Asynchronous Online Federated Learning with Reduced Communication Requirements

Francois Gauthier, *Member*, *IEEE*, Vinay Chakravarthi Gogineni, *Senior Member*, *IEEE*, Stefan Werner, *Fellow*, *IEEE*, Yih-Fang Huang, *Life Fellow*, *IEEE*, Anthony Kuh, *Fellow*, *IEEE*

***Abstract*—Online federated learning (FL) enables geographically distributed devices to learn a global shared model from locally available streaming data. Most online FL literature considers a best-case scenario regarding the participating clients and the communication channels. However, these assumptions are often not met in real-world applications. Asynchronous settings can reflect a more realistic environment, such as heterogeneous client participation due to available computational power and battery constraints, as well as delays caused by communication channels or straggler devices. Further, in most applications, energy efficiency must be taken into consideration. Using the principles of partial-sharing-based communications, we propose a communication-efficient asynchronous online federated learning (PAO-Fed) strategy. By reducing the communication load of the participants, the proposed method renders participation more accessible and efficient. In addition, the proposed aggregation mechanism accounts for random participation, handles delayed updates and mitigates their effect on accuracy. We study the first and second-order convergence of the proposed PAO-Fed method and obtain an expression for its steady-state mean square deviation. Finally, we conduct comprehensive simulations to study the performance of the proposed method on both synthetic and real-life datasets. The simulations reveal that in asynchronous settings, the proposed PAO-Fed is able to achieve the same convergence properties as that of the online federated stochastic gradient while reducing the communication by 98 percent.**

***Index Terms*—Asynchronous behavior, communication efficiency, online federated learning, partial-sharing-based communications, nonlinear regression.**

I. Introduction

A myriad of intelligent devices, such as smartphones, smartwatches, and smart home appliances, are becoming an integral part of our daily lives, and an enormous amount of data is available on those devices. Unfortunately, this data is primarily unused, and we need to develop tools that can process this data to extract information that can improve our daily lives while, at the same time, ensuring our privacy. Federated learning (FL) [1] provides an adaptive large-scale collaborative learning framework suitable for this task. In FL, a server aggregates information received from distributed devices referred to as clients to train a global shared model; the clients do not share any private data with the server, only their local model parameters or gradients learned from this data [1], [2]. When data becomes progressively available to clients, it is possible to perform decentralized learning in real-time (implementing, e.g., online FL [3]) for applications that include environmental monitoring and condition monitoring using sensor networks [4], internet-of-medical-things (IoMT) based healthcare applications [5] (e.g., cardio rhythm monitoring), and autonomous vehicles [6]. In online FL, the server aggregates the local models learned on the streaming data of the clients [7]. However, in many applications, the participating clients might have heterogeneous energy supply and limited communication capacity that can be intermittently unavailable or subject to failure. Therefore, such edge devices cannot participate in typical federated learning implementations.

In most real-world implementations of FL, it is essential to consider statistical heterogeneity, system heterogeneity, and imperfect communication channels between clients and the server. Statistical heterogeneity implies that data are imbalanced and not independent and identically distributed (non-i.i.d.) [8] across devices, while system heterogeneity refers to their various computational and communication capacities. Finally, imperfect communication channels cause delays in the exchanged messages. Although many FL approaches can handle statistical heterogeneity, there is relatively little research addressing the remaining complications above. In particular, existing FL methods commonly assume a best-case scenario concerning the client availability and performance as well as perfect channel conditions [1], [9]–[18]. However, several additional aspects need attention for efficient FL in a realistic setting. First, clients cannot be expected to have the same participation frequency, e.g., due to diverse resource constraints, channel availability, or concurrent solicitations [19]–[22]. Furthermore, clients may become unavailable for a certain period during the learning process, i.e., some clients are malfunctioning or not reachable by the server [19], [20]. In addition, physical constraints such as distance or overload introduce delays in the communication between the clients and the server, making their contribution arrive later than expected [20]–[23]. These constraints, frequently occurring in practice, impair the efficiency of FL and complicate the design of methods tailored for asynchronous settings [19]–[25].

This work was supported by the Research Council of Norway.

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

Vinay Chakravarthi Gogineni was with the Norwegian University of Science and Technology, Trondheim, Norway. Now, he is with the SDU Applied AI and Data Science, the Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark (e-mail: vigo@mmmi.sdu.dk).

Yih-Fang Huang is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: huang@nd.edu).

Anthony Kuh is with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI 96822 USA (e-mail: kuh@hawaii.edu).
Anthony Kuh acknowledges support in part by NSF Grant 2142987

A conference precursor was published in the proceedings of the IEEE International Conference on Communications, 2022.

Energy efficiency is an essential aspect of distributed machine learning algorithms and one of the original motivations for FL [26]. The communication of high-dimensional models is energy-onerous for distributed devices. For this reason, it is crucial to cut the communication cost for clients [14], [18]. Further, such reduction can facilitate more frequent participation of resource-constrained devices, or stragglers, in the learning process. In addition, in asynchronous settings, where power and communication are restricted, ensuring communication efficiency reduces the risk of bottlenecks in the communication channels or power-related shutdown of clients due to excessive resource usage.

We can find a considerable amount of research in the literature on communication-efficient FL [13]–[18], [27]–[30] and asynchronous FL [19]–[25], [31]–[39]; however, only a few works consider both aspects within the same framework. The classical federated averaging (FedAvg) [18], developed for ideal conditions, reduces the communication cost by selecting a subset of the clients to participate at each iteration. In a perfect setting, this allows clients to space out their participation while maintaining a consistent participation rate. In asynchronous settings, however, clients may already participate sporadically because of their inherent limitations. Hence, subsampling comes with an increased risk of discarding valuable information. The work of [34] proposes a smart selection system to address this issue, but this is associated with an additional computational burden on the server, and only lessens the information loss associated with client scheduling. The works in [23], [28] reduce communication in uplink via compressed client updates. Aside from the accuracy penalty associated with the sparsification and projection used, the resulting extra computational burden on the clients of these non-trivial operations is not appealing for resource-constrained clients. Moreover, the work in [28] did not consider asynchronous settings. Although the work in [23] considers various participation frequencies for the clients, it assumes they are constant throughout the learning process. The works in [24], [39] reduce the communication load of clients in asynchronous settings; however, they are specific to neural networks and lack mathematical analysis. In addition, the considered asynchronous settings do not include communication delays. We note that structure and sketch update methods suffer the same accuracy cost and additional computational burden as compressed updates; and in all three, the simultaneous unpacking of all the received updates at the server can form a computational bottleneck. Another option explored recently for distributed learning is the partial-sharing of model parameters [40]. The partial-sharing-based online FL (PSO-Fed) algorithm [27] features reduced communications in FL, but only in ideal settings.

