= \underbrace{\frac{\mu^2}{K}\boldsymbol{\phi}^\intercal\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}_{\mathcal{E}_{\boldsymbol{\phi}}} \\ &+ \underbrace{\frac{1}{K}\boldsymbol{\omega}^\intercal\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^\intercal\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}_{\mathcal{E}_{\boldsymbol{\omega}}} + \underbrace{\frac{1}{K}\mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)}_{\mathcal{E}_{\boldsymbol{\Theta}}}.\end{aligned} \tag{35}$$

The partial sharing of model parameters, scheduling of clients, statistics of input data, and value of stepsize impact the first and second terms on the RHS of (35), $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{E}_{\boldsymbol{\omega}}$. The observation noise influences the first and third terms, $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{E}_{\boldsymbol{\Theta}}$, with the latter term being solely due to this noise. Moreover, the effect of model-poisoning attacks by Byzantine clients on the steady-state MSE of PSO-Fed is confined to the second term, $\mathcal{E}_{\boldsymbol{\omega}}$, which is induced by these attacks.

Remark 1: Considering (27) and the calculation of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{C}}}$ in Appendix C, both partial sharing and client scheduling have a diminishing effect on $\boldsymbol{\omega}$. The lower the probability of sharing each entry of the model parameter vector, $p_e = \frac{M}{D}$, the smaller the entries of $\boldsymbol{\omega}$ are. Therefore, (35) suggests that partial sharing can indeed endow online FL with enhanced resilience to model-poisoning attacks. However, partial sharing also affects $\mathcal{E}_{\boldsymbol{\phi}}$ and increases its value as shown in [11, Fig. 1(a)]. This makes it hard to straightforwardly determine the overall impact of partial sharing on the steady-state MSE of PSO-Fed in scenarios involving model-poisoning attacks, by only analyzing (35). To gain deeper insights and validate these theoretical predictions, we undertake comprehensive simulations in section IV.

D. Optimal Stepsize

In the absence of model-poisoning attacks where $\mathcal{E}_{\boldsymbol{\omega}} = 0$, $\mathcal{E}$ is a monotonically increasing function of the stepsize μ, given (31) is satisfied. However, under model-poisoning attacks, the existence of $\mathcal{E}_{\boldsymbol{\omega}}$ disrupts this monotonicity. It amplifies $\mathcal{E}$ for smaller values of μ and shifts the point of minimal $\mathcal{E}$ from $\mu = 0$ to a larger $\mu > 0$. Consequently, this leads to the emergence of a non-trivial optimal stepsize that results in the minimum $\mathcal{E}$ when facing model-poisoning attacks. We denote this optimal stepsize as $\mu^\star$.

Assuming (31) is satisfied hence $\rho(\boldsymbol{\mathcal{F}}^\intercal) < 1$, approximating $\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1}$ with its corresponding J-term truncated Neumann series [40, Eq. (186)] gives

$$\left(\mathbf{I} - \boldsymbol{\mathcal{F}}^\intercal\right)^{-1} \approx \sum_{j=0}^{J}\left(\boldsymbol{\mathcal{F}}^\intercal\right)^j. \tag{36}$$

We consider $J \geq 3$. Moreover, for simplicity, we rewrite (19) as

$$\boldsymbol{\mathcal{F}}^{\intercal} = \mathbf{A}_0 - \mu\mathbf{A}_1 + \mu^2\mathbf{A}_2,$$

by defining $\mathbf{A}_0 = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}^{\intercal}$, $\mathbf{A}_1 = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}^{\intercal}$, and $\mathbf{A}_2 = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}^{\intercal}$. Using (36) in (35), we have

$$\mathcal{E} \approx \frac{1}{K}\left[\left(\mu^2\boldsymbol{\phi}^{\intercal} + \boldsymbol{\omega}^{\intercal}\right)\left\{\sum_{j=0}^{J}\left(\boldsymbol{\mathcal{F}}^{\intercal}\right)^j\right\}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\} + \mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)\right].$$

Since a valid stepsize is typically much smaller than one, we can further approximate $\mathcal{E}$ by excluding terms beyond μ^2 as

$$\mathcal{E} \approx \frac{1}{K}\Big[\left[\boldsymbol{\omega}^{\intercal}\mathbf{B}_{0,J} - \mu\boldsymbol{\omega}^{\intercal}\mathbf{B}_{1,J} + \mu^2(\boldsymbol{\phi}^{\intercal}\mathbf{B}_{0,J} + \boldsymbol{\omega}^{\intercal}\mathbf{B}_{2,J})\right]\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\} + \mathrm{tr}\left(\boldsymbol{\Theta}_{\boldsymbol{\nu}}\right)\Big], \tag{37}$$

where

$$\mathbf{B}_{0,J} = \sum_{j=0}^{J}\mathbf{A}_0^j,\ \mathbf{B}_{1,J} = \sum_{j=1}^{J}\mathbf{A}_0^{j-1}\mathbf{A}_1\mathbf{B}_{0,J-j},$$

$$\mathbf{B}_{2,J} = \mathbf{B}_{1,J-1}\mathbf{A}_1 + \sum_{j=1}^{J}\mathbf{A}_0^{j-1}\mathbf{A}_2\mathbf{B}_{0,J-j} + \sum_{j=3}^{J}\mathbf{C}_{2,J},$$

$$\mathbf{C}_{2,J} = \mathbf{C}_{2,J-1}\mathbf{A}_0 + \left(\sum_{j=1}^{J}\mathbf{A}_0^{j-1}\mathbf{A}_1\mathbf{A}_0^{J-j}\right)\mathbf{A}_1,\ \mathbf{C}_{2,2} = \mathbf{A}_1^2.$$

Differentiating $\mathcal{E}$ in (37) with respect to μ, setting the derivative equal to zero, and solving for μ yields the approximate optimal stepsize as

$$\mu^{\star} \approx \frac{\boldsymbol{\omega}^{\intercal}\mathbf{B}_{1,J}\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}{2(\boldsymbol{\phi}^{\intercal}\mathbf{B}_{0,J} + \boldsymbol{\omega}^{\intercal}\mathbf{B}_{2,J})\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}^{\intercal}\mathrm{bvec}\{\boldsymbol{\mathcal{R}}\}}. \tag{38}$$

Note that, when no model-poisoning attack occurs, i.e., $\boldsymbol{\omega} = 0$, the numerator on the RHS of (38) becomes zero, leading to $\mu^{\star} = 0$.

IV. Simulation Results

To verify our theoretical findings, we conduct several numerical experiments. We also compare the performance of the PSO-Fed algorithm with that of the Online-Fed, SignSGD, CS-Fed, and QS-Fed algorithms within a federated network comprising $K = 50$ or 100 clients and a model parameter vector of size $D = 5$. Each client possesses non-IID data vectors $\mathbf{x}_{k,t}$ and their associated response values $y_{k,t}$, which are interconnected as per (1) with model parameter vector $\mathbf{w} = \frac{1}{\sqrt{D}}[1, \cdots, 1]^{\intercal} \in \mathbb{R}^D$. Each entry of $\mathbf{x}_{k,t}$ for each client k is drawn from a zero-mean Gaussian distribution with variance ς_k^2, where ς_k^2 itself is sampled from a uniform distribution between 0.2 and 1.2, denoted as $\mathcal{U}(0.2, 1.2)$. In addition, the observation noise $\nu_{k,t}$ is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$ drawn from $\mathcal{U}(0.005, 0.025)$. In the conducted experiments, only $M = 1$ entry of the model parameter vector is shared per iteration, unless specified differently.

We evaluate the performance of the considered algorithms on the server's side by employing a test dataset consisting of $N_t = 50$ instances $\{\check{\mathbf{X}}, \check{\mathbf{y}}\}$ and calculating the test MSE at the server as

$$\frac{1}{N_t}\|\check{\mathbf{y}} - \check{\mathbf{X}}^{\intercal}\mathbf{w}_t\|_2^2. \tag{39}$$

We also evaluate the performance of PSO-Fed on the client side using (4a) and calculating the network-wide average steady-state MSE as

$$\frac{1}{K}\sum_{k=1}^{K}\lim_{t\to\infty}\epsilon_{k,t}^2. \tag{40}$$

In our first experiment, we examine the effect of different numbers of Byzantine clients $|\mathcal{S}_B|$ on the steady-state test MSE for PSO-Fed, Online-Fed, SignSGD, CS-Fed, and QS-Fed. In this experiment, We have $K = 100$ clients. In every iteration t, the server randomly selects a set of $|\mathcal{S}_t| = 5$ clients to

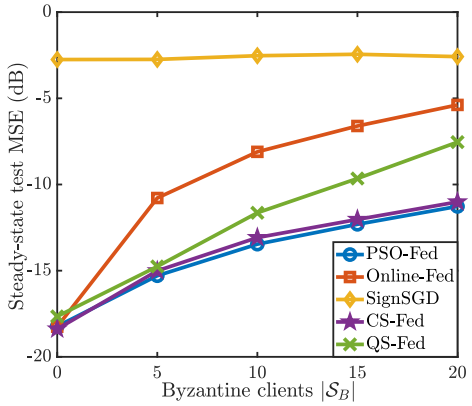

Fig. 1. Steady-state test MSE for different algorithms with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 1$.

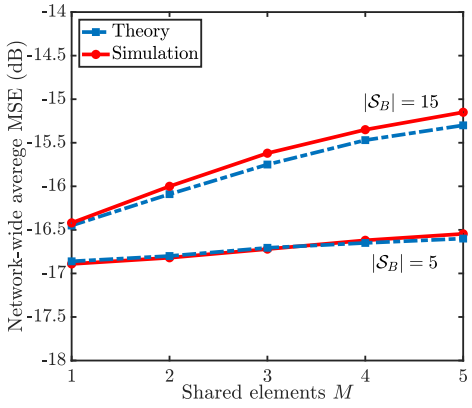

Fig. 2. Steady-state test MSE of PSO-Fed for different numbers of shared elements M with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

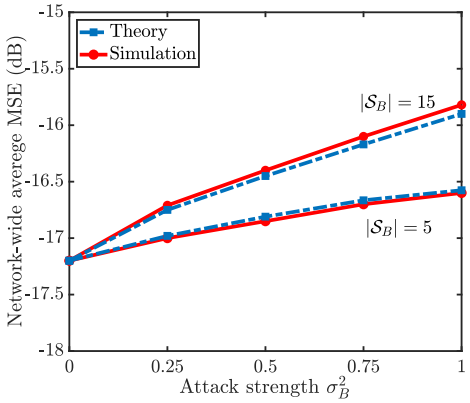

Fig. 3. Network-wide average MSE of PSO-Fed for different values of attack strengths σ_B^2 and Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack probability $p_a = 0.2$.

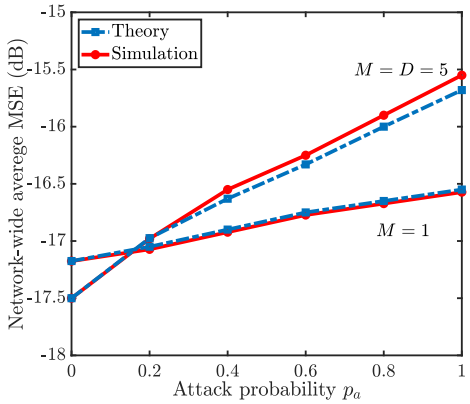

Fig. 5. Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of shared elements $M \in \{1, 5\}$, number of Byzantine clients $|\mathcal{S}_B| = 5$ and attack strength $\sigma_B^2 = 0.25$.

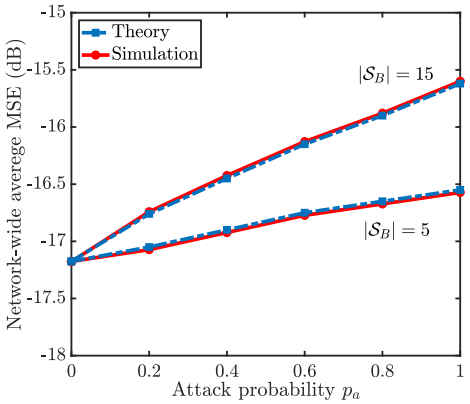

Fig. 4. Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack strength $\sigma_B^2 = 0.25$.

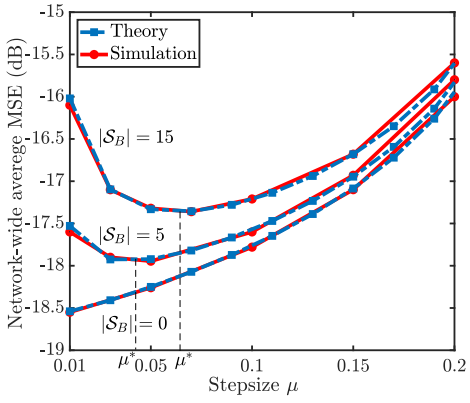

Fig. 6. Network-wide average MSE of PSO-Fed for different values of stepsize μ, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 0.25$.

participate in FL, with each client having an equal probability of being selected. In addition, all Byzantine clients are characterized by model-poisoning noise variance of $\sigma_B^2 = 0.25$ and attack probability of $p_a = 1$. In each iteration, we allocate the same communication budget to all algorithms. Therefore, to allow all algorithms converge to their lowest steady-state MSD within 3000 iterations, we set $\mu = 0.15$ for PSO-Fed, Online-Fed, CS-Fed and QS-Fed, and $\mu = 0.08$ for SignSGD at all clients. According to (31), the maximum stepsize ensuring the mean-square convergence of PSO-Fed in this experiment is $\mu_{max} = 0.245$. The results displayed in Fig. 1 indicate that PSO-Fed outperforms the existing algorithms across all considered number of Byzantine clients. Notably, Online-Fed and PSO-Fed perform similarly when there is no Byzantine client or poisoning attack (i.e., $\sigma_B^2 = 0$). However, the performance of Online-Fed deteriorates more than PSO-Fed as the number of Byzantine clients increases. While CS-Fed performs closely to PSO-Fed in our experiment, it is important to note that the computational complexity of CS-Fed exceeds that of PSO-Fed.

In our second experiment, we examine the impact of the number of shared entries, M, on the steady-state test MSE of PSO-Fed. We simulate PSO-Fed with $K = 50$ clients, $D = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$, and attack probability $p_a = 0.2$. The results are shown in Fig. 3. We observe that increasing M leads to higher steady-state test MSE. In addition, PSO-Fed exhibits greater resilience to model-poisoning attacks compared to Online-Fed, without incurring any extra computational or communication burden on the clients. This experiment corroborates our theoretical findings discussed in section III-C.

In our third experiment, we investigate the effect of varying

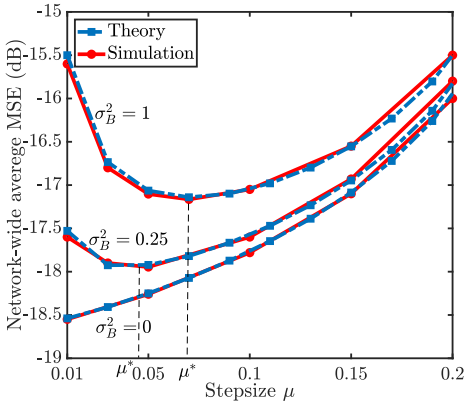

Fig. 7. Network-wide average MSE of PSO-Fed for different values of stepsize μ, numbers of Byzantine clients $|\mathcal{S}_B| = 5$ and attack probability $p_a = 0.25$.

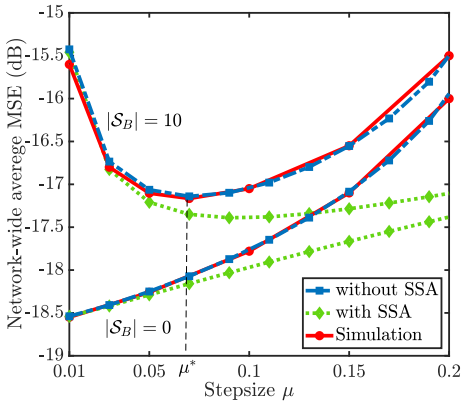

Fig. 8. Effect of small stepsize approximation (SSA) on network-wide average MSE of PSO-Fed, numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.25$.

the attack strength σ_B^2 on the network-wide average steady state MSE. We simulate PSO-Fed with $K = 50$ clients, $D = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 \in \{0, 0.25, 0.5, 0.75, 1\}$, and attack probability $p_a = 0.2$. The server randomly selects 5 clients in each iteration. The results depicted in Fig.2 align closely with our theoretical predictions. We notice an upward trend in the network-wide average steady-state MSE as the attack strength or the number of Byzantine clients increases. In addition, the presence of $|\mathcal{S}_B| = \alpha_1$ Byzantine clients with an attack strength of $\sigma_B^2 = \beta_1$ results in the same MSE as having $|\mathcal{S}_B| = \alpha_2$ Byzantine clients with an attack strength $\sigma_B^2 = \beta_2$, provided that the condition $\alpha_1\beta_1 = \alpha_2\beta_2$ is met.

In our fourth experiment, we explore how the attack probability p_a influences the steady-state test MSE of PSO-Fed, considering different numbers of shared entries $M \in \{1, 5\}$,

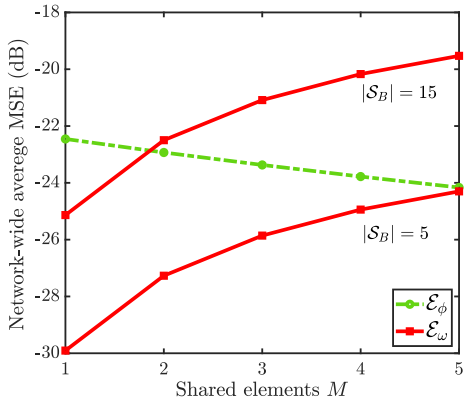

Fig. 9. Effect of number of shared elements M on $\mathcal{E}_\omega$ and $\mathcal{E}_\phi$ in (35) for $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

with the number of Byzantine clients set to $|\mathcal{S}_B| = 5$ and attack strength to $\sigma_B^2 = 0.25$. The results, presented in Fig. 5, align with observations from the previous experiment, showing that an increase in the attack probability leads to a higher MSE. Moreover, partial sharing (i.e., when $M < D$) enhances the resilience against model-poisoning attacks as elaborated upon in section III-C.

In our fifth experiment, we investigate the impact of attack probability p_a on the steady-state test MSE of PSO-Fed, considering different numbers of Byzantine clients $|\mathcal{S}_B| \in \{5, 15\}$ and setting the attack strength as $\sigma_B^2 = 0.25$. The results shown in Fig. 4 indicate that increasing the attack probability increases the steady-state MSE.

In our sixth experiment, we evaluate the performance of PSO-Fed using different stepsize values and Byzantine client numbers in the presence or absence of model-poisoning attacks. We simulate PSO-Fed with $K = 50$ clients, $D = 5$, attack strength $\sigma_B^2 = 0.25$, attack probability $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results illustrated in Fig. 6, show that with no model-poisoning attack ($|\mathcal{S}_B| = 0$), the larger the stepsize μ, the worse PSO-Fed performs, a trend consistent with gradient descent optimization methods. However, with model-poisoning attacks ($|\mathcal{S}_B| > 0$), the performance first improves with an increase in μ, then deteriorates when μ grows larger. This observation confirms the existence of an optimal stepsize $\mu^\star$ that ensures the best performance of PSO-Fed under model-poisoning attacks. By calculating the optimal stepsize using (38), we determine $\mu^\star \approx 0.03$, which corresponds to the experimental results.

In our seventh experiment, we analyze the performance of PSO-Fed using different stepsizes in the presence or absence of model-poisoning attacks, considering different attack strengths σ_B^2. We simulate PSO-Fed with $K = 50$ clients, $D = 5$, $|\mathcal{S}_B| = 5$ Byzantine clients, and attack probability of $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results,

presented in Fig. 7, echo the observations from the previous experiment: without a poisoning attack ($\sigma_B^2 = 0$), PSO-Fed's performance decreases as the stepsize μ increases. Conversely, with model-poisoning attacks ($\sigma_B^2 > 0$), performance initially improves with an increase in μ but begins to decline as μ is further increased. This pattern confirms the existence of an optimal stepsize $\mu^\star$ for scenarios with model-poisoning attacks. Using (38) to estimate the optimal stepsize, we find $\mu^\star \approx 0.03$, corresponding to the experiment's findings. This experiment also reinforces our conclusion following the third experiment that identical attack properties result in the same MSE.

In our eighth experiment, we investigate the impact of the commonly-adopted small stepsize approximation (SSA) on predicting the performance of PSO-Fed in the presence or absence of model-poisoning attacks. Specifically, we assume that higher orders of μ are negligible when computing $\boldsymbol{\mathcal{F}}$ via (19). We then compare the simulated network-wide average MSE of PSO-Fed with the corresponding theoretical predictions with or without SSA. We simulate PSO-Fed with $K = 50$ clients, $D = 5$, attack probability $p_a = 0.25$, attack strength $\sigma_B^2 = 0.5$, and numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$. The server randomly selects 5 clients in each iteration. The results in Fig. 8 reveal that, when μ is sufficiently small, SSA does not hinder the accurate prediction of steady-state MSE even without the knowledge of the 4th moment of the input vector, $\boldsymbol{\mathcal{H}}$. However, as the stepsize increases, discrepancies arise in both scenarios, with and without model-poisoning attacks, diverging from the ideal case.

In our final experiment, we investigate the effect of partial sharing on different terms of MSE (35), namely, $\mathcal{E}_{\boldsymbol{\omega}}$ and $\mathcal{E}_{\boldsymbol{\phi}}$, in the presence of model-poisoning attacks. We compute both terms using the same setup as in our second experiment. The results depicted in Fig. 9 show that decreasing the number of shared parameters considerably lowers $\mathcal{E}_{\boldsymbol{\omega}}$, significantly mitigating the impact of model-poisoning attacks. However, sharing fewer entries increases $\mathcal{E}_{\boldsymbol{\phi}}$. Still, the benefits of partial sharing in reducing $\mathcal{E}_{\boldsymbol{\omega}}$ are more substantial, leading to reduced MSE. Note that $\mathcal{E}_{\boldsymbol{\phi}}$ remains unchanged across both considered cases, as it depends solely on scheduling and data noise. Finally, this experiment corroborates our theoretical findings and confirms our assertions regarding the effect of partial sharing on $\mathcal{E}_{\boldsymbol{\omega}}$ and $\mathcal{E}_{\boldsymbol{\phi}}$, as discussed at the end of section III-C.

V. Discussion

The PSO-Fed algorithm enhances the communication efficiency of online FL while maintaining robustness against model-poisoning attacks without placing any additional computational demand on clients. By sharing only subsets of model parameters, PSO-Fed inherently preserves model integrity without supplementary defense mechanisms, positioning it advantageously for resource-aware and security-sensitive FL applications, such as healthcare.

Our theoretical analysis, featuring a novel intermittent attack model formalized in (6), yields a key insight: the errors induced by Byzantine attacks and those arising from observational noise exhibit a counteracting relationship as a function of the stepsize, as explicitly demonstrated in (35). Exploiting this relationship, we derive a closed-form expression for the optimal stepsize, eliminating the need for extensive numerical parameter tuning. Furthermore, our analysis is based on the assumptions that the input data is wide-sense stationary and that the noise is independent. However, these assumptions may not accurately reflect the realities of FL applications, where input distributions may exhibit correlations and noise can be dependent among clients or over time. Such discrepancies may compromise the theoretical guarantees presented in this study. Future research endeavors can focus on relaxing these assumptions to better represent realistic FL scenarios and on investigating alternative techniques, such as random Fourier features (RFF) [41], to achieve enhanced generalization beyond conventional linear regression models.

VI. Conclusion

We conducted a theoretical analysis of the recently proposed PSO-Fed algorithm to examine its resilience to model-poisoning (Byzantine) attacks imparted by partial sharing. In our analysis, we considered a linear regression task with the local objective function of each client defined as an empirical risk. We showed that, even under Byzantine attacks, PSO-Fed converges in both mean and mean-square senses, given an appropriate choice of stepsize. Notably, we showed that, in the presence of Byzantine clients, the steady-state MSE of PSO-Fed is significantly smaller than that of the Online-Fed algorithm, which does not feature partial sharing. Our theoretical analysis also uncovered the existence of a non-trivial optimal stepsize for PSO-Fed under model-poisoning attacks. Simulation results corroborated our theoretical findings, highlighting PSO-Fed's effectiveness against Byzantine attacks and validating the accuracy of the theoretically predicted values of its steady-state MSE and optimal stepsize.

Appendix A
Evaluation of matrices $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}$ and $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}$

Let us define

$$\boldsymbol{\mathcal{A}}_t = \begin{bmatrix} \mathbf{A}_{1,1,t} & \mathbf{A}_{1,2,t} & \cdots & \mathbf{A}_{1,K+1,t} \\ \mathbf{A}_{2,1,t} & \mathbf{A}_{2,2,t} & \cdots & \mathbf{A}_{2,K+1,t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{K+1,1,t} & \mathbf{A}_{K+1,2,t} & \cdots & \mathbf{A}_{K+1,K+1,t} \end{bmatrix},$$

where

$$\mathbf{A}_{i,j,t} = \begin{cases} \mathbf{I}_D, & \text{if } i,j = 1 \\ a_{i,t}\mathbf{S}_{i,t}, & \text{if } i = 2,\ldots,K+1, j = 1 \\ \mathbf{I}_D - a_{i,t}\mathbf{S}_{i,t}, & \text{if } (i=j) \neq 1 \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Thus, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{A}}}$ is given by

$$\mathbb{E}\left[\boldsymbol{\mathcal{A}}_t \otimes_b \boldsymbol{\mathcal{A}}_t\right] = \mathbb{E}\begin{bmatrix} \mathbf{A}_{1,1,t} \otimes_b \boldsymbol{\mathcal{A}}_t & \cdots & \mathbf{A}_{1,K+1,t} \otimes_b \boldsymbol{\mathcal{A}}_t \\ \mathbf{A}_{2,1,t} \otimes_b \boldsymbol{\mathcal{A}}_t & \cdots & \mathbf{A}_{2,K+1,t} \otimes_b \boldsymbol{\mathcal{A}}_t \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{K+1,1,t} \otimes_b \boldsymbol{\mathcal{A}}_t & \cdots & \mathbf{A}_{K+1,K+1,t} \otimes_b \boldsymbol{\mathcal{A}}_t \end{bmatrix}$$

where

$$\mathbb{E}\left[\mathbf{A}_{i,j,t} \otimes_b \boldsymbol{\mathcal{A}}_t\right] = \mathbb{E}\begin{bmatrix} \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{1,1,t} & \cdots & \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{1,K+1,t} \\ \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{2,1,t} & \cdots & \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{2,K+1,t} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{K+1,1,t} & \cdots & \mathbf{A}_{i,j,t} \otimes \mathbf{A}_{K+1,K+1,t} \end{bmatrix}.$$

Recall that the probability of any model parameter being shared with the server in any iteration is $p_e = \frac{M}{D}$. In addition, the probability of any client being selected by the server in any iteration is $p_c = \frac{|\mathcal{S}_t|}{K}$. Therefore, we have

$$\mathbb{E}\left[a_{i,t}a_{i',t}\right] = \begin{cases} p_c & \text{if } i = i' \\ p_c\left(\frac{|\mathcal{S}_t|-1}{K-1}\right) & \text{if } i \neq i' \end{cases}$$

and $\mathbb{E}\left[\mathbf{S}_{i,t} \otimes \mathbf{S}_{i,t}\right]$ is a diagonal matrix with the zth diagonal entry being

$$\begin{cases} p_e & \text{if } z = (n-1)D + n,\ n = 1, \cdots, D \\ p_e\left(\frac{M-1}{D-1}\right) & \text{if } z \neq (n-1)D + n,\ n = 1, \cdots, D. \end{cases}$$

Subsequently, the zth diagonal entry of the diagonal matrix $\boldsymbol{\mathcal{S}} = \mathbb{E}\left[a_{i,t}\mathbf{S}_{i,t} \otimes a_{i',t}\mathbf{S}_{i,t}\right] \in \mathbb{R}^{D^2 \times D^2}$ is

$$\begin{cases} p_c p_e & \text{if } i = i' \text{ and } \big(z = (n-1)D + n, \\ & \quad n = 1, \cdots, D\big) \\ p_c p_e\left(\frac{M-1}{D-1}\right) & \text{if } i = i' \text{ and } \big(z \neq (n-1)D + n, \\ & \quad n = 1, \cdots, D\big) \\ p_c p_e\left(\frac{|\mathcal{S}_t|-1}{K-1}\right) & \text{if } i \neq i' \text{ and } \big(z = (n-1)D + n, \\ & \quad n = 1, \cdots, D\big) \\ p_c p_e\left(\frac{M-1}{D-1}\right)\left(\frac{|\mathcal{S}_t|-1}{K-1}\right) & \text{if } i \neq i' \text{ and } \big(z \neq (n-1)D + n, \\ & \quad n = 1, \cdots, D\big). \end{cases}$$

Therefore, we have

$$\mathbb{E}\left[\mathbf{A}_{i,j,t} \otimes \mathbf{A}_{l,m,t}\right] = \begin{cases} \mathbf{I}_{D^2} & \text{if } i,j = 1 \text{ and } l,m = 1 \\ p_e p_c \mathbf{I}_{D^2} & \text{if } i,j = 1 \text{ and } l \geq 2, m = 1 \\ (1 - p_e p_c)\mathbf{I}_{D^2} & \text{if } i,j = 1 \text{ and } (l = m) \neq 1 \\ p_e p_c \mathbf{I}_{D^2} & \text{if } i \geq 2, j = 1 \text{ and } l,m = 1 \\ (1 - p_e p_c)\mathbf{I}_{D^2} & \text{if } (i = j) \neq 1 \text{ and } l,m = 1 \\ \boldsymbol{\mathcal{S}} & \text{if } (i = l) \geq 2 \text{ and } (j = m) = 1 \\ p_e p_c \mathbf{I}_{D^2} - \boldsymbol{\mathcal{S}} & \text{if } i \geq 2, j = 1 \text{ and } (l = m) \neq 1 \\ p_e p_c \mathbf{I}_{D^2} - \boldsymbol{\mathcal{S}} & \text{if } (i = j) \neq 1 \text{ and } l \geq 2, m = 1 \\ (1 - 2p_e p_c)\mathbf{I}_{D^2} + \boldsymbol{\mathcal{S}} & \text{if } (i = j, l = m) \neq 1 \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

We similarly have

$$\boldsymbol{\mathcal{B}}_t = \begin{bmatrix} \mathbf{B}_{1,1,t} & \mathbf{B}_{1,2,t} & \cdots & \mathbf{B}_{1,K+1,t} \\ \mathbf{B}_{2,1,t} & \mathbf{B}_{2,2,t} & \cdots & \mathbf{B}_{2,K+1,t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{K+1,1,t} & \mathbf{B}_{K+1,2,t} & \cdots & \mathbf{B}_{K+1,K+1,t} \end{bmatrix},$$

where

$$\mathbf{B}_{i,j,t} = \begin{cases} \mathbf{I}_D - \sum_{k=1}^{K} \frac{a_{k,t}}{|\mathcal{S}_t|}\mathbf{S}_{k,t}, & \text{if } i,j = 1 \\ \frac{a_{j,t}}{|\mathcal{S}_t|}\mathbf{S}_{j,t}, & \text{if } i = 1, j = 2, \ldots, K+1 \\ \mathbf{I}_D, & \text{if } (i = j) \neq 1 \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Hence, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{B}}}$ is calculated as

$$\mathbb{E}\left[\boldsymbol{\mathcal{B}}_t \otimes_b \boldsymbol{\mathcal{B}}_t\right] = \mathbb{E}\begin{bmatrix} \mathbf{B}_{1,1,t} \otimes_b \boldsymbol{\mathcal{B}}_t & \cdots & \mathbf{B}_{1,K+1,t} \otimes_b \boldsymbol{\mathcal{B}}_t \\ \mathbf{B}_{2,1,t} \otimes_b \boldsymbol{\mathcal{B}}_t & \cdots & \mathbf{B}_{2,K+1,t} \otimes_b \boldsymbol{\mathcal{B}}_t \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{K+1,1,t} \otimes_b \boldsymbol{\mathcal{B}}_t & \cdots & \mathbf{B}_{K+1,K+1,t} \otimes_b \boldsymbol{\mathcal{B}}_t \end{bmatrix},$$

where

$$\mathbb{E}\left[\mathbf{B}_{i,j,t} \otimes_b \boldsymbol{\mathcal{B}}_t\right] = \mathbb{E}\begin{bmatrix} \mathbf{B}_{i,j,t} \otimes \mathbf{B}_{1,1,t} & \cdots & \mathbf{B}_{i,j,t} \otimes \mathbf{B}_{1,K+1,t} \\ \mathbf{B}_{i,j,t} \otimes \mathbf{B}_{2,1,t} & \cdots & \mathbf{B}_{i,j,t} \otimes \mathbf{B}_{2,K+1,t} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{i,j,t} \otimes \mathbf{B}_{K+1,1,t} & \cdots & \mathbf{B}_{i,j,t} \otimes_b \mathbf{B}_{K+1,K+1,t} \end{bmatrix}$$

and

$$\mathbb{E}\left[\mathbf{B}_{i,j,t} \otimes \mathbf{B}_{l,m,t}\right] = \begin{cases} (1 - \frac{2K}{|\mathcal{S}_t|}p_e p_c)\mathbf{I}_{D^2} + \frac{K}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_1 + \frac{K(K-1)}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_2 & \text{if } i,j = 1 \text{ and } l,m = 1 \\ \frac{p_e}{K}\mathbf{I}_{D^2} - \frac{1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_1 - \frac{K-1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_2 & \text{if } i,j = 1 \text{ and } l = 1, m \geq 2 \\ \frac{p_e}{K}\mathbf{I}_{D^2} - \frac{1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_1 - \frac{K-1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}}_2 & \text{if } i = 1, j \geq 2 \text{ and } l,m = 1 \\ (1 - p_e)\mathbf{I}_{D^2} & \text{if } i,j = 1 \text{ and } (l = m) \neq 1 \\ (1 - p_e)\mathbf{I}_{D^2} & \text{if } (i = j) \neq 1 \text{ and } l,m = 1 \\ \frac{1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}} & \text{if } i = 1, j \geq 2 \text{ and } l = 1, m \geq 2 \\ \frac{p_e}{K}\mathbf{I}_{D^2} & \text{if } i = 1, j \geq 2 \text{ and } (l = m) \neq 1 \\ \frac{p_e}{K}\mathbf{I}_{D^2} & \text{if } (i = j) \neq 1 \text{ and } l = 1, m \geq 2 \\ \mathbf{I}_{D^2} & \text{if } (i = j) \neq 1 \text{ and } (l = m) \neq 1 \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Here, $\boldsymbol{\mathcal{S}}_1$ and $\boldsymbol{\mathcal{S}}_2$ are diagonal matrices with their zth diagonal entry being

$$\begin{cases} p_c p_e & \text{if } z = (n-1)D + n,\ n = 1, \cdots, D \\ p_c p_e\left(\frac{M-1}{D-1}\right) & \text{if } z \neq (n-1)D + n,\ n = 1, \cdots, D, \end{cases}$$

and

$$\begin{cases} p_c p_e \left(\frac{|\mathcal{S}_t|-1}{K-1}\right) & \text{if } z=(n-1)D+n, \\ & n=1,\cdots,D) \\ p_c p_e \left(\frac{M-1}{D-1}\right)\left(\frac{|\mathcal{S}_t|-1}{K-1}\right) & \text{if } z\neq(n-1)D+n, \\ & n=1,\cdots,D), \end{cases}$$

respectively.

APPENDIX B
EVALUATION OF MATRIX $\mathcal{H}$

We have

$$\boldsymbol{\mathcal{H}} = \mathbb{E}[\mathbf{X}_t\mathbf{X}_t^\intercal \otimes_b \mathbf{X}_t\mathbf{X}_t^\intercal] = \mathbb{E}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_{1,t}\mathbf{x}_{1,t}^\intercal \otimes_b \mathbf{X}_t\mathbf{X}_t^\intercal & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{x}_{K,t}\mathbf{x}_{K,t}^\intercal \otimes_b \mathbf{X}_t\mathbf{X}_t^\intercal \end{bmatrix}, \tag{41}$$

with

$$\mathbb{E}[\mathbf{x}_{1,t}\mathbf{x}_{1,t}^\intercal \otimes_b \mathbf{X}_t\mathbf{X}_t^\intercal] = \mathbb{E}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_{1,t}\mathbf{x}_{1,t}^\intercal \otimes \mathbf{x}_{1,t}\mathbf{x}_{1,t}^\intercal & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{x}_{1,t}\mathbf{x}_{1,t}^\intercal \otimes \mathbf{x}_{K,t}\mathbf{x}_{K,t}^\intercal \end{bmatrix}. \tag{42}$$

Given that the input vectors $\mathbf{x}_{k,t}$ of each client arise from a wide-sense stationary multivariate random process with the covariance matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal]$, we have

$$\mathbb{E}[\mathbf{x}_{i,t}\mathbf{x}_{i,t}^\intercal \otimes \mathbf{x}_{j,t}\mathbf{x}_{j,t}^\intercal] = \begin{cases} \mathbb{E}[\mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal \otimes \mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal] & \text{if } i=j=k \\ \mathbf{R}_i \otimes \mathbf{R}_j & \text{if } i\neq j. \end{cases} \tag{43}$$

Each entry of $\mathbb{E}[\mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal \otimes \mathbf{x}_{k,t}\mathbf{x}_{k,t}^\intercal] \in \mathbb{R}^{D\times D}$ can be computed by utilizing the Isserlis' theorem [42] as

$$\begin{cases} \mathbb{E}[x_{k,l,t}^4] = 3\sigma_{ll}^4 & \forall k,l,t \\ \mathbb{E}[x_{k,l,t}^2 x_{k,m,t}^2] = \sigma_{ll}^2\sigma_{mm}^2 + 2\sigma_{lm}^2 & \text{if } l\neq m \\ \mathbb{E}[x_{k,l,t}^3 x_{k,m,t}] = 3\sigma_{ll}^2\sigma_{lm}^2 & \text{if } l\neq m \\ \mathbb{E}[x_{k,l,t}^2 x_{k,m,t}x_{k,q,t}] = \sigma_{ll}^2\sigma_{mq}^2 + 2\sigma_{lq}^2\sigma_{lm}^2 & \text{if } l\neq m\neq q. \end{cases} \tag{44}$$

Considering that entries of $\mathbf{x}_{i,t}$ are independent of each other, i.e., $\sigma_{lm}=0$, and assuming an identical variance among all entries, i.e., $\sigma_{ll}=\sigma_k$, we can simplify (44) as

$$\begin{cases} \mathbb{E}[x_{k,l,t}^4] = 3\sigma_k^4 & \forall k,l,t \\ \mathbb{E}[x_{k,l,t}^2 x_{k,m,t}^2] = \sigma_k^4 & \text{if } l\neq m \\ \mathbb{E}[x_{k,l,t}^3 x_{k,m,t}] = 0 & \text{if } l\neq m \\ \mathbb{E}[x_{k,l,t}^2 x_{k,m,t}x_{k,q,t}] = 0 & \text{if } l\neq m\neq q. \end{cases} \tag{45}$$

APPENDIX C
EVALUATION OF MATRIX $\mathcal{Q}_{\mathcal{C}}$

Similar to $\mathcal{Q}_{\mathcal{B}}$, we can write $\mathcal{Q}_{\mathcal{C}}$ as

$$\mathbb{E}[\mathcal{C}_t \otimes_b \mathcal{C}_t] = \mathbb{E}\begin{bmatrix} \mathbf{C}_{1,1,t}\otimes_b \mathcal{C}_t & \dots & \mathbf{C}_{1,K+1,t}\otimes_b \mathcal{C}_t \\ \mathbf{C}_{2,1,t}\otimes_b \mathcal{C}_t & \dots & \mathbf{C}_{2,K+1,t}\otimes_b \mathcal{C}_t \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{K+1,1,t}\otimes_b \mathcal{C}_t & \dots & \mathbf{C}_{K+1,K+1,t}\otimes_b \mathcal{C}_t \end{bmatrix},$$

where

$$\mathbf{C}_{i,j,t} = \begin{cases} \frac{a_{j,t}}{|\mathcal{S}_t|}\mathbf{S}_{j,t}, & \text{if } i=1, j=2,\dots,K+1 \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

and

$$\mathbb{E}[\mathbf{C}_{i,j,t}\otimes_b \mathcal{C}_t] = \mathbb{E}\begin{bmatrix} \mathbf{C}_{i,j,t}\otimes \mathbf{C}_{1,1,t} & \dots & \mathbf{C}_{i,j,t}\otimes \mathbf{C}_{1,K+1,t} \\ \mathbf{C}_{i,j,t}\otimes \mathbf{C}_{2,1,t} & \dots & \mathbf{C}_{i,j,t}\otimes \mathbf{C}_{2,K+1,t} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{i,j,t}\otimes \mathbf{C}_{K+1,1,t} & \dots & \mathbf{C}_{i,j,t}\otimes_b \mathbf{C}_{K+1,K+1,t} \end{bmatrix}.$$

We then have

$$\mathbb{E}[\mathbf{C}_{i,j,t}\otimes \mathbf{C}_{l,m,t}] = \begin{cases} \frac{1}{|\mathcal{S}_t|^2}\boldsymbol{\mathcal{S}} & \text{if } i=1, j\geq 2 \text{ and } l=1, m\geq 2 \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

with $\boldsymbol{\mathcal{S}}$ as calculated above.

APPENDIX D
EVALUATION OF VECTOR $\boldsymbol{\phi_\nu}$

Recall $\boldsymbol{\phi_\nu} = \text{bvec}\{\mathbb{E}[\mathbf{X}_t\boldsymbol{\Theta}_\nu\mathbf{X}_t^\intercal]\}$. Since $\mathbf{X}_t$ is not a square matrix, we cannot use the block vectorization properties. Therefore, to facilitate the calculation of $\boldsymbol{\phi_\nu}$, we introduce the modified versions of $\mathbf{X}_t$ and $\boldsymbol{\Theta_\nu}$ as $\hat{\mathbf{X}}_t = \text{bdiag}\{\mathbf{0}, \text{diag}\{\mathbf{x}_{1,t}\},\dots,\text{diag}\{\mathbf{x}_{K,t}\}\}$ and $\hat{\boldsymbol{\Theta}}_{\boldsymbol{\nu}} = \boldsymbol{\Theta_\nu}\otimes \mathbf{I}_D$. Therefore, we have

$$\boldsymbol{\phi_\nu} = \mathbb{E}[\hat{\mathbf{X}}_t \otimes_b \hat{\mathbf{X}}_t]\text{bvec}\{\hat{\boldsymbol{\Theta}}_{\boldsymbol{\nu}}\}, \tag{46}$$

where $\mathbb{E}[\hat{\mathbf{X}}_t \otimes_b \hat{\mathbf{X}}_t]$ can be written as

$$\mathbb{E}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_{1,t}\otimes_b \hat{\mathbf{X}}_t & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{x}_{K,t}\otimes_b \hat{\mathbf{X}}_t \end{bmatrix}, \tag{47}$$

with

$$\mathbb{E}[\mathbf{x}_{1,t}\otimes_b \hat{\mathbf{X}}_t] = \mathbb{E}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_{1,t}\otimes \mathbf{x}_{1,t} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{x}_{1,t}\otimes \mathbf{x}_{K,t} \end{bmatrix}. \tag{48}$$

We can calculate $\mathbb{E}[\mathbf{x}_{i,t} \otimes \mathbf{x}_{j,t}]$ as

$$\begin{cases} \mathbb{E}[\mathbf{x}_{k,t} \otimes \mathbf{x}_{k,t}] & \text{if } i = j = k \\ \mathbf{0} & \text{if } i \neq j, \end{cases} \tag{49}$$

where each entry of $\mathbb{E}[\mathbf{x}_{k,t} \otimes \mathbf{x}_{k,t}] \in \mathbb{R}^{D \times D}$ can be computed by utilizing the Isserlis' theorem [42] as

$$\begin{cases} \mathbb{E}[x_{k,l,t}^2] = \sigma_{ll}^2 & \forall k, l, t \\ \mathbb{E}[x_{k,l,t} x_{k,m,t}] = \sigma_{lm} & \text{if } l \neq m. \end{cases} \tag{50}$$

Given that the entries of $\mathbf{x}_{i,t}$ are independent of each other, i.e., $\sigma_{lm} = 0$ and assuming an identical variance among all entries of $\mathbf{x}_{i,t}$, i.e., $\sigma_{ll} = \sigma_k$, we can write (50) as

$$\begin{cases} \mathbb{E}[x_{k,l,t}^2] = \sigma_k^2 & \forall k, l, t \\ \mathbb{E}[x_{k,l,t} x_{k,m,t}] = 0 & \text{if } l \neq m. \end{cases} \tag{51}$$

REFERENCES

[1] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "On the resilience of online federated learning to model poisoning attacks through partial sharing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2024, pp. 9201–9205.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[3] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Adv. Neural Inf. Process. Syst.*, vol. 30, Jan. 2017.

[4] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[5] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[6] Q. Yang, Y. L., T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Feb. 2019.

[7] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated learning with non-IID data in wireless networks," *" IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[8] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, Aug. 2023, pp. 265–269.

[9] E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, "Noise-robust and resource-efficient admm-based federated learning," *arXiv preprint arXiv:2409.13451*, 2024.

[10] F. Hu, W. Zhou, K. Liao, H. Li, and D. Tong, "Toward federated learning models resistant to adversarial attacks," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16 917–16 930, Oct. 2023.

[11] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4531–4544, Mar. 2023.

[12] ——, "Communication-efficient online federated learning framework for nonlinear regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232.

[13] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, Nov. 2023, pp. 1199–1203.

[14] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 560–569.

[15] R. Jin, Y. Huang, X. He, H. Dai, and T. Wu, "Stochastic-sign SGD for federated learning with theoretical guarantees," *arXiv preprint arXiv:2002.10940*, Feb. 2020.

[16] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "1-bit compressive sensing for efficient federated learning over the air," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Oct. 2022.

[17] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2020, pp. 8253–8265.

[18] A. Kuh, "Real time kernel learning for sensor networks using principles of federated learning," in *Proc. IEEE Int. Conf. Asia-Pacific Signal Info. Process. Assoc.*, Dec. 2021, pp. 2089–2093.

[19] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3521–3530.

[20] X. Cao and N. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2022, pp. 3395–3403.

[21] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, Dec. 2017.

[22] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "SignSGD with majority vote is communication efficient and fault tolerant," *arXiv preprint arXiv:1810.05291*, Oct. 2018.

[23] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *USENIX Security Symp.*, Aug. 2020, pp. 1605–1622.

[24] Z. Wang, Q. Kang, X. Zhang, and Q. Hu, "Defense strategies toward model poisoning attacks in federated learning: A survey," in *Proc. IEEE Wireless Comm. Net. Conf.*, May 2022, pp. 548–553.

[25] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 5650–5659.

[26] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, Dec. 2020.

[27] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "FedRecover: Recovering from poisoning attacks in federated learning using historical information," in *Proc. IEEE Symp. Security Privacy*, Jul. 2023, pp. 326–343.

[28] H. Zhu and Q. Ling, "Byzantine-robust distributed learning with compression," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 9, pp. 280–294, 2023.

[29] Z. Yang and W. U. Bajwa, "Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 611–627, 2019.

[30] C. Fang, Z. Yang, and W. U. Bajwa, "Bridge: Byzantine-resilient decentralized gradient descent," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 610–626, 2022.

[31] Z. Liu, K. Zheng, L. Hou, H. Yang, and K. Yang, "A novel blockchain-assisted aggregation scheme for federated learning in IoT networks," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17 544–17 556, Oct. 2023.

[32] L. Cui, Y. Qu, G. Xie, D. Zeng, R. Li, S. Shen, and S. Yu, "Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures," *IEEE Trans. Industr. Inform.*, vol. 18, no. 5, pp. 3492–3500, 2022.

[33] Z. Song, H. Sun, H. H. Yang, X. Wang, Y. Zhang, and T. Q. S. Quek, "Reputation-based federated learning for secure wireless networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1212–1226, 2022.

[34] R. Jin, Y. Liu, Y. Huang, X. He, T. Wu, and H. Dai, "Sign-based gradient descent with heterogeneous data: Convergence and byzantine resilience," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, Jan. 2024.

[35] R. H. Koning, H. Neudecker, and T. Wansbeek, "Block kronecker products and the vecb operator," *Linear algebra and its applications*, vol. 149, pp. 165–184, Apr. 1991.

[36] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 472–484, Jan. 2014.

[37] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014.

[38] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 145–158, Mar. 2017.

[39] H. C. Shin and A. Sayed, "Mean-square performance of a family of affine projection algorithms," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 90–102, Jan. 2004.

[40] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, Nov. 2012, version 20121115. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html
[41] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, Dec. 2007, pp. 1–8.
[42] L. Isserlis, "On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables," *Biometrika*, vol. 12, pp. 134–139, Nov. 1918.

Ehsan Lari (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in Electrical Engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2016 and 2018, respectively. He is currently pursuing a Ph.D. degree at the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His research interests include federated and distributed learning, wireless communications, statistical signal processing, internet of things (IoT), and wireless sensor networks, with a focus on practical applications and the integration of advanced machine learning techniques into these domains.

Reza Arablouei received the Ph.D. degree in telecommunications engineering from the Institute for Telecommunications Research, University of South Australia, Mawson Lakes, SA, Australia, in 2013. He was a Research Fellow with the University of South Australia from 2013 to 2015. He is currently a Senior Research Scientist with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Pullenvale, QLD, Australia. His research interests include signal processing and machine learning on embedded systems.

Vinay Chakravarthi Gogineni (Senior Member, IEEE) received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Anantapur, India, in 2005, the master's degree in communication engineering from the Vellore Institute of Technology University, Vellore, India, in 2008, and the Ph.D. degree in electronics and electrical communication engineering from the Indian Institute of Technology Kharagpur, India, in 2019. He is currently an Assistant Professor with SDU Applied AI and Data Science, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark. Prior to this, he was a postdoctoral research Fellow with the Norwegian University of Science and Technology, Trondheim, Norway, and Simula Metropolitan Center for Digital Engineering (SimulaMet), Oslo, Norway. From 2008 to 2011, he was with a couple of MNCs in India.

His research interests include deep learning, decentralized machine learning, geometric deep learning, and their application in healthcare, internet-of-things and fusion energy. He was the recipient of the ERCIM Alain Bensoussan Fellowship in 2019, Best Paper Award at APSIPA ASC-2021, Tokyo, Japan, and HC Ørsted Research Talent Award, Denmark, 2024. He is also a member of the Editorial Board for the IEEE Sensors Journal.

Stefan Werner (Fellow, IEEE) received the M.Sc. Degree in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, in 1998, and the D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is currently a Professor with the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, Director of IoT@NTNU, and Adjunct Professor with Aalto University, Aalto, Finland. He was a visiting Melchor Professor with the University of Notre Dame during the summer of 2019 and an Adjunct Senior Research Fellow with the Institute for Telecommunications Research, University of South Australia, from 2014 to 2020. From 2009 to 2014, he held an Academy Research Fellowship, funded by the Academy of Finland. His research interests include adaptive and statistical signal processing, wireless communications, and security and privacy in cyber-physical systems. He is a Member of the Editorial boards of the EURASIP Journal of Signal Processing and IEEE Transactions on Signal and Information Processing over Networks.

Appendix C

Publication 3

P3 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.

RESOURCE-EFFICIENT FEDERATED LEARNING ROBUST TO COMMUNICATION ERRORS

Ehsan Lari[⋆]*, Vinay Chakravarthi Gogineni*[⋆]*, Reza Arablouei*[†] *and Stefan Werner*[⋆]

[⋆]Department of Electronic System, Norwegian University of Science and Technology, Trondheim, Norway
[†]Data61, CSIRO, Pullenvale QLD 4069, Australia

ABSTRACT

The effectiveness of federated learning (FL) in leveraging distributed datasets is highly contingent upon the accuracy of model exchanges between clients and servers. Communication errors caused by noisy links can negatively impact learning accuracy. To address this issue, we present an FL algorithm that is robust to communication errors while reducing the communication load on clients. To derive the proposed algorithm, we consider a weighted least-squares regression problem as a motivating example. We cast the considered problem as a distributed optimization problem over a federated network, which employs random scheduling to enhance communication efficiency, and solve it using the alternating direction method of multipliers. To improve robustness, we eliminate the local dual parameters and reduce the number of global model exchanges via a change of variable. We analyze the mean convergence of our proposed algorithm and demonstrate its effectiveness compared with related existing algorithms via simulations.

1. INTRODUCTION

With the increasing prevalence of smart devices, big data is becoming more ubiquitous. Learning from big data can enhance the decision-making capability of the end-users [1, 2]. However, this is challenging as the data is stored locally on edge devices and moving it to the cloud or a central server may raise privacy/security or excessive resource utilization concerns. Federated learning (FL) is a distributed learning paradigm that allows edge devices to collaboratively learn a shared global model using their locally-stored data without compromising their data privacy [3, 4]. FL is increasingly popular due to its ability to handle heterogeneous data and devices [5]. Data heterogeneity may refer to data being non-IID or imbalance in client data used to learn the global model [6, 7]. Device heterogeneity relates to diversity in storage, energy, computational, or communication resources of the clients participating in FL [8, 9].

The federated average (FedAvg) algorithm [3] is a popular baseline FL method. In FedAvg, the global iteration round begins with the server sharing its aggregated model with a subset of all clients selected uniformly at random, typically over a wireless network. After receiving the aggregated model from the server, the clients perform local learning to update the model and share the updated model with the server. Finally, the server receives the local models and aggregates them to produce a new global model. This process repeats until a specific convergence criterion is met. Many FL methods, including FedAvg, have been studied in the literature considering different aspects such as data privacy [10], model poisoning attacks [11], and communication efficiency [12]. However, most FL algorithms assume ideal communication links and do not take communication noise or error into account [13–17].

This work was supported by the Research Council of Norway.

When the communication channels between the server and the edge devices are noisy, the server receives noisy local updates due to uplink noise, and each client receives a noisy version of the aggregated model from the server due to downlink noise [18–20]. Using models that are corrupted by communication noise/error can deteriorate the quality of the learned model. Many works on FL have primarily focused on the uplink noise [21, 22]. In [23], the impact of downlink noise on FL is investigated. These studies show that the performance of gradient-descent-based FL algorithms can degrade when noise is present in the communication channels. In [19], a new loss function is proposed for FL using the first-order derivative of the loss function as a regularizer to overcome the problem of additive noise in communication links. In [23], two approaches are proposed to make FL more robust to the downlink noise. The first approach is based on using a quantization technique alongside transmitting the global model update via digital links and employing channel coding with a common rate. The other approach is based on an analog downlink transmission scheme where the server transmits an uncoded global model update.

The authors of [24, 25] propose that by controlling the scale of the communication signal-to-noise ratio, the noise can be tolerated and the convergence rate of FedAvg with perfect communication links can be maintained. However, they do not consider any countermeasure for the effects of the noise. In [26], the authors use precoding and scaling upon transmissions to alleviate the ill effects of noisy channels and ensure the convergence of their algorithm. These and some other similar methods proposed to deal with link noise in FL usually require additional resources on the client side. This can be counterproductive as, in FL, clients often operate with limited resources in terms of power/energy, memory, or computational capacity.

FL approaches based on the alternating direction method of multipliers (ADMM) can exhibit some robustness to additive communication noise due to the nature of their design [27]. However, they require all clients to participate in every FL round, which may be impractical in real-life scenarios when the clients are resource-constrained or heterogeneous edge devices. Therefore, there is a great demand for FL algorithms that are robust to noise in communication links with minimal extra communication or computational requirements.

In this paper, we propose a resource-efficient ADMM-based FL algorithm that is robust to communication noise/error while imposing no additional computational burden on the participating clients. We consider the presence of noise in both uplink and downlink communications. Considering the weighted least-squares (WLS) regression problem as a motivating example, we develop our proposed FL algorithm by iteratively solving an appropriately-formulated distributed convex optimization problem via the ADMM. To achieve communication efficiency, we employ random scheduling of the clients. Furthermore, to prevent error accumulation from degrading the learning, we communicate a linear combination of the last two global model updates as well as eliminating the dual model parameters at all participating edge devices. Through theoretical analysis, we show that the convergence of the proposed algorithm is ensured when the server chooses a

random subset of the clients at each iteration even with noisy communication links. Our simulation results also attest to the effectiveness of the proposed algorithm in comparison with the state of the art as well as corroborating our theoretical findings.

2. PROBLEM FORMULATION

We consider a federated network consisting of N edge devices (clients), each directly connected to a server. Each client i has a dataset denoted by $\mathcal{D}_i = \{\mathbf{H}_i, \mathbf{y}_i\}$ where $\mathbf{y}_i$ is a column vector with d_i entries and $\mathbf{H}_i$ is a matrix of size $d_i \times L$. For every client i, a linear regression model relating $\mathbf{H}_i$ to $\mathbf{y}_i$ can be described as

$$\mathbf{y}_i = \mathbf{H}_i\mathbf{x} + \boldsymbol{\nu}_i, \tag{1}$$

where $\mathbf{x}$ is the global regression parameter vector of size $L \times 1$ and $\boldsymbol{\nu}_i$ represents perturbation/noise.

The main goal of FL is to estimate the parameter vector $\mathbf{x}$ by collaboratively minimizing a global objective function over the federated network. To this end, we define a global WLS estimation problem in the federated setting as

$$\min_{\{\mathbf{x}_i\}} \sum_{i=1}^{N} \mathcal{J}_i(\mathbf{x}_i) \;\; \text{s.t. } \mathbf{x}_i = \mathbf{x},\; i = 1, 2, \cdots, N, \tag{2}$$

where $\mathcal{J}_i(\mathbf{x}) = \|\mathbf{y}_i - \mathbf{H}_i\mathbf{x}\|^2_{\mathbf{W}_i}$ is the local objective function for estimating $\mathbf{x}$ at client i and $\mathbf{W}_i$ is the error weight matrix of client i. In addition, $\mathbf{x}_i$ represents the local model estimate at client i and $\mathbf{x}$ denotes the global model estimate.

We use the ADMM to solve (2) whose associated augmented Lagrangian function can be expressed as

$$\sum_{i=1}^{N} \mathcal{L}_i(\mathbf{x}, \mathbf{x}_i, \boldsymbol{\pi}_i) = \sum_{i=1}^{N} \mathcal{J}_i(\mathbf{x}_i) + \langle \mathbf{x}_i - \mathbf{x}, \boldsymbol{\pi}_i \rangle + \frac{\rho_i}{2}\|\mathbf{x}_i - \mathbf{x}\|_2^2, \tag{3}$$

where $\boldsymbol{\pi}_i$ and $\rho_i > 0$ are the Lagrange multiplier vector and the penalty parameter associated with client i, respectively. Therefore, we obtain the following recursions at each client

$$\boldsymbol{\pi}_{i,k} = \boldsymbol{\pi}_{i,k-1} + \rho_i(\mathbf{x}_{i,k} - \mathbf{x}_k) \tag{4a}$$

$$\mathbf{x}_{i,k+1} = \hat{\mathbf{x}}_i - \mathbf{N}_i^{-1}(\boldsymbol{\pi}_{i,k} - \rho_i\mathbf{x}_k) \tag{4b}$$

and at the server

$$\mathbf{x}_{k+1} = \frac{1}{\sum_{i=1}^{N} \rho_i} \sum_{i=1}^{N} (\rho_i\mathbf{x}_{i,k+1} + \boldsymbol{\pi}_{i,k}) \tag{5}$$

where we define $\mathbf{N}_i = 2\mathbf{H}_i^\intercal\mathbf{W}_i\mathbf{H}_i + \rho_i\mathbf{I}$ and $\hat{\mathbf{x}}_i = 2\mathbf{N}_i^{-1}\mathbf{H}_i^\intercal\mathbf{W}_i\mathbf{y}_i$, and the index k denotes the iteration number. In the above algorithm, after performing local learning, i.e., (4a) and (4b), each client shares the local estimate $\rho_i\mathbf{x}_{i,k+1} + \boldsymbol{\pi}_{i,k}$ with the server. The server then obtains the global estimate as in (5) and broadcasts it to every client while the FL process continues.

In the formulation (4) and (5), there is a need to send both primal and dual model updates to the server in order for the server to be able to aggregate the global model update. However, the information in the dual update can be incorporated inside the primal update using a careful choice of the initial value. Therefore, we can reformulate the recursions (4)-(5) as

$$\mathbf{x}_{i,k+1} = (\mathbf{I} - \rho_i\mathbf{N}_i^{-1})\mathbf{x}_{i,k} + \rho_i\mathbf{N}_i^{-1}(2\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{6a}$$

$$\mathbf{x}_{k+1} = \frac{1}{\sum_{i=1}^{N} \rho_i} \sum_{i=1}^{N} \rho_i\mathbf{x}_{i,k+1}. \tag{6b}$$

via initializing $\mathbf{x}_{-1} = \mathbf{0}$, $\boldsymbol{\pi}_{i,-1} = \mathbf{0}$, and $\mathbf{x}_{i,0} = \hat{\mathbf{x}}_i$ and eliminating the Lagrange multipliers $\boldsymbol{\pi}_{i,k}$. Defining $\mathbf{s}_{i,k} = 2\mathbf{x}_{i,k} - \mathbf{x}_{i,k-1}$ and $\mathbf{s}_k = 2\mathbf{x}_k - \mathbf{x}_{k-1}$, we can further rewrite (6) as

$$\mathbf{x}_{i,k+1} = (\mathbf{I} - \rho_i\mathbf{N}_i^{-1})\mathbf{x}_{i,k} + \rho_i\mathbf{N}_i^{-1}\mathbf{s}_k \tag{7a}$$

$$\mathbf{s}_{i,k+1} = 2\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k} \tag{7b}$$

$$\mathbf{s}_{k+1} = \frac{1}{\sum_{i=1}^{N} \rho_i} \sum_{i=1}^{N} \rho_i\mathbf{s}_{i,k+1}. \tag{7c}$$

In this algorithm, the clients share $\mathbf{s}_{i,k+1}$ with the server, and the server broadcasts $\mathbf{s}_{k+1}$ to the clients. As we will show later, this formulation is favorable when the communication links are unideal.

The parameter exchanges between the clients and the server are often carried out over wireless communication channels. Therefore, both uplink and downlink channels may be corrupted by noise. In the downlink, the clients receive noisy versions of the aggregated model updates from the server, i.e., $\tilde{\mathbf{s}}_k^i = \mathbf{s}_k + \boldsymbol{\zeta}_k^i$ where $\boldsymbol{\zeta}_k^i$ denotes the downlink noise of client i at iteration k. In the uplink, the server receives a noisy version of the local model update of each client, i.e., $\tilde{\mathbf{s}}_{i,k} = \mathbf{s}_{i,k} + \boldsymbol{\eta}_{i,k}$ where $\boldsymbol{\eta}_{i,k}$ denotes the uplink noise of client i at iteration k.

Comparing the recursions (4)-(5) and (6) with (7), we hypothesize that introducing $\mathbf{s}_{i,k+1}$ as a linear combination of $\mathbf{x}_{i,k+1}$ and $\mathbf{x}_{i,k}$ and sending it instead of $\mathbf{x}_{i,k+1}$ can result in less noise corruption in the estimates of the clients due to using a single noisy global estimate rather that two. Hence, it can lead to improved performance in terms of robustness against additive communication noise.

However, the recursions (7) require all clients to take part in a global model update iteration. In FL, the clients may have different communication capabilities due to having limited resources. Therefore, the participation of all clients at each global update round may come at a considerable cost, e.g., slow convergence time or increased resource utilization. To tackle this, the server may implement a random scheduling of the clients and have only a subset of the clients denoted by $\mathcal{S}_k$ participate in model aggregation at each iteration k. The scheduling can lower the communication overhead of FL significantly. Due to the choosing of a random subset of the clients, some clients may not be selected at two consecutive iterations. As a result, the recursions (7) will fail to work as two consecutive updates $\mathbf{x}_{i,k+1}$ and $\mathbf{x}_{i,k}$ may not be available at the client and it is not always possible to calculate the signal $\mathbf{s}_{i,k+1}$ at the clients, i.e., if client i is selected at $k+1$ and $k' \neq k$, then $2\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k'} \neq \mathbf{s}_{i,k+1}$. Therefore, we consider sending the local model updates $\mathbf{x}_{i,k+1}$ instead of $\mathbf{s}_{i,k+1}$ in order to guarantee convergence.

3. RESOURCE-EFFICIENT FEDERATED LEARNING OVER NOISY CHANNELS

Communication efficiency is essential for FL in real-world applications, as it directly affects its scalability and cost-effectiveness. The efficiency of communication is closely tied to the amount of data that needs to be transmitted among the clients and the server during the model training process. When communication load is high, it can lead to increased resource usage and longer training times, resulting in higher costs and decreased system utility. Therefore, minimizing data transmissions while maintaining high accuracy is a critical challenge in FL.

Random scheduling of the clients for communication is one way to improve the communication efficiency in FL. However, as it is managed by the server, at any given time, two consecutive updates may not be available at the client. Hence, it may not be possible to calculate $\mathbf{s}_{i,k+1}$ at all clients and iterations. Therefore, the recursions (7) may fail to converge. As a solution, we propose to calculate the local model

Algorithm 1 : `RERCE-Fed`. N clients, penalty parameters ρ_i, set of all clients $\mathcal{S}$.

Initialization: global model $\mathbf{x}_0 = \mathbf{x}_{-1} = \mathbf{0}$, local models $\mathbf{x}_{i,0} = \hat{\mathbf{x}}_i$

For $k = 1, ...$

The server randomly selects a subset $\mathcal{S}_k$ of its clients and sends the aggregated global model $\mathbf{s}_k = 2\mathbf{x}_k - \mathbf{x}_{k-1}$ to them.

Client Local Update:

Each client $i \in \mathcal{S}_k$ receives a noisy version of $\mathbf{s}_k$ and updates its model via (8a), then sends its updated model $\mathbf{x}_{i,k+1}$ to the server.

Aggregation at the Server:

The server receives noisy versions of the locally updated models and aggregates them via (8b).

EndFor

updates at the clients selected by random scheduling $i \in \mathcal{S}_k$ and send them to the server. The server then aggregates the received local updates and sends $\mathbf{s}_k = 2\mathbf{x}_k - \mathbf{x}_{k-1}$ to the new set of selected clients. Note that $\mathbf{s}_k$ is corrupted with different noise for two different sets of clients and the last two global iterations.

In each global iteration k, the selected clients receive $\tilde{\mathbf{s}}_k^i$ from the server and update their model by (8a). Then, the server receives the signal $\tilde{\mathbf{x}}_{i,k+1}$ from the selected clients and aggregates the received signal via (8b) and broadcasts the latest global update to the selected client in the next iteration. The clients that are not selected maintain their last update until they are selected again. Therefore, the recursions of the proposed resource-efficient FL algorithm robust to communication errors, called RERCE-Fed, are expressed as

$$\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} - a_{i,k}\rho_i \mathbf{N}_i^{-1}\mathbf{x}_{i,k} + a_{i,k}\rho_i \mathbf{N}_i^{-1}\tilde{\mathbf{s}}_k^i \tag{8a}$$

$$\mathbf{x}_{k+1} = \frac{1}{\sum_{i=1}^{N} a_{i,k}\rho_i} \sum_{i=1}^{N} a_{i,k}\rho_i \tilde{\mathbf{x}}_{i,k+1}, \tag{8b}$$

where $a_{i,k}$ is the variable that represents random scheduling such that $a_{i,k} = 1$ when the client i in selected by the server in iteration k, i.e., $i \in \mathcal{S}_k$, and $a_{i,k} = 0$ otherwise. We summarize the proposed RERCE-Fed in Algorithm 1. In the following section, we study its mean convergence.