This paper proposes a partial-sharing-based asynchronous online federated learning (PAO-Fed) algorithm for nonlinear regression in asynchronous settings. The proposed approach reduces communication significantly while retaining fast convergence. In order to perform nonlinear regression, we use random Fourier feature space (RFF) [41], [42], where inner products in a fixed-dimensional space approximate the nonlinear relationship between the input and output data. Consequently, given the constant communication and computational load, RFF is more suitable for decentralized learning than traditional dictionary-based solutions whose model order depends on the sample size. In addition, RFF presents the advantage of being resilient to model change during the learning process, which is key in online FL. Further, we implement partial-sharing-based communications to reduce the communication load of the algorithm. Compared to the other available methods, partial-sharing does not incur an additional computational load and only transfers a fraction of the model parameters between clients and the server. This allows clients to participate more frequently while maintaining minimal communication without additional computational burden. The proposed aggregation mechanism handles delayed updates and calibrates their contribution to the global shared model. We provide first- and second-order convergence analyses of the PAO-Fed algorithm in a setting where client participation is random, and communication links suffer delays. Finally, we conducted simulation studies using synthetic and real-life data to examine and compare the proposed algorithm with existing methods.

The paper is organized as follows. Section II introduces FL for nonlinear regression as well as partial-sharing-based communications. Section III defines the considered asynchronous settings and introduces the proposed method. Section IV provides the first and second-order convergence analysis of the PAO-Fed algorithm. Section V presents numerical results for the proposed method and compares it with existing ones. Finally, Section VI concludes the paper.

II. Preliminaries and Problem Formulation

This section presents the nonlinear regression problem in the context of FL. Further, a brief overview of the most closely related existing algorithms is proposed. Finally, the behavior of partial-sharing-based communications is presented.

A. Online Federated Learning for Nonlinear Regression

We consider a federated network where a server is connected to a set $\mathcal{K}$ of $|\mathcal{K}| = K$ geographically distributed devices, referred to as clients. In the online FL setting [3], used when real-time computation is desirable, the entire dataset of a client is not immediately available. Instead, it is made available to the client progressively throughout the learning process. We denote the continuous streaming data appearing at client $k \in \mathcal{K}$ at iteration n by $\mathbf{x}_{k,n} \in \mathbb{R}^L$, the corresponding output $y_{k,n}$ is given by:

$$y_{k,n} = f(\mathbf{x}_{k,n}) + \eta_{k,n}, \tag{1}$$

where $f(\cdot) : \mathbb{R}^L \longrightarrow \mathbb{R}$ is a nonlinear model and $\eta_{k,n}$ is the observation noise. The objective is that the server and clients learn a global shared nonlinear model from the data available at each client, without this data being shared amongst clients or with the server. To this aim, the clients periodically share with the server their local model, learned from local data, and the server shares its global model with the clients.

Several adaptive methods can be used to handle nonlinear model estimation problems, e.g., [41]–[44]. The conventional

kernel least-mean-square (KLMS) algorithm [43] is one of the most popular choices but suffers from a growing dimensionality problem, leading to prohibitive computation and communication requirements. Coherence-check-based methods [44] sparsify the original dictionary by selecting the regressors using a coherence measure. Although feasible, this method is not attractive for online FL, especially in asynchronous settings, since it requires that each new dictionary element be made available throughout the network, inducing a significant communication overhead, especially if the underlying model changes. The random Fourier feature (RFF) space method [41], [42] approximates the kernel function evaluation by projecting the model into a pre-selected fixed-dimensional space. The selected RFF space does not change throughout the computation, and, given that the chosen dimension is large enough, the obtained linearizations can be as precise as desired. Therefore, we use RFF-based KLMS for the nonlinear regression task, as it is data-independent, resilient to model change, and does not require extra communication overhead, unlike conventional or coherence-check-based KLMS.

In the following, we approximate the nonlinear model by projecting it on a D-dimensional RFF-space, in which the function $f(\cdot)$ is approximated by the linear model $\mathbf{w}^*$. To estimate the global shared model using the local streaming data, we solve the following problem:

$$\min_{\mathbf{w}} \mathcal{J}(\mathbf{w}), \tag{2}$$

where $\mathcal{J}(\mathbf{w})$ is given by:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K}\sum_{k\in\mathcal{K}} \mathcal{J}_k(\mathbf{w}) \tag{3}$$
$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E}[|y_{k,n} - \mathbf{w}^\mathsf{T}\mathbf{z}_{k,n}|^2],$$

and $\mathbf{z}_{k,n}$ is the mapping of $\mathbf{x}_{k,n}$ into the D-dimentional RFF-space.

B. Existing Algorithms

The Online-Fed algorithm, an online FL version of the conventional FedAvg algorithm [18] solves the above estimation problem as follows. At each iteration, n, the server selects a subset of the clients $\mathcal{K}_n \subseteq \mathcal{K}$ to participate in the learning task and shares the global shared model $\mathbf{w}_n$ with them. Then the selected clients in $\mathcal{K}_n$ perform the local learning process on their local estimates $\mathbf{w}_{k,n}$ as

$$\mathbf{w}_{k,n+1} = \mathbf{w}_n + \mu\mathbf{z}_{k,n}e_{k,n}, \tag{4}$$

where μ is the learning rate and $e_{k,n}$ is the *a priori* error of the global model on the local data given by

$$e_{k,n} = y_{k,n} - \mathbf{w}_n^\mathsf{T}\mathbf{z}_{k,n}. \tag{5}$$

The clients then share their updated models with the server, which aggregates them as

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{K}_n|}\sum_{k\in\mathcal{K}_n} \mathbf{w}_{k,n+1}, \tag{6}$$

where $|\mathcal{K}_n|$ denotes the cardinality of $\mathcal{K}_n$. In the particular case where $\forall n, \mathcal{K}_n = \mathcal{K}$, i.e., all the clients participate at each iteration, we denote the algorithm Online-FedSGD.

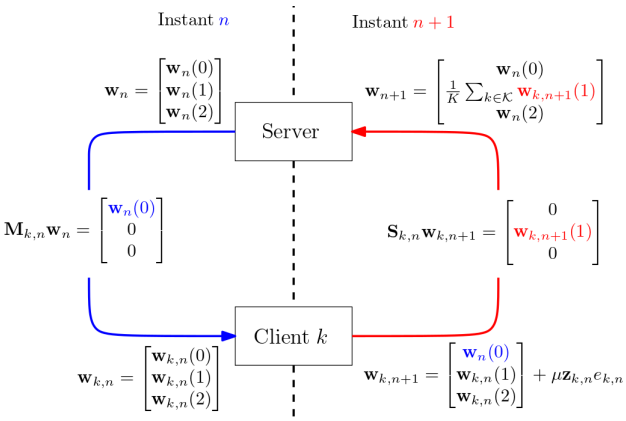


Fig. 1: Partial sharing in a simple scenario.