4. CONVERGENCE ANALYSIS

To facilitate the analysis, we define the extended optimal global model as $\mathbf{x}_e^\star = \mathbf{1}_{2N} \otimes \mathbf{x}^\star$ and the vector containing the client model estimate as $\tilde{\mathbf{x}}_{e,k} = \mathrm{col}\{\tilde{\mathbf{x}}_{1,k}, \cdots, \tilde{\mathbf{x}}_{N,k}\}$, where $\mathbf{1}_{2N}$ is the $2N \times 1$ vector of all ones, $\mathbf{x}^\star$ is the optimal solution to (2), $\otimes$ is the Kronecker product, and $\mathrm{col}\{\cdot\}$ denotes column-wise stacking.

With ideal communication links, it can be shown that the iterates $\mathbf{x}_{i,k}$ and $\mathbf{x}_k$ converge as $k \to \infty$. Our goal here is to show that they still converge when the communication channels are noisy.

Substituting (8b) in (8a), the global recursion of the proposed algorithm can be stated as

$$\begin{bmatrix} \mathbf{x}_{e,k+1} \\ \mathbf{x}_{e,k} \end{bmatrix} = \boldsymbol{\mathcal{A}}_k \begin{bmatrix} \mathbf{x}_{e,k} \\ \mathbf{x}_{e,k-1} \end{bmatrix} + \boldsymbol{\zeta}_k + \boldsymbol{\eta}_k, \tag{9}$$

where

$$\boldsymbol{\mathcal{A}}_k = \left[\begin{array}{c:c} \boldsymbol{\mathcal{A}}_{k,1} & \boldsymbol{\mathcal{A}}_{k,2} \\ \hdashline \boldsymbol{\mathcal{A}}_{k,3} & \boldsymbol{\mathcal{A}}_{k,4} \end{array}\right] \tag{10}$$

and the noise vectors $\boldsymbol{\zeta}_k$ and $\boldsymbol{\eta}_k$ stack

$$a_{i,k}\rho_i \mathbf{N}_i^{-1}\boldsymbol{\zeta}_{i,k} \tag{11}$$

and

$$a_{i,k}\rho_i \mathbf{N}_i^{-1} \sum_{j=1}^{N} \left(\frac{2a_{j,k-1}\rho_j \boldsymbol{\eta}_{j,k-1}}{\sum_{u=1}^{N} a_{u,k-1}\rho_u} - \frac{a_{j,k-2}\rho_j \boldsymbol{\eta}_{j,k-2}}{\sum_{u=1}^{N} a_{u,k-2}\rho_u} \right), \tag{12}$$

respectively, for $1 \le i \le N$ and zero for $N+1 \le i \le 2N$.

The value of $\boldsymbol{\mathcal{A}}_k \in \mathbb{R}^{2LN \times 2LN}$ depends on the iteration number k as the server selects a random number of clients at each iteration. Its sub-matrices of size $LN \times LN$ are block matrices whose $L \times L$ client-wise sub-matrices are

$$[\boldsymbol{\mathcal{A}}_{k,1}]_{ii} = \mathbf{I} - a_{i,k}\rho_i \mathbf{N}_i^{-1} + 2a_{i,k}a_{i,k-1} \frac{\rho_i^2 \mathbf{N}_i^{-1}}{\sum_{u=1}^{N} a_{u,k-1}\rho_u}, \tag{13a}$$

$$[\boldsymbol{\mathcal{A}}_{k,1}]_{ij} = 2a_{i,k}a_{j,k-1} \frac{\rho_i \rho_j \mathbf{N}_i^{-1}}{\sum_{u=1}^{N} a_{u,k-1}\rho_u}, \tag{13b}$$

$$[\boldsymbol{\mathcal{A}}_{k,2}]_{ij} = -a_{i,k}a_{j,k-2} \frac{\rho_i \rho_j \mathbf{N}_i^{-1}}{\sum_{u=1}^{N} a_{u,k-2}\rho_u}, \tag{13c}$$

$\boldsymbol{\mathcal{A}}_{k,3} = \mathbf{I}$, and $\boldsymbol{\mathcal{A}}_{k,4} = \mathbf{0}$.

Applying the expectation operator $\mathbb{E}[\cdot]$ to both sides of (9) and considering the fact that the noises are zero-mean, we obtain

$$\mathbb{E}\begin{bmatrix} \mathbf{x}_{e,k+1} \\ \mathbf{x}_{e,k} \end{bmatrix} = \boldsymbol{\mathcal{A}}_k \mathbb{E}\begin{bmatrix} \mathbf{x}_{e,k} \\ \mathbf{x}_{e,k-1} \end{bmatrix}. \tag{14}$$

Since $\boldsymbol{\mathcal{A}}_k$ is a right-stochastic matrix as the entries of all its columns add up to one, (14) can be recursively unfolded as

$$\mathbb{E}\begin{bmatrix} \mathbf{x}_{e,k+1} \\ \mathbf{x}_{e,k} \end{bmatrix} = \boldsymbol{\mathcal{A}}'_k \mathbb{E}\begin{bmatrix} \mathbf{x}_{e,1} \\ \mathbf{x}_{e,0} \end{bmatrix},$$

where $\boldsymbol{\mathcal{A}}'_k = \prod_{i=1}^{k} \boldsymbol{\mathcal{A}}_i$. The matrix $\boldsymbol{\mathcal{A}}'_k$ is right-stochastic as the multiplication of right-stochastic matrices produces a right-stochastic matrix, i.e., $\boldsymbol{\mathcal{A}}'_k \mathbf{1} = \left(\prod_{i=1}^{k} \boldsymbol{\mathcal{A}}_i\right) \mathbf{1} = \mathbf{1}$. An important property of a right-stochastic matrix is that all its eigenvalues λ_i satisfy $|\lambda_i| \le 1$. Therefore, $\boldsymbol{\mathcal{A}}'_k$ is stable and the recursions (9) converge.

Furthermore, defining $\mathbb{E}[\epsilon_{e,k+1}] = \mathbf{x}_e^\star - \mathbb{E}[\mathbf{x}_{e,k+1}^\intercal \mathbf{x}_{e,k}^\intercal]^\intercal$, and utilizing the fact that $\mathbf{x}_e^\star = \boldsymbol{\mathcal{A}}'_k \mathbf{x}_e^\star$, $\mathbb{E}[\epsilon_{e,k+1}]$ can be recursively expressed as $\mathbb{E}[\epsilon_{e,k+1}] = \boldsymbol{\mathcal{A}}'_k \mathbb{E}[\epsilon_{e,1}]$. As both $\boldsymbol{\mathcal{A}}'_k$ and $\mathbb{E}[\epsilon_{e,1}]$ are bounded, the expected error $\mathbb{E}[\epsilon_{e,k+1}]$ is bounded and the proposed RERCE-Fed algorithm converges. In the next section, we verify the convergence and robustness of the proposed algorithm to link noise via numerical simulations.

5. SIMULATION RESULTS

In this section, we conduct a series of experiments to examine the performance of the proposed RERCE-Fed. We consider a scenario having $N = 100$ clients connected to a server. The clients aim to learn a model $\mathbf{x}$ of dimension $L = 128$. To induce data imbalance, we draw the local dataset size of each client, d_i, randomly from a uniform distribution, i.e., $d_i \in \mathcal{U}(50, 90)$. Every client i has imbalanced synthetic non-IID data $\{\mathbf{H}_i, \mathbf{y}_i\}$ with the matrices $\mathbf{H}_i$ drawn from a normal distribution $\mathcal{N}(\mu_{\mathbf{H}_i}, \sigma^2_{\mathbf{H}_i})$ where $\mu_{\mathbf{H}_i} \in \mathcal{U}(-0.5, 0.5)$ and $\sigma^2_{\mathbf{H}_i} \in \mathcal{U}(0.5, 1.5)$. We set the weight matrices at each client i to the inverse covariance matrix of $\mathbf{y}_i$, i.e., $\mathbf{W}_i = \Sigma_{\mathbf{y}_i}^{-1} = \mathbb{E}[(\mathbf{y}_i - \mathbb{E}[\mathbf{y}_i])(\mathbf{y}_i - \mathbb{E}[\mathbf{y}_i])^\intercal]^{-1}$. The parameter vector $\mathbf{x}$ is drawn from a normal distribution $\mathcal{N}(0, 1)$. The observation noise $\boldsymbol{\nu}_i$ is taken as zero-mean IID Gaussian with variance

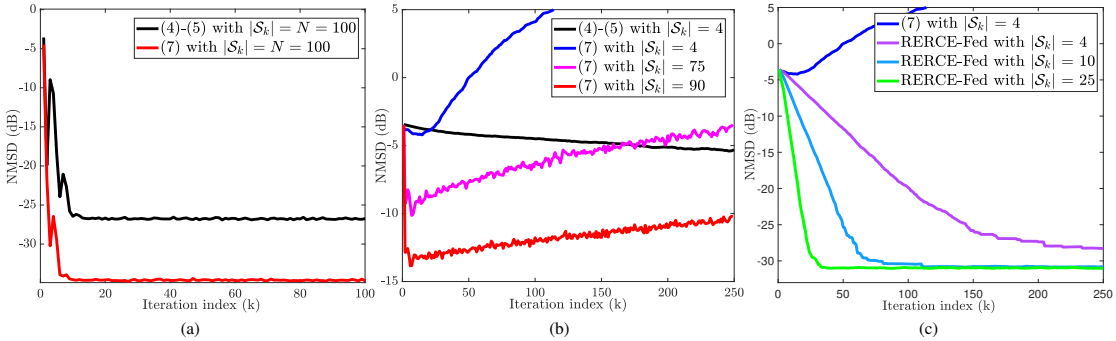

Fig. 1. Normalized mean-square deviation (NMSD) versus iteration index: (a) Learning curves of (4)-(5) and (7) for $|\mathcal{S}_k| = N = 100$. (b). Learning curves of (4)-(5) and (7) for different values of $|\mathcal{S}_k|$. (c). Learning curves of the proposed RERCE-Fed for different values of $|\mathcal{S}_k|$.

10^{-4} for each client. The additive noise in both uplink and downlink is zero-mean IID white Gaussian with variance 6.25×10^{-4}. In all simulated algorithms, the penalty parameter is set to $\rho_i = 1$ for all clients. The server randomly selected a subset of clients with uniform probability in every iteration k. We evaluate the performance of the considered algorithms via the network-wide average normalized mean-square deviation (NMSD) defined at iteration k as

$$\text{NMSD}(k) = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathbf{x}_{i,k} - \mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}. \tag{15}$$

The learning curves (i.e., NMSD in dB vs. iteration number k) presented in the following figures are obtained by averaging over 100 independent experiments.

In our first experiment, we simulate (4)-(5) and (7) solving the regression problem outlined in section 2 when all clients participate in FL, i.e., $|\mathcal{S}_k| = N = 100$. We present the corresponding learning curves in Fig. 1(a). We notice that (7) exhibits improved performance (7dB improvement) over (4)-(5) in the presence of noisy communication links.

In many practical applications, the FL clients operate under resource constraints. Thus, we next examine the performance of the considered algorithms when only a small subset of the clients participate in every communication and learning round. Hence, we simulate (7) when the server chooses only a subset of the clients, e.g., $|\mathcal{S}_k| \in \{4, 75, 90\}$. We also simulate (4)-(5) when $|\mathcal{S}_k| = 4$. We present the corresponding learning curves in Fig. 1(b). We observe that, when only a small subset of the clients participate in each FL round, (4)-(5) exhibit poor performance at the presence of link noise. In addition, (7), which exhibited good performance in the previous experiment, fails to converge. It appears to diverge due to error accumulation even when the majority of the clients participate in every FL round, e.g., $|\mathcal{S}_k| \in \{75, 90\}$. Therefore, it is evident that (7) cannot cope with noise in the communication links when only a small number of clients are selected during each iteration.

In our last experiment, we evaluate the performance of the proposed RERCE-Fed algorithm at the presence of noise in the communication links while incorporating random scheduling for communication efficiency. We simulate RERCE-Fed when $|\mathcal{S}_k| \in \{4, 10, 20\}$. We present the corresponding learning curves in Fig. 1(c). We observe that the proposed algorithm exhibits robustness against communication noise/error despite even when a small portion of the clients participate in every FL round. It is also clear that there is a trade-off between $|\mathcal{S}_k|$ and NMSD. Moreover, as the number of participating clients increases, the convergence rate increases. As the number of the participating clients increases, their number becomes less important, i.e., by setting the number of the participating clients to $|\mathcal{S}_k| \geq 10$, the performance is close to when all clients participate. Therefore, a desired performance can be attained with a relatively low number of clients participating at every iteration. This means, using the proposed algorithm, it is possible to achieve accurate model estimates in FL while making efficient use of the available communication resources, even when the communication links are imperfect. The proposed algorithm also delivers an effective trade-off between estimation accuracy and convergence rate on one side and communication resource utilization on the other. This trade-off can be easily governed by controlling the number of clients that participate in FL at every iteration. The participation rate need not necessarily be uniform. That is, depending on resource availability or conditions of the communication links, different numbers of clients may be summoned for FL at different iterations.

6. CONCLUSIONS

We developed a resource-efficient FL algorithm that has improved robustness against noise/error in communication links. To motivate the developed algorithm, we considered a weighted least-squares regression problem. To achieve the robustness, we proposed to combine the last two global model updates and send them together alongside eliminating the dual model update performed at each participating edge device. The proposed algorithm, called RERCE-Fed, ensures that clients receive a less corrupted global model update from the server even when the server uses random scheduling to achieve communication efficiency. We proved the convergence of RERCE-Fed in the mean sense at the presence of link noise. We also verified the desirable performance of RERCE-Fed via simulations, particularly, its robustness against additive communication link noise in comparison to existing related algorithms. In future work, we will analyze the mean-square performance of RERCE-Fed and consider applying it to different applications with potentially non-linear/non-quadratic or non-convex objective functions. We will also study the case when different numbers of clients may participate in different iterations of FL.

7. REFERENCES

[1] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[2] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb 2021.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. and Stat.*, Apr. 2017, pp. 1273–1282.

[4] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Proc. Advances in Neural Info. Process. Syst.*, vol. 30, Jan. 2017.

[5] Q. Yang, Y. L., T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, Feb. 2019.

[6] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated learning with non-iid data in wireless networks," *" IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[8] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, Apr. 2020.

[9] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor, "Federated learning: A signal processing perspective," *IEEE Signal Process. Mag.*, vol. 39, no. 3, pp. 14–41, May 2022.

[10] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[11] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, Aug. 2018.

[12] V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, pp. 1–1, Nov 2022.

[13] ——, "Communication-efficient online federated learning framework for nonlinear regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232.

[14] ——, "Decentralized graph federated multitask learning for streaming data," in *Proc. 56th Annu. Conf. Inf. Sci. Syst.*, Apr. 2022, pp. 101–106.

[15] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. on Selected Areas in Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

[16] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *Proc. IEEE Global Commun. Conf.*, Feb. 2018, pp. 1–7.

[17] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. on Commun.*, Jul. 2019, pp. 1–7.

[18] S. Zhou and G. Y. Li, "Commun.-efficient ADMM-based federated learning," *arXiv preprint arXiv:2110.15318*, Jan. 2021.

[19] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Mar. 2020.

[20] S. Zheng, C. Shen, and X. Chen, "Design and analysis of uplink and downlink communications for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2150–2167, Jul. 2021.

[21] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[22] H. Guo, A. Liu, and V. K. N. Lau, "Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 197–210, Jan. 2021.

[23] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of federated learning over a noisy downlink," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1422–1437, Mar. 2022.

[24] X. Wei and C. Shen, "Federated learning over noisy channels," in *n Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[25] ——, "Federated learning over noisy channels: Convergence analysis and design examples," *IEEE Trans. Cognitive Commun. and Networking*, vol. 8, no. 2, pp. 1253–1268, Jun. 2022.

[26] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "COTAF: Convergent over-the-air federated learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[27] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

Appendix D

Publication 4

P4 E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.

Continual Local Updates for Federated Learning with Enhanced Robustness to Link Noise

Ehsan Lari[1], Vinay Chakravarthi Gogineni[2], Reza Arablouei[3], Stefan Werner[1]
[1]Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
[2]The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark
[3]CSIRO's Data61, Pullenvale QLD 4069, Australia

***Abstract*—Communication errors caused by noisy links can negatively impact the accuracy of federated learning (FL) algorithms. To address this issue, we introduce an FL algorithm that is robust to communication errors while concurrently reducing the communication load on clients. To formulate the proposed algorithm, we consider a weighted least-squares regression problem as a motivating example. We recast this problem as a distributed optimization problem over a federated network, which employs random scheduling to enhance communication efficiency, and solve the reformulated problem via the alternating direction method of multipliers. Unlike conventional FL approaches employing random scheduling, the proposed algorithm grants the clients the ability to continually update their local model estimates even when they are not selected by the server to participate in FL. This allows for more frequent and ongoing client involvement, resulting in performance improvement and enhanced robustness to communication errors compared to when the local updates are only performed when the respective clients are selected by the server. We demonstrate the effectiveness and performance gains of the proposed algorithm through simulations.**

I. Introduction

Federated learning (FL) [1]–[6] is a distributed machine-learning approach that enables the training of models across multiple decentralized devices without transferring the data to any global server. FL allows devices to share their knowledge in terms of model estimates or gradients without revealing their raw data, thereby improving data privacy and security [7]. FL holds significant potential in a range of critical applications, including healthcare [8], finance [9], and industrial IoT [10], [11]. These data-sensitive domains demand utmost priority on data privacy, making FL an ideal approach to safeguard sensitive information while enabling effective machine learning.

The literature on FL encompasses a diverse array of methods that have been extensively investigated to address a multitude of aspects and challenges such as preserving privacy [9], [12]–[14], handling Byzantine attacks [15], [16], and improving communication efficiency [7], [17]. However, a significant number of these works operate under the assumption of ideal communication links, neglecting the presence of communication errors or noise [18]–[22]. In practical real-world applications, however, the communication links connecting clients and the server are susceptible to noise corruption, posing a potential threat to the performance of the model [23]. To tackle this challenge, several studies have introduced different techniques aimed at enhancing the accuracy of FL models when confronted with communication noise. Some works focus on the uplink noise and neglect the downlink noise [24], [25] while some other also consider the effects of the downlink noise [26].

Distributed algorithms that are based on the alternating direction method of multipliers (ADMM) can exhibit a certain degree of robustness to additive communication noise [27]. This is due to the inherent characteristics of ADMM that allow it to alleviate the impact of noise present in the communication channel. However, it requires complete collaboration among all clients, which stands in contrast to the objectives of FL, particularly those associated with system heterogeneity, such as varying computation and storage capacities across different clients. Hence, there exists a need for noise-robust ADMM-based algorithms that achieve convergence even when only a subset of clients participate in each iteration.

In this paper, we introduce a new FL algorithm that exhibits both communication efficiency and robustness to communication noise/errors. Additionally, our algorithm facilitates continual local updates at the clients, even when they are not selected by the server. This results in improved accuracy with no extra communication. We derive the proposed algorithm by using ADMM to solve a weighted least-squares (WLS) regression problem. We consider the communication links in both uplink and downlink to be noisy. To achieve noise robustness, we eliminate the dual variable update step at each client and transmit a linear combination of the last two global model updates. Our extensive simulation results corroborate the effectiveness of the proposed algorithm.

II. Background

Let us consider a federated network with K clients and a server. Each client k has an exclusive dataset denoted by $\mathcal{D}_k = \{\mathbf{H}_k, \mathbf{y}_k\}$ where $\mathbf{y}_k \in d_k$ is a column vector and $\mathbf{H}_k$ is a matrix of size $d_k \times L$. Each client independently trains a local model on its respective dataset using an FL algorithm. The process entails exchanging model updates with the global server, allowing for collaborative model training while preserving data privacy. For client k, a linear regression model relating $\mathbf{H}_k$ to $\mathbf{y}_k$ can be described as

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}^* + \boldsymbol{\nu}_k, \tag{1}$$

where $\mathbf{x}^* \in \mathbb{R}^L$ is the global regression parameter vector and $\boldsymbol{\nu}_k$ represents noise or perturbation.

The goal of FL is to find an estimate of $\mathbf{x}^*$ that is the optimal solution to

$$\min_{\{\mathbf{x}_k\}} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{x}_k) \;\; \text{s.t. } \mathbf{x}_k = \mathbf{x}, \;\; k = 1, 2, \cdots, K, \tag{2}$$

where $\mathcal{J}_k(\mathbf{x}) = \|\mathbf{y}_k - \mathbf{H}_k\mathbf{x}\|^2_{\mathbf{W}_k}$ is the local objective function at client k and $\mathbf{W}_k$ is the appropriate weight matrix of client k. In addition, $\mathbf{x}_k$ represents the local model estimate at client k, and $\mathbf{x}$ denotes the global model estimate.

We use an ADMM-based approach to solve (2). In this approach, primal and dual updates are computed by each client and transmitted to the global server via a noisy communication channel. Each client implements local iterations to update its local model and shares its updated local model estimate with the global server. The global server aggregates the received model estimates from the clients and updates the global model estimate. The server then sends the updated global model estimate to the clients over a noisy channel. The clients utilize the received global model estimate in updating their local estimates and the process continues until a convergence criterion is met.

To solve (2) using ADMM, we express the related augmented Lagrangian function as

$$\sum_{k=1}^{K} \mathcal{L}_k(\mathbf{x}, \mathbf{x}_k, \boldsymbol{\pi}_k) = \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{x_k}) + \langle \mathbf{x}_k - \mathbf{x}, \boldsymbol{\pi}_k \rangle + \frac{\rho_k}{2} \|\mathbf{x}_k - \mathbf{x}\|_2^2, \tag{3}$$

where $\boldsymbol{\pi}_k \in \mathbb{R}^L$ and $\rho_k > 0$ are, respectively, the Lagrange multiplier vector and the penalty parameter associated with client k. Hence, the corresponding ADMM iterations at each client k are given as

$$\boldsymbol{\pi}_{k,n} = \boldsymbol{\pi}_{k,n-1} + \rho_k(\mathbf{x}_{k,n} - \mathbf{x}_n) \tag{4a}$$

$$\mathbf{x}_{k,n+1} = \hat{\mathbf{x}}_k - \mathbf{N}_k^{-1}(\boldsymbol{\pi}_{k,n} - \rho_k \mathbf{x}_n) \tag{4b}$$

and at the global server as

$$\mathbf{x}_{n+1} = \frac{1}{\sum\limits_{k=1}^{K} \rho_k} \sum_{k=1}^{K} (\rho_k \mathbf{x}_{k,n+1} + \boldsymbol{\pi}_{k,n}) \tag{5}$$

where we define $\mathbf{N}_k = 2\mathbf{H}_k^\intercal \mathbf{W}_k \mathbf{H}_k + \rho_k \mathbf{I}$ and $\hat{\mathbf{x}}_k = 2\mathbf{N}_k^{-1}\mathbf{H}_k^\intercal \mathbf{W}_k \mathbf{y}_k$, and the index n denotes the iteration number.

In the above iterations, each client shares the local estimate $\rho_k \mathbf{x}_{k,n+1} + \boldsymbol{\pi}_{k,n}$ with the global server after computing (4a) and (4b). The global server then aggregates the received estimates to obtain the global estimate as in (5).

As it is evident from the recursions (4) and (5), both primal and dual model updates are sent to the server for it to estimate the global model update. However, by making a careful selection of the initial values, the information of the dual update can be incorporated into the primal update. Therefore, using the initial values $\mathbf{x}_{-1} = \mathbf{0}$, $\boldsymbol{\pi}_{k,-1} = \mathbf{0}$, and $\mathbf{x}_{k,0} = \hat{\mathbf{x}}_k$, we can eliminate the Lagrange multiplier vectors $\boldsymbol{\pi}_{k,n}$ and restate the recursions (4)-(5) as

$$\mathbf{x}_{k,n+1} = (\mathbf{I} - \rho_k \mathbf{N}_k^{-1})\mathbf{x}_{k,n} + \rho_k \mathbf{N}_k^{-1}(2\mathbf{x}_n - \mathbf{x}_{n-1}) \tag{6a}$$

$$\mathbf{x}_{n+1} = \frac{1}{\sum\limits_{k=1}^{K} \rho_k} \sum_{k=1}^{K} \rho_k \mathbf{x}_{k,n+1}. \tag{6b}$$

By defining $\mathbf{s}_{k,n} = 2\mathbf{x}_{k,n} - \mathbf{x}_{k,n-1}$ and $\mathbf{s}_n = 2\mathbf{x}_n - \mathbf{x}_{n-1}$, we can further rewrite (6) as

$$\mathbf{x}_{k,n+1} = (\mathbf{I} - \rho_k \mathbf{N}_k^{-1})\mathbf{x}_{k,n} + \rho_k \mathbf{N}_k^{-1}\mathbf{s}_n \tag{7a}$$

$$\mathbf{s}_{k,n+1} = 2\mathbf{x}_{k,n+1} - \mathbf{x}_{k,n} \tag{7b}$$

$$\mathbf{s}_{n+1} = \frac{1}{\sum\limits_{k=1}^{K} \rho_k} \sum_{k=1}^{K} \rho_k \mathbf{s}_{k,n+1}. \tag{7c}$$

In this algorithm, the clients share $\mathbf{s}_{k,n+1}$ with the server, and the server broadcasts $\mathbf{s}_{n+1}$ to the clients. When the exchange of model parameters occurs over noisy communication channels, the iterations (7) are more robust to communication noise compared to (6) as they utilize a single noisy global estimate received from the server to update the local estimate, i.e., $\mathbf{s}_n$, rather than two in (6), i.e., $\mathbf{x}_n$ and $\mathbf{x}_{n-1}$.

In the downlink, noisy versions of the aggregated global model updates are received by the clients as $\tilde{\mathbf{s}}_n^k = \mathbf{s}_n + \boldsymbol{\zeta}_n^k$ where $\boldsymbol{\zeta}_n^k$ denotes the downlink noise of client k at iteration n. In the uplink, a noisy version of the local model update of each client is received by the server as $\tilde{\mathbf{s}}_{k,n} = \mathbf{s}_{k,n} + \boldsymbol{\eta}_{k,n}$ where $\boldsymbol{\eta}_{k,n}$ denotes the uplink noise of client k at iteration n.

In recursions (7), all clients are required to take part in each global model update. However, FL clients often have limited and diverse communication capabilities. Hence, participation of all clients in each global update round can be costly. Therefore, FL servers usually employ random scheduling of their clients by selecting only a subset of the clients, denoted by $\mathcal{S}_n$, to participate in model aggregation during each iteration n. The scheduling reduces the communications required at each iteration, leading to improved efficiency and better resource management. Utilizing random scheduling and considering noisy communications, we can rewrite the recursions (7) as mentioned in [28] as

$$\mathbf{x}_{k,n+1} = \begin{cases} \left(\mathbf{I} - \rho_k \mathbf{N}_k^{-1}\right) \mathbf{x}_{k,n} + \rho_k \mathbf{N}_k^{-1} \tilde{\mathbf{s}}_n^k, & k \in \mathcal{S}_n \\ \mathbf{x}_{k,n}, & \text{otherwise} \end{cases} \tag{8a}$$

$$\mathbf{s}_{k,n+1} = 2\mathbf{x}_{k,n+1} - \mathbf{x}_{k,n}, \quad k \in \mathcal{S}_n \tag{8b}$$

$$\mathbf{s}_{n+1} = \frac{1}{\sum\limits_{k \in \mathcal{S}_n} \rho_k} \sum_{k \in \mathcal{S}_n} \rho_k \tilde{\mathbf{s}}_{k,n+1}. \tag{8c}$$

III. Proposed Algorithm

It is important to enhance the energy efficiency of FL by reducing its communication load, particularly considering that clients often have constraints on their energy resources in real-world scenarios. Lowering the communication burden on the clients can also improve the scalability and cost-effectiveness of

FL algorithms. The efficiency of communication during model training is related to the amount of data exchanged between the clients and the server. Thus, minimizing the exchanges of models, parameters, gradients, or other relevant information while upholding high accuracy levels presents a significant challenge in FL.

Random scheduling of clients is a practical method to enhance communication efficiency within FL, using which the overall communication load can be effectively reduced and resource utilization can be optimized more efficiently. It helps alleviate the potential bottlenecks that can arise from simultaneous communication by all clients. As a result, it improves the efficiency of data transmissions and plays a role in enhancing the overall performance of FL. However, in the conventional random scheduling approach as in (8), the clients that are not selected during each iteration do not carry out any local update and their most recent local model estimates are not incorporated into the global model aggregation process.

Here, we propose to allow all clients, including those not selected through random scheduling, to update their local model estimates during every iteration. As we will show later, this can improve the performance without introducing any additional communication overhead or imposing any significant increase in computations on the clients or the server. To realize the proposed algorithm, we let the clients store the most recent global model estimate received from the server and the server store the latest local model estimates received from the clients. Hence, the clients continually update their local models using their most recent global model estimate and the server updates the global model using the latest local updates from all clients. When a client is selected at iteration n, its latest local model estimate is made up-to-date at the server and the global model estimate received from the server replaces its older version at the client.

Therefore, the recursions of the proposed resource-efficient and noise-robust FL algorithm featuring continual local updates are given by

$$\begin{aligned}\mathbf{x}_{k,n+1} &= (\mathbf{I}-\rho_k\mathbf{N}_k^{-1})\mathbf{x}_{k,n} \\ &\quad + \rho_k\mathbf{N}_k^{-1}\left[a_{k,n}\tilde{\mathbf{s}}_n^k + (1-a_{k,n})\tilde{\mathbf{s}}_m^k\right] \qquad (9a)\\ \mathbf{s}_{k,n+1} &= 2\mathbf{x}_{k,n+1} - \mathbf{x}_{k,n}, \quad k\in\mathcal{S}_n \qquad (9b)\\ \mathbf{s}_{n+1} &= \frac{1}{\sum_{k=1}^{K}\rho_k}\sum_{k=1}^{K}\rho_k\left[a_{k,n}\tilde{\mathbf{s}}_{k,n+1} + (1-a_{k,n})\tilde{\mathbf{s}}_{k,m}\right], \qquad (9c)\end{aligned}$$

where $a_{k,n}$ represents random scheduling, i.e., $a_{k,n}=1$ when $k\in\mathcal{S}_n$ and $a_{k,n}=0$ otherwise. In addition, $\tilde{\mathbf{s}}_m^k$ represents the most recent global model estimate received from the server and stored in client k, which is utilized when the client is not chosen by the server. Moreover, $\tilde{\mathbf{s}}_{k,m}$ denotes the most recent local model estimate associated with client k, which is stored at the server and utilized during iterations when the client is not selected through random scheduling. We summarize the proposed algorithm in Algorithm 1.

Algorithm 1 The proposed communication-efficient and noise-robust FL algorithm with continual local updates.

Parameters: penalty parameters ρ_k, number of clients K, set of clients $\mathcal{S}$

Initialization: global model $\mathbf{x}_0 = \mathbf{x}_{-1} = \mathbf{0}$, local models $\mathbf{x}_{k,0} = \hat{\mathbf{x}}_k$

For $n = 1, \cdots, N$

The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{s}_n$ to them.

Client Local Update:

If $k\in\mathcal{S}_n$

Receive $\tilde{\mathbf{s}}_n^k$, a noisy version of $\mathbf{s}_n$, from the server.
Store the latest global model $\tilde{\mathbf{s}}_m^k = \tilde{\mathbf{s}}_n^k$.
Update the local model as
$\mathbf{x}_{k,n+1} = (\mathbf{I}-\rho_k\mathbf{N}_k^{-1})\mathbf{x}_{k,n} + \rho_k\mathbf{N}_k^{-1}\tilde{\mathbf{s}}_n^k$.
Send $\mathbf{s}_{k,n+1} = 2\mathbf{x}_{k,n+1} - \mathbf{x}_{k,n}$ to the server.

Else

Update the local model as
$\mathbf{x}_{k,n+1} = (\mathbf{I}-\rho_k\mathbf{N}_k^{-1})\mathbf{x}_{k,n} + \rho_k\mathbf{N}_k^{-1}\tilde{\mathbf{s}}_m^k$.

EndIf

Aggregation at the Server:

The server receives $\tilde{\mathbf{s}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k\in\mathcal{S}_n$ and aggregates them with $\tilde{\mathbf{s}}_{k,m}$, the stored local model estimates of the non-selected clients via

$$\mathbf{s}_{n+1} = \frac{1}{\sum_{k=1}^{K}\rho_k}\sum_{k=1}^{K}\rho_k\left[a_{k,n}\tilde{\mathbf{s}}_{k,n+1} + (1-a_{k,n})\tilde{\mathbf{s}}_{k,m}\right].$$

EndFor

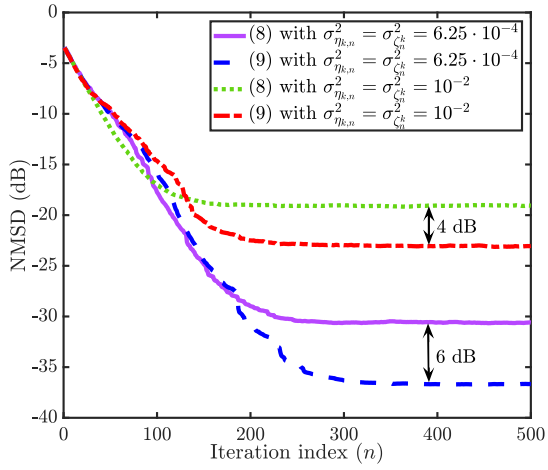

Fig. 1. NMSD of (8) and (9) versus iteration number for $|\mathcal{S}_n| = 4$ and different link noise variances.

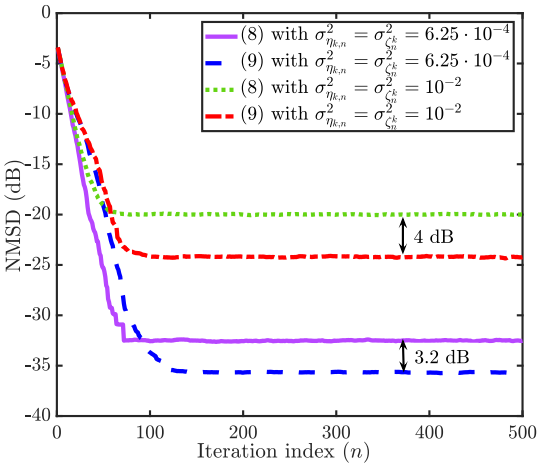

Fig. 2. NMSD of (8) and (9) versus iteration number for $|\mathcal{S}_n| = 10$ and different link noise variances.