The PSO-Fed algorithm proposed in [27] uses partial-sharing-based communications to reduce further the communication load of the Online-Fed algorithm. Additionally, PSO-Fed allows clients who are not participating in the current iteration to perform local learning on their new data. By doing so, this algorithm drastically reduces communication without compromising the convergence speed.

C. Partial-sharing-based Communications

In partial-sharing-based communications, as defined in [40], the server and the clients exchange only a portion of their respective models instead of the entire model. The portion is extracted prior to communication by multiplication with a diagonal selection matrix with main diagonal elements being either 0 or 1, where the locations of the latter specify the model parameters to share. This operation is computationally trivial and, therefore, does not induce delay on the communication, unlike compressed update methods, e.g., [23], [28]. Here, m denotes the number of nonzero elements in the selection matrices; this is the number of model parameters shared at each iteration. The selection matrix $\mathbf{M}_{k,n}$ is used for server-to-client communication at time n and the selection matrix $\mathbf{S}_{k,n}$ for client k's response, as can be seen on Fig. 1 where the simple case where $m = D/3$ is illustrated.

The usual aggregation step in (6) cannot be used with partial-sharing-based communications, and needs to be adapted. The expression of $\mathbf{w}_{n+1}$ in Fig. 1 is the aggregation step for coordinated partial-sharing in perfect settings. Coordinated partial-sharing is the special case where all clients send the same portion of the model at a given iteration.

For the clients to participate in the learning of the whole model, and to ensure consistency across models, it is necessary that the selection matrices evolve. To this aim, we set:

$$\text{diag}(\mathbf{M}_{k,n+1}) = \text{circshift}(\text{diag}(\mathbf{M}_{k,n}), m) \tag{7}$$
$$\mathbf{S}_{k,n} = \mathbf{M}_{k,n+1} \tag{8}$$

where circshift denotes a circular shift operator. $\mathbf{S}_{k,n}$ is set to be equal to $\mathbf{M}_{k,n+1}$ rather than $\mathbf{M}_{k,n}$ in order to share a portion of the client's model further refined by the local learning process. As can be seen in Fig. 1, $\mathbf{w}_{k,n+1}(0)$ contains information from a single local learning step of client k, while

$\mathbf{w}_{k,n+1}(1)$ contains information from three (since it is equal to $\mathbf{w}_{n-2}(1)$ refined thrice by the local learning process). For smaller values of m, the additional information is greater, but the original value of the portion is also older.

D. Motivation

The above-mentioned algorithms offer significant communication savings but do not consider practical network environments and client resources. When performing federated learning in real-world applications, clients may be unavailable for various reasons, message exchanges may be delayed or blocked, and straggler clients may be present. For this reason, it is essential to tailor the developed algorithms to asynchronous settings. Those environments impact the algorithm design and optimization. For instance, we will see that many choices made for the PSO-Fed algorithm in an ideal setting are unsuitable for asynchronous settings.

III. Proposed Method

This section presents the proposed communication-efficient Partial-sharing-based Asynchronous Online Federated Learning (PAO-Fed) algorithm and the asynchronous settings for which it is developed.

A. Asynchronous Settings

The following features are necessary for an online FL method to operate successfully in realistic environments.

- The capability to handle non-IID and unevenly distributed data.
- The capability to handle heterogeneous, time-varying, and unpredictable client participation, including possible downtimes. In most real-world applications, the computation and communication capacity of a specific task are heterogeneous and time-varying. In addition, clients are unreliable as they may experience many issues (low battery, software failure, physical threat, etc.). Moreover, when dealing with many clients, an infrequently occurring failure is likely experienced at least once. Lastly, it is unlikely for the server to know in advance when a client will be unavailable or suffer a failure, so even the most reliable clients may suffer downtimes.
- The capability to weigh the importance of delayed messages. Model parameters with the same timestamp may arrive at different instants at the server. In practice, communication channels are unreliable, and although most messages arrive within a short window, some may take longer, especially when the communication channels are strained. In addition, straggler clients may not be able to complete the learning task in the given time frame, and although their update may not be delayed, it will arrive late at the server. Therefore, the developed method must be robust to a delay spread in the received parameters.
- The capability to reduce the likelihood of straggler-like behavior. Resource-constrained devices may induce latency or run out of power, resulting in reduced information sharing. It is, therefore, not sufficient to consider stragglers-like behavior [21]; it is preferable to improve their operational environments, e.g., by reducing their computation and communication load.

The first step to address those challenges is to model the presented behaviors properly. To this aim, the clients' participation is modeled by participation probabilities. At an iteration n, the Bernoulli trial on the probability $p_{k,n}$ dictates if client k is able to participate. The use of probabilities for participation allows the model to address all the behaviors presented in the second point, unlike the commonly used tier-based model for participation (e.g., [23]), where each tier is expected to behave optimally given a tailored frequency. In fact, heterogeneity and time-dependency are handled by giving clients various evolving probabilities $p_{k,n}$, and unpredictability and downtimes are naturally present when ensuring that all probabilities are lower than one. In addition, any communication sent by a client to the server may be delayed by one or several iterations.

With the proposed model, the limitations of real-world applications and the heterogeneity of the computational power and communication capacity of the available devices are taken into consideration. Those asynchronous settings diminish the potential performance of an FL method, especially in the online setting, where data not shared in time is lost. The proposed method ensures communication efficiency and, in turn, some extend energy efficiency in order, notably, to avoid downward cycles in the asynchronous behavior of the participating clients. For instance, a weaker device may take longer to perform the learning process, struggle to send a long message, and need time to save enough power to participate again. Therefore, performing less computation and exchanging shorter messages will reduce the burden on the clients and the communication channels, making further complications or delays less likely. For this reason, a communication-efficient method tailored for the asynchronous settings can perform above its expectations in a real-life scenario.

B. Delayed Updates

The consequence of the introduced delays is that not all updates sent by clients participating at a given iteration will arrive at the server simultaneously. Precisely, we denote $\mathcal{K}_n$ the set of all the clients who sent an update that arrived at the server at iteration n. This set can be decomposed as:

$$\mathcal{K}_n = \bigcup_{l=0}^{\infty} \mathcal{K}_{n,l} \tag{9}$$

where $\mathcal{K}_{n,l}$ denotes the set of the clients who sent an update at iteration $n-l$ which reached the server at iteration n, the subscript l corresponds to the number of iterations during which the update was delayed. A delayed update will naturally lose value the longer it is delayed, as it becomes outdated. To improve the learning accuracy of the proposed algorithm, we propose a weight-decreasing mechanism that weights down delayed updates. By doing so, we diminish the negative impact of outdated data on the convergence. This mechanism is different from age of update mechanisms found in [45]–[47] where weights are dictated by the amount of

data, independently from communication delays. We denote $\alpha_l \in [0,1]$ the weight given to the updates sent by the clients in $\mathcal{K}_{n,l}$. This work only considers potential delays in client-to-server communications. Although delays in server-to-client communications also affect performance, they do not require further modification of the aggregation mechanism. Additionally, such delays are less likely to occur in IoT/CPS applications, where the server is typically a powerful device that broadcasts messages to resource-constrained clients.

C. PAO-Fed

The proposed PAO-Fed algorithm is tailored to the asynchronous settings; notably, its novel aggregation step is designed to handle delayed updates. PAO-Fed makes use of all the available clients at a given iteration. To reduce the amount of communication associated with the learning, it uses partial-sharing-based communications, which is well adapted to the asynchronous settings as it does not lay any additional computational burden on the participating clients. Further, the aggregation step is refined with a weight-decreasing mechanism to diminish the negative impact of delayed updates on convergence. The algorithm is as follows.

During iteration n, the server shares a portion of the global shared model, i.e., $\mathbf{M}_{k,n}\mathbf{w}_n$, to all the available clients. The selection matrix $\mathbf{M}_{k,n}$ dictates which portion of the model is sent to client k. The available client k receives its portion of the global shared model, and uses it to update its local model, the new local model is given by $\mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n}$. Afterward, the available client k refines its local model by performing the process of local learning on its newly available data as follows.

$$\mathbf{w}_{k,n+1} = \mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n} + \mu\mathbf{z}_{k,n}e_{k,n}, \quad (10)$$

where $e_{k,n}$ is the *a priori* error of the local model on the local data given by:

$$e_{k,n} = y_{k,n} - (\mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n})^{\mathsf{T}}\mathbf{z}_{k,n}. \quad (11)$$

When a client is unavailable at a given iteration but receives new data and is not malfunctioning, it refines its local model autonomously. For example, this can be a case where a client is well functioning but does not have communication capacity at the time. This local update step, identical to the one used in [27], is performed as

$$\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu\mathbf{z}_{k,n}e_{k,n}, \quad (12)$$

where $e_{k,n}$ in that case is given by

$$e_{k,n} = y_{k,n} - \mathbf{w}_{k,n}^{\mathsf{T}}\mathbf{z}_{k,n}. \quad (13)$$

This update is computationally trivial for most devices and does not involve communication. Its purpose is for the client to share better-refined model parameters during the next participation. Naturally, this additional information only reaches the server if the model parameters are not overwritten before being communicated, further motivating the choice of selection matrices made in (8).

After this local update step, all available clients communicate a portion of their updated local models to the server. A client k communicates the portion of the model dictated by the selection matrix $\mathbf{S}_{k,n}$, that is, $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$. Those updates may arrive at the present iteration or at a later one if they are delayed.

At the server, we consider the previously introduced set $\mathcal{K}_n$ consisting of the clients whose updates arrive at the current iteration. This set comprises the sets $\mathcal{K}_{n,l}, 0 \leqslant l < \infty$ that consist of the clients whose update was sent at iteration $n-l$ and arrives at the current iteration. The set $\mathcal{K}_{n,0}$ consists of the available clients at the current iteration whose updates have not been delayed. Note that a client may appear twice in the set $\mathcal{K}_n$ if two of its updates arrive at the same iteration. The deviation from the current global model engendered by the updates received from a non-empty set $\mathcal{K}_{n,l}$ is given by

$$\Delta_{n,l} = \frac{1}{|\mathcal{K}_{n,l}|}\sum_{k\in\mathcal{K}_{n,l}} \mathbf{S}_{k,n-l}(\mathbf{w}_{k,n+1-l} - \mathbf{w}_n). \quad (14)$$

If a set $\mathcal{K}_{n,l}$ is empty, we set by convention $\Delta_{n,l} = \mathbf{0}$

The aggregation step of the proposed algorithm uses a weight-decreasing mechanism for delayed updates. A client's participation that has been delayed for l iterations will be given the weight $\alpha_l \in [0,1]$. By convention, we set the weight of the updates that are not delayed to $\alpha_0 = 1$. The resulting aggregation mechanism is given by:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \sum_{l=0}^{\infty}\alpha_l\Delta_{n,l}. \quad (15)$$

When $l > l_{\max}$, the maximum effective delay, the aggregation mechanism discards the corresponding updates by setting $\alpha_l = 0,\ l > l_{\max}$. It is possible to replace ∞ by $l_{\max}$ in (15) without changing the aggregation mechanism. Note that in the eventuality where several updates from clients in $\mathcal{K}_n$ update the same model parameter, only the most recent updates are considered, the selection matrices of the remaining updates are adjusted accordingly prior to computing (15). The resulting algorithm is presented in Algorithm 1.

D. Partial-sharing in Asynchronous Settings

In coordinated partial sharing, all participating clients share the same portion of the model so that the server's model is aggregated from a large number of clients, thus improving accuracy. For this reason, coordinated partial-sharing is used in most algorithms assuming perfect settings. In practice, however, delayed updates partially overwrite the previously aggregated portion, as can be seen in (15), thus negating the added value of coordination.

To tackle this issue, one can either use a weight-decreasing mechanism such as the one presented above or use uncoordinated partial sharing. Besides, uncoordinated partial-sharing is ideal when dealing with underlying model changes, as the server's model uniformly steers towards its new steady-state value, instead of doing so portion by portion as with coordinated partial-sharing.

IV. Convergence Analysis

In this section, we examine the convergence behavior of the proposed PAO-Fed algorithm that uses partial-sharing-based

Algorithm 1 PAO-Fed

Initialization: $\mathbf{w}_0$ and $\mathbf{w}_{k,0}, k \in \mathcal{K}$ set to $\mathbf{0}$
Procedure at Local client k
for iteration $n = 1, 2, \ldots, N$ **do**
  if Client k receives new data at time n **then**
    if k is available **then**
      Receive $\mathbf{M}_{k,n}\mathbf{w}_n$ from the server.
      Compute $\mathbf{w}_{k,n+1}$ as in (10).
      Share $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$ with the server.
    else
      Update $\mathbf{w}_k$ as in (12).
    end if
  end if
end for
Procedure at Central Server
for iteration $n = 1, 2, \ldots, N$ **do**
  Receive client updates from subset $\mathcal{K}_n \subset \mathcal{K}$.
  Compute $\mathbf{w}_{n+1}$ as in (15).
  Share $\mathbf{M}_{k,n+1}\mathbf{w}_{n+1}$ with the available clients.
end for

communications and evolves in asynchronous settings such as the ones presented in Section III. We prove mathematically that the proposed PAO-Fed algorithm converges to the exact model in the RFF space and exhibits stable extended mean square displacement under certain general assumptions.