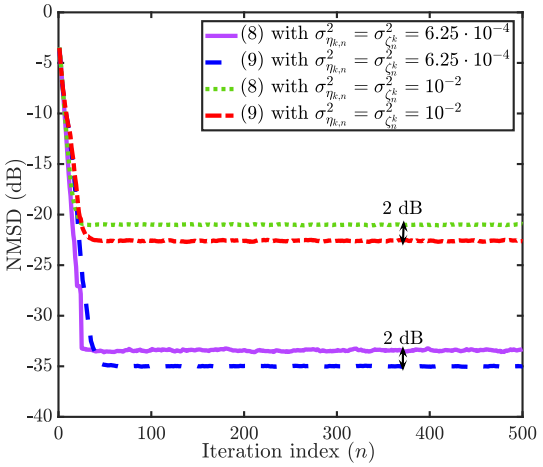

Fig. 3. NMSD of (8) and (9) versus iteration number for $|\mathcal{S}_n| = 25$ and different link noise variances.

IV. Simulation Results

In this section, we conduct a series of numerical experiments to examine the performance of the proposed algorithm. We consider a scenario with $K = 100$ clients directly connected to a global server. The goal of the federated network is to estimate a global model $\mathbf{x}^*$ of dimension $L = 128$. To induce data imbalance among the clients, we draw the size of each client's local dataset, d_k, from a uniform distribution, i.e., $d_k \in \mathcal{U}(50, 90)$. Each client k has synthetic and imbalanced non-IID data $\{\mathbf{H}_k, \mathbf{y}_k\}$ with the matrices $\mathbf{H}_k$ drawn from a multivariate normal distribution $\mathcal{N}(\mu_{\mathbf{H}_k}, \sigma^2_{\mathbf{H}_k})$ where $\mu_{\mathbf{H}_k} \in \mathcal{U}(-0.5, 0.5)$ and $\sigma^2_{\mathbf{H}_k} \in \mathcal{U}(0.5, 1.5)$. The weight matrices are set to the inverse covariance matrix of $\mathbf{y}_k$ at each client k, i.e., $\mathbf{W}_k = \Sigma^{-1}_{\mathbf{y}_k} = \mathbb{E}[(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])^\intercal]^{-1}$. We set the global parameter vector $\mathbf{x}^*$ arbitrarily by drawing each entry from a standard normal distribution $\mathcal{N}(0, 1)$. The observation noise $\boldsymbol{\nu}_k$ at each client k is zero-mean IID Gaussian with variance 10^{-4}. The additive noise in both uplink and downlink are zero-mean IID white Gaussian. In all experiments, we set the penalty parameter to $\rho_k = 1$ for all clients. At each iteration n, the server selects a subset of the clients with equal probability of selection assigned to each client. We evaluate the performance using the network-wide average normalized mean-square deviation (NMSD) defined at each iteration n as

$$\text{NMSD}(n) = \frac{1}{K} \sum_{k=1}^{K} \frac{\|\mathbf{x}_{k,n} - \mathbf{x}^\star\|_2^2}{\|\mathbf{x}^\star\|_2^2}. \tag{10}$$

We obtain the learning curves (i.e., NMSD in dB vs. iteration number n) by averaging over 100 independent trials.

In Figs. 1-3, we present a performance comparison between the proposed algorithm, i.e., (9), and its closest contender, i.e. (8), which does not feature continual local updates. We obtain the results by including noise in the communication links and utilizing random scheduling for communication efficiency. We set the number of clients selected at each iteration to $|\mathcal{S}_n| \in \{4, 10, 25\}$ and use different link noise variances of $\sigma^2_{\boldsymbol{\eta}_{k,n}} = \sigma^2_{\boldsymbol{\zeta}^k_n} \in \{6.25 \cdot 10^{-4}, 10^{-2}\}$. We plot the corresponding learning curves in the figures. We observe that the proposed algorithm exhibits robustness against communication noise/error even when a small portion of the clients participate in every FL round. It also outperforms (8) significantly in terms of the steady-state NMSD in all considered cases. In summary, the results demonstrate that permitting the clients, which are not selected by random scheduling, to continually update their local models leads to a notable reduction in the steady-state NMSD without compromising the convergence rate. Moreover, as expected, when the variance of the noise in the communication links increases, the learning accuracy decreases.

V. Conclusions

We proposed a new FL algorithm that leverages random scheduling to enhance communication efficiency while being robust to additive noise in the communication links, in part, due to eliminating the dual parameters and minimizing the use of noisy estimates in update equations. The key novel feature of the proposed algorithm is that it allows the clients to update their model estimates locally even when they are not selected by the global server for participation in FL as per random scheduling. Our simulation results showed that the continual local updates lead to performance improvements in terms of both learning accuracy and robustness to link noise.

Acknowledgement

The Research Council of Norway supported this work.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. and Stat.*, Apr. 2017, pp. 1273–1282.

[2] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Proc. Adv. Neural Inf. Process. Sys.*, vol. 30, Jan. 2017.

[3] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[4] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Automat. Contr.*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[5] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb 2021.

[6] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[7] H. Zhao, W. Du, F. Li, P. Li, and G. Liu, "FedPrompt: Communication-efficient and privacy-preserving prompt tuning in federated learning," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, 2023, pp. 1–5.

[8] T. Xia, J. Han, A. Ghosh, and C. Mascolo, "Cross-device federated learning for mobile health diagnostics: A first study on COVID-19 detection," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, 2023, pp. 1–5.

[9] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Mag.*, vol. 9, no. 3, pp. 8–16, 2020.

[10] R. Xie, C. Li, X. Zhou, and Z. Dong, "Asynchronous federated learning for real-time multiple licence plate recognition through semantic communication," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, 2023, pp. 1–5.

[11] Z. Zheng, Y. Zhou, Y. Sun, Z. Wang, B. Liu, and K. Li, "Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges," *Connection Science*, vol. 34, no. 1, pp. 1–28, 2022.

[12] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.

[13] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[14] Y. Chang, K. Zhang, J. Gong, and H. Qian, "Privacy-preserving federated learning via functional encryption, revisited," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 1855–1869, 2023.

[15] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, Aug. 2018.

[16] X. He, J. Zhang, and Q. Ling, "Byzantine-robust and communication-efficient personalized federated learning," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, 2023, pp. 1–5.

[17] V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, pp. 4531–4544, Nov 2022.

[18] ——, "Communication-efficient online federated learning framework for nonlinear regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232.

[19] ——, "Decentralized graph federated multitask learning for streaming data," in *Proc. 56th Annu. Conf. Inf. Sci. Syst.*, Apr. 2022, pp. 101–106.

[20] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. on Selected Areas in Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

[21] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *Proc. IEEE Global Commun. Conf.*, Feb. 2018, pp. 1–7.

[22] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. on Commun.*, Jul. 2019, pp. 1–7.

[23] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Mar. 2020.

[24] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[25] H. Guo, A. Liu, and V. K. N. Lau, "Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 197–210, Jan. 2021.

[26] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of federated learning over a noisy downlink," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1422–1437, Mar. 2022.

[27] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[28] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, 2023, pp. 265–269.

Appendix E

Publication 5

P5 E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, “Noise-Robust and Resource-Efficient ADMM-based Federated Learning for WLS Regression," submitted to *Elsevier Signal Processing*.

Noise-Robust and Resource-Efficient ADMM-based Federated Learning for WLS Regression*

Ehsan Lari[a,*], Reza Arablouei[b], Vinay Chakravarthi Gogineni[c], Stefan Werner[a]

[a]Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
[b]CSIRO's Data61, Pullenvale, QLD, Australia
[c]Applied AI and Data Science Unit, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark

Abstract

Federated learning (FL) leverages client-server communications to train global models on decentralized data. However, communication noise or errors can impair model accuracy. To address this challenge, we propose a novel FL algorithm that enhances robustness against communication noise while also reducing communication load. We derive the proposed algorithm by solving the weighted least-squares (WLS) regression problem, framed as a distributed convex optimization problem over a federated network with random client scheduling, via the alternating direction method of multipliers (ADMM). To counteract the detrimental effects of cumulative communication noise, we introduce a key modification by eliminating the dual variable and implementing a new local model update at each participating client. This subtle yet effective change results in using a single noisy global model update at each client instead of two, improving robustness against additive communication noise. Furthermore, we incorporate another modification enabling clients to continue local updates even when not selected by the server, leading to substantial performance improvements. Our theoretical analysis confirms the convergence of the proposed algorithm in both mean and mean-square senses, even when the server communicates with a random subset of clients over noisy links. Numerical results validate the effectiveness of our algorithm and corroborate theoretical findings.

Keywords:
Alternating direction method of multipliers, federated learning, noisy communication links, weighted least-squares.

1. Introduction

The proliferation of smart devices has led to the widespread availability of big data, which has the potential to enhance decision-making processes for end-users [1, 2, 3, 4, 5]. However, leveraging this data effectively presents several challenges, particularly in terms of privacy, security, and resource management. A key issue is that data is typically stored locally on edge devices, and transferring it to a central server or the cloud raises privacy and security concerns and can lead to excessive resource utilization. Federated learning (FL) has emerged as a promising machine learning paradigm that addresses these challenges by enabling edge devices to collaboratively train a global model without sharing their locally stored data [1, 6]. FL faces two significant challenges: data heterogeneity and device heterogeneity [7]. Data heterogeneity arises from the non-independent and identically distributed (non-IID) nature of data or imbalances among client datasets used in training the global model [8, 9]. Diversity in data distribution across participants can significantly impact model convergence and performance. Additionally, device heterogeneity refers to the disparities in storage capacity, energy resources, computational power, and communication capabilities among participating clients [10, 11]. Variations in device characteristics can affect the efficiency and fairness of the FL process. Despite these challenges, FL emerges as a promising approach for managing heterogeneous data and devices in distributed learning settings.

The FL literature frequently employs the FedAvg algorithm [1] as a standard benchmark. FedAvg begins with the server broadcasting its aggregated global model to a randomly chosen subset of clients, typically over a wireless network. These clients then perform local training to refine their local models before sending the updates back to the server. The server aggregates these local models into a new global model, repeating this process until a specific convergence criterion is met. Building on FedAvg, numerous FL approaches have been developed to address various challenges inherent in distributed learning. These challenges include preserving privacy [12, 13, 14, 15, 16, 17, 18], mitigating Byzantine attacks [19, 20, 21, 22], and improving communication efficiency [23, 24, 25, 26, 27, 28]. However, many of these approaches assume ideal communication links, overlooking the potential impact of communication errors or noise [29, 30, 31, 32]. In practical deployments, the communication channels be-

*This work was supported by the Research Council of Norway. Parts of this work have been presented at the 2023 IEEE Statistical Signal Processing Workshop (SSP) [DOI: 10.1109/SSP53291.2023.10208024] and the 2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) [DOI: 10.1109/APSIPAASC58517.2023.10317446].

*Corresponding author
Email addresses: ehsan.lari@ntnu.no (Ehsan Lari), reza.arablouei@csiro.au (Reza Arablouei), vigo@mmmi.sdu.dk (Vinay Chakravarthi Gogineni), stefan.werner@ntnu.no (Stefan Werner)

tween clients and the server are often subject to noise, which can degrade model performance [33]. To address this issue, various techniques have been proposed to enhance the robustness of FL models in the presence of communication noise. While some studies focus on the uplink noise and neglect the downlink noise [34, 35], others account for noise in both communication directions [36, 37].

In real-world scenarios, communication channels between the central server and edge devices are often imperfect, leading to noise in both uplink and downlink [38, 33, 39, 40, 41]. This leads to the server receiving noisy versions of local model updates and clients obtaining noisy versions of the aggregated model from the server, potentially compromising the quality of the learned model. The presence of noise in communication channels can severely impair the performance of gradient-descent-based FL algorithms. Several solutions have been proposed to address this challenge. One approach involves utilizing a new loss function that incorporates the first-order derivative as a regularizer to counteract additive noise [33]. Additionally, resilience to downlink noise can be enhanced through techniques such as digital transmission with quantization and channel coding or analog downlink transmission using uncoded global model updates [36]. In [42], the authors utilize precoding and scaling during transmissions to mitigate the adverse effects of noisy channels, ensuring the convergence of their algorithm. Moreover, [43, 44] propose that controlling the scale of the communication signal-to-noise ratio (SNR) can help tolerate noise while maintaining the convergence rate of FedAvg.

In noisy communication environments, gradient-descent-based methods, such as FedAvg, require the noise variance to decrease at a rate of $O(1/n^2)$ as a sufficient condition for convergence, as demonstrated in [44, Theorem 1]. As stated in [40], this condition often necessitates additional resources on the client side, such as increased transmission power or bandwidth, which is impractical in many real-world FL scenarios with resource-constrained clients. Therefore, a robust variant of FedAvg is introduced in [40]. However, the corresponding analysis does not consider the impact of random client participation, resulting in significant performance degradation, even when 25% of clients participate in training. Our approach aims to effectively address these challenges, providing a robust and resource-efficient solution for FL under noisy communication conditions.

Several recent studies have explored the application of the alternating direction method of multipliers (ADMM) [45] in the context of FL. For instance, in [46], an ADMM-based FL framework is introduced that employs an inexact primal variable update scheme to alleviate the computational burden associated with solving local optimization problems. In [47], an algorithm is proposed that modifies the order of dual variable updates at the client level, enabling them to mitigate server drift. To address heterogeneity in federated networks, in [48], an ADMM-based FL algorithm is proposed. In [49], a hierarchical structured ADMM-based FL algorithm is introduced to enhance both privacy and learning performance. Furthermore, in [50], a decentralized FL algorithm is presented to cope with model inconsistency arising from non-identically distributed data across clients. Building on these efforts, the work in [51] utilizes ADMM to develop an FL algorithm that explicitly addresses system heterogeneity. Despite these advancements, the impact of additive communication noise on the convergence and accuracy of ADMM-based FL algorithms remains largely unexplored.

In [52, 53, 54, 55], it is shown that the algorithms based on ADMM can achieve convergence even in the presence of additive communication noise. This robustness positions ADMM as a promising candidate for FL applications where communication noise is a concern. However, conventional ADMM algorithms require the participation of all clients in each FL round, which may be impractical in real-world FL scenarios due to resource constraints and the heterogeneity of edge devices. Therefore, there is a pressing need for developing ADMM-based FL algorithms that can effectively cope with communication noise with minimal additional communication or computational overhead.

In this paper, we propose a new communication-efficient FL algorithm based on ADMM that is resilient against communication noise and errors in both uplink and downlink channels, without imposing any additional computational burden on participating clients. We narrow the focus to the weighted least-squares (WLS) regression problem to ensure robust theoretical guarantees. This scope allows us to establish the mean convergence of both global and local models toward the optimal solution while presenting a novel mean-square convergence analysis and a closed-form expression for the mean-square error. The WLS problem is widely applicable in numerous contexts involving linear models and maximum likelihood or maximum a posteriori estimation, particularly when error variance varies across observations. To enhance communication efficiency, we incorporate random client scheduling, where the server selects a subset of clients during each FL iteration. To alleviate the adverse effects of link noise, we communicate a linear combination of the last two global model updates and eliminate the dual model parameters at all participating edge devices. Moreover, we enable clients to continue performing model updates, even when not scheduled for participation in any FL round. We evaluate the proposed algorithm and showcase its effectiveness through comprehensive theoretical performance analysis and numerical simulations. In summary, our main contributions are:

- We propose an ADMM-based FL algorithm for solving the WLS regression problem that is robust to communication noise without imposing any additional computational overhead on clients.
- To effectively reduce communication overhead during the learning process, we implement a randomized client scheduling mechanism. Subsequently, we utilize the statistical properties of discrete uniform random processes to facilitate our performance analysis. To further enhance performance and efficiency, we allow continual local updates at clients not selected by the server in any FL round.
- As a key advancement over the related conference precursors [56, 57], we present a rigorous first- and second-order

theoretical analysis of the proposed algorithm, proving its convergence in both mean and mean-square senses, even in the presence of link noise and random scheduling. In addition, we derive the theoretical steady-state mean-square error (MSE) as a function of various parameters, including the number of participating clients in each iteration and the uplink and downlink noise variances. We validate the accuracy of our analysis through comparisons with simulation results.

The remainder of this paper is organized as follows. In section 2, we provide an overview of the system model and the background for our problem. In section 3, we introduce our proposed noise-robust and communication-efficient FL algorithm. In addition, we introduce a modification to allow continual local updates at clients regardless of random client scheduling to improve performance. In section 4, we evaluate the robustness of the proposed algorithm by analyzing its theoretical mean and mean-square convergence and calculating its theoretical steady-state MSE. In section 5, we validate our theoretical findings through comprehensive numerical experiments, assessing algorithm performance through both theoretical predictions and numerical simulations. In section 6, we provide an overview of our key contributions, highlight the potential benefits, address the limitations, and outline promising directions for future research. Finally, in section 7, we present some concluding remarks.

Mathematical Notations: We denote scalars by lowercase letters, column vectors by bold lowercase letters, and matrices by bold uppercase letters. The superscripts $(\cdot)^\intercal$ and $(\cdot)^{-1}$ denote the transpose and inverse operations, respectively, while $\|\cdot\|$ signifies the Euclidean norm. Additionally, $\mathbf{1}_K$ denotes a column vector with K entries, all set to one, and $\mathbf{I}_K$ is the $K \times K$ identity matrix. The operators col$\{\cdot\}$, bcol$\{\cdot\}$, diag$\{\cdot\}$ and bdiag$\{\cdot\}$ represent column-wise stacking, block column-wise stacking, diagonalization and block diagonalization, respectively. Furthermore, $\otimes_b$ denotes the block Kronecker product, bvec$\{\cdot\}$ is the block vectorization operator, and bvec$^{-1}\{\cdot\}$ is its inverse operator. Lastly, tr$(\cdot)$ denotes the trace of a matrix.

2. Federated Learning over Noisy Channels

We consider a federated network consisting of a server and K clients, where the clients communicate with the server over wireless channels. Each client k has access to a local dataset denoted by $\mathcal{D}_k = \{\mathbf{X}_k, \mathbf{y}_k\}$, which comprises a column response vector $\mathbf{y}_k$ with d_k entries, and a data matrix $\mathbf{X}_k$ of size $d_k \times L$. For each client k, a linear model is employed to relate the data matrix $\mathbf{X}_k$ to the response vector $\mathbf{y}_k$ as

$$\mathbf{y}_k = \mathbf{X}_k\boldsymbol{\omega} + \boldsymbol{\nu}_k, \tag{1}$$

where $\boldsymbol{\omega}$ denotes the global regression model parameter vector of size L, and $\boldsymbol{\nu}_k$ represents the observation noise or perturbation, which is a vector of size d_k with each entry assumed to be zero-mean Gaussian.

2.1. Federated Weighted Least-Squares Regression

Weighted least-squares (WLS) regression is a natural extension of least-squares regression that offers significant advantages in various signal processing applications, such as power system state estimation [58], position estimation [59], and image noise reduction [60]. In WLS, different observations are assigned weights based on their reliability, allowing the model to more accurately reflect the data when observations vary in quality. This approach can effectively mitigate the impact of less reliable data, resulting in more precise and dependable models.

In federated WLS regression, the goal is to collaboratively estimate the global model parameter vector $\boldsymbol{\omega}$ by minimizing a global objective function across a federated network. This is framed as a global WLS estimation problem within the FL framework as

$$\begin{aligned} &\min_{\{\mathbf{w}_k\}} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}_k) \\ &\text{s.t. } \mathbf{w}_k = \mathbf{w},\ k \in \{1, 2, \cdots, K\}, \end{aligned} \tag{2}$$

where $\mathcal{J}_k(\mathbf{w}_k) = \|\mathbf{y}_k - \mathbf{X}_k\mathbf{w}_k\|^2_{\boldsymbol{\mathcal{W}}_k}$ is the local objective function for estimating $\boldsymbol{\omega}$ at client k, $\boldsymbol{\mathcal{W}}_k$ is the weight matrix specific to client k, $\mathbf{w}_k$ is the local model estimate at client k, and $\mathbf{w}$ serves as the global model estimate. The optimal solution to (2), which can be viewed as a federated version of distributed Pareto optimization [61, 62] for WLS regression, is stated as

$$\mathbf{w}^\star = \left(\sum_{k=1}^{K} \mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{X}_k\right)^{-1} \left(\sum_{k=1}^{K} \mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{y}_k\right). \tag{3}$$

To solve (2) within an FL framework, we utilize the ADMM algorithm [45]. Therefore, we express the augmented Lagrangian function corresponding to (2) as

$$\begin{aligned} \mathcal{L}(\mathbf{w}_k, \mathbf{w}, \mathbf{z}_k) &= \sum_{k=1}^{K} \mathcal{L}_k(\mathbf{w}_k, \mathbf{w}, \mathbf{z}_k) \\ &= \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{w}_k) + \langle \mathbf{w}_k - \mathbf{w}, \mathbf{z}_k \rangle + \frac{\rho}{2}\|\mathbf{w}_k - \mathbf{w}\|_2^2, \end{aligned} \tag{4}$$

where $\mathbf{z}_k$ is the Lagrange multiplier vector and and $\rho > 0$ is the penalty parameter. Consequently, we derive the recursive update equations at each client k and iteration number n as

$$\mathbf{z}_{k,n} = \mathbf{z}_{k,n-1} + \rho(\mathbf{w}_{k,n} - \mathbf{w}_n) \tag{5a}$$

$$\mathbf{w}_{k,n+1} = \widehat{\mathbf{w}}_k - \mathbf{N}_k(\mathbf{z}_{k,n} - \rho\mathbf{w}_n), \tag{5b}$$

along with the recursive update equation at the server as

$$\mathbf{w}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{w}_{k,n+1} + \rho^{-1}\mathbf{z}_{k,n}\right), \tag{6}$$

where we define

$$\mathbf{N}_k = \left(2\mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{X}_k + \rho\mathbf{I}\right)^{-1} \tag{7}$$

and

$$\widehat{\mathbf{w}}_k = 2\mathbf{N}_k\mathbf{X}_k^\intercal \boldsymbol{\mathcal{W}}_k \mathbf{y}_k. \tag{8}$$

In this solution, after performing local training, i.e., (5a) and (5b), each client shares its local estimate of $\mathbf{w}_{k,n+1} + \rho^{-1}\mathbf{z}_{k,n}$ with the server. The server then obtains the global estimate as in (6) and broadcasts it to all clients while the FL process continues.

2.2. Dual Variable Elimination

We posit that, in the recursions (5)-(6), it is necessary to transmit a combination of the primal and dual model parameter estimates to the server to enable the aggregation that produces the global model estimate. However, the dual update information can be integrated into the primal update by judiciously selecting the initial estimates and introducing a new local primal update at clients. Accordingly, we reformulate (5)-(6) as

$$\mathbf{w}_n = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}_{k,n} \tag{9a}$$

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k(2\mathbf{w}_n - \mathbf{w}_{n-1}) \tag{9b}$$

by initializing with $\mathbf{w}_{-1} = \mathbf{0}$, $\mathbf{z}_{k,-1} = \mathbf{0}$, and $\mathbf{w}_{k,0} = \widehat{\mathbf{w}}_k$, hence eliminating the Lagrange multipliers $\mathbf{z}_{k,n}$. The recursion begins with clients sharing their $\widehat{\mathbf{w}}_k$ with the server, which then aggregates them and broadcasts the resulting global model estimate to the clients. We further define $\mathbf{s}_n = 2\mathbf{w}_n - \mathbf{w}_{n-1}$ and modify (9b) as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\mathbf{s}_n. \tag{10}$$

A key advantage of (10) over (9b) is that the linear combination of the two most recent global model estimates, required for local model updates, is performed at the server rather than by individual clients. This can significantly reduce the impact of communication noise as the combination occurs before transmission to clients.

2.3. Communication Noise

Clients and the server often communicate via wireless channels, where both uplink and downlink channels are susceptible to noise. In the downlink, clients receive noisy versions of the aggregated model updates from the server. Specifically, at iteration n, client k receives $\tilde{\mathbf{s}}_{k,n} = \mathbf{s}_n + \boldsymbol{\zeta}_{k,n}$ where $\boldsymbol{\zeta}_{k,n}$ represents the downlink noise affecting the transmission. In the uplink, the server receives a noisy version of each client's local model update, i.e., $\tilde{\mathbf{w}}_{k,n+1} = \mathbf{w}_{k,n+1} + \boldsymbol{\eta}_{k,n}$ where $\boldsymbol{\eta}_{k,n}$ denotes the uplink noise for client k at iteration n. Considering the impact of communication noise and allowing client updates to occur before server aggregation, we obtain

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n} \tag{11a}$$

$$\mathbf{w}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\tilde{\mathbf{w}}_{k,n+1} \tag{11b}$$

3. Resource-efficient FL over Noisy Channels

In this section, we introduce RERCE-Fed, our proposed FL algorithm, that is both resource-efficient and robust to communication errors.

3.1. Client Participation

In (11), there is an implicit assumption that all clients participate in each global model update iteration. However, in FL, clients often have limited communication and computational resources. Therefore, requiring participation of all clients in every global update round can lead to significant drawbacks, such as prolonged convergence time or excessive resource utilization. To address this challenge, we enable the server to implement a mechanism known as random scheduling [1, 3]. This approach involves randomly selecting a subset of clients, denoted by $\mathcal{S}_n$, to participate in model aggregation at each iteration n. We consider the cardinality of this subset, $C = |\mathcal{S}_n|$, to be fixed throughout the FL process.

3.2. RERCE-Fed

In our proposed algorithm, during each global iteration n, the selected clients, $k \in \mathcal{S}_n$, receive $\tilde{\mathbf{s}}_{k,n}$ from the server and update their models. The server then receives $\tilde{\mathbf{w}}_{k,n+1}$ from these clients and aggregates them. Subsequently, it sends the most recent global update to a new set of selected clients in the next iteration. Clients not selected by the server in a given round retain their latest local update until they are selected again. Therefore, the recursions of the proposed RERCE-Fed algorithm are expressed as [56]

$$\mathbf{w}_{k,n+1} = \left(\mathbf{I} - a_{k,n}\rho\mathbf{N}_k\right)\mathbf{w}_{k,n} + a_{k,n}\rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n} \tag{12a}$$

$$\mathbf{w}_{n+1} = \frac{1}{C}\sum_{k=1}^{K} a_{k,n}\tilde{\mathbf{w}}_{k,n+1}, \tag{12b}$$

where $a_{k,n}$ is the indicator variable for random scheduling, with $a_{k,n} = 1$ when client k is selected by the server in iteration n (i.e., $k \in \mathcal{S}_n$) and $a_{k,n} = 0$ otherwise. We summarize RERCE-Fed in Algorithm 1.

3.3. RERCE-Fed with Continual Local Updates

Unlike the conventional random scheduling approach, described in section 3.1, where non-selected clients refrain from local updates and their latest model estimates are not integrated into the global aggregation process, we propose a new approach where all clients, regardless of their selection status, continually update their local model estimates during each iteration. This new approach can enhance overall performance without introducing any additional communication overhead or imposing any notable increase in computational load of clients or the server, as we will demonstrate later.

To implement this approach, clients store the most recent global model estimate received from the server, while the server holds the latest local model estimates received from all clients. Consequently, clients continually update their local models using the most recent global model estimate, and the server updates the global model using the latest local updates from all clients, irrespective of random scheduling. When a client is selected at iteration n, its latest local model estimate is synchronized at the server, and the global model estimate received from the server supersedes the previous version at the client.

Algorithm 1 RERCE-Fed.

Parameters: penalty parameter ρ, number of clients K, set of clients $\mathcal{S}$

Initialization: global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \widehat{\mathbf{w}}_k$

For $n = 1, \cdots$, *Until Convergence*

The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{s}_n$ to them.

Client Local Update:

If $k \in \mathcal{S}_n$

Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of $\mathbf{s}_n$, from the server.
Update the local model as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n}$$

Send $\mathbf{w}_{k,n+1}$ to the server.

EndIf

Aggregation at the Server:

The server receives $\tilde{\mathbf{w}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them via

$$\mathbf{w}_{n+1} = \frac{1}{C}\sum_{k\in\mathcal{S}_n}\tilde{\mathbf{w}}_{k,n+1}$$
$$\mathbf{s}_{n+1} = 2\mathbf{w}_{n+1} - \mathbf{w}_n$$

EndFor

Therefore, the recursions of the RERCE-Fed algorithm with continual local updates are given by [57]

$$\begin{aligned}\mathbf{w}_{k,n+1} &= (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} \\ &\quad + \rho\mathbf{N}_k\left[a_{k,n}\tilde{\mathbf{s}}_{k,n} + (1-a_{k,n})\tilde{\mathbf{s}}_{k,m}\right] \qquad (13a)\end{aligned}$$

$$\mathbf{w}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left[a_{k,n}\tilde{\mathbf{w}}_{k,n+1} + (1-a_{k,n})\tilde{\mathbf{w}}_{k,m}\right], \qquad (13b)$$

where $\tilde{\mathbf{s}}_{k,m}$ denotes the most recent global model estimate received from the server and stored in client k. This estimate is utilized when the client is not selected by the server. Additionally, $\tilde{\mathbf{w}}_{k,m}$ represents the most recent local model estimate associated with client k, which is stored at the server and utilized during iterations when the client is not chosen through random scheduling. Defining $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$, we can restate (13b) as

$$\mathbf{s}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left[a_{k,n}\tilde{\mathbf{t}}_{k,n+1} + (1-a_{k,n})\tilde{\mathbf{t}}_{k,m}\right]. \qquad (13c)$$

We summarize RERCE-Fed with continual local updates in Algorithm 2.

Remark 1: Our selective transmission of global model updates to only the chosen clients at each iteration addresses critical practical requirements in real-world FL deployments. This design prioritizes energy efficiency by ensuring that non-participating clients avoid unnecessary operations and communication overhead, conserving their resources. Such an approach aligns well with resource-aware environments, including those leveraging WiFi or LTE networks, where reducing redundant transmissions can also help lower operational costs for clients and system operators. In scenarios where all clients perform local updates regardless of their selection status, we consider a dedicated communication channel between the server and each client. This channel activates only when a client is selected, ensuring minimal resource usage while maintaining flexibility in communication protocols. This selective approach reflects real-world constraints, where clients often operate under limited energy, bandwidth, or computational resources, making it impractical to broadcast updates universally.

Remark 2: Although employing a selective transmission may diverge from conventional broadcast assumptions in FL literature, our proposed framework is intentionally adaptable. It can seamlessly operate in both selective and broadcast modes, enabling it to cater to a wide range of deployment contexts and use cases. By balancing resource efficiency with adaptability, this approach ensures robust applicability while addressing practical limitations often encountered in large-scale FL systems. Furthermore, the adoption of a hybrid strategy that incorporates periodic synchronization of the global model, such as every $\mathcal{T}$ iteration, can effectively mitigate the negative impact of outdated updates that may lead to model drift at clients. It can also enhance the convergence rate and promote fairness across clients. By striking a balance between maintaining model accuracy and optimizing resource efficiency, this strategy enhances the overall effectiveness of our approach. We will explore various downlink strategies through simulations in section 5.2.

4. Performance Analysis

In this section, we analyze the performance of RERCE-Fed theoretically. We establish the convergence of the iterates $\mathbf{w}_{k,n}$ in both mean and mean-square senses as $n \to \infty$, even in the presence of noisy communication links. Given that $\mathbf{w}_n$ represents the average of client estimates $\mathbf{w}_{k,n}$, its convergence follows accordingly.

To facilitate the analysis, we define the extended optimal global model as $\mathbf{w}_e^\star = \mathbf{1}_{2K} \otimes \mathbf{w}^\star$ and the vector containing the client local model estimates as

$$\mathbf{w}_{e,n} = \mathrm{col}\{\mathbf{w}_{1,n}, \cdots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \cdots, \mathbf{w}_{K,n-1}\},$$

where $\mathbf{1}_{2K}$ is the $2K \times 1$ vector of all ones, $\otimes$ is the Kronecker product, and $\mathrm{col}\{\cdot\}$ denotes column-wise stacking.

Substituting (12b) into (12a), the global recursion of the proposed algorithm can be stated as

$$\mathbf{w}_{e,n+1} = \mathcal{A}_n\mathbf{w}_{e,n} + \boldsymbol{\zeta}_n + \boldsymbol{\eta}_n, \qquad (14)$$

where

$$\mathcal{A}_n = \left[\begin{array}{c:c} \mathcal{A}_{n,1} & \mathcal{A}_{n,2} \\ \hdashline \mathbf{I} & \mathbf{0} \end{array}\right], \qquad (15)$$

and the extended noise vectors $\boldsymbol{\zeta}_n$ and $\boldsymbol{\eta}_n$ stack the vectors

$$a_{k,n}\rho\mathbf{N}_k\boldsymbol{\zeta}_{k,n} \qquad (16)$$

Algorithm 2 RERCE-Fed with continual local updates.

Parameters: penalty parameters ρ, number of clients K, set of clients $\mathcal{S}$

Initialization: global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \widehat{\mathbf{w}}_k$

For $n = 1, \cdots$, *Until Convergence*

The server randomly selects a subset $\mathcal{S}_n$ of its clients and sends the aggregated global model $\mathbf{s}_n$ to them.

Client Local Update:

If $k \in \mathcal{S}_n$

Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of $\mathbf{s}_n$, from the server.
Store the latest global model $\tilde{\mathbf{s}}_{k,m} = \tilde{\mathbf{s}}_{k,n}$.
Update the local model as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n}$$

Send $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$ to the server.

Else

Update the local model as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho\mathbf{N}_k)\mathbf{w}_{k,n} + \rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,m}$$

EndIf

Aggregation at the Server:

The server receives $\tilde{\mathbf{t}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them with $\tilde{\mathbf{t}}_{k,m}$, the stored local model estimates of the non-selected clients via

$$\mathbf{s}_{n+1} = \frac{1}{K}\sum_{k=1}^{K}\left[a_{k,n}\tilde{\mathbf{t}}_{k,n+1} + (1 - a_{k,n})\tilde{\mathbf{t}}_{k,m}\right]$$

EndFor

and

$$a_{k,n}\frac{\rho}{C}\mathbf{N}_k\sum_{j=1}^{K}\left(2a_{j,n-1}\boldsymbol{\eta}_{j,n-1} - a_{j,n-2}\boldsymbol{\eta}_{j,n-2}\right) \tag{17}$$

at their top halves, respectively, and zeros at their bottom halves. The value of $\mathcal{A}_n \in \mathbb{R}^{2LK\times 2LK}$ depends on the iteration number n as the server selects a random number of clients at each iteration. Its sub-matrices of size $LK \times LK$ are block matrices whose $L \times L$ blocks are calculated as

$$[\mathcal{A}_{n,1}]_{ii} = \mathbf{I} - a_{i,n}\rho\mathbf{N}_i + 2a_{i,n}a_{i,n-1}\frac{\rho}{C}\mathbf{N}_i \tag{18a}$$

$$[\mathcal{A}_{n,1}]_{ij} = 2a_{i,n}a_{j,n-1}\frac{\rho}{C}\mathbf{N}_i,\ i \neq j \tag{18b}$$

$$[\mathcal{A}_{n,2}]_{ij} = -a_{i,n}a_{j,n-2}\frac{\rho}{C}\mathbf{N}_i. \tag{18c}$$

To make the analysis tractable, we adopt the following assumptions:

A1: The communication noise vectors of both uplink and downlink, $\boldsymbol{\eta}_{k,n}$ and $\boldsymbol{\zeta}_{k,n}$ $\forall k, n$, are independently and identically distributed. In addition, they are independent of each other and all other stochastic variables.

A2: The random scheduling variables, $a_{k,n}$ $\forall k, n$, are independent and follow the same Bernoulli distribution with parameter $\bar{a} = \frac{C}{K}$, i.e., $a_{k,n} = 1$ with probability $\bar{a}$ and $a_{k,n} = 0$ with probability $1 - \bar{a}$.

4.1. Mean Convergence

Taking the expected value of both sides of (14), while considering A1-A2, yields

$$\mathbb{E}[\mathbf{w}_{e,n+1}] = \bar{\mathcal{A}}\,\mathbb{E}[\mathbf{w}_{e,n}], \tag{19}$$

where

$$\bar{\mathcal{A}} = \mathbb{E}[\mathcal{A}_n] = \left[\begin{array}{c:c} \mathbf{I} - \boldsymbol{O} + 2\boldsymbol{\mathcal{P}} & -\boldsymbol{\mathcal{P}} \\ \hdashline \mathbf{I} & \mathbf{0} \end{array}\right], \tag{20}$$

$$\boldsymbol{O} = \bar{a}\rho\ \mathrm{bdiag}\{\mathbf{N}_1, \cdots, \mathbf{N}_K\}, \tag{21a}$$

$$\boldsymbol{\mathcal{P}} = \frac{\bar{a}\rho}{K}\mathbf{1}_K^{\mathsf{T}} \otimes \mathrm{bcol}\{\mathbf{N}_1, \cdots, \mathbf{N}_K\}, \tag{21b}$$

and bcol$\{\cdot\}$ and bdiag$\{\cdot\}$ represent block column-wise stacking and block diagonalization, respectively. Given that the initial estimate of each client is deterministic, (19) can be recursively unfolded over n iterations as

$$\mathbb{E}[\mathbf{w}_{e,n+1}] = \bar{\mathcal{A}}^n\mathrm{col}\{\widehat{\mathbf{w}}_e, \mathbf{0}\}, \tag{22}$$

where $\widehat{\mathbf{w}}_e = \mathrm{col}\{\widehat{\mathbf{w}}_1, \cdots, \widehat{\mathbf{w}}_K\}$.

Proposition 1: The RERCE-Fed algorithm is unbiased and converges in the mean to the optimal solution of the WLS regression problem, (3), i.e.,

$$\lim_{n\to\infty}\mathbb{E}\left[\mathbf{w}_{e,n}\right] = \mathbf{w}_e^{\star}. \tag{23}$$

Proof: See Appendix A.

4.2. Mean-Square Convergence

Let us define the deviation vector as $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_{e,n} - \mathbf{w}_e^{\star}$. Since $\mathcal{A}_n$ is block right-stochastic, i.e., its block rows add up to the identity matrix, we have $\mathcal{A}_n\mathbf{w}_e^{\star} = \mathbf{w}_e^{\star}$. Therefore, by defining the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ as $\|\tilde{\mathbf{w}}_{e,n}\|_{\boldsymbol{\Sigma}}^2 = \tilde{\mathbf{w}}_{e,n}^{\mathsf{T}}\boldsymbol{\Sigma}\tilde{\mathbf{w}}_{e,n}$, where $\boldsymbol{\Sigma}$ is an arbitrary positive semi-definite matrix, we obtain the variance relation as

$$\begin{aligned}&\mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n+1}\|_{\boldsymbol{\Sigma}}^2\right] \\ &= \mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n}\|_{\boldsymbol{\Sigma}'}^2\right] + \mathbb{E}\left[\boldsymbol{\zeta}_n^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\zeta}_n\right] + \mathbb{E}\left[\boldsymbol{\eta}_n^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\eta}_n\right],\end{aligned} \tag{24}$$

where the cross terms vanish as they are independent and uncorrelated with one another and all other variables. The matrix $\boldsymbol{\Sigma}'$ is given by

$$\boldsymbol{\Sigma}' = \mathbb{E}\left[\mathcal{A}_n^{\mathsf{T}}\boldsymbol{\Sigma}\mathcal{A}_n\right]. \tag{25}$$

We can describe the relationship between $\boldsymbol{\Sigma}'$ and $\boldsymbol{\Sigma}$ as $\boldsymbol{\sigma}' = \mathrm{bvec}\{\boldsymbol{\Sigma}'\} = \boldsymbol{\mathcal{Q}}\boldsymbol{\sigma}$, where $\boldsymbol{\mathcal{Q}} = \mathbb{E}[\mathcal{A}_n^{\mathsf{T}} \otimes_b \mathcal{A}_n^{\mathsf{T}}]$ and $\boldsymbol{\sigma} = \mathrm{bvec}\{\boldsymbol{\Sigma}\}$. Here, $\otimes_b$ is the block Kronecker product (Tracy–Singh product [63]) and bvec$\{\cdot\}$ denotes the block vectorization operation [64]. We evaluate $\boldsymbol{\mathcal{Q}}$ in Appendix B.

Using the relationship between the matrix trace operator, denoted by tr(·), and the block Kronecker product, we evaluate the second term on the right-hand side (RHS) of (24) as

$$\mathbb{E}[\boldsymbol{\zeta}_n^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\zeta}_n] = \mathrm{tr}(\mathbb{E}[\boldsymbol{\zeta}_n\boldsymbol{\zeta}_n^{\mathsf{T}}]\boldsymbol{\Sigma}) = \boldsymbol{\phi}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{26}$$

where $\boldsymbol{\phi} = \mathrm{bvec}\{\mathbb{E}[\boldsymbol{\zeta}_n\boldsymbol{\zeta}_n^{\mathsf{T}}]\}$,

$$\mathbb{E}[\boldsymbol{\zeta}_n\boldsymbol{\zeta}_n^{\mathsf{T}}] = \bar{a}\rho^2\mathrm{bdiag}\{\sigma_{\zeta_1}^2\mathbf{N}_1^2, \cdots, \sigma_{\zeta_K}^2\mathbf{N}_K^2, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{K}\}, \tag{27}$$

and $\sigma_{\zeta_k}^2$ is the variance of the downlink noise for client k.

Following a similar procedure, we evaluate the third term on the RHS of (24) as

$$\mathbb{E}[\boldsymbol{\eta}_n^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\eta}_n] = \mathrm{tr}(\mathbb{E}[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^{\mathsf{T}}]\boldsymbol{\Sigma}) = \boldsymbol{\varphi}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{28}$$

where $\boldsymbol{\varphi} = \mathrm{bvec}\{\mathbb{E}[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^{\mathsf{T}}]\}$,

$$\mathbb{E}[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^{\mathsf{T}}] = \frac{5\rho^2}{K^2}\sum_{k=1}^{K}\sigma_{\eta_k}^2\mathrm{bdiag}\{\mathbf{N}_1^2, \cdots, \mathbf{N}_K^2, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{K}\}, \tag{29}$$

and $\sigma_{\eta_k}^2$ is the variance of the uplink noise for client k. Note that in (29), $\mathbb{E}[a_{i,n}a_{j,n}] \ll 1\ \forall i \neq j$, and can be neglected.

Utilizing the above results, we can write the global recursion for the weighted mean-squared error (MSE) of RERCE-Fed as

$$\begin{aligned}&\mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right]\\ &= \mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\mathcal{Q}}\boldsymbol{\sigma}\}}\right] + \boldsymbol{\phi}^{\mathsf{T}}\boldsymbol{\sigma} + \boldsymbol{\varphi}^{\mathsf{T}}\boldsymbol{\sigma}.\end{aligned} \tag{30}$$

By defining $\boldsymbol{\psi} = \boldsymbol{\phi} + \boldsymbol{\varphi}$ and iterating (30) backward to $n = 1$, we obtain

$$\begin{aligned}&\mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,n+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right]\\ &= \mathbb{E}\left[\|\tilde{\mathbf{w}}_{e,1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\mathcal{Q}}^n\boldsymbol{\sigma}\}}\right] + \boldsymbol{\psi}^{\mathsf{T}}\sum_{i=0}^{n-1}\boldsymbol{\mathcal{Q}}^i\boldsymbol{\sigma}.\end{aligned} \tag{31}$$

Using the Jordan canonical form of $\boldsymbol{\mathcal{Q}}$, we have

$$\boldsymbol{\mathcal{Q}}^i = \boldsymbol{\mathcal{U}}\boldsymbol{\mathcal{J}}^i\boldsymbol{\mathcal{U}}^{-1} = \sum_{\ell=1}^{4L^2K^2}\mu_\ell^i\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\mathsf{T}}, \tag{32}$$

where μ_ℓ, $\boldsymbol{u}_\ell$ and $\boldsymbol{v}_\ell$ denote the ℓth eigenvalue of $\boldsymbol{\mathcal{Q}}$ and its corresponding right and left eigenvectors, respectively. In Appendix C, we show that the spectral radius of $\boldsymbol{\mathcal{Q}}$ is one with the geometric and algebraic multiplicity of L^2, i.e., $\mu_\ell = 1\ \forall \ell \in \{1, \ldots, L^2\}$.

Proposition 2: $\forall \ell \in \{1, \ldots, L^2\}$, we have

$$\boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\mathsf{T}}\boldsymbol{\sigma} = 0. \tag{33}$$

Proof: See Appendix D.

4.3. Steady-State Mean-Square Error

Setting $\boldsymbol{\sigma} = \mathrm{bvec}\{\mathbf{I}_{2KL}\}$, letting $n \to \infty$ on both sides of (31), and using Proposition 1, we compute the steady-state MSE of RERCE-Fed, denoted by $\mathcal{E}$, as

$$\begin{aligned}\mathcal{E} &= \lim_{n\to\infty}\mathbb{E}\left[\tilde{\mathbf{w}}_{e,n}^{\mathsf{T}}\tilde{\mathbf{w}}_{e,n}\right]\\ &= \underbrace{\tilde{\mathbf{w}}_{e,1}^{\mathsf{T}}\boldsymbol{\Sigma}_\infty\tilde{\mathbf{w}}_{e,1}}_{\mathcal{E}_\nu} + \underbrace{\sum_{\ell=L^2+1}^{4L^2K^2}(1-\mu_\ell)^{-1}\boldsymbol{\psi}^{\mathsf{T}}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\mathsf{T}}\boldsymbol{\sigma}}_{\mathcal{E}_\psi},\end{aligned} \tag{34}$$

where

$$\begin{aligned}\boldsymbol{\Sigma}_\infty &= \mathrm{bvec}^{-1}\left\{\boldsymbol{\mathcal{Q}}^\infty\boldsymbol{\sigma}\right\}\\ &= \mathrm{bvec}^{-1}\left\{\sum_{\ell=1}^{L^2}\boldsymbol{u}_\ell\boldsymbol{v}_\ell^{\mathsf{T}}\mathrm{bvec}\{\mathbf{I}_{2KL}\}\right\}.\end{aligned} \tag{35}$$

Remark 3: The observation noise $\boldsymbol{\nu}_k$ in (1) and the initial client estimates $\mathbf{w}_{k,0}$ affect the first term on the RHS of (34), $\mathcal{E}_\nu$, resulting in a noise floor. Additionally, scheduling parameters, including the client participation probability and the number of participating clients in each iteration, along with the penalty parameter, the number of clients, the local data at each client, and the variance of noise in the uplink and downlink communications impact the second term on the RHS of (34), $\mathcal{E}_\psi$. We will explore these factors further through simulations in section 5.3.

5. Simulation Results

In this section, we conduct several numerical experiments to evaluate the performance of the proposed algorithm and validate our theoretical findings. We consider a federated network consisting of K clients. Each client has non-IID data $\{\mathbf{X}_k, \mathbf{y}_k\}$, where the entries of $\mathbf{X}_k$ are drawn from a normal distribution $\mathcal{N}(\mu_k, \sigma_k^2)$ with $\mu_k \in \mathcal{U}(-0.5, 0.5)$ and $\sigma_k^2 \in \mathcal{U}(0.5, 1.5)$. Here, $\mathcal{U}(\cdot,\cdot)$ denotes a uniform distribution with the specified lower and upper bounds. The local data size for each client is randomly selected from a uniform distribution, i.e., $d_k \in \mathcal{U}(50, 90)$. We set the weight matrices at each client k to the inverse covariance matrix of $\mathbf{y}_k$, i.e., $\boldsymbol{\mathcal{W}}_k = \mathbb{E}[(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])^{\mathsf{T}}]^{-1}$.

The local data is related as per (1) given the model parameter vector $\boldsymbol{\omega}$ with its entries drawn from a normal distribution $\mathcal{N}(0, 1)$. The observation noise $\boldsymbol{\nu}_k$ for each client is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$. The additive noise in both the uplink and downlink channels is zero-mean IID Gaussian with variances $\sigma_{\eta_k}^2$ and $\sigma_{\zeta_k}^2$, respectively. We set the penalty parameter as $\rho = 1$. The server randomly selects a subset of C clients with uniform probability in each iteration. We evaluate the algorithm performance on the client side via the normalized MSE (NMSE) defined at iteration n as

$$\frac{1}{K}\sum_{k=1}^{K}\frac{\|\mathbf{w}_{k,n} - \mathbf{w}^\star\|_2^2}{\|\mathbf{w}^\star\|_2^2}. \tag{36}$$

We average the NMSE learning curves over 100 independent trials to obtain our simulation results.

We present the results of our numerical experiments in three subsections. In the first subsection, we evaluate the performance of the proposed RERCE-Fed algorithm in comparison

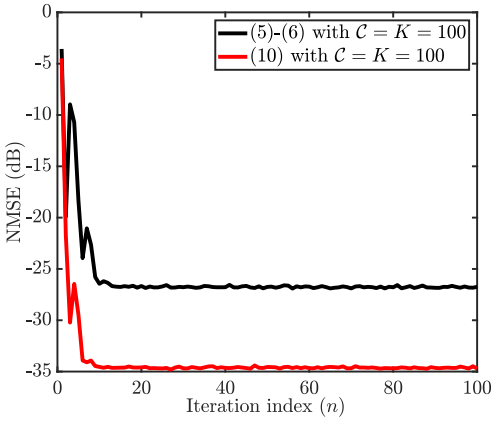

Figure 1: NMSE of (5)-(6) and (10) for $C = K = 100$.

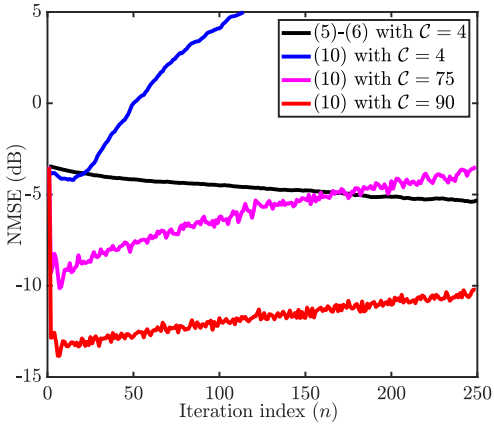

Figure 2: NMSE of (5)-(6) with $C = 4$ and (10) with $C \in \{4, 75, 90\}$.

to its predecessors. In the second subsection, we evaluate the performance of RERCE-Fed with continual local updates and compare it with RERCE-Fed without continual local updates. In the last subsection, we corroborate our theoretical findings by demonstrating mean convergence and comparing theoretical predictions with simulation results, confirming the accuracy of our theoretical expression for MSE of RERCE-Fed.

5.1. Performance of RERCE-Fed

In our first experiment, we simulate the algorithms described by (5)-(6) and (10) to solve the considered WLS problem. We run these simulations with $K = 100$ clients, model parameter vector size of $L = 128$, and link noise variance of $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 6.25 \times 10^{-4}$. All clients participate in the FL process, i.e., $C = K = 100$. The corresponding learning curves are shown in Figure 1. We observe that (10) exhibits a 7dB improvement over (5)-(6) in the presence of noisy communication links when all clients are involved in each iteration of the FL process.

In our second experiment, we examine the performance of (5)-(6) and (10) under similar conditions as the first experiment, but with the server randomly selecting a subset of clients to participate in each iteration. We simulate (5)-(6) with $C = 4$ and (10) with $C \in \{4, 75, 90\}$. The corresponding learning curves are shown in Figure 2. Unlike the first experiment, Figure 2 illustrates that (10) fails to converge due to error accumulation, even when a majority of clients participate in every FL round, as observed for $C \in \{75, 90\}$. Additionally, the performance of (5)-(6) degrades significantly when only a small subset of clients participate in each FL round. Consequently, both (5)-(6) and (10) exhibit an inability to cope with additive noise in communication links when the server selects only a subset of clients in each iteration.

In our third experiment, we assess the performance of FedAvg [1], FedADMM [46] and the proposed RERCE-Fed algorithm in the presence of link noise and random client scheduling by the server, aiming to enhance communication efficiency. We

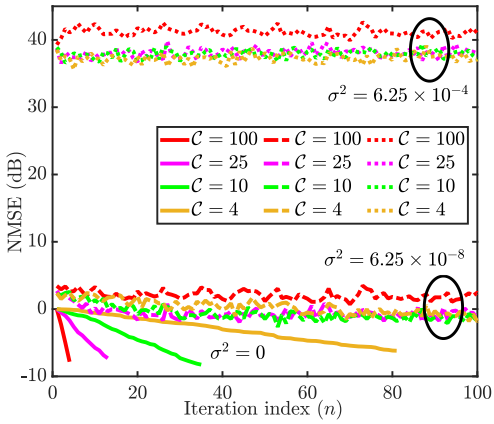

Figure 3: NMSE of FedADMM [46] for different numbers of participating clients $C \in \{4, 10, 25, 100\}$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{0, 6.25 \times 10^{-8}, 6.25 \times 10^{-4}\}$.

simulate RERCE-Fed with $K = 100$ clients, $L = 128$, link noise variance of $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 6.25 \times 10^{-4}$, and different numbers of participating clients $C \in \{4, 10, 25\}$. Additionally, we simulate FedAvg [1], FedADMM [46] with a setup similar to that of RERCE-Fed. The corresponding learning curves are depicted in Figures 4 and 3. As shown in Figure 4, RERCE-Fed exhibits robustness against communication noise, even when only a subset of clients participate in each FL round, in contrast to the results observed with (5)-(6) and (10) in the second experiment. In addition, Figure 4 clearly illustrates that the FedAvg algorithm diverges under similar conditions. As shown in Figure 3, the FedADMM algorithm also fails to converge, with the associated error reaching around 40dB even under low noise levels. This outcome is expected, as neither FedAvg nor FedADMM is designed to be robust against communication noise. Furthermore, additional experiments (not shown here to maintain brevity) indicate that FedAND [47] is also considerably sensi-

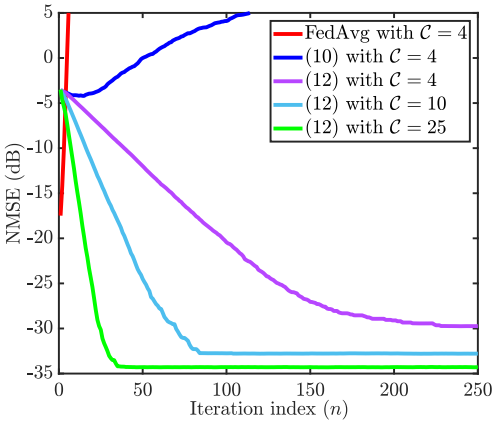

Figure 4: NMSE of FedAvg [1] and (10) with $C = 4$ and RERCE-Fed (12) for different numbers of participating clients $C \in \{4, 10, 25\}$.

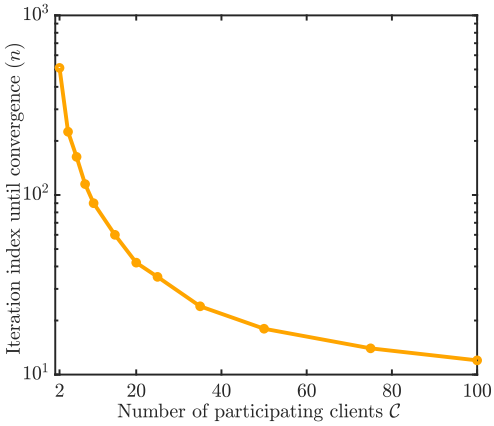

Figure 5: Number of FL iterations until convergence for RERCE-Fed (12) with different numbers of clients participating at each iteration, $2 \leq C \leq 100$.

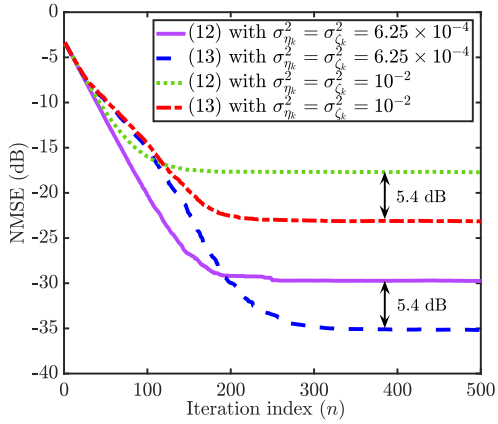

Figure 6: NMSE of RERCE-Fed (12) and RERCE-Fed with continual local updates (13) with $C = 4$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

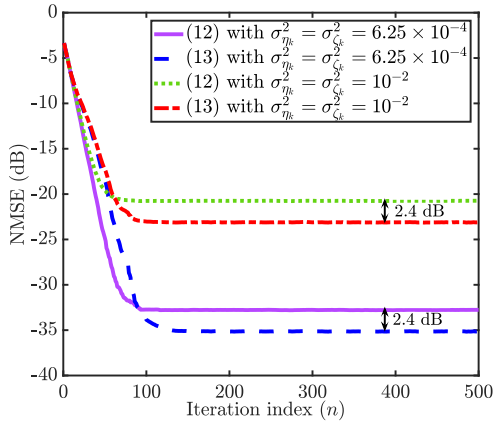

Figure 7: NMSE of RERCE-Fed (12) and RERCE-Fed with continual local updates (13) with $C = 10$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

tive to link noise. In contrast, RERCE-Fed converges under the same conditions, highlighting its effectiveness.

Another important observation from Figure 4 is the trade-off between the number of participating clients C and the performance and convergence rate of RERCE-Fed. Specifically, increasing the number of participating clients leads to faster convergence and lower NMSE. However, this benefit diminishes once $C \geq 10$, as the performance of RERCE-Fed with $C \geq 10$ approaches that of the scenario where all clients participate in each FL round. This implies that RERCE-Fed can achieve accurate model parameter estimation while making more efficient use of available communication resources, even in the presence of noisy communication links. Note that results for RERCE-Fed (12) with full client participation, i.e., $C = 100$, are similar to that of (10) with $C = K = 100$ in Figure 1.

Additionally, we analyze the number of active communication links, focusing specifically on uplink communication until convergence for RERCE-Fed (12), across varying numbers of participating clients $2 \leq C \leq 100$, as depicted in Figure 5. The results indicate that while increasing the number of participating clients C initially accelerates convergence, the rate of improvement diminishes as C grows. This observation highlights a trade-off between enhancing computational efficiency and managing the communication overhead imposed on the clients.

5.2. *Performance of RERCE-Fed with Continual Local Updates*

In our fourth experiment, we compare the performance of RERCE-Fed with and without continual local updates by plotting their corresponding learning curves given different numbers of participating clients and link noise variances. We

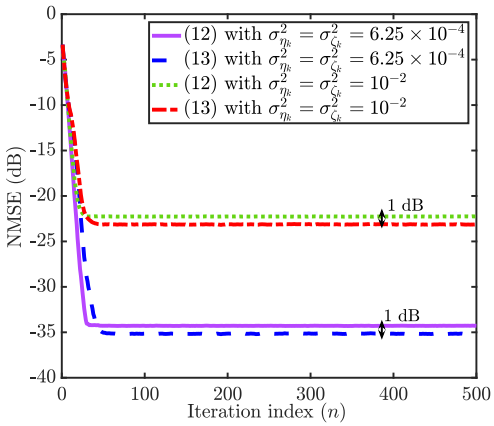

Figure 8: NMSE of RERCE-Fed (12) and RERCE-Fed with continual local updates (13) with $C = 25$ and different uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

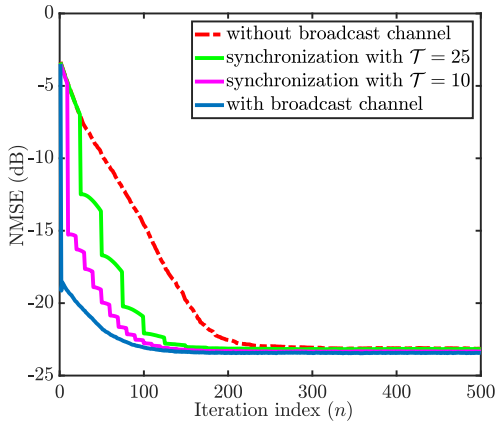

Figure 9: NMSE of RERCE-Fed with continual local updates (13) with $C = 4$, uplink and downlink noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 10^{-2}$, and different downlink strategies.

present the results in Figures 6-8. The number of clients selected at each iteration is $C \in \{4, 10, 25\}$, and the link noise variances are $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$. From these figures, we observe that RERCE-Fed with continual local updates consistently exhibits robustness against communication noise, even when only a small subset of clients participate in every FL round. In addition, RERCE-Fed with continual local updates significantly outperforms its counterpart without continual local updates in all considered scenarios. These results indicate that allowing clients to continually update their local models, even when not selected by the server, leads to a substantial improvement in steady-state NMSE. Moreover, it can be seen that the extent of improvement diminishes as the number of participating clients C increases. Additionally, as anticipated, increasing the link noise variance results in performance degradation.

Another observation from Figures 6-8 is that RERCE-Fed with continual local updates (13) initially converges at a slightly slower rate compared to RERCE-Fed without continual local updates (12), especially when the number of participating clients C is small. This difference stems from the nature of continual local updates, where clients may perform updates using older versions of the global model, and the server aggregates the global model using less recent versions of the local models of non-selected clients. While this approach introduces additional variability in the updates, potentially slowing the convergence slightly, it effectively leverages the computational resources available at non-selected clients. This trade-off highlights the enhanced resource utilization offered by continual local updates, which improve overall system performance despite a marginal reduction in convergence rate.

Furthermore, we investigate various downlink strategies as discussed in Remark 2, including the baseline scenario without a broadcast channel, downlink channels with periodic synchronization intervals of $\mathcal{T} \in \{10, 25\}$ iterations, and the broadcast channel as the best-case scenario. We present the results in Figure 9. We observe that as the synchronization period decreases, the convergence rate improves. However, this enhancement comes at the cost of increased communication resource usage on the client side.

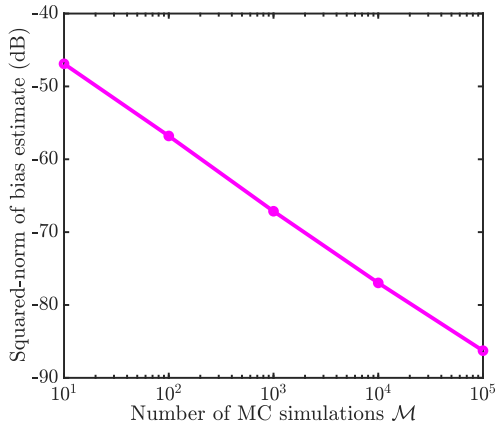

Figure 10: Squared-norm of bias estimate of RERCE-Fed (12) with $K = 6$, $L = 6$, $C = 3$, and $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 10^{-4}$ for different numbers of MC runs $\mathcal{M} \in \{10, 10^2, 10^3, 10^4, 10^5\}$.

5.3. *Comparison of Theory and Experiment*

In our fifth experiment, we demonstrate the mean convergence of RERCE-Fed. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$, and link noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} = 10^{-4}$. In addition, the server randomly selects $C = 3$ clients in each iteration. In Figure 10, we plot the squared ℓ_2-norm of the global model parameter bias estimated for different numbers of Monte Carlo (MC) runs, denoted by $\mathcal{M}$, specifically

$$\frac{1}{L}\|\frac{1}{\mathcal{M}}\sum_{i=1}^{\mathcal{M}} \mathbf{w}_n^i - \mathbf{w}^\star\|_2^2,$$

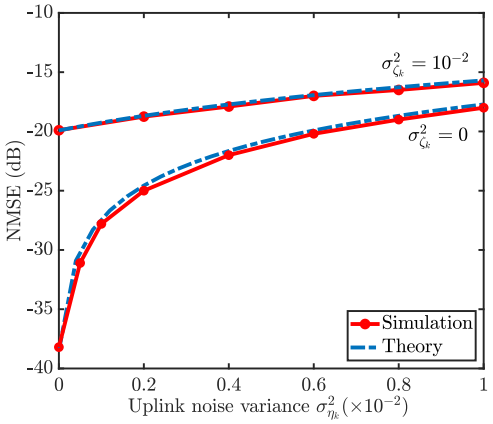

Figure 11: NMSE of RERCE-Fed (12) with $C = 3$ for different uplink noise variances.

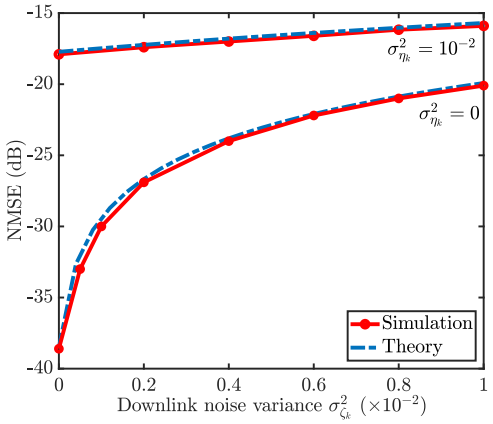

Figure 12: NMSE of RERCE-Fed (12) with $C = 3$ for different downlink noise variances.

where $\mathbf{w}_n^i$ is the global model parameter estimate of the ith MC run. We can observe from Figure 10 that the squared-norm of the multivariate bias estimate decreases log-linearly with the number of MC runs, indicating the unbiasedness shown in section 4.1.

In our sixth experiment, we validate the accuracy of our theoretical expression for the steady-state MSE of RERCE-Fed in (35) and explore the impact of varying uplink and downlink noise variances, $\sigma^2_{\eta_k}$ and $\sigma^2_{\zeta_k}$, on performance. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$ parameters, and different values for uplink and downlink noise variances. The server randomly selects $C = 3$ clients in each iteration. We present the theoretical predictions of steady-sate NMSE using (34) alongside the corresponding experimental values in Figures 11-12 as functions of $\sigma^2_{\eta_k}$, $\sigma^2_{\zeta_k}$, and C. The results show a close alignment between theory and experiment. Furthermore, we observe an upward trend in the steady-state NMSE as either uplink or downlink noise variance increases.

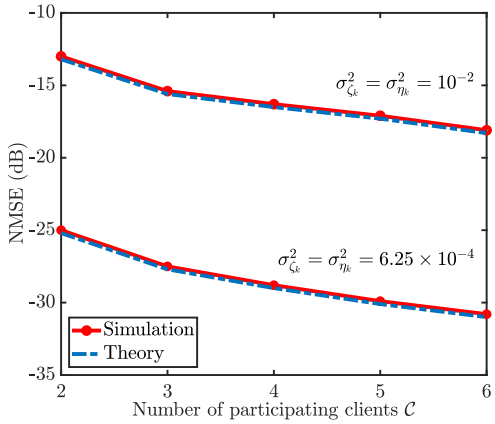

Figure 13: NMSE of RERCE-Fed (12) for different numbers of participating clients $C \in \{2, 3, 4, 5, 6\}$ and different link noise variances.

The uplink and downlink noise variances exhibit distinct effects depending on the number of participating clients C. While the error induced by uplink noise remains constant, the impact of downlink noise intensifies with an increasing number of participating clients. This observation is consistent with the intuition that averaging the model parameter estimates at the server can mitigate the adverse effect of uplink noise on the performance of RERCE-Fed. However, changing C also affects $\boldsymbol{Q}$. To obtain a better understanding, we investigate the impact of C on the performance of RERCE-Fed in our final experiment, where we consider $K = 6$, $L = 6$, link noise variances $\sigma^2_{\eta_k} = \sigma^2_{\zeta_k} \in \{6.25 \times 10^{-4}, 10^{-2}\}$, and $C \in \{2, 3, 4, 5, 6\}$. The results presented in Figure 13 illustrate that increasing the number of participating clients improves the performance of RERCE-Fed. However, this improvement comes at the cost of higher resource utilization on the client side, which can be limiting in real-world FL scenarios.