Before proceeding to the analysis, we introduce auxiliary matrices to express an entire iteration of the algorithm in the matrix form. Similar to [48], we define the extended model vector $\mathbf{w}_{e,n}$, local update matrix $\mathbf{A}_{e,n}$, and mapping of the data into the RFF-space $\mathbf{Z}_{e,n}$ as

$$\begin{aligned}\mathbf{w}_{e,n} &= \text{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n} \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \\ &\qquad \ldots, \mathbf{w}_{K,n-1}, \ldots, \mathbf{w}_{1,n-l_{\max}}, \ldots, \mathbf{w}_{K,n-l_{\max}}\}, \\ \mathbf{A}_{e,n} &= \text{blockdiag}\{\mathbf{A}_n, \mathbf{I}_{DK}, \ldots, \mathbf{I}_{DK}\}, \\ \mathbf{Z}_{e,n} &= \text{blockdiag}\{\mathbf{Z}_n, \mathbf{0}_{DK\times K}, \ldots, \mathbf{0}_{DK\times K}\}, \end{aligned} \tag{16}$$

with

$$\mathbf{A}_n = \begin{bmatrix} \mathbf{I} & \mathbf{0}_D & \cdots & \mathbf{0}_D \\ a_{1,n}\mathbf{M}_{1,n} & \mathbf{I} - a_{1,n}\mathbf{M}_{1,n} & & \vdots \\ \vdots & \mathbf{0}_D & \ddots & \mathbf{0}_D \\ a_{K,n}\mathbf{M}_{K,n} & \vdots & & \mathbf{I} - a_{K,n}\mathbf{M}_{K,n} \end{bmatrix},$$

$$\mathbf{Z}_n = \text{blockdiag}\{\mathbf{0}_D, \mathbf{z}_{1,n}, \ldots, \mathbf{z}_{K,n}\}, \tag{17}$$

where $a_{k,n} = 1$ if the client k is available at iteration n and 0 otherwise, col$\{\cdot\}$ and blockdiag$\{\cdot\}$ represent column-wise stacking and block diagonalization operators, respectively. We can now express the extended observation vector $\mathbf{y}_{e,n} = \text{col}\{0, y_{1,n}, y_{2,n}, \ldots, y_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$ as

$$\mathbf{y}_{e,n} = \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{w}_e^* + \boldsymbol{\eta}_{e,n}, \tag{18}$$

where $\mathbf{w}_e^* = \mathbf{1}_{(K+1)l_{\max}+1} \otimes \mathbf{w}^*$ and the extended observation noise $\boldsymbol{\eta}_{e,n} = \text{col}\{0, \eta_{1,n}, \eta_{2,n}, \ldots, \eta_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$. We then can express the extended estimation error vector as

$$\mathbf{e}_{e,n} = \mathbf{y}_{e,n} - \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{A}_{e,n}\mathbf{w}_{e,n}. \tag{19}$$

Therefore, the recursion of the extended model vector $\mathbf{w}_{e,n}$ is given by

$$\mathbf{w}_{e,n+1} = \mathbf{B}_{e,n}(\mathbf{A}_{e,n}\mathbf{w}_{e,n} + \mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n}), \tag{20}$$

with

$$\mathbf{B}_{e,n} = \begin{bmatrix} \mathbf{B}_n & \mathbf{B}_{0,n} & \mathbf{0}_{D\times DK} & \mathbf{B}_{1,n} & \cdots & \mathbf{B}_{l_{\max},n} \\ \mathbf{0}_{D\times 1} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \mathbf{0}_{DK} \\ \mathbf{0}_{D\times 1} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} \end{bmatrix}$$

$$\begin{aligned}\mathbf{B}_n &= \mathbf{I} - \sum_{l=0}^{l_{\max}} \alpha_l \sum_{k\in\mathcal{K}_{n,l}} \frac{b_{k,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{k,n-l} \\ \mathbf{B}_{l,n} &= [\frac{\alpha_l b_{1,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{1,n-l}, \cdots, \frac{\alpha_l b_{K,n,1}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{K,n-l}]. \end{aligned} \tag{21}$$

where $b_{k,n,l} = 1$ if $k \in \mathcal{K}_{n,l}$ and 0 otherwise.

In the following, we present a detailed convergence analysis of the PAO-Fed algorithm both in mean and mean-square senses. To this end, we make the following assumptions:

Assumption 1: The mapped data vectors $\mathbf{z}_{k,n}$ are drawn at each time step from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{z}_{k,n}\mathbf{z}_{k,n}^{\mathsf{T}}]$.

Assumption 2: The observation noise $\eta_{k,n}$ is assumed to be zero mean white Gaussian, and independent of all input and output data.

Assumption 3: At each client, the model parameter vector is assumed to be independent of the input data.

Assumption 4: The selection matrices are assumed independent from each other, and of any other data.

Assumption 5: The learning rate μ is small enough for terms involving higher-order powers of μ to be neglected.

It is important to note that no assumption is taken on the α_l variables because $l_{\max}$ is a fixed value in our analysis.

A. First-order Analysis

This subsection examines the mean convergence of the proposed PAO-Fed algorithm.

Theorem 1. *Let **Assumptions 1–4** hold true. Then, The proposed PAO-Fed converges in mean if and only if*

$$0 < \mu < \frac{2}{\max\limits_{\forall k,i} \lambda_i(\mathbf{R}_k)}. \tag{22}$$

Proof: Denoting the model error vector $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^* - \mathbf{w}_{e,n}$, and using the fact that $\mathbf{w}_e^* = \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_e^*$ (by construction, all rows in $\mathbf{B}_{e,n}$ and $\mathbf{A}_{e,n}$ sum to 1), from (20), we can recursively express $\tilde{\mathbf{w}}_{e,n}$ as

$$\begin{aligned}\tilde{\mathbf{w}}_{e,n+1} &= \mathbf{w}_e^* - \mathbf{w}_{e,n+1} \\ &= \mathbf{w}_e^* - \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n} \\ &= \mathbf{B}_{e,n}\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n} \\ &\quad - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}}(\mathbf{w}_e^* - \mathbf{A}_{e,n}\mathbf{w}_{e,n}) \\ &= \mathbf{B}_{e,n}(\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}})\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} \\ &\quad - \mu\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n}. \end{aligned} \tag{23}$$

Taking the statistical expectation $\mathbb{E}[\cdot]$ on both sides of (23) and using **Assumptions** 1–4, we obtain

$$\begin{aligned}\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] &= \mathbb{E}[\mathbf{B}_{e,n}]\mathbb{E}[\mathbf{I}-\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}}]\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}]\\ &= \mathbb{E}[\mathbf{B}_{e,n}](\mathbf{I}-\mu\mathbf{R}_e)\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}], \end{aligned} \tag{24}$$

where $\mathbf{R}_e = \text{blockdiag}\{\mathbf{0}_D, \mathbf{R}_1, \mathbf{R}_1, \cdots, \mathbf{R}_K, \mathbf{0}_{DKl_{\max}}\}$. The quantities $\mathbb{E}[\mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{B}_{e,n}]$ are evaluated in Appendix A.

Further, we consider the vectors and matrices reduced to the subspace between the index $D+1$ and $D(K+1)$. We denote the reduction of $\mathbf{x}$ by $\mathbf{x}|_{\text{sel}}$. Using the reduced definitions, (24) becomes: $\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}|_{\text{sel}}] = (\mathbf{I}-\mu\mathbf{R}_e|_{\text{sel}})\mathbb{E}[\mathbf{A}_{e,n}|_{\text{sel}}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$, where the block $\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}$ is defined as a linear sequence of order 1 in a normed algebra. To prove the convergence of $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$, we use the properties of the block maximum norm [49]. From Appendix A, we have $||\mathbb{E}[\mathbf{A}_{e,n}|_{\text{sel}}]||_{b,\infty} = 1$. Then the convergence condition reduces to $||\mathbf{I}-\mu\mathbf{R}_e|_{\text{sel}}||_{b,\infty} < 1$, equivalently, $|1-\mu\lambda_i(\mathbf{R}_k)| < 1$, $\forall k, i$, where $\lambda_i(\cdot)$ is the ith eigenvalue of the argument matrix. This leads to the convergence condition given by (22).

□

B. Second-order Analysis

In this subsection, we present the second-order analysis of the proposed PAO-Fed algorithm. For the given arbitrary positive semidefinite matrix $\mathbf{\Sigma}$, the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ is given by $||\tilde{\mathbf{w}}_{e,n}||^2_{\mathbf{\Sigma}} = \tilde{\mathbf{w}}_{e,n}^{\mathsf{T}}\mathbf{\Sigma}\tilde{\mathbf{w}}_{e,n}$. From (23), we can obtain

$$\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\mathbf{\Sigma}}] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\mathbf{\Sigma}'}] + \mu^2\mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\mathbf{Y}_n^{\mathbf{\Sigma}}\boldsymbol{\eta}_{e,n}], \tag{25}$$

where the cross terms are null under **Assumption 2** and the matrices $\mathbf{\Sigma}'$ and $\mathbf{Y}^{\mathbf{\Sigma}}$ are given by

$$\begin{aligned}\mathbf{\Sigma}' = \mathbb{E}[&\mathbf{A}_{e,n}^{\mathsf{T}}(\mathbf{I}-\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}})\mathbf{B}_{e,n}^{\mathsf{T}}\,\mathbf{\Sigma}\\ &\mathbf{B}_{e,n}(\mathbf{I}-\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}})\mathbf{A}_{e,n}],\end{aligned} \tag{26}$$

$$\mathbf{Y}_n^{\mathbf{\Sigma}} = \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{B}_{e,n}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}. \tag{27}$$

Using **Assumption 3** and the properties of the block Kronecker product, and the block vectorization operator bvec$\{\cdot\}$ [50], we can establish a relationship between $\boldsymbol{\sigma} = \text{bvec}\{\mathbf{\Sigma}\}$ and $\boldsymbol{\sigma}' = \text{bvec}\{\mathbf{\Sigma}'\}$ as

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{28}$$

where

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{I}\otimes_b\mathbf{R}_e)\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{R}_e\otimes_b\mathbf{I})\boldsymbol{\mathcal{Q}}_{\mathbf{A}},$$

where the higher-order powers of μ are neglected under **Assumption 5**. In the above

$$\begin{aligned}\boldsymbol{\mathcal{Q}}_{\mathbf{A}} &= \mathbb{E}[\mathbf{A}_{e,n}\otimes_b\mathbf{A}_{e,n}],\\ \boldsymbol{\mathcal{Q}}_{\mathbf{B}} &= \mathbb{E}[\mathbf{B}_{e,n}\otimes_b\mathbf{B}_{e,n}].\end{aligned} \tag{29}$$

In Appendix B, we evaluate the matrices $\boldsymbol{\mathcal{Q}}_{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}_{\mathbf{B}}$, and prove that all their entries are real, non-negative, and add up to unity on each row. This implies that both matrices are right-stochastic, and thus, their spectral radius is equal to one.

We will now evaluate the term $\mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\mathbf{Y}_n^{\mathbf{\Sigma}}\boldsymbol{\eta}_{e,n}]$ as follows:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\mathbf{Y}_n^{\mathbf{\Sigma}}\boldsymbol{\eta}_{e,n}] &= \mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{B}_{e,n}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n}]\\ &= \mathbb{E}[\text{trace}(\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{B}_{e,n}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n})]\\ &= \text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\boldsymbol{\eta}_{e,n}]\mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{B}_{e,n}^{\mathsf{T}}]\mathbf{\Sigma})\\ &= \text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\mathbf{\Phi}_n\mathbf{B}_{e,n}^{\mathsf{T}}]\mathbf{\Sigma}),\end{aligned} \tag{30}$$

with $\mathbf{\Phi}_n = \mathbf{Z}_{e,n}\mathbf{\Lambda}_\eta\mathbf{Z}_{e,n}^{\mathsf{T}}$, where $\mathbf{\Lambda}_\eta = \mathbb{E}[\boldsymbol{\eta}_{e,n}^{\mathsf{T}}\boldsymbol{\eta}_{e,n}]$ is a diagonal matrix having the noise variances of all clients on its main diagonal. Note that we used **Assumption 2** in the last line of (30). Finally, using the properties of the block Kronecker product, we have

$$\text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\mathbf{\Phi}_n\mathbf{B}_{e,n}^{\mathsf{T}}]\mathbf{\Sigma}) = \mathbf{h}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{31}$$

with

$$\begin{aligned}\mathbf{h} &= \text{bvec}\{\mathbb{E}[\mathbf{B}_{e,n}\mathbf{\Phi}_n\mathbf{B}_{e,n}^{\mathsf{T}}]\}\\ &= \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\text{bvec}\{\mathbb{E}[\mathbf{\Phi}_n]\}.\end{aligned} \tag{32}$$

Combining (25), (28), and (30), we can write the recursion for the weighted extended mean square displacement of the PAO-Fed algorithm as:

$$\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\mathcal{F}}^{\mathsf{T}}\boldsymbol{\sigma}\}}] + \mu^2\mathbf{h}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{33}$$

where bvec$^{-1}\{\cdot\}$ represents the reverse operation of block vectorization.