6. Discussion

In this section, we discuss our key findings, their potential impact, associated limitations, and future research directions.

The RERCE-Fed algorithm addresses the WLS regression problem while enhancing the communication efficiency of FL without imposing any additional communication or computational burden on clients. This aligns with the foundational goal of FL research, which is to minimize computational demands on clients while avoiding the need for any additional mechanism often required in existing alternative approaches.

Our focus on the WLS regression problem is deliberate. While related works such as [54] offer generalized formulations, our approach tackles the specific challenges posed by random client participation in noisy FL settings. Concentrating on the WLS problem enables us to guarantee mean convergence for both global and local models to the optimal solution, an assurance not typically provided in more generalized

formulations. Moreover, this focus allows for a novel mean-square convergence analysis and a closed-form expression for the mean-square error, which are central contributions of this work. The relevance of WLS extends to numerous linear-model applications, particularly in maximum likelihood or maximum a posteriori estimation problems where error variance is non-constant. Although extending our methodology to encompass broader strongly convex objective functions presents an exciting avenue for future exploration, our current scope prioritizes rigorous analytical treatment and practical applicability.

To facilitate our theoretical analysis, we adopted a random client scheduling mechanism that ensures a balance between fairness and simplicity in client selection and participation while leveraging the tractable statistical properties of the discrete uniform processes. While this approach may not lead to the fastest convergence, it provides a robust foundation for integrating more advanced client selection and scheduling mechanisms. For example, client selection probabilities can be customized to prioritize specific clients based on importance or other relevant criteria, thereby enhancing both efficiency and accuracy. Additionally, advanced techniques, such as diverse client selection [65] and lazy model aggregation [66], hold promise for optimizing communication efficiency and convergence rate. Incorporating such techniques into our system model is an important future research direction to enhance the practicality and robustness of our algorithm.

Another key direction for future research is conducting a comprehensive non-asymptotic convergence analysis of the proposed algorithm. Such an analysis will provide stronger theoretical guarantees and yield deeper insights into the behavior of our algorithm across diverse FL settings. Furthermore, we plan to investigate the effects of various adversarial attacks, such as model-poisoning, data-poisoning, and label-poisoning, on the performance of RERCE-Fed. By examining the impact of these attacks, assessing the algorithm's resilience, and devising effective defense mechanisms, we aim to further enhance the robustness and practical applicability of RERCE-Fed in real-world FL settings.

Balancing memory usage and performance presents a critical trade-off in resource-constrained settings. Storing the global model locally enhances performance by enabling continual updates but comes at the expense of increased memory demands. Strategies such as compressing the global model, using quantized updates, or leveraging external storage solutions offer viable approaches to mitigate this challenge. Future extensions of this work can explore modifications, such as limiting local updates to a subset of model parameters (e.g., via partial-sharing [19, 20]), enabling an optimal balance between memory efficiency and performance while enhancing the practicality and scalability of our approach.

7. Conclusion

We proposed RERCE-Fed, an FL algorithm designed to effectively reduce communication load while maintaining robustness against additive communication noise or errors. By employing ADMM to solve the WLS problem, we introduced a new local model update at the clients. This innovative solution mitigates the effects of communication noise without imposing additional computational burden on clients. Furthermore, we improved the communication efficiency by randomly selecting a subset of clients to participate in each learning round. To further optimize performance, we enabled non-selected clients to continue with their local updates, resulting in a modified version of RERCE-Fed. Our theoretical analysis confirmed the convergence of RERCE-Fed in both mean and mean-square sense, even with random client scheduling and communication over noisy communication links. In addition, we derived a closed-form expression for the steady-state MSE of the RERCE-Fed algorithm. Comprehensive numerical analysis substantiated our theoretical findings and confirmed the accuracy of our theoretical predictions.

Appendix A. Proof of Proposition 1

Considering (20) and (21), the matrix $\bar{\mathcal{A}}$ is right-stochastic, as its block rows add up to the identity matrix, and has a spectral radius of 1 with the geometric and algebraic multiplicity of L [52, Lemma 6(a)]. Therefore, using the Jordan canonical form $\bar{\mathcal{A}} = \mathbf{U}\mathbf{J}\mathbf{U}^{-1}$, we have

$$\bar{\mathcal{A}}^n = \mathbf{U}\mathbf{J}^n\mathbf{U}^{-1} \tag{A.1}$$

and, as $n \to \infty$, we obtain

$$\bar{\mathcal{A}}^\infty = \sum_{i=1}^{L} \mathbf{u}_i \mathbf{v}_i^\mathsf{T}. \tag{A.2}$$

where $\mathbf{u}_i$ and $\mathbf{v}_i^\mathsf{T}$ denote the ith right and left eigenvectors of matrix $\bar{\mathcal{A}}$ corresponding to the eigenvalue 1, respectively.

The L dominant right eigenvectors (corresponding to the eigenvalue 1) have the form

$$\mathbf{u}_i = \mathrm{col}\{\boldsymbol{\epsilon}_i, \cdots, \boldsymbol{\epsilon}_i\} \;\; \forall i \in \{1, \cdots, L\},$$

where $\boldsymbol{\epsilon}_i$ denotes an L-dimensional vector with one in its ith entry and zero in all other entries [52, Lemma 6(b)]. Additionally, the L dominant left eigenvectors (corresponding to the eigenvalue 1) satisfy

$$\mathbf{v}_i^\mathsf{T} \bar{\mathcal{A}} = \mathbf{v}_i^\mathsf{T} \;\; \forall i \in \{1, \cdots, L\}$$

that results in

$$\mathbf{v}_{i,1}^\mathsf{T}(\mathbf{I} - \boldsymbol{O} + 2\boldsymbol{\mathcal{P}}) + \mathbf{v}_{i,2}^\mathsf{T} = \mathbf{v}_{i,1}^\mathsf{T} \tag{A.3a}$$

$$\mathbf{v}_{i,1}^\mathsf{T}\boldsymbol{\mathcal{P}} + \mathbf{v}_{i,2}^\mathsf{T} = \mathbf{0}, \tag{A.3b}$$

where $\mathbf{v}_i$ can be written as $\mathrm{col}\{\mathbf{v}_{i,1}, \mathbf{v}_{i,2}\}$. Therefore, considering $\mathbf{v}_i^\mathsf{T}\mathbf{u}_i = 1$, we have

$$\mathbf{v}_{i,1}^\mathsf{T}(\mathbf{I} - \boldsymbol{O} + \boldsymbol{\mathcal{P}}) = \mathbf{v}_{i,1}^\mathsf{T} \tag{A.4a}$$

$$\mathbf{v}_{i,1}^\mathsf{T}(\mathbf{I} - \boldsymbol{\mathcal{P}})\mathbf{u}_{i,1} = 1, \tag{A.4b}$$

where $\mathbf{u}_{i,1}$ contains the first KL entries of $\mathbf{u}_i$.

Following the same procedure as in [52, Lemma 6(c)], $\mathbf{v}_{i,1}\ \forall i \in \{1,\cdots,L\}$ can be determined from (A.4). Subsequently, $\mathbf{v}_{i,2}$ can be computed using (A.3b). Hence, we have

$$\mathbf{v}_{i,1}^{\intercal} = \frac{\boldsymbol{\epsilon}_i^{\intercal}}{2}\left(\sum_{k=1}^{K}\mathbf{X}_k^{\intercal}\boldsymbol{\curlyvee}\boldsymbol{\mathcal{W}}_k\mathbf{X}_k\right)^{-1}\left[\mathbf{N}_1^{-1},\cdots,\mathbf{N}_K^{-1}\right] \tag{A.5a}$$

$$\mathbf{v}_{i,2}^{\intercal} = -\mathbf{v}_{i,1}^{\intercal}\boldsymbol{\mathcal{P}}\ \ \forall i \in \{1,\cdots,L\}. \tag{A.5b}$$

Consequently, in view of [52, Proposition 4], taking the limit on both sides of (22) leads to

$$\lim_{n\to\infty}\mathbb{E}\left[\mathbf{w}_{e,n}\right] = \sum_{i=1}^{L}\mathbf{u}_i\mathbf{v}_{i,1}^{\intercal}\widehat{\mathbf{w}}_e \tag{A.6}$$

$$= \sum_{i=1}^{L}\mathbf{u}_i\boldsymbol{\epsilon}_i^{\intercal}\left[\left(\sum_{k=1}^{K}\mathbf{X}_k^{\intercal}\boldsymbol{\curlyvee}\boldsymbol{\mathcal{W}}_k\mathbf{X}_k\right)^{-1}\sum_{k=1}^{K}\mathbf{X}_k^{\intercal}\boldsymbol{\curlyvee}\boldsymbol{\mathcal{W}}_k\mathbf{y}_k\right] = \mathbf{w}_e^{\star}.$$

Therefore, the RERCE-Fed algorithm is unbiased and the proof for mean convergence is complete.

Appendix B. Evaluation of $\boldsymbol{\mathcal{Q}}$

Let us define

$$\boldsymbol{\mathcal{A}}_n = \begin{bmatrix} \mathbf{A}_{1,1,n} & \mathbf{A}_{1,2,n} & \cdots & \mathbf{A}_{1,2K,n} \\ \mathbf{A}_{2,1,n} & \mathbf{A}_{2,2,n} & \cdots & \mathbf{A}_{2,2K,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{2K,1,n} & \mathbf{A}_{2K,2,n} & \cdots & \mathbf{A}_{2K,2K,n} \end{bmatrix},$$

where

$$\mathbf{A}_{i,j,n} = \begin{cases} \mathbf{I}_L - a_{i,n}\rho\mathbf{N}_i & i = j \in \{1,\ldots,K\} \\ \quad +2a_{i,n}a_{i,n-1}\frac{\rho}{C}\mathbf{N}_i & \\ 2a_{i,n}a_{j,n-1}\frac{\rho}{C}\mathbf{N}_i & i \neq j \in \{1,\ldots,K\} \\ -a_{i,n}a_{j-K,n-2}\frac{\rho}{C}\mathbf{N}_i & i \in \{1,\ldots,K\}, \\ & j \in \{K+1,\ldots,2K\} \\ \mathbf{I}_L & i \in \{K+1,\ldots,2K\}, \\ & j \in \{1,\ldots,K\} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Thus, $\boldsymbol{\mathcal{Q}}^{\intercal}$ is given by

$$\mathbb{E}\left[\boldsymbol{\mathcal{A}}_n \otimes_b \boldsymbol{\mathcal{A}}_n\right] = \mathbb{E}\begin{bmatrix} \mathbf{A}_{1,1,n}\otimes_b\boldsymbol{\mathcal{A}}_n & \cdots & \mathbf{A}_{1,2K,n}\otimes_b\boldsymbol{\mathcal{A}}_n \\ \mathbf{A}_{2,1,n}\otimes_b\boldsymbol{\mathcal{A}}_n & \cdots & \mathbf{A}_{2,2K,n}\otimes_b\boldsymbol{\mathcal{A}}_n \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{2K,1,n}\otimes_b\boldsymbol{\mathcal{A}}_n & \cdots & \mathbf{A}_{2K,2K,n}\otimes_b\boldsymbol{\mathcal{A}}_n \end{bmatrix},$$

where

$$\mathbb{E}\left[\mathbf{A}_{i,j,n}\otimes_b\boldsymbol{\mathcal{A}}_n\right] = \mathbb{E}\begin{bmatrix} \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{1,1,n} & \cdots & \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{1,2K,n} \\ \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{2,1,n} & \cdots & \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{2,2K,n} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{2K,1,n} & \cdots & \mathbf{A}_{i,j,n}\otimes\mathbf{A}_{2K,2K,n} \end{bmatrix}.$$

Recall that the probability of any client being selected by the server in any iteration n is $\bar{a} = C/K$. Consequently, we have

$$\mathbb{E}\left[a_{i,n}a_{j,n}\right] = \begin{cases} \bar{a} = \frac{C}{K} & i = j \\ \check{a} = \frac{C}{K}\frac{C-1}{K-1} & i \neq j. \end{cases}$$

Furthermore, we define

$$\begin{aligned} \boldsymbol{\mathcal{N}}_{il} &= \mathbf{N}_i \otimes \mathbf{N}_l \\ \boldsymbol{\mathcal{N}}_{0i} &= \mathbf{I}_L \otimes \mathbf{N}_i \\ \boldsymbol{\mathcal{N}}_{i0} &= \mathbf{N}_i \otimes \mathbf{I}_L \\ \boldsymbol{\mathcal{N}}_i &= \boldsymbol{\mathcal{N}}_{0i} + \boldsymbol{\mathcal{N}}_{i0}. \end{aligned}$$

Therefore, we can calculate $\mathbb{E}\left[\mathbf{A}_{i,j,n}\otimes\mathbf{A}_{l,m,n}\right]$ as shown at the top of the following page.

Appendix C. Evaluation of the spectral radius of $\boldsymbol{\mathcal{Q}}$

Recall that

$$\boldsymbol{\mathcal{A}}_n = \left[\begin{array}{c:c} \boldsymbol{\mathcal{A}}_{n,1} & \boldsymbol{\mathcal{A}}_{n,2} \\ \hdashline \mathbf{I} & \mathbf{0} \end{array}\right]$$

and $\boldsymbol{\mathcal{Q}}^{\intercal} = \mathbb{E}[\boldsymbol{\mathcal{A}}_n \otimes_b \boldsymbol{\mathcal{A}}_n]$. In addition, using the definition of the block Kronecker product [63], we can further write $\boldsymbol{\mathcal{A}}_n \otimes_b \boldsymbol{\mathcal{A}}_n$ as

$$\begin{bmatrix} \boldsymbol{\mathcal{A}}_{n,1}\otimes\boldsymbol{\mathcal{A}}_{n,1} & \boldsymbol{\mathcal{A}}_{n,1}\otimes\boldsymbol{\mathcal{A}}_{n,2} & \boldsymbol{\mathcal{A}}_{n,2}\otimes\boldsymbol{\mathcal{A}}_{n,1} & \boldsymbol{\mathcal{A}}_{n,2}\otimes\boldsymbol{\mathcal{A}}_{n,2} \\ \boldsymbol{\mathcal{A}}_{n,1}\otimes\mathbf{I} & \mathbf{0} & \boldsymbol{\mathcal{A}}_{n,2}\otimes\mathbf{I} & \mathbf{0} \\ \mathbf{I}\otimes\boldsymbol{\mathcal{A}}_{n,1} & \mathbf{I}\otimes\boldsymbol{\mathcal{A}}_{n,2} & \mathbf{0} & \mathbf{0} \\ \mathbf{I}\otimes\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

By employing the bilinearity property of the Kronecker product and the fact that $\boldsymbol{\mathcal{A}}_{n,1} + \boldsymbol{\mathcal{A}}_{n,2} = \mathbf{I}$, we can sum up each row to an identity matrix. Therefore, $\boldsymbol{\mathcal{Q}}^{\intercal}$ is a block right-stochastic matrix with a spectral radius of 1. Hence, the spectral radius of $\boldsymbol{\mathcal{Q}}$ is also 1.

Appendix D. Proof of Proposition 2

We want to show that

$$\lim_{n\to\infty}\boldsymbol{\psi}^{\intercal}\boldsymbol{\mathcal{Q}}^n = \lim_{n\to\infty}\mathrm{bvec}^{\intercal}\{\mathbb{E}\left[\boldsymbol{\mathcal{A}}_n\cdots\boldsymbol{\mathcal{A}}_1\boldsymbol{\psi}\boldsymbol{\mathcal{A}}_1^{\intercal}\cdots\boldsymbol{\mathcal{A}}_n^{\intercal}\right]\} = \mathbf{0}^{\intercal}, \tag{D.1}$$

where $\boldsymbol{\psi} = \mathrm{bvec}^{-1}\{\boldsymbol{\psi}\}$. Given that $\boldsymbol{\mathcal{A}}_n$ is a right-stochastic matrix, we can verify that

$$\lim_{n\to\infty}\prod_{i=1}^{n}\boldsymbol{\mathcal{A}}_i = \lim_{n\to\infty}\tilde{\boldsymbol{\mathcal{A}}}^n = \left[\begin{array}{c:c} \boldsymbol{\mathcal{B}} & \mathbf{O} \\ \hdashline \boldsymbol{\mathcal{B}} & \mathbf{O} \end{array}\right] \tag{D.2}$$

is a constant matrix [67], where $\mathbf{O}$ denotes the zero matrix. This requires

$$\begin{aligned} \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}_{n,1} &= \boldsymbol{\mathcal{A}}_{n,1} \\ \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}_{n,2} &= \mathbf{O}, \end{aligned} \tag{D.3}$$

$$\mathbb{E}\left[\mathbf{A}_{i,j,n}\otimes\mathbf{A}_{l,m,n}\right]=$$

$$\begin{cases}
\mathbf{I}_{L^2}-\rho\bar{a}\boldsymbol{\mathcal{N}}_i+\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_i+\rho^2\bar{a}\boldsymbol{\mathcal{N}}_{ii}-\frac{4\rho^2\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{ii} & i=j=l=m\in\{1,\dots,K\}\\
\quad+\frac{4\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{ii} & \\
\mathbf{I}_{L^2}-\rho\bar{a}\boldsymbol{\mathcal{N}}_{i0}-\rho\bar{a}\boldsymbol{\mathcal{N}}_{0l}+\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}+\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l} & i=j\in\{1,\dots,K\},\ l=m\in\{1,\dots,K\},\ i\neq l\\
\quad+\rho^2\check{a}\boldsymbol{\mathcal{N}}_{il}-\frac{4\rho^2\bar{a}\check{a}}{C}\boldsymbol{\mathcal{N}}_{il}+\frac{4\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & \\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0i}-\frac{2\rho^2\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{ii}+\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i=j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i=l,\ j\neq m\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l}-\frac{2\rho^2\bar{a}\check{a}}{C}\boldsymbol{\mathcal{N}}_{il}+\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{il} & i=j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i\neq l,\ j=m\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l}-\frac{2\rho^2\check{a}\bar{a}}{C}\boldsymbol{\mathcal{N}}_{il}+\frac{4\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i=j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i\neq l,\ j\neq m\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0i}+\frac{\rho^2\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{ii}-\frac{2\rho^2\bar{a}^3}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i=j\in\{1,\dots,K\},\ l\in\{1,\dots,K\},\ m\in\{K+1,\dots,2K\},\ i=l\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l}+\frac{\rho^2\check{a}\bar{a}}{C}\boldsymbol{\mathcal{N}}_{il}-\frac{2\rho^2\bar{a}\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i=j\in\{1,\dots,K\},\ l\in\{1,\dots,K\},\ m\in\{K+1,\dots,2K\},\ i\neq l\\
\mathbf{I}_{L^2}-\rho\bar{a}\boldsymbol{\mathcal{N}}_{i0}+\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0} & i=j\in\{1,\dots,K\},\ m\in\{1,\dots,K\},\ l=m+K\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}-\frac{2\rho^2\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{ii}+\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\neq j\in\{1,\dots,K\},\ l=m\in\{1,\dots,K\},\ i=l,\ j\neq m\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}-\frac{2\rho^2\bar{a}\check{a}}{C}\boldsymbol{\mathcal{N}}_{il}+\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq j\in\{1,\dots,K\},\ l=m\in\{1,\dots,K\},\ i\neq l,\ j=m\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}-\frac{2\rho^2\check{a}\bar{a}}{C}\boldsymbol{\mathcal{N}}_{il}+\frac{4\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq j\in\{1,\dots,K\},\ l=m\in\{1,\dots,K\},\ i\neq l,\ j\neq m\\
\frac{4\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\neq j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i=l,\ j=m\\
\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\neq j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i=l,\ j\neq m\\
\frac{4\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i\neq l,\ j=m\\
\frac{4\rho^2\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq j\in\{1,\dots,K\},\ l\neq m\in\{1,\dots,K\},\ i\neq l,\ j\neq m\\
\frac{-2\rho^2\bar{a}^3}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\neq j\in\{1,\dots,K\},\ l\in\{1,\dots,K\},\ m\in\{K+1,\dots,2K\},\ i=l\\
\frac{-2\rho^2\bar{a}\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq j\in\{1,\dots,K\},\ l\in\{1,\dots,K\},\ m\in\{K+1,\dots,2K\},\ i\neq l\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0} & i\neq j\in\{1,\dots,K\},\ m\in\{1,\dots,K\},\ l=m+K\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}+\frac{\rho^2\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{ii}-\frac{2\rho^2\bar{a}^3}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\in\{1,\dots,K\},\ j\in\{K+1,\dots,2K\},\ l=m\in\{1,\dots,K\},\ i=l\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0}+\frac{\rho^2\check{a}\bar{a}}{C}\boldsymbol{\mathcal{N}}_{il}-\frac{2\rho^2\bar{a}\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\in\{1,\dots,K\},\ j\in\{K+1,\dots,2K\},\ l=m\in\{1,\dots,K\},\ i\neq l\\
\frac{-2\rho^2\bar{a}^3}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i\in\{1,\dots,K\},\ j\in\{K+1,\dots,2K\},\ l\neq m\in\{1,\dots,K\},\ i=l\\
\frac{-2\rho^2\bar{a}\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\in\{1,\dots,K\},\ j\in\{K+1,\dots,2K\},\ l\neq m\in\{1,\dots,K\},\ i\neq l\\
\frac{\rho^2\bar{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i=l\in\{1,\dots,K\},\ j=m\in\{K+1,\dots,2K\}\\
\frac{\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{ii} & i=l\in\{1,\dots,K\},\ j\neq m\in\{K+1,\dots,2K\}\\
\frac{\rho^2\bar{a}\check{a}}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq l\in\{1,\dots,K\},\ j=m\in\{K+1,\dots,2K\}\\
\frac{\rho^2\check{a}^2}{C^2}\boldsymbol{\mathcal{N}}_{il} & i\neq l\in\{1,\dots,K\},\ j\neq m\in\{K+1,\dots,2K\}\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{i0} & i\in\{1,\dots,K\},\ j\in\{K+1,\dots,2K\},\ m\in\{1,\dots,K\},\ l=m+K\\
\mathbf{I}_{L^2}-\rho\bar{a}\boldsymbol{\mathcal{N}}_{0l}+\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l} & j\in\{1,\dots,K\},\ i=j+K,\ l=m\in\{1,\dots,K\}\\
\frac{2\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l} & j\in\{1,\dots,K\},\ i=j+K,\ l\neq m\in\{1,\dots,K\}\\
\frac{-\rho\bar{a}^2}{C}\boldsymbol{\mathcal{N}}_{0l} & j\in\{1,\dots,K\},\ i=j+K,\ l\in\{1,\dots,K\},\ m\in\{K+1,\dots,2K\}\\
\mathbf{I}_{L^2} & j\in\{1,\dots,K\},\ i=j+K,\ m\in\{1,\dots,K\},\ l=m+K\\
\mathbf{0} & \text{otherwise.}
\end{cases}$$

---

which, in light of (18), results in

$$\boldsymbol{\mathcal{B}}\tilde{\mathbf{N}}=\mathbf{O}, \tag{D.4}$$

where $\tilde{\mathbf{N}}=\text{bdiag}\{\mathbf{N}_1,\cdots,\mathbf{N}_K\}$. Finally, considering (27) and (29), we can write the expected value in (D.1) as

$$\begin{bmatrix}\boldsymbol{\Gamma}\boldsymbol{\mathcal{B}}\tilde{\mathbf{N}}^2\boldsymbol{\mathcal{B}}^\intercal & \boldsymbol{\Gamma}\boldsymbol{\mathcal{B}}\tilde{\mathbf{N}}^2\bar{\boldsymbol{\mathcal{B}}}^\intercal\\ \bar{\boldsymbol{\Gamma}}\bar{\boldsymbol{\mathcal{B}}}\tilde{\mathbf{N}}^2\boldsymbol{\mathcal{B}}^\intercal & \bar{\boldsymbol{\Gamma}}\bar{\boldsymbol{\mathcal{B}}}\tilde{\mathbf{N}}^2\bar{\boldsymbol{\mathcal{B}}}^\intercal\end{bmatrix}=\mathbf{O}, \tag{D.5}$$

where the equality is due to (D.4) and we have

$$\boldsymbol{\Gamma}=\bar{a}\rho^2\text{bdiag}\left\{\sigma_{\zeta_1}^2\mathbf{I},\cdots,\sigma_{\zeta_K}^2\mathbf{I}\right\}+\frac{5\rho^2}{K^2}\sum_{k=1}^{K}\sigma_{\eta_k}^2\mathbf{I}. \tag{D.6}$$

This concludes the proof.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. Y. Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. Int. Conf. Artif. Intell. and Stat., 2017, pp. 1273–1282.

[2] V. Smith, C. Chiang, M. Sanjabi, A. S. Talwalkar, Federated multi-task learning, in: Proc. Adv. Neural Inf. Process. Syst., Vol. 30, 2017, pp. 1–11.

[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in: Proc. Mach. Learn. Syst., Vol. 2, 2020, pp. 429–450.

[4] X. Wang, C. Wang, X. Li, V. C. M. Leung, T. Taleb, Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching, IEEE Internet Things J. 7 (10) (2020) 9441–9455. doi:10.1109/JIOT.2020.2986803.

[5] E. Lari, R. Arablouei, S. Werner, Privacy-preserving distributed nonnegative matrix factorization, in: Proc. Eur. Signal Process. Conf., 2024, pp. 1022–1026.

[6] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, Y. Liu, Privacy-preserving blockchain-based federated learning for IoT devices, IEEE Internet Things J. 8 (3) (2021) 1817–1829. doi:10.1109/JIOT.2020.3017377.

[7] Q. Yang, Y. L., T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.

[8] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, M. Peng, Federated learning with non-iid data in wireless networks, " IEEE Trans. Wireless Commun. 21 (3) (2022) 1927–1942. doi:10.1109/TWC.2021.3108197.

[9] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60. doi:10.1109/MSP.2020.2975749.

[10] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tut. 22 (3) (2020) 2031–2063. doi:10.1109/COMST.2020.2986024.

[11] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, H. V. Poor, Federated learning: A signal processing perspective, IEEE Signal Process. Mag. 39 (3) (2022) 14–41. doi:10.1109/MSP.2021.3125282.

[12] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, Q. Yang, Decentralized federated learning: A survey on security and privacy, IEEE Trans. Big Data 10 (2) (2024) 194–213. doi:10.1109/TBDATA.2024.3362191.

[13] Z. A. E. Houda, A. S. Hafid, L. Khoukhi, Mitfed: A privacy preserving collaborative network attack mitigation framework based on federated learning using SDN and blockchain, IEEE Trans. Netw. Sci. Eng. 10 (4) (2023) 1985–2001. doi:10.1109/TNSE.2023.3237367.

[14] Y. Chang, K. Zhang, J. Gong, H. Qian, Privacy-preserving federated learning via functional encryption, revisited, IEEE Trans. Inf. Forensics Secur. 18 (2023) 1855–1869. doi:10.1109/TIFS.2023.3255171.

[15] M. S. Jere, T. Farnan, F. Koushanfar, A taxonomy of attacks on federated learning, IEEE Secur. Privacy 19 (2) (2021) 20–28. doi:10.1109/MSEC.2020.3039941.

[16] Z. Li, V. Sharma, S. P. Mohanty, Preserving data privacy via federated learning: Challenges and solutions, IEEE Consum. Electron. Mag. 9 (3) (2020) 8–16. doi:10.1109/MCE.2019.2959108.

[17] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, H. Vincent Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469. doi:10.1109/TIFS.2020.2988575.

[18] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, A. Kuh, Personalized graph federated learning with differential privacy, IEEE Trans. Signal Inf. Process. Netw. 9 (2023) 736–749. doi:10.1109/TSIPN.2023.3325963.

[19] E. Lari, R. Arablouei, V. C. Gogineni, S. Werner, Resilience in online federated learning: Mitigating model-poisoning attacks via partial sharing, IEEE Trans. Signal Inf. Process. Netw. 11 (2025) 388–400. doi:10.1109/TSIPN.2025.3559444.

[20] E. Lari, V. C. Gogineni, R. Arablouei, S. Werner, On the resilience of online federated learning to model poisoning attacks through partial sharing, in: Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2024, pp. 9201–9205.

[21] C. Fung, C. J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, arXiv preprint arXiv:1808.04866 (Aug. 2018).

[22] X. He, J. Zhang, Q. Ling, Byzantine-robust and communication-efficient personalized federated learning, in: Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2023, pp. 1–5. doi:10.1109/ICASSP49357.2023.10095468.

[23] L. Chen, W. Liu, Y. Chen, W. Wang, Communication-efficient design for quantized decentralized federated learning, IEEE Trans. Signal Process. 72 (2024) 1175–1188. doi:10.1109/TSP.2024.3363887.

[24] Y. Oh, N. Lee, Y.-S. Jeon, H. V. Poor, Communication-efficient federated learning via quantized compressed sensing, IEEE Trans. Wireless Commun. 22 (2) (2023) 1087–1100. doi:10.1109/TWC.2022.3201207.

[25] H. Zhao, W. Du, F. Li, P. Li, G. Liu, FedPrompt: Communication-efficient and privacy-preserving prompt tuning in federated learning, in: Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2023, pp. 1–5. doi:10.1109/ICASSP49357.2023.10095356.

[26] S. Vargaftik, R. B. Basat, A. Portnoy, G. Mendelson, Y. B. Itzhak, M. Mitzenmacher, EDEN: Communication-efficient and robust distributed mean estimation for federated learning, in: Proc. Int. Conf. Mach. Learn., 2022, pp. 21984–22014.

[27] V. C. Gogineni, S. Werner, Y. Huang, A. Kuh, Communication-efficient online federated learning strategies for kernel regression, IEEE Internet Things J. (2022) 4531–4544doi:10.1109/JIOT.2022.3218484.

[28] V. C. Gogineni, S. Werner, Y. Huang, A. Kuh, Communication-efficient online federated learning framework for nonlinear regression, in: Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2022, pp. 5228–5232. doi:10.1109/ICASSP43922.2022.9746228.

[29] V. C. Gogineni, S. Werner, Y. Huang, A. Kuh, Decentralized graph federated multitask learning for streaming data, in: Proc. Conf. Inf. Sci. Syst., 2022, pp. 101–106. doi:10.1109/CISS53076.2022.9751160.

[30] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1205–1221. doi:10.1109/JSAC.2019.2904348.

[31] S. Samarakoon, M. Bennis, W. Saad, M. Debbah, Federated learning for ultra-reliable low-latency V2V communications, in: Proc. IEEE Global Commun. Conf., 2018, pp. 1–7. doi:10.1109/GLOCOM.2018.8647927.

[32] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: Proc. IEEE Int. Conf. Commun., 2019, pp. 1–7. doi:10.1109/ICC.2019.8761315.

[33] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, F. R. Yu, Robust federated learning with noisy communication, IEEE Trans. Commun. 68 (6) (2020) 3452–3464. doi:10.1109/TCOMM.2020.2979149.

[34] M. M. Amiri, D. Gündüz, Federated learning over wireless fading channels, IEEE Trans. Wireless Commun. 19 (5) (2020) 3546–3557. doi:10.1109/TWC.2020.2974748.

[35] H. Guo, A. Liu, V. K. N. Lau, Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis, IEEE Internet Things J. 8 (1) (2021) 197–210. doi:10.1109/JIOT.2020.3002925.

[36] M. M. Amiri, D. Gündüz, S. R. Kulkarni, H. V. Poor, Convergence of federated learning over a noisy downlink, IEEE Trans. Wireless Commun. 21 (3) (2022) 1422–1437. doi:10.1109/TWC.2021.3103874.

[37] L. Qu, Y. Mao, S. Song, C.-Y. Tsui, Energy-efficient channel decoding for wireless federated learning: Convergence analysis and adaptive design, IEEE Trans. Wireless Commun. 23 (11) (2024) 17222–17235. doi:10.1109/TWC.2024.3451949.

[38] S. Zhou, G. Y. Li, Communication-efficient ADMM-based federated learning, arXiv preprint arXiv:2110.15318 (Jan. 2021).

[39] S. Zheng, C. Shen, X. Chen, Design and analysis of uplink and downlink communications for federated learning, IEEE J. Sel. Areas Commun. 39 (7) (2021) 2150–2167. doi:10.1109/JSAC.2020.3041388.

[40] S. M. Shah, L. Su, V. K. N. Lau, Robust federated learning over noisy fading channels, IEEE Internet Things J. 10 (9) (2023) 7993–8013. doi:10.1109/JIOT.2022.3230452.

[41] J. Yao, W. Xu, Z. Yang, X. You, M. Bennis, H. V. Poor, Wireless federated learning over resource-constrained networks: Digital versus analog transmissions, IEEE Trans. Wireless Commun. 23 (10) (2024) 14020–14036. doi:10.1109/TWC.2024.3407822.

[42] T. Sery, N. Shlezinger, K. Cohen, Y. C. Eldar, COTAF: Convergent over-the-air federated learning, in: Proc. IEEE Global Commun. Conf., 2020, pp. 1–6. doi:10.1109/GLOBECOM42002.2020.9322580.

[43] X. Wei, C. Shen, Federated learning over noisy channels, in: Proc. IEEE Int. Conf. Commun., 2021, pp. 1–6. doi:10.1109/ICC42927.2021.9500833.

[44] X. Wei, C. Shen, Federated learning over noisy channels: Convergence analysis and design examples, IEEE Trans. Cogn. Commun. Netw. 8 (2) (2022) 1253–1268. doi:10.1109/TCCN.2022.3140788.

[45] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, Now Publishers, Inc., 2011.

[46] S. Zhou, G. Y. Li, Federated learning via inexact ADMM, IEEE Trans. Pattern Anal. Mach. Intell. 45 (8) (2023) 9699–9708. doi:10.1109/TPAMI.2023.3243080.

[47] H. Kang, M. Kim, B. Lee, H. Kim, FedAND: Federated learning exploiting consensus ADMM by nulling drift, IEEE Trans. Ind. Inform. 20 (7) (2024) 9837–9849. doi:10.1109/TII.2024.3380742.