Theorem 2. *Let **Assumptions 1–5** hold true. Then, the PAO-Fed algorithm exhibits stable mean square displacement if and only if:*

$$0 < \mu < \frac{1}{\max\limits_{\forall k,i}\lambda_i(\mathbf{R}_k)}. \tag{34}$$

Proof: Iterating (33) backwards to $n = 0$, we get

$$\begin{aligned}\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] =&\mathbb{E}[||\tilde{\mathbf{w}}_{e,0}||^2_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^{\mathsf{T}})^{n+1}\boldsymbol{\sigma}\}}]\\ &+\mu^2\mathbf{h}^{\mathsf{T}}(\mathbf{I}+\sum_{j=1}^{n}(\boldsymbol{\mathcal{F}}^{\mathsf{T}})^j)\boldsymbol{\sigma}.\end{aligned} \tag{35}$$

To prove the convergence of $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\mathbf{\Sigma}}] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$, we need to prove that the spectral radius of $\boldsymbol{\mathcal{F}}$ is less than one, i.e. $\rho(\boldsymbol{\mathcal{F}}) < 1$. Using the properties of the block maximum norm [49], we have

$$\begin{aligned}\rho(\boldsymbol{\mathcal{F}}) &\leqslant ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{I}-\mu(\mathbf{I}\otimes_b\mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b\mathbf{I}))\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty},\\ &\leqslant ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}||_{b,\infty}||\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty}\\ &\quad ||(\mathbf{I}-\mu(\mathbf{I}\otimes_b\mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b\mathbf{I}))||_{b,\infty}.\end{aligned} \tag{36}$$

Since the matrices $\boldsymbol{\mathcal{Q}}_{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}_{\mathbf{B}}$ are right stochastic, we have $||\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty} = ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}||_{b,\infty} = 1$. Therefore the condition $||(\mathbf{I}-\mu(\mathbf{I}\otimes_b\mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b\mathbf{I}))||_{b,\infty} < 1$, equivalently, $|1-\mu(\lambda_i(\mathbf{R}_e)+\lambda_j(\mathbf{R}_e))| < 1$, $\forall i, j$, is sufficient to guarantee the convergence of $||\tilde{\mathbf{w}}_{e,n}||^2_{\mathbf{\Sigma}}$. This simplification leads to the convergence condition in (34). □

C. Transient and Steady-state Mean Square Deviation

From (33), we can express the relation between $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ and $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ as

$$\begin{aligned}\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] &= \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] \\ &+ \mathbb{E}[||\tilde{\mathbf{w}}_{e,0}||^2_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^\mathsf{T}-\mathbf{I})(\boldsymbol{\mathcal{F}}^\mathsf{T})^n\boldsymbol{\sigma}\}}] \\ &+ \mu^2\mathbf{h}^\mathsf{T}(\boldsymbol{\mathcal{F}}^\mathsf{T})^n\boldsymbol{\sigma}. \end{aligned} \tag{37}$$

If we set $\boldsymbol{\sigma} = \text{bvec}\{\text{blockdiag}\{\mathbf{I_D}, \mathbf{0}, \ldots, \mathbf{0}\}\}$, we obtain the transient expression for the mean square deviation of the global model at iteration n: $\mathbb{E}[||\tilde{\mathbf{w}}_n||^2] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$.

Under (34), by letting $n \to \infty$ in (33), we obtain the expression of the steady-state mean square deviation (MSD) for the PAO-Fed algorithm.

$$\lim_{n\to\infty} \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{(\mathbf{I}-\boldsymbol{\mathcal{F}}^\mathsf{T})\boldsymbol{\sigma}\}}] = \mu^2\mathbf{h}^\mathsf{T}\boldsymbol{\sigma}. \tag{38}$$

By setting $\boldsymbol{\sigma} = (\mathbf{I} - \boldsymbol{\mathcal{F}}^\mathsf{T})^{-1}\text{bvec}\{\text{blockdiag}\{\mathbf{I_D}, \mathbf{0}, \ldots, \mathbf{0}\}\}$, the steady-state MSD expression of the global model can be obtained.

V. Numerical Simulations

This section demonstrates the performance of the proposed PAO-Fed algorithm through a series of numerical experiments. In these experiments, we compare the performance of the PAO-Fed algorithm with existing methods, specifically, Online-FedSGD, Online-Fed [18], and PSO-Fed [27].

A. Simulation Setup

We considered a federated network comprising $K = 256$ clients connected to a server. Synthetic data is progressively made available to the clients in an imbalanced and non-IID manner. For this purpose, the clients are separated into 4 data groups for which training sets are composed of 500, 1000, 1500, and 2000 samples, respectively. A single data sample is of the form $\{\mathbf{x}_{k,n}, y_{k,n}\}$, and related by the following nonlinear relation $\mathbb{R}^4 \longrightarrow \mathbb{R}$:

$$\begin{aligned} y_{k,n} =& \sqrt{\mathbf{x}^2_{k,n}[1] + \sin^2(\pi\mathbf{x}_{k,n}[4])} \\ &+ (0.8 - 0.5\exp(-\mathbf{x}^2_{k,n}[2])\mathbf{x}_{k,n}[3]) + \eta_{k,n}, \end{aligned} \tag{39}$$

where $\mathbf{x}_{k,n}[i]$ denotes the ith element of vector $\mathbf{x}_{k,n} = [x_{k,n}, x_{k,n-1}, x_{k,n-4}, x_{k,n-3}]$. A first-order autoregressive model is used to produce the non-IID input signal $x_{k,n} = \theta_k\, x_{k,n-1} + \sqrt{1-\theta_k^2}\, u_{k,n}$, with $u_{k,n} \in \mathcal{N}(\mu_k, \sigma^2_{u_k})$, and, for a given client k, $\theta_k \in \mathcal{U}(0.2, 0.9)$, $\mu_k \in \mathcal{U}(-0.2, 0.2)$, and $\sigma^2_{u_k} \in \mathcal{U}(0.2, 1.2)$. The observation noise $\nu_{k,n}$ is assumed to be white Gaussian with variance $\sigma^2_{\nu_k} \in \mathcal{U}(0.005, 0.03)$. Further, the cosine feature function is used to map $\mathbf{x}_{k,n}$ from dimension $L = 4$ into the RFF space of dimension $D = 200$.

As discussed in Section III.A, client participation is modeled using the probabilities $p_{k,n}, k \in \mathcal{K}$. Note that a client can only participate in an iteration if it receives new data; otherwise, the probability is set to 0. The clients of each data group are further separated into 4 availability groups, dictating their probability $p_{k,n}$ of participating at each iteration. The Bernoulli trial on $p_{k,n}$ dictates if a client is available or not at a given iteration. Unless stated otherwise, the participation probabilities given to the four availability groups are 0.25, 0.1, 0.025, and 0.005. Finally, each communication to the server will be delayed by more than l iterations with probability $\delta^l, 0 < l < l_{\text{max}}$, with, unless stated otherwise, $\delta = 0.2$ and $l_{\text{max}} = 10$. This probability is assumed to be the same for all clients.