[48] S. He, J. Zheng, M. Feng, Y. Chen, Communication-efficient federated learning with adaptive consensus ADMM, Appl. Sci. 13 (9) (2023) 5270.

[49] S. M. Azimi-Abarghouyi, N. Bastianello, K. H. Johansson, V. Fodor, Hierarchical federated ADMM, IEEE Netw. Lett. 7 (1) (2025) 11–15. doi:10.1109/LNET.2025.3527161.

[50] Q. Li, L. Shen, G. Li, Q. Yin, D. Tao, DFedADMM: Dual constraint controlled model inconsistency for decentralize federated learning, IEEE Trans. Pattern Anal. Mach. Intell. (2025) 1–12doi:10.1109/TPAMI.2025.3546659.

[51] S. Wang, Y. Xu, Z. Wang, T.-H. Chang, T. Q. Quek, D. Sun, Beyond ADMM: A unified client-variance-reduced adaptive federated learning framework, in: Proc. AAAI Conf. Artif. Intell., Vol. 37, 2023, pp. 10175–10183.

[52] I. D. Schizas, A. Ribeiro, G. B. Giannakis, Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals, IEEE Trans. Signal Process. 56 (1) (2008) 350–364. doi:10.1109/TSP.2007.906734.

[53] H. Zhu, G. B. Giannakis, A. Cano, Distributed in-network channel decoding, IEEE Trans. Signal Process. 57 (10) (2009) 3970–3983. doi:10.1109/TSP.2009.2023936.

[54] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, P. K. Varshney, Robust decentralized learning using ADMM with unreliable agents, IEEE Trans. Signal Process. 70 (2022) 2743–2757. doi:10.1109/TSP.2022.3178655.

[55] E. Lari, R. Arablouei, N. K. Venkategowda, S. Werner, Distributed maximum consensus over noisy links, in: Proc. Eur. Signal Process. Conf., 2024, pp. 2247–2251.

[56] E. Lari, V. C. Gogineni, R. Arablouei, S. Werner, Resource-efficient federated learning robust to communication errors, in: Proc. IEEE Stat. Signal Process. Workshop, 2023, pp. 265–269.

[57] E. Lari, V. C. Gogineni, R. Arablouei, S. Werner, Continual local updates for federated learning with enhanced robustness to link noise, in: Proc. Asia-Pacific Signal Inf. Process. Assoc., 2023, pp. 1199–1203.

[58] Y.-F. Huang, S. Werner, J. Huang, N. Kashyap, V. Gupta, State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid, IEEE Signal Process. Mag. 29 (5) (2012) 33–43. doi:10.1109/MSP.2012.2187037.

[59] L. Li, J. Zhong, M. Zhao, Doppler-aided GNSS position estimation with weighted least squares, IEEE Trans. Veh. Technol. 60 (8) (2011) 3615–3624. doi:10.1109/TVT.2011.2163738.

[60] Z. Farbman, R. Fattal, D. Lischinski, R. Szeliski, Edge-preserving decompositions for multi-scale tone and detail manipulation, ACM Trans. Graph. 27 (3) (2008) 1–10.

[61] R. Arablouei, K. Doğançay, S. Werner, Y.-F. Huang, On the asymptotic bias of the diffusion-based distributed Pareto optimization, Signal Process. 130 (2017) 337–342.

[62] J. Chen, A. H. Sayed, Distributed pareto optimization via diffusion strategies, IEEE J. Sel. Top. Signal Process. 7 (2) (2013) 205–220. doi:10.1109/JSTSP.2013.2246763.

[63] D. S. Tracy, R. P. Singh, A new matrix product and its applications in partitioned matrix differentiation, Statistica Neerlandica 26 (4) (1972) 143–157.

[64] R. H. Koning, H. Neudecker, T. Wansbeek, Block kronecker products and the vecb operator, Linear algebra and its applications 149 (1991) 165–184.

[65] R. Balakrishnan, T. Li, T. Zhou, N. Himayat, V. Smith, J. Bilmes, Diverse client selection for federated learning via submodular maximization, in: Proc. Int. Conf. Learn. Represent., 2022.

[66] G. Xu, D.-L. Kong, X.-B. Chen, X. Liu, Lazy aggregation for heterogeneous federated learning, Appl. Sci. 12 (17) (2022) 8515.

[67] J. M. Anthonisse, H. Tijms, Exponential convergence of products of stochastic matrices, J. Math. Anal. Appl. 59 (2) (1977) 360–364.

Appendix F

Publication 6

P6 E. Lari, R. Arablouei, and S. Werner, "Privacy-Preserving Distributed Nonnegative Matrix Factorization," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 1022–1026.

Privacy-Preserving Distributed Nonnegative Matrix Factorization

Ehsan Lari[1], Reza Arablouei[2], Stefan Werner[1]
[1]Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
[2]CSIRO's Data61, Pullenvale QLD 4069, Australia

***Abstract*—Nonnegative matrix factorization (NMF) is an effective data representation tool with numerous applications in signal processing and machine learning. However, deploying NMF in a decentralized manner over ad-hoc networks introduces privacy concerns due to the conventional approach of sharing raw data among network agents. To address this, we propose a privacy-preserving algorithm for fully-distributed NMF that decomposes a distributed large data matrix into left and right matrix factors while safeguarding each agent's local data privacy. It facilitates collaborative estimation of the left matrix factor among agents and enables them to estimate their respective right factors without exposing raw data. To ensure data privacy, we secure information exchanges between neighboring agents utilizing the Paillier cryptosystem, a probabilistic asymmetric algorithm for public-key cryptography that allows computations on encrypted data without decryption. Simulation results conducted on synthetic and real-world datasets demonstrate the effectiveness of the proposed algorithm in achieving privacy-preserving distributed NMF over ad-hoc networks.**

I. Introduction

Nonnegative matrix factorization (NMF) [1]–[5] is a specific case of constrained low-rank matrix approximation [6] and a linear dimensionality reduction (LDR) technique aimed at representing nonnegative data more compactly through nonnegative factors. NMF, originally introduced as positive matrix factorization in [7], has gained significant research interest, particularly after being popularized by [8]. It has found widespread applications in various fields such as signal and image processing, data mining and analytics, machine learning, and federated learning. Examples include air emission control [7], visual object recognition [9], video background-foreground separation [10], spectral unmixing [11], text mining [12], blind source separation [13], clustering [14], collaborative filtering [15], computational biology [16], music analysis [17], molecular pattern discovery [3], efficient implementation of deep neural networks [18], and detecting malware activities [19]. Its popularity stems from its utility in identifying and extracting meaningful features from data in addition to serving as a powerful LDR technique.

Distributed optimization and estimation algorithms have garnered significant attention due to the ubiquity of data dispersed across multiple agents within network environments. The existing distributed NMF algorithms align with this trend, addressing scenarios where data is distributed among a network of agents. However, the conventional approach of sharing raw data among neighboring agents poses inherent security risks and compromises the privacy of sensitive information [20]–[22]. Consequently, there is a pressing need for privacy-preserving distributed NMF algorithms that ensure the security and confidentiality of each agent's local data.

The Paillier cryptosystem [23] is a fundamental tool for enhancing privacy in distributed algorithms. As a probabilistic asymmetric algorithm for public-key cryptography, it is specifically designed to provide secure homomorphic encryption. Its primary advantage lies in its homomorphic properties, which enable computations to be performed on encrypted data without decryption. The Paillier cryptosystem has proven to be effective in improving privacy in various applications, including smart grids [24]–[26], machine learning [27], smart homes [28], and federated learning [29], [30]. However, the potential advantages of its utilization in addressing the distributed NMF problem have not been explored in the literature.

In this paper, we introduce a privacy-preserving distributed NMF (PPDNMF) algorithm tailored for scenarios where the data matrix to be factorized is distributed among agents within an ad-hoc network. Each agent holds a subset of the columns of the data matrix. Our goal is to perform NMF of the entire data dispersed over the network in a fully distributed and secure manner. Specifically, agents participate in a distributed and collaborative process to estimate both the left and right factors, exchanging information exclusively with their immediate neighbors over secure communication links. While taking part in this collaborative process, agents maintain the privacy of their local data and the corresponding right factor estimates. We utilize the block coordinate-descent (BCD) algorithm and the alternating direction method of multipliers (ADMM) to develop our distributed NMF algorithm. Furthermore, to ensure privacy preservation, we integrate the Paillier cryptosystem into our algorithm. We evaluate the performance of the proposed algorithm through simulations using synthetic and real data, demonstrating its efficacy in achieving results comparable to those obtained by the centralized alternative.

II. Distributed NMF

The objective of NMF is to approximate a data matrix $\mathbf{Z} \in \mathbb{R}^{L \times M}$ consisting of nonnegative entries using the product of left and right factor matrices, both with nonnegative entries. That is, $\mathbf{Z} = \mathbf{X}\mathbf{Y}$ where $\mathbf{X} \in \mathbb{R}^{L \times K}$ and $\mathbf{Y} \in \mathbb{R}^{K \times M}$, typically with $K \leq \min(L, M)$. This approximation represents the

This work was partially supported by the Research Council of Norway and the Research Council of Finland (Grant 354523).

L-dimensional datapoints (columns of the data matrix) within a K-dimensional linear subspace spanned by the columns of the left factor, whose coordinates are given by the columns of the right factor. The nonnegativity constraint on the factors induces sparsity, further enhancing the compactness of the representation. Moreover, in many applications, the factors' nonnegativity is essential to their physical plausibility and intuitive interpretability.

We utilize the least-squares criterion, which is appropriate when the perturbation in the data matrix $\mathbf{Z}$ can be modeled as a Gaussian process. Therefore, the NMF problem can be formulated as

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Y}} \ & \frac{1}{2}\left\|\mathbf{Z}-\mathbf{X}\mathbf{Y}\right\|_{\mathsf{F}}^2 \\ \text{s.t. } & \mathbf{X}\geq 0, \mathbf{Y}\geq 0, \end{aligned} \tag{1}$$

where $\|\cdot\|_{\mathsf{F}}$ denotes the Frobenius norm. We consider the scenario where $\mathbf{Z}$ is distributed over a network with N agents such that we have $\mathbf{Z} = [\mathbf{Z}_1, \cdots, \mathbf{Z}_N]$ and consequently $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]$ with $\mathbf{Y}_i \in \mathbb{R}^{K\times M_i}$ and $\sum_{i=1}^N M_i = M$. Therefore, we rewrite (1) as

$$\begin{aligned} \min_{\mathbf{X},\{\mathbf{Y}_i\}} \ & \frac{1}{2}\sum_{i=1}^N \left\|\mathbf{Z}_i-\mathbf{X}\mathbf{Y}_i\right\|_{\mathsf{F}}^2 \\ \text{s.t. } & \mathbf{X}\geq 0, \mathbf{Y}_i\geq 0. \end{aligned} \tag{2}$$

In a fully distributed approach, every agent, indexed by i, aims to estimate $\mathbf{X}$ and its own $\mathbf{Y}_i$ using its local data $\mathbf{Z}_i$ and by exchanging information solely with its immediate neighbors through single-hop communication. To this end, we utilize the BCD algorithm and iteratively solve two optimization subproblems for $\mathbf{X}$ and $\mathbf{Y}$. That is, we repeat the following alternating minimizations until convergence is achieved:

$$\begin{aligned} \mathbf{X}^{(n)} &= \min_{\mathbf{X}} \ \frac{1}{2}\sum_{i=1}^N \left\|\mathbf{Z}_i-\mathbf{X}\mathbf{Y}_i^{(n-1)}\right\|_{\mathsf{F}}^2 \\ &\quad \text{s.t. } \mathbf{X}\geq 0, \end{aligned} \tag{3}$$

$$\begin{aligned} \mathbf{Y}_i^{(n)} &= \min_{\mathbf{Y}_i} \ \frac{1}{2}\left\|\mathbf{Z}_i-\mathbf{X}^{(n)}\mathbf{Y}_i\right\|_{\mathsf{F}}^2, \forall i\in\{1,\ldots,N\} \\ &\quad \text{s.t. } \mathbf{Y}_i\geq 0. \end{aligned} \tag{4}$$

The superscript (n) denotes the estimate of its respective parameter at the nth BCD iteration.

The solution of (4) can be localized straightforwardly, provided that each agent has access to the estimate $\mathbf{X}^{(n)}$. To solve (3) in a fully distributed manner, we introduce the variable $\mathbf{X}_i$ at each agent i as a local copy of $\mathbf{X}$ and enforce it to be equal to those of the agents within the immediate neighborhood of agent i, thereby achieving consensus across the network. Thus, we reformulate (3) into the following equivalent form

$$\begin{aligned} \mathbf{X}_i^{(n)} &= \min_{\mathbf{X}_i} \ \frac{1}{2}\left\|\mathbf{Z}_i-\mathbf{X}_i\mathbf{Y}_i^{(n-1)}\right\|_{\mathsf{F}}^2+\imath(\mathbf{X}_i) \\ &\quad \text{s.t. } \mathbf{X}_i=\mathbf{X}_j \ \ \forall j\in\mathcal{N}_i, \ \forall i\in\{1,\ldots,N\} \end{aligned} \tag{5}$$

where $\imath(\cdot)$ denotes the indicator function accounting for the nonegativity constraint and $\mathcal{N}_i$ denotes the set of neighbors of agent i with cardinality $d_i = |\mathcal{N}_i|$. Subsequently, we decompose and decouple the optimization problems at the agents by introducing the auxiliary variables $\mathbf{U}_i, \mathbf{S}_{i,j} \in \mathbb{R}^{L\times K}$ and rewriting the optimization in (5) as

$$\begin{aligned} \min_{\mathbf{x}_i,\mathbf{U}_i,\mathbf{S}_{i,j}} \ & \frac{1}{2}\left\|\mathbf{Z}_i-\mathbf{U}_i\mathbf{Y}_i^{(n-1)}\right\|_{\mathsf{F}}^2+\imath(\mathbf{X}_i) \\ & \mathbf{U}_i=\mathbf{X}_i \\ \text{s.t. } & \mathbf{S}_{i,j}=\mathbf{U}_i \ \ \forall j\in\mathcal{N}_i, \ \forall i\in\{1,\ldots,N\} \\ & \mathbf{S}_{j,i}=\mathbf{S}_{i,j}. \end{aligned} \tag{6}$$

We can express the corresponding aggregate augmented Lagrangian function as

$$\begin{aligned} &\mathcal{L}\left(\{\mathbf{X}_i\},\{\mathbf{U}_i\},\{\mathbf{S}_{i,j}\},\{\mathbf{P}_i\},\{\mathbf{Q}_{i,j}\}\right) \\ &= \frac{1}{2}\sum_{i=1}^N \left\|\mathbf{Z}_i-\mathbf{U}_i\mathbf{Y}_i^{(n-1)}\right\|_{\mathsf{F}}^2+\sum_{i=1}^N \imath(\mathbf{X}_i) \\ &+ \frac{\mu}{2}\sum_{i=1}^N \left\|\mathbf{X}_i-\mathbf{U}_i-\mathbf{P}_i\right\|_{\mathsf{F}}^2 \\ &+ \frac{1}{2}\sum_{i=1}^N\sum_{j\in\mathcal{N}_i} \rho_{i,j}\left\|\mathbf{U}_i-\mathbf{S}_{i,j}-\mathbf{Q}_{i,j}\right\|_{\mathsf{F}}^2, \end{aligned} \tag{7}$$

where μ and $\rho_{i,j}$ are penalty parameters and $\mathbf{P}_i, \mathbf{Q}_{i,j} \in \mathbb{R}^{L\times K}$ are scaled Lagrange multipliers. We maintain μ consistent across all agents and iterations. However, we allow each $\rho_{i,j}$, unique to the edge connecting agents i and j, to vary over iterations [31]. Additionally, we consider $\rho_{i,j} = \rho_{j,i}\ \forall i,j$.

A. Estimating the Left Factor

Minimizing (7) using ADMM, which leads to the elimination of the auxiliary variables $\{\mathbf{S}_{i,j}\}$, yields the following iterations at each agent i [32]:

$$\mathbf{X}_i^{(n,m)} = \Pi_{\geq 0}\left\{\mathbf{U}_i^{(m-1)}+\mathbf{P}_i^{(m-1)}\right\} \tag{8}$$

$$\begin{aligned} \mathbf{U}_i^{(m)} = &\Big[\mathbf{Z}_i\mathbf{Y}_i^{(n-1)\intercal}+\mu\left(\mathbf{X}_i^{(n,m)}-\mathbf{P}_i^{(m-1)}\right) \\ &+\rho_i^{(m)}\left(\mathbf{U}_i^{(m-1)}+2\mathbf{Q}_i^{(m-1)}-\mathbf{Q}_i^{(m-2)}\right)\Big] \\ &\times\left[\mathbf{Y}_i^{(n-1)}\mathbf{Y}_i^{(n-1)\intercal}+\left(\mu+\rho_i^{(m)}\right)\mathbf{I}\right]^{-1} \end{aligned} \tag{9}$$

$$\mathbf{P}_i^{(m)} = \mathbf{P}_i^{(m-1)}-\left(\mathbf{X}_i^{(n,m)}-\mathbf{U}_i^{(m)}\right) \tag{10}$$

$$\mathbf{Q}_i^{(m)} = \mathbf{Q}_i^{(m-1)}+\sum_{j\in\mathcal{N}_i}\rho_{i,j}^{(m)}\left(\mathbf{U}_j^{(m)}-\mathbf{U}_i^{(m)}\right). \tag{11}$$

Here, (m) denotes the ADMM iteration index, $\Pi_{\geq 0}$ represents the projection onto the nonnegative orthant, and $(\cdot)^\intercal$ stands for matrix transpose. In addition, we define $\rho_i^{(m)} = \sum_{j\in\mathcal{N}_i}\rho_{i,j}^{(m)}$ and $\mathbf{Q}_i^{(m)} = \sum_{j\in\mathcal{N}_i}\rho_{i,j}^{(m)}\mathbf{Q}_{i,j}^{(m)}$. These ADMM iterations can be executed in a fully distributed manner, relying solely on locally available information and single-hop communications. Upon convergences of the algorithm, we utilize the latest estimates $\mathbf{X}_i^{(n,m)}$ for optimizing $\mathbf{Y}_i$ in the subsequent BCD iteration, i.e., $\mathbf{X}_i^{(n)} \leftarrow \mathbf{X}_i^{(n,m)}$. Note that we enforce the nonnegativity constraint and consensus simultaneously.

B. Estimating the Right Factor

Similarly, we can employ ADMM to iteratively solve (4) as follows:

$$\mathbf{Y}_i^{(n,k)} = \Pi_{\geq 0}\left\{\mathbf{V}_i^{(k-1)} + \mathbf{R}_i^{(k-1)}\right\} \tag{12}$$

$$\mathbf{V}_i^{(k)} = \left(\mathbf{X}_i^{(n)\intercal}\mathbf{X}_i^{(n)} + \eta\mathbf{I}\right)^{-1} \times \left[\mathbf{X}_i^{(n)\intercal}\mathbf{Z}_i + \eta\left(\mathbf{Y}_i^{(n,k)} - \mathbf{R}_i^{(k-1)}\right)\right] \tag{13}$$

$$\mathbf{R}_i^{(k)} = \mathbf{R}_i^{(k-1)} - \left(\mathbf{Y}_i^{(n,k)} - \mathbf{V}_i^{(k)}\right). \tag{14}$$

Here, (k) represents the ADMM iteration index and η is the penalty parameter. Once convergence is attained, we utilize the latest estimates $\mathbf{Y}_i^{(n,k)}$ to update $\mathbf{X}_i$ estimates in the subsequent BCD iteration, i.e., $\mathbf{Y}_i^{(n)} \leftarrow \mathbf{Y}_i^{(n,k)}$.

Note that, we employ warm start in both ADMM algorithms for estimating the left and right factors. At the onset of each BCD iteration, we initialize both ADMM inner iterations using the most recent estimates from the preceding iterations.

C. Synchronization and Stopping

Our algorithm does not require waiting for all agents to converge in either BCD or ADMM iterations. During $\mathbf{X}_i$ updates, the first agent to converge or reach a predetermined maximum number of ADMM iterations stops updating and raises a flag, signaling its neighbors to stop as well. This message is then propagated through the network until all agents stop updating. After completing the collaborative $\mathbf{X}_i$ update iterations, each agent can immediately start updating its $\mathbf{Y}_i$ estimate independently. When an agent stops $\mathbf{Y}_i$ update due to convergence or reaching the corresponding iteration limit, it begins the first iteration of $\mathbf{X}_i$ update and shares its $\mathbf{U}_i$ with its neighbors. Using warm start, in the first iteration of $\mathbf{X}_i$ update, each agent utilizes the neighbor $\mathbf{U}_j$ values from the previous BCD iteration. Afterwards, if an agent has not received all required $\mathbf{U}_j$ updates from its neighbors, subsequent iterations are postponed until they are all acquired, ensuring synchronization of $\mathbf{X}_i$ updates among all agents. To decide when to terminate the BCD iterations, one can employ the same strategy as described for $\mathbf{X}_i$ updates. That is, the initial agent to discern convergence or reach a predefined maximum number of BCD iterations halts its updates and notifies its neighbors. This notification propagates across the network until all agents cease updating.

D. Convergence Analysis

We can express the optimization problem (6) as

$$\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{U}}} f(\boldsymbol{\mathcal{X}}) + g(\boldsymbol{\mathcal{U}}) \tag{15}$$

$$\text{s.t. } \mathbf{A}\boldsymbol{\mathcal{X}} + \mathbf{B}\boldsymbol{\mathcal{U}} = \mathbf{0}, \tag{16}$$

where

$$f(\boldsymbol{\mathcal{X}}) = \sum_{i=1}^{N} \imath(\mathbf{X}_i),\ g(\boldsymbol{\mathcal{U}}) = \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{Z}_i - \mathbf{U}_i\mathbf{Y}_i^{(n-1)}\right\|_{\mathsf{F}}^2,$$

$$\boldsymbol{\mathcal{X}} = \left[\mathbf{X}_1, \cdots, \mathbf{X}_N\right]^{\intercal},\ \boldsymbol{\mathcal{U}} = \left[\mathbf{U}_1, \cdots, \mathbf{U}_N \middle| \mathbf{S}_{1,1}, \cdots, \mathbf{S}_{N,N}\right]^{\intercal},$$

Algorithm 1: The PPDNMF algorithm as agent i.

$\mathbf{X}_i^{(0)} = \mathbf{U}_i^{(0)} = \mathbf{1}_{L\times K}$, $\mathbf{P}_i^{(0)} = \mathbf{Q}_i^{(0)} = \mathbf{Q}_i^{(-1)} = \mathbf{0}_{L\times K}$,
$\mathbf{Y}_i^{(0)} = \mathbf{V}_i^{(0)} = \mathbf{R}_i^{(0)} = \mathbf{0}_{K\times M_i}$
for $n = 1, 2, \ldots,$ *until convergence* **do**
  for $m = 1, 2, \ldots,$ *until convergence* **do**
    $\mathbf{X}_i^{(n,m)} = \Pi_{\geq 0}\left\{\mathbf{U}_i^{(m-1)} + \mathbf{P}_i^{(m-1)}\right\}$
    $\mathbf{U}_i^{(m)} = \left[\mathbf{Z}_i\mathbf{Y}_i^{(n-1)\intercal} + \mu\left(\mathbf{X}_i^{(n,m)} - \mathbf{P}_i^{(m-1)}\right) + \rho_i^{(m)}\left(\mathbf{U}_i^{(m-1)} + 2\mathbf{Q}_i^{(m-1)} - \mathbf{Q}_i^{(m-2)}\right)\right] \times \left[\mathbf{Y}_i^{(n-1)}\mathbf{Y}_i^{(n-1)\intercal} + \left(\mu + \rho_i^{(m)}\right)\mathbf{I}\right]^{-1}$
    $\mathbf{P}_i^{(m)} = \mathbf{P}_i^{(m-1)} - \left(\mathbf{X}_i^{(n,m)} - \mathbf{U}_i^{(m)}\right)$
    encrypt $-\mathbf{U}_i^{(m)}$ using the public key k_{pi} as $\mathcal{E}_i\left(-\mathbf{U}_i^{(m)}\right)$
    send $\mathcal{E}_i\left(-\mathbf{U}_i^{(m)}\right)$ and k_{pi} to all neighbors in $\mathcal{N}_i$
    for $j \in \mathcal{N}_i$ **do**
      encrypt $\mathbf{U}_j^{(m)}$ using k_{pi} as $\mathcal{E}_i\left(\mathbf{U}_j^{(m)}\right)$
      send $\left[\mathcal{E}_i\left(\mathbf{U}_j^{(m)}\right)\mathcal{E}_i\left(-\mathbf{U}_i^{(m)}\right)\right]^{g_{j\to i}^{(m)}}$ to agent i
    decrypt messages received from neighbors using the private key k_{si} and multiply them by $g_{i\to j}^{(m)}$
    $\mathbf{Q}_i^{(m)} = \mathbf{Q}_i^{(m-1)} + \sum_{j\in\mathcal{N}_i} g_{i\to j}^{(m)} g_{j\to i}^{(m)}\left(\mathbf{U}_j^{(m)} - \mathbf{U}_i^{(m)}\right)$.
  $\mathbf{X}_i^{(n)} \leftarrow \mathbf{X}_i^{(n,m)}$, $\mathbf{U}_i^{(0)} \leftarrow \mathbf{U}_i^{(m)}$, $\mathbf{P}_i^{(0)} \leftarrow \mathbf{P}_i^{(m)}$,
  $\mathbf{Q}_i^{(0)} \leftarrow \mathbf{Q}_i^{(m)}$, $\mathbf{Q}_i^{(-1)} \leftarrow \mathbf{Q}_i^{(m-1)}$
  for $k = 1, 2, \ldots,$ *until convergence* **do**
    $\mathbf{Y}_i^{(n,k)} = \Pi_{\geq 0}\left\{\mathbf{V}_i^{(k-1)} + \mathbf{R}_i^{(k-1)}\right\}$
    $\mathbf{V}_i^{(k)} = \left(\mathbf{X}_i^{(n)\intercal}\mathbf{X}_i^{(n)} + \eta\mathbf{I}\right)^{-1} \times \left[\mathbf{X}_i^{(n)\intercal}\mathbf{Z}_i + \eta\left(\mathbf{Y}_i^{(n,m)} - \mathbf{R}_i^{(k-1)}\right)\right]$
    $\mathbf{R}_i^{(k)} = \mathbf{R}_i^{(k-1)} - \left(\mathbf{Y}_i^{(n,k)} - \mathbf{V}_i^{(k)}\right)$
  $\mathbf{Y}_i^{(n)} \leftarrow \mathbf{Y}_i^{(n,k)}$, $\mathbf{V}_i^{(0)} \leftarrow \mathbf{V}_i^{(k)}$, $\mathbf{R}_i^{(0)} \leftarrow \mathbf{R}_i^{(k)}$

$$\mathbf{A} = \left[\begin{array}{c}\mathbf{I}\\\hline\mathbf{0}\\\hline\mathbf{0}\end{array}\right],\ \mathbf{B} = \left[\begin{array}{c|c}-\mathbf{I} & \mathbf{0}\\\hline\boldsymbol{\mathcal{D}} & -\boldsymbol{\mathcal{A}}\\\hline\mathbf{0} & \boldsymbol{\mathcal{I}}\end{array}\right],$$

and $\boldsymbol{\mathcal{D}} = \mathrm{bdiag}(d_1\mathbf{I}_{L\times K}, \cdots, d_N\mathbf{I}_{L\times K})$. In addition, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{NL\times N^2K}$ and $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{NL\times N^2K}$ represent modified versions of the adjacency and edge-node incidence matrices of the network, respectively [33]. Consequently, the convergence of the ADMM iterations (8)-(11) can be proven using the approach proposed in [34]–[36]. The convergence of (12)-(14) can also be verified in a similar manner.

III. Privacy-Preserving Distributed NMF

In this section, we provide a brief overview of the Paillier cryptosystem, which we employ to enhance the privacy of the distributed NMF algorithm developed in section II. Subsequently, we introduce our proposed privacy-preserving distributed NMF (PPDNMF) algorithm.

A. Paillier Cryptosystem

In the Paillier cryptosystem, a public key and a private key are utilized. The public key is broadcast publicly, allowing other users to encrypt messages. However, decrypting the messages is only possible with the private key, which remains unknown to other users. This cryptosystem exhibits additive homomorphism [31], [37], [38], i.e., $\mathcal{E}(m_3(m_1+m_2)) = (\mathcal{E}(m_1)\mathcal{E}(m_2))^{m_3}$.

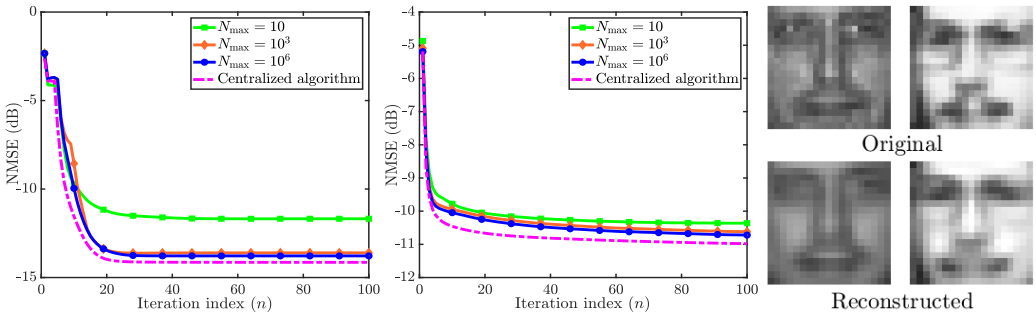

Fig. 1. The normalized mean-square error (NMSE) values of PPDNMF and centralized algorithm versus the BCD iteration index for different values of $N_{\max}$ on (a) synthetic data and (b) MIT-CBCL database. (c) The original and reconstructed faces #1 and #2429 from the MIT-CBCL database.

B. Privacy Preservation

The only update equation among (8)-(14) that relies on information received from neighboring agents is (11). To protect the privacy of agents at this step, we adopt a similar approach to [31] and enable the agents to encrypt all messages communicated with their neighbors. To this end, we decompose each edge-specific penalty parameter as $\rho_{i,j}^{(m)} = g_{i\to j}^{(m)} g_{j\to i}^{(m)}$ where $g_{i\to j}^{(m)}$ and $g_{j\to i}^{(m)}$ are exclusively known to agents i and j, respectively. In addition, we implement a secure data exchange procedure as outlined in lines 10-15 of Algorithm 1, which provides a summary of the proposed PPDNMF algorithm. Consequently, note the following:

- Data exchanged between agents i and j is encrypted, rendering it inaccessible to other agents or eavesdroppers, even if intercepted.
- The parameter $g_{i\to j}^{(m)}$ is unique to each edge and iteration. Therefore, an agent cannot infer the private information $\mathbf{U}_j^{(m)}$ of any of its neighbors by decrypting the messages it receives from them, as each neighbor j uses its unique $g_{j\to i}^{(m)}$ in its encrypted message to agent i.
- The Paillier cryptosystem is intended for encrypting scalar unsigned integers. To encrypt the entries of $\mathbf{U}_i^{(m)}$, which are typically floating-point values, we initially quantize them. This involves multiplying each entry by a positive integer $N_{\max}$, which determines the quantization resolution, and then rounding the result to the nearest integer. To undo the quantization, we divide the decrypted values by $N_{\max}$.
- To guarantee convergence, we ensure that the parameters $g_{i\to j}^{(m)}$ increase monotonically over iterations without becoming unbounded [31]. Thus, we select each parameter uniformly from the interval $\left(g_{i\to j}^{(m-1)}, g_i\right]$, where g_i is a predefined positive constant, known only to agent i, and $g_{i\to j}^{(0)} = 0$.

IV. Simulation Results

In this section, we conduct a series of numerical experiments to evaluate the performance of our PPDNMF algorithm. We consider a network consisting of $N = 10$ agents, interconnected arbitrarily, with each agent having three neighbors on average. We test our algorithm on two datasets, namely, a synthetic dataset and the MIT-CBCL face database [39]. The agents collaboratively factorize a data matrix $\mathbf{Z} \in \mathbb{R}^{L\times M}$ to left and right factor matrices $\mathbf{X} \in \mathbb{R}^{L\times K}$ and $\mathbf{Y} \in \mathbb{R}^{K\times M}$, where $L \in \{30, 361\}, M \in \{200, 2429\}$, and $K \in \{5, 49\}$ in our two experiments. We set the number of BCD iterations to 100 and the number of ADMM iterations to 30. In our implementation of the Paillier cryptosystem, we use 128-bit public and private keys. To handle the encryption of negative quantized values (note line 10 in Algorithm 1), we convert them to positive integers by adding the public key to them [37].

We evaluate the performance of PPDNMF in comparison with the centralized algorithm, i.e., where all data is available at a central hub. To quantify the performance, we utilize the normalized mean-square error (NMSE) at each BCD iteration, defined as $\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{Z}_i - \mathbf{X}_i^{(n)}\mathbf{Y}_i^{(n)}\|_{\mathsf{F}}/\|\mathbf{Z}_i\|_{\mathsf{F}}$. In addition, we average the presented results over 100 independent trials.

In our first experiment utilizing synthetic data, we draw the entries of the nonnegative factor matrices $\mathbf{X} \in \mathbb{R}^{30\times 5}$ and $\mathbf{Y} \in \mathbb{R}^{5\times 200}$ independently from exponential distributions with parameter values 0.033 and 0.8, respectively. We calculate the data matrix as $\mathbf{Z} = \mathbf{XY} + \mathbf{\Gamma}$, where we draw the entries of $\mathbf{\Gamma}$ independently from a Gaussian distribution with zero mean and variance 3.6×10^{-4}, resulting in an SNR of approximately 20dB. We set $\mu = 0.1$, $\eta = 1$, and $g_i = 0.033$ for all agents. We consider $\mathbf{Z}$ to be distributed among the agents such that each agent has a varying number of columns between four and 40. We present the NMSE learning curves of PPDNMF for different values of $N_{\max}$ alongside that of the corresponding centralized algorithm in Fig. 1(a). We observe from Fig. 1(a) that the proposed PPDNMF algorithm closely approximates the performance of the centralized algorithm in terms of both convergence rate and steady-state NMSE. Additionally, it is also evident that a higher value of $N_{\max}$ leads to a lower steady-state NMSE.

Our second experiment involves the MIT-CBCL face

database, which comprises 2429 monochromic face images in its training set. We distribute the associated data matrix among the agents such that each agent has between 224 and 245 columns. For this experiment, we set $\mu = 2$, $\eta = 2$, and $g_i = 0.05$ for all agents. The results presented in Fig. 1(b) underscore the effectiveness of PPDNMF. Notably, PPDNMF exhibits robust performance even with $N_{\max} = 10$. Furthermore, we compare the original faces #1 and #2429 and their reconstructed versions by PPDNMF using $N_{\max} = 10^6$ in Fig. 1(c). The reconstructed faces closely resemble their original counterparts.

V. Conclusion

We introduced a novel privacy-preserving distributed nonnegative matrix factorization algorithm that employs the Paillier cryptosystem to enable secure collaboration among agents, thereby safeguarding their privacy and mitigating the risk of sensitive data leakage over ad-hoc networks. Our simulation results, based on both synthetic and real data, confirmed the efficacy of the proposed algorithm. In future work, we plan to conduct a comprehensive theoretical privacy analysis of the proposed algorithm, exploring its resilience across various attack scenarios.

References

[1] N. Gillis, "The why and how of nonnegative matrix factorization," *Connections*, vol. 12, no. 2, 2014.

[2] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[3] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Comput. Stat. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.

[4] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, 2013.

[5] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13. MIT Press, 2000.

[6] M. Udell, C. Horn, R. Zadeh, S. Boyd *et al.*, "Generalized low rank models," *Found. Trends Mach. Learn.*, vol. 9, no. 1, pp. 1–118, 2016.

[7] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[9] M. W. Spratling and P. Dayan, "Learning image components for object recognition." *J. Mach. Learn. Res.*, vol. 7, no. 5, 2006.

[10] A. Kumar and V. Sindhwani, "Near-separable non-negative matrix factorization with ℓ_1 and Bregman loss functions," in *Proc. SIAM Int. Conf. Data Min.*, 2015, pp. 343–351.

[11] W.-K. Ma, J. M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi, and C.-Y. Chi, "A signal processing perspective on hyperspectral unmixing: Insights from remote sensing," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 67–81, 2013.

[12] D. Godfrey, C. Johns, C. Meyer, S. Race, and C. Sadek, "A case study in text mining: Interpreting twitter data from world cup tweets," *arXiv preprint arXiv:1408.5427*, 2014.

[13] T.-H. Chan, W.-K. Ma, C.-Y. Chi, and Y. Wang, "A convex analysis framework for blind separation of non-negative sources," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 5120–5134, 2008.

[14] A. C. Türkmen, "A review of nonnegative matrix factorization methods for clustering," *arXiv preprint arXiv:1507.03194*, 2015.

[15] P. Melville and V. Sindhwani, "Recommender systems." *Encycl. Mach. Learn.*, vol. 1, pp. 829–838, 2010.

[16] K. Devarajan, "Nonnegative matrix factorization: an analytical and interpretive tool in computational biology," *PLoS Comput. Biol.*, vol. 4, no. 7, p. e1000029, 2008.

[17] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Comput.*, vol. 21, no. 3, pp. 793–830, 2009.

[18] S. Bhattacharya and N. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2016, pp. 176–189.

[19] Y.-W. Chang, H.-Y. Chen, C. Han, T. Morikawa, T. Takahashi, and T.-N. Lin, "FINISH: Efficient and scalable NMF-based federated learning for detecting malware activities," *IEEE Trans. Emerg. Top. Comput.*, vol. 11, no. 4, pp. 934–949, 2023.

[20] Y. Qian, C. Tan, D. Ding, H. Li, and N. Mamoulis, "Fast and secure distributed nonnegative matrix factorization," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 2, pp. 653–666, 2022.

[21] P. Mai and Y. Pang, "Privacy-preserving multiview matrix factorization for recommender systems," *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 267–277, 2024.

[22] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed maximum consensus," *IEEE Signal Process. Lett.*, vol. 27, pp. 1839–1843, 2020.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.* Springer, 1999, pp. 223–238.

[24] Y. Yan, Z. Chen, V. Varadharajan, M. J. Hossain, and G. E. Town, "Distributed consensus-based economic dispatch in power grids using the paillier cryptosystem," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3493–3502, 2021.

[25] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, pp. 1621–1631, 2012.

[26] H. Shen, M. Zhang, and J. Shen, "Efficient privacy-preserving cube-data aggregation scheme for smart grids," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1369–1381, 2017.

[27] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities," *IEEE Internet Things J.*, vol. 6, pp. 7702–7712, 2019.

[28] S. M. Errapotu, J. Wang, Y. Gong, J.-H. Cho, M. Pan, and Z. Han, "Safe: Secure appliance scheduling for flexible and efficient energy consumption for smart home IoT," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4380–4391, 2018.

[29] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyberphysical systems," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5615–5624, 2021.

[30] Q. Xu, Y. Lan, Z. Su, D. Fang, and H. Zhang, "Verifiable and privacy-preserving cooperative federated learning in uav-assisted vehicular networks," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 2288–2293.

[31] C. Zhang, M. Ahmad, and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 565–580, 2019.

[32] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, *Decentralized learning for wireless communications and networking*. Springer, 2017.

[33] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Proc. IEEE Conf. Decis. Control*, 2012, pp. 5445–5450.

[34] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *J. Sci. Comput.*, vol. 66, pp. 889–916, 2016.

[35] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, pp. 29–63, 2019.

[36] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.

[37] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *Proc. IEEE Conf. Decis. Control*, 2015, pp. 6836–6843.

[38] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Trans. Automat. Control*, vol. 64, pp. 4035–4049, 2019.

[39] R. Fischer, J. Skelley, and B. Heisele, "The MIT-CBCL facial expression database." [Online]. Available: http://cbcl.mit.edu/software-datasets/FaceData2.html

Appendix G

Publication 7

P7 E. Lari, R. Arablouei, N. K. D. Venkategowda, and S. Werner, "Distributed Maximum Consensus over Noisy Links," in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 2247–2251.

Distributed Maximum Consensus over Noisy Links

Ehsan Lari[1], Reza Arablouei[2], Naveen K. D. Venkategowda[3], Stefan Werner[1]
[1]Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
[2]CSIRO's Data61, Pullenvale QLD 4069, Australia
[3]Department of Science and Technology, Linköping University, Norrköping, Sweden

***Abstract*—We introduce a distributed algorithm, termed noise-robust distributed maximum consensus (RD-MC), for estimating the maximum value within a multi-agent network in the presence of noisy communication links. Our approach entails redefining the maximum consensus problem as a distributed optimization problem, allowing a solution using the alternating direction method of multipliers. Unlike existing algorithms that rely on multiple sets of noise-corrupted estimates, RD-MC employs a single set, enhancing both robustness and efficiency. To further mitigate the effects of link noise and improve robustness, we apply moving averaging to the local estimates. Through extensive simulations, we demonstrate that RD-MC is significantly more robust to communication link noise compared to existing maximum-consensus algorithms.**

I. Introduction

Distributed learning algorithms have garnered significant attention in recent years for addressing data-centric challenges across large-scale multi-agent networks. These algorithms find diverse applications across various analytics tasks [1]–[7]. Distributed algorithms not only enjoy enhanced resilience against node or link failures, compared to centralized algorithms, but also obviate the need for central data collection and processing.

Consensus algorithms play a pivotal role in a variety of distributed computing and optimization applications, including those related to distributed learning [8]–[12]. These algorithms facilitate coordination and consensus formation among multiple agents within a distributed system, enabling them to collaboratively achieve a common goal. Therefore, they serve as a fundamental component in systems reliant on distributed decision-making [13]–[15]. Several studies [16]–[24] delve into the problem of attaining network-wide consensus on various values such as average, minimum, and median in a distributed manner. Achieving consensus in a multi-agent network mandates local computations by agents, coupled with data exchange among neighboring agents. Hence, the presence of communication noise, especially over wireless links, necessitates careful consideration.

The distributed maximum consensus problem pertains to identifying the maximum value within a network. Extensive research has been conducted on this problem in various contexts [25]–[31]. For instance, [25] presents a distributed algorithm for maximum consensus, albeit assuming noiseless links. In addition, [26] derives bounds on the expected convergence time for maximum consensus in asynchronous networks without considering communication noise. The approach in [27] addresses the maximum consensus problem by approximating the maximum function with the soft-max function. However, its performance is limited by a trade-off between estimation error and convergence speed. While [28] proposes a noise-robust distributed maximum consensus algorithm, its error variance increases linearly with the network size.

In this paper, we introduce a fully-distributed algorithm, called noise-robust distributed maximum consensus (RD-MC), devised to accurately estimate the maximum value across a multi-agent network, particularly in scenarios where communication channels are corrupted by noise. In developing RD-MC, we draw inspiration from previous research [6], [32], [33] that highlight the benefits of strategically designed parameter exchanges. We substantiate the effectiveness of RD-MC by conducting extensive simulations and comparing its performance with existing algorithms.

Mathematical Notations: The sets of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. Scalars and column vectors are denoted by lowercase and bold lowercase, respectively. The indicator function $\mathcal{I}_a(x)$ is defined as $\mathcal{I}_a(x) = 0$, if $x \geq a$ and ∞ otherwise.

II. Preliminaries

We consider a connected network comprising $J \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges, modeled by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Here, the set of vertices $\mathcal{V} = \{1, 2, \cdots, J\}$ corresponds to the agents and the edge set $\mathcal{E}$ represents the communication links between the agents. Agent $i \in \mathcal{V}$ communicates with its neighbors, indexed in $\mathcal{N}_i$ with cardinality $d_i = |\mathcal{N}_i|$. The set $\mathcal{N}_i$ does not include the agent i itself. We consider only simple graphs, devoid of self-loops or multiple edges. The structure of $\mathcal{G}$ is described by its adjacency matrix $\mathbf{A}$ with entries a_{ij}, where $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. Furthermore, the degree matrix $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_J)$ contains the number of nodes in each agent's neighborhood.

The conventional maximum consensus algorithm, which relies on the network agents communicating solely with their immediate neighbors, is expressed as

$$x_i(k+1) = \max(x_i(k), \{x_j(k)\}_{j \in \mathcal{N}_i}), \qquad \forall i \in \mathcal{V}, \quad (1)$$

where $x_i(k)$ is the estimate of the ith agent at time instant k, and the initial value is $x_i(0) = a_i \ \forall i \in \mathcal{V}$. We denote the solution to (1) by $\max(\{a_i\}_{i \in \mathcal{V}}) = a^\star$. When the inter-node communications are noiseless, there exists a finite K such that $x_i(k) = a^\star \ \forall k \geq K$ and $i \in \mathcal{V}$ [29].

This work was supported by the Research Council of Norway.

We consider the communication links to be noisy. We denote the noise in the message received by agent i from agent j at time instant k as $w_j^i(k) \in \mathbb{R}$ and model it as zero-mean additive white Gaussian noise with variance σ^2. We assume that the noise is uncorrelated across different time instants and agents. By accounting for additive link noise, (1) becomes

$$x_i(k+1) = \max(x_i(k), \{x_j(k) + w_j^i(k)\}_{j\in\mathcal{N}_i}), \qquad \forall i \in \mathcal{V}. \tag{2}$$

While (1) converges to the maximum value under ideal communication conditions, (2) may fail to converge due to potential noise-induced drift in the estimated maximum value during each iteration. This drift, compounded over time, can lead to significant inaccuracies. To address this challenge, we reformulate the maximum consensus problem as a distributed optimization problem with an aggregate global objective function. The proposed RD-MC algorithm, developed to solve this problem, converges even in the presence of additive noise in the communication links.

III. Noise-Robust Maximum Consensus Algorithm

In this section, we present a reformulation of the maximum consensus problem that enables its solution via ADMM. Subsequently, we describe two subtle modifications to the algorithm resulting from solving the reformulated problem through ADMM. These modifications are aimed at enhancing the robustness of distributed maximum consensus to communication noise and lead to the proposed RD-MC algorithm.

The consensus-based reformulation of the maximum consensus problem (1) has been discussed in [29]. However, the effect of noisy links has not been investigated in that work. In [29], it is demonstrated that (1) can be equivalently expressed as

$$\begin{aligned} \min_{\{x_i,y_i,q_i^j\}} \quad & \frac{1}{J}\sum_{i=1}^{J} x_i + \frac{1}{J}\sum_{i=1}^{J} \mathcal{I}_{a_i}(y_i) \\ \text{s.t.} \quad & x_i = y_i \quad \forall i \in \mathcal{V} \\ & x_i = q_i^j, x_j = q_i^j \quad \forall i \in \mathcal{V}, j \in \mathcal{N}_i, \end{aligned} \tag{3}$$

where the indicator function $\mathcal{I}_{a_i}(y_i)$ imposes an inequality constraint to seek the maximum value and the auxiliary variables $\mathcal{Q} = \{q_i^j\}_{i\in\mathcal{V},j\in\mathcal{N}_i}$ facilitate consensus within each agent's neighborhood and, consequently, across the network. The optimization problem (3) can be tackled using various methods, including those based on subgradients or ADMM. However, distributed subgradient methods applied to affine objective functions are known to converge slowly [34]. Therefore, we opt for ADMM to solve (3).

Let $\mathcal{L}_\rho(\{x_i, y_i\}_{i\in\mathcal{V}}, \mathcal{Q}, \mathcal{M})$ denote the augmented Lagrangian function associated with (3), where $\mathcal{M} = \{u_i, \mu_i^j, \pi_i^j\}_{i\in\mathcal{V},j\in\mathcal{N}_i}$ represents the respective Lagrange multipliers. Minimizing $\mathcal{L}_\rho$, while applying the Karush-Kuhn-Tucker optimality conditions [35] to (3) and defining $v_i(k) = 2\sum_{j\in\mathcal{N}_i}\mu_i^j(k)$, leads to the following iterative updates at the ith agent along with the elimination of $\{\pi_i^j\}_{i\in\mathcal{V},j\in\mathcal{N}_i}$ and $\mathcal{Q}$ [29], [36]:

$$\begin{aligned} x_i(k+1) &= n_i\big(-J^{-1} + \rho_y[y_i(k) - \bar{u}_i(k)] - v_i(k) \\ &\quad + \rho_z \sum_{j\in\mathcal{N}_i} [x_i(k) + \tilde{x}_j(k)]\big), \end{aligned} \tag{4a}$$

$$y_i(k+1) = \max(x_i(k+1) + \bar{u}_i(k), a_i), \tag{4b}$$

$$\bar{u}_i(k+1) = \bar{u}_i(k) + x_i(k+1) - y_i(k+1), \tag{4c}$$

$$v_i(k+1) = v_i(k) + \rho_z \sum_{j\in\mathcal{N}_i} [x_i(k+1) - \tilde{x}_j(k+1)]. \tag{4d}$$

Here, k is the iteration index, $n_i = (\rho_y + 2\rho_z d_i)^{-1}$, $\rho_y > 0$ and $\rho_z > 0$ are the penalty parameters, and $\bar{u}_i = u_i/\rho_y$. In addition, all initial values $\{x_i(0), y_i(0), \bar{u}_i(0), v_i(0)\}_{i\in\mathcal{V}}$ are set to zero. Note that, in (4a), agent i has access to $\tilde{x}_j(k) = x_j(k) + w_i^j(k)$ rather than $x_j(k)$. The iterations (4) can be implemented locally at each agent in a fully distributed fashion, as the required information is available within each agent's neighborhood. We refer to this algorithm, originally proposed in [29], as distributed maximum consensus (D-MC).

Using the initial values $v_i(0) = 0\ \forall i \in \mathcal{V}$, we obtain

$$v_i(k) = \rho_z \sum_{\ell=1}^{k} \sum_{j\in\mathcal{N}_i} [x_i(\ell) - \tilde{x}_j(\ell)] \tag{5}$$

from (4d). Substituting (5) into (4a) while using the initial values $x_i(0) = 0$ and $x_i(1) = -J^{-1}n_i\ \forall i \in \mathcal{V}$, we can eliminate $v_i(k)$ and modify (4a) as

$$\begin{aligned} x_i(k+1) &= (1 - \rho_y n_i)x_i(k) - \rho_z d_i n_i x_i(k-1) \\ &\quad + n_i\Big[\rho_y z_i(k) + \rho_z \sum_{j\in\mathcal{N}_i} \tilde{s}_j(k)\Big], \end{aligned} \tag{6a}$$

$$\bar{x}_i(k+1) = \sum_{\ell=0}^{\mathcal{C}-1} \alpha_\ell x_i(k+1-\ell), \tag{6b}$$

$$z_i(k+1) = 2y_i(k+1) - y_i(k), \tag{6c}$$

$$s_i(k+1) = 2\bar{x}_i(k+1) - x_i(k). \tag{6d}$$

Note that, in (6b), to enhance robustness against spurious noise, we compute the convex combination of $\mathcal{C}$ past local estimates, utilizing the weights α_ℓ that sum to one.

In this alternative formulation, instead of $x_i(k)$, agents exchange $s_i(k)$, which is a smoothed version of $x_i(k)$. However, due to communication noise, they receive noisy versions from their neighbors, i.e., agents $j \in \mathcal{N}_i$ receive $\tilde{s}_i(k) = s_i(k) + w_i^j(k)$ from agent i. The recursions (6) alongside (4b) and (4c) constitute the proposed noise-robust distributed maximum consensus (RD-MC) algorithm, summarized in Algorithm 1.

We mitigate the effect of noisy links in RD-MC through two key modifications to D-MC. First, the introduction of $s_i(k)$, a linear combination of $\bar{x}_i(k)$ and $x_i(k-1)$, offers a strategic advantage in alleviating the adverse effects of communication noise. By exchanging $s_i(k)$ instead of $x_i(k)$ over noisy links, we enhance robustness. Notably, while the aggregation of two sets of noisy estimates received from neighbors at consecutive iterations [i.e., $\tilde{x}_j(k)$ in (4a) and $\tilde{x}_j(k+1)$ in (4d)] renders

Algorithm 1 The RD-MC algorithm.

Parameters: penalty parameters ρ_z and ρ_y

Initialization: $x_i(0) = 0$, $x_i(1) = -J^{-1}n_i$, $\bar{u}_i(1) = 0$, $z_i(1) = 0$, $s_i(1) = -2J^{-1}n_i$, $\forall i \in \mathcal{V}$

For $k = 1, \cdots, K$ *until convergence*

Receive $\tilde{s}_j(k)$ from neighbors $j \in \mathcal{N}_i$

$$x_i(k+1) = (1-\rho_y n_i)x_i(k) - \rho_z d_i n_i x_i(k-1) + n_i\Big[\rho_y z_i(k) + \rho_z \textstyle\sum_{j\in\mathcal{N}_i} \tilde{s}_j(k)\Big]$$

$$\bar{x}_i(k+1) = \textstyle\sum_{\ell=0}^{\mathcal{C}-1} \alpha_\ell x_i(k+1-\ell),\ \sum_\ell \alpha_\ell = 1$$

$$y_i(k+1) = \max(x_i(k+1) + \bar{u}_i(k), a_i)$$

$$\bar{u}_i(k+1) = \bar{u}_i(k) + x_i(k+1) - y_i(k+1)$$

$$z_i(k+1) = 2y_i(k+1) - y_i(k)$$

Send $s_i(k+1) = 2\bar{x}_i(k+1) - x_i(k)$ to neighbors $j \in \mathcal{N}_i$

EndFor

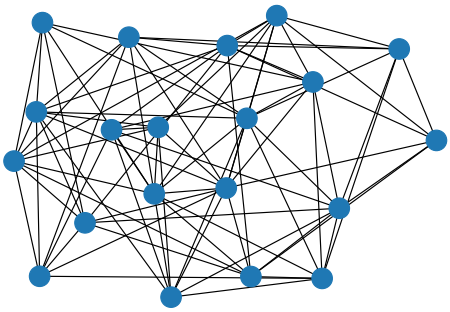

Fig. 1. The considered network with an arbitrary topology and $J = 20$ agents.

D-MC vulnerable to noise accumulation, RD-MC's reliance on a single set of noisy estimates [i.e., $\tilde{s}_j(k)$ in (6a)] enhances its resilience to link noise. Second, we further enhance robustness to link noise by applying a weighted averaging of $x_i(k+1)$ over a sliding window of size $\mathcal{C}$ as in (6b).

IV. Simulation Results

We conduct a series of experiments to examine the performance of the proposed RD-MC algorithm. We consider a network of $J = 20$ agents as depicted in Fig. 1. We independently draw the initial values (estimates) of the agents from a standard normal distribution, i.e., $a_i \sim \mathcal{N}(0,1)\ \forall i \in \mathcal{V}$, and set $a^\star = \max(\{a_i\}_{i\in\mathcal{V}})$. In addition, we set the penalty parameter values to $\rho_z = \rho_y = 1$ and the weights in (6b) to $\alpha_\ell = 1/\mathcal{C}$ in all our experiments. We obtain the results by averaging over 1000 independent realizations of communication noise. To model the noise in the communication links, we employ a truncated zero-mean normal distribution, truncating the noise at $\pm 3\sigma$ to ensure it remains within a reasonable bound.

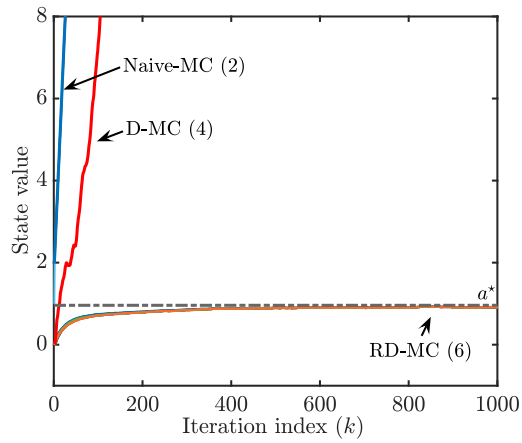

Fig. 2. The impact of noise on the performance of naive-MC (2), D-MC algorithm (4) and RD-MC algorithm (6) with window size $\mathcal{C} = 3$ and noise variance $\sigma^2 = 0.1$.

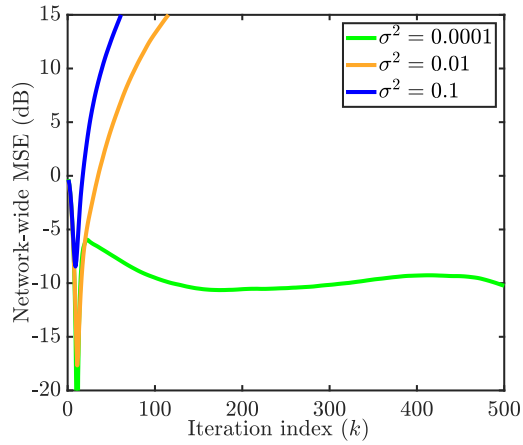

Fig. 3. The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 1$ and different noise variances $\sigma^2 \in \{0.0001, 0.01, 0.1\}$.

In our first experiment, we examine the impact of noise on the performance of RD-MC, D-MC, and the naive solution (2), referred to as naive-MC, using a noise variance of $\sigma^2 = 0.1$ and a window size of $\mathcal{C} = 3$. Fig. 2 shows the evolution of the estimates of all agents using the considered algorithms over 1000 iterations. It is evident that RD-MC converges to the maximum value with a bounded error, whereas the other two algorithms diverge.

In our second experiment, we study the effect of noise variance on the network-wide mean square error (MSE) of RD-

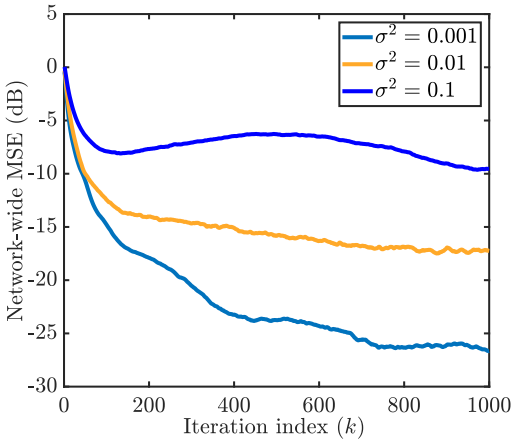

Fig. 4. The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 2$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

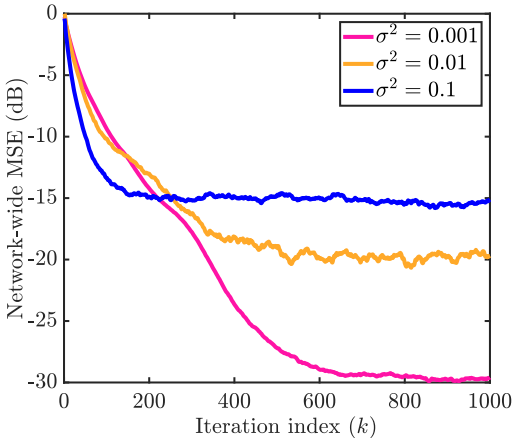

Fig. 5. The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

MC calculated as

$$\frac{1}{J}\sum_{i=1}^{J}\mathbb{E}\left[(x_i(k) - a^\star)^2\right].$$

We conduct simulations of RD-MC using different window sizes $\mathcal{C} \in \{1, 2, 3\}$ and noise variances σ^2, and present the results in Figs. 3-5. We observe that increasing σ^2 results in higher steady-state network-wide error across all experiments. However, the choice of window size profoundly influences RD-MC's efficacy in mitigating communication noise. While RD-MC struggles to converge with $\mathcal{C} = 1$, it maintains convergence with $\mathcal{C} \geq 2$ and increasing $\mathcal{C}$ enhances its robustness to noise. Figs. 2 and 5 show that RD-MC with $\mathcal{C} = 3$ exhibits significantly greater resilience to link noise compared to D-MC, without imposing any additional computational or communication overhead.

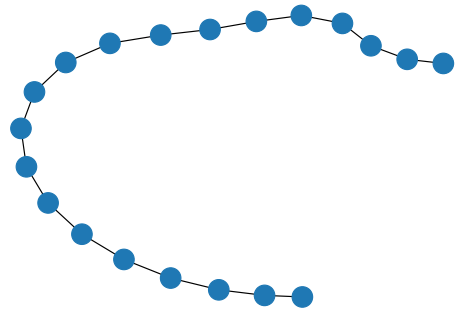

Fig. 6. The considered network with linear topology and $J = 20$ agents.

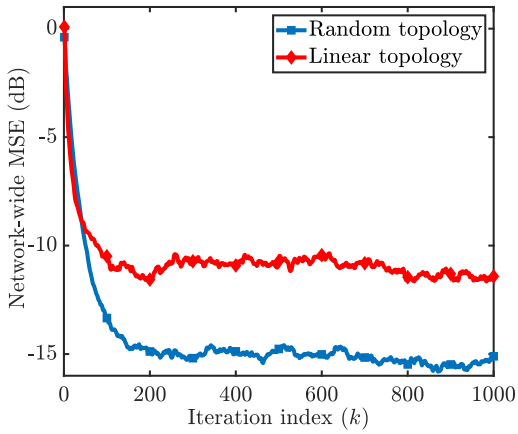

Fig. 7. The impact of network connectivity on the network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ in the presence of link noise with variance $\sigma^2 = 0.1$.

In our final experiment, we assess the sensitivity of RD-MC's performance to network topology in the presence of link noise. We simulate RD-MC with a window size of $\mathcal{C} = 3$ for two networks, namely, the network in 1 and a network with linear topology depicted in Fig. 6. We also set the noise variance to $\sigma^2 = 0.1$. We present the results in Fig. 7. We observe that, with a linear network topology, the network-wide steady-state MSE of RD-MC is larger compared to a network with an arbitrary topology and higher average degree. However, RD-MC continues to perform well in the linear network topology with low connectivity.

V. Conclusion

We developed a distributed algorithm, called noise-robust distributed maximum consensus (RD-MC), to tackle the challenge of identifying the maximum value within an ad-hoc multi-agent network utilizing noisy communication channels. Unlike existing algorithms designed for ad-hoc networks, RD-MC exhibits robustness against additive communication noise. Our extensive simulation results demonstrated the effectiveness of RD-MC in different scenarios.

References

[1] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Automat. Contr.*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[2] A. Papachristodoulou, A. Jadbabaie, and U. Münz, "Effects of delay in multi-agent consensus and oscillator synchronization," *IEEE Trans. Automat. Contr.*, vol. 55, no. 6, pp. 1471–1477, June 2010.

[3] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3863–3877, Sep. 2013.

[4] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, 2009.

[5] S. Liu, M. Fardad, E. Masazade, and P. K. Varshney, "Optimal periodic sensor scheduling in networks of dynamical systems," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3055–3068, 2014.

[6] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[7] L. M. Borges, F. J. Velez, and A. S. Lebres, "Survey on the characterization and classification of wireless sensor network applications," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 4, pp. 1860–1890, 2014.

[8] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Trans. Automat. Contr.*, vol. 64, no. 10, pp. 4035–4049, 2019.

[9] Y. Zhang, Z. Peng, G. Wen, J. Wang, and T. Huang, "Privacy preserving-based resilient consensus for multiagent systems via state decomposition," *IEEE Trans. Control. Netw. Syst.*, vol. 10, no. 3, pp. 1172–1183, 2023.

[10] J. Zhang, J. Lu, J. Liang, and K. Shi, "Privacy-preserving average consensus in multiagent systems via partial information transmission," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, pp. 2781–2791, 2023.

[11] X. Chen, L. Huang, K. Ding, S. Dey, and L. Shi, "Privacy-preserving push-sum average consensus via state decomposition," *IEEE Trans. Automat. Contr.*, vol. 68, no. 12, pp. 7974–7981, 2023.

[12] A.-R. Lagos, H. E. Psillakis, and A. K. Gkesoulis, "Almost-sure finite-time stochastic min-max consensus," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 9, pp. 3509–3513, 2023.

[13] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[14] Y. Zhang and S. Li, "Distributed biased min-consensus with applications to shortest path planning," *IEEE Trans. Automat. Contr.*, vol. 62, no. 10, pp. 5429–5436, Oct. 2017.

[15] J. Hu, Q. Sun, M. Zhai, and B. Wang, "Privacy-preserving consensus strategy for secondary control in microgrids against multilink false data injection attacks," *IEEE Trans. Ind. Inform.*, vol. 19, no. 10, pp. 10 334–10 343, 2023.

[16] H. Rezaee and F. Abdollahi, "Average consensus over high-order multi-agent systems," *IEEE Trans. Automat. Contr.*, vol. 60, no. 11, pp. 3047–3052, Nov. 2015.

[17] G. Oliva, R. Setola, and C. N. Hadjicostis, "Distributed finite-time average-consensus with limited computational and storage capability," *IEEE Trans. Control. Netw. Syst.*, vol. 4, no. 2, pp. 380–391, June 2017.

[18] W. Chen, L. Liu, and G.-P. Liu, "Privacy-preserving distributed economic dispatch of microgrids: A dynamic quantization-based consensus scheme with homomorphic encryption," *IEEE Trans. Smart Grid*, vol. 14, no. 1, pp. 701–713, 2023.

[19] D. Deplano, N. Bastianello, M. Franceschelli, and K. H. Johansson, "A unified approach to solve the dynamic consensus on the average, maximum, and median values with linear convergence," in *Proc. IEEE Conf. Decis. Control*, 2023, pp. 6442–6448.

[20] L. Rong, Y. Kan, X. Xie, G.-P. Jiang, and S. Xu, "Edge-preserving consensus via non-recursive filters: A parallel system design," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 1, pp. 181–185, 2023.

[21] L. Gao, Y. Zhou, X. Chen, R. Cai, G. Chen, and C. Li, "Privacy-preserving dynamic average consensus via random number perturbation," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 4, pp. 1490–1494, 2023.

[22] E. Montijano, J. I. Montijano, C. Sagüés, and S. Martínez, "Robust discrete time dynamic average consensus," *Automatica*, vol. 50, no. 12, pp. 3131–3138, 2014.

[23] M. Franceschelli, A. Giua, and A. Pisano, "Finite-time consensus on the median value with robustness properties," *IEEE Trans. Automat. Contr.*, vol. 62, no. 4, pp. 1652–1667, 2017.

[24] S. Yu, Y. Chen, and S. Kar, "Dynamic median consensus over random networks," in *Proc. IEEE Conf. Decis. Control*, 2021, pp. 5695–5702.

[25] M. Abdelrahim, J. M. Hendrickx, and W. Heemels, "Max-consensus in open multi-agent systems with gossip interactions," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 4753–4758.

[26] A. Nowzari and M. G. Rabbat, "Improved bounds for max consensus in wireless networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 2, pp. 305–319, June 2019.

[27] S. Zhang, C. Tepedelenlioğlu, M. K. Banavar, and A. Spanias, "Max consensus in sensor networks: Non-linear bounded transmission and additive noise," *IEEE Sens. J.*, vol. 16, no. 24, pp. 9089–9098, Dec. 2016.

[28] G. Muniraju, C. Tepedelenlioglu, and A. Spanias, "Analysis and design of robust max consensus for wireless sensor networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 779–791, Dec. 2019.

[29] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed maximum consensus," *IEEE Signal Process. Lett.*, vol. 27, pp. 1839–1843, Oct. 2020.

[30] M. Lippi, A. Furchì, A. Marino, and A. Gasparri, "An adaptive distributed protocol for finite-time infimum or supremum dynamic consensus," *IEEE Control Syst. Lett.*, vol. 7, pp. 401–406, 2023.

[31] D. Deplano, M. Franceschelli, and A. Giua, "Dynamic min and max consensus and size estimation of anonymous multiagent networks," *IEEE Trans. Automat. Contr.*, vol. 68, no. 1, pp. 202–213, 2023.

[32] E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, "Resource-efficient federated learning robust to communication errors," in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.

[33] ——, "Continual local updates for federated learning with enhanced robustness to link noise," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.

[34] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[35] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[36] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, *Decentralized learning for wireless communications and networking*. Springer, 2017.