The performance of the algorithms is evaluated on a test dataset with the mean squared error (MSE) given at iteration n by:

$$\text{MSE-test} = \frac{1}{\text{MC}}\sum_{e=1}^{\text{MC}} \frac{||\mathbf{y}^e_{\text{test}} - (\mathbf{Z}^e_{\text{test}})^\mathsf{T}\mathbf{w}^e_n||_2^2}{T}, \tag{40}$$

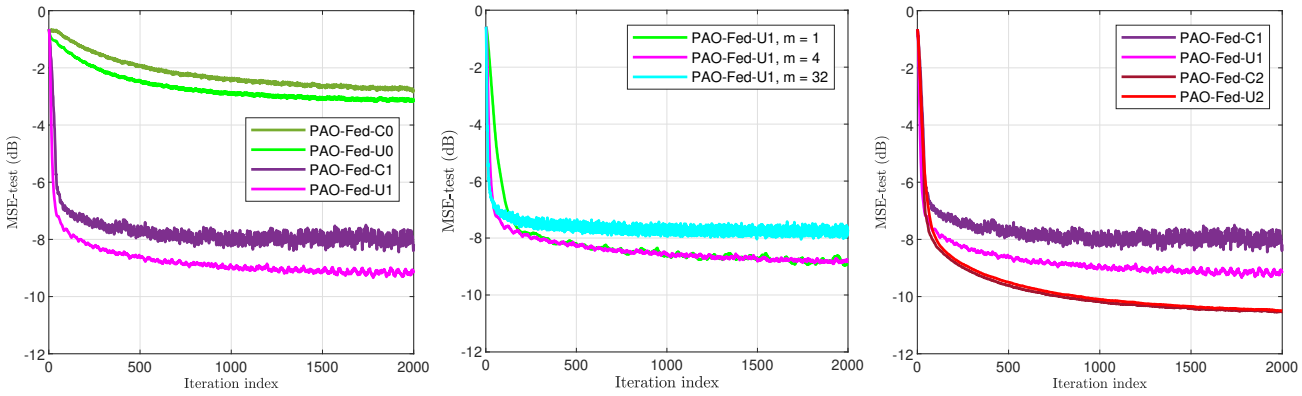
where MC is the number of Monte Carlo iterations, T is the size of the test dataset, $\mathbf{y}^e_{\text{test}}$ and $\mathbf{Z}^e_{\text{test}}$ are the realization of the data for a given Monte Carlo iteration, and $\mathbf{w}^e_n$ is the server's model vector for the considered method. When comparing the PAO-Fed algorithm with other methods, the learning rates were set to yield identical initial convergence rates so that steady-state values may be compared. Some algorithms were not able to reach this common convergence rate, but since their steady-state accuracy is lower, comparison is still possible. All the learning rates satisfy the convergence conditions obtained in Section IV for PAO-Fed, and are available in [18], [27] for Online-Fed, Online-FedSGD, and PSO-Fed. For instance, in Fig. 2, 3, and 4, the step-size for the PAO-Fed algorithm is set to $\mu = 0.4$ with $\max_{\forall k,i} \lambda_i(\mathbf{R}_k) = 1.02$.

In the simulations, we implement uncoordinated partial-sharing-based communications from the server to the clients with $\text{diag}(\mathbf{M}_{k,n}) = \text{circshift}(\text{diag}(\mathbf{M}_{1,n}), mk)$ and $\text{diag}(\mathbf{M}_{1,n}) = \text{circshift}(\text{diag}(\mathbf{M}_{1,0}), mn)$. This, in turn, dictates the portion of the model sent by the clients to the server (see Section II C) so that, on average, all portions are equally represented in the aggregation. We recall that m is the number of model parameters shared at each iteration by both the server and the clients, and dictates the communication savings in partial-sharing-based communications.

We consider different versions of the PAO-Fed algorithm.

- **PAO-Fed-C0** and **PAO-Fed-U0** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (15), that is, $\alpha_l = 1, 0 \leqslant l \leqslant l_{\text{max}}$. Further, the clients share the last received server model portion, refined once by the local update process.
- **PAO-Fed-C1** and **PAO-Fed-U1** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (15). Their selection matrices evolve as described in Section II C.
- **PAO-Fed-C2** and **PAO-Fed-U2** utilize coordinated and uncoordinated partial-sharing, respectively, and employ the weight-decreasing mechanism in (15) with $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\text{max}}$. Their selection matrices evolve as described in Section II C.

Unless explicitly specified, each PAO-Fed implementation shares $m = 4$ model parameters per communication round, resulting in a 98% reduction in communication.



(a) (b) (c)

Fig. 2: Optimization of the PAO-Fed method. (a) Utilizing local updates and coordinated/uncoordinated partial-sharing, (b) Communication savings, (c) Utilizing a weight-decreasing mechanism for delayed updates.

B. Hyper Parameters Selection

In the first experiments, we study the impact of the hyperparameters on the convergence properties of the PAO-Fed algorithm. Specifically, we investigate the impact of the choice of the selection matrices, the number of model parameters shared, and the scale of the weight-decreasing mechanism for delayed updates. The corresponding learning curves in Fig. 2 display the MSE-test in dB versus the iteration index.

First, we examined how the choice of the selection matrices $\mathbf{M}_{k,n}$ and $\mathbf{S}_{k,n}$ impact the convergence properties of the PAO-Fed algorithm. These matrices select the model portion to be shared between the server and clients (see Section II C). The versions PAO-Fed-C0 and PAO-Fed-U0 are set with $\mathbf{S}_{k,n} = \mathbf{M}_{k,n}$; that is, the last received portion from the server is updated once by the local learning process at the clients before being sent back to the server. On the contrary, the versions PAO-Fed-C1 and PAO-Fed-U1 are set as in (7) and (8); that is, the received portions from the server will be updated several times by the local learning process to accumulate information, in a manner similar to batch learning, before being sent back to the server. We observe in Fig. 2 (a) that the versions PAO-Fed-(C/U)1 outperform the versions PAO-Fed-(C/U)0. For this reason, we will only consider the versions of the PAO-Fed algorithm making full use of the local updates in the following. We also notice in this experiment that it is best to use uncoordinated partial-sharing in asynchronous settings, this contradicts the behavior of partial-sharing-based communications in ideal settings, where coordinated partial-sharing performs slightly better than uncoordinated, as explained in [27].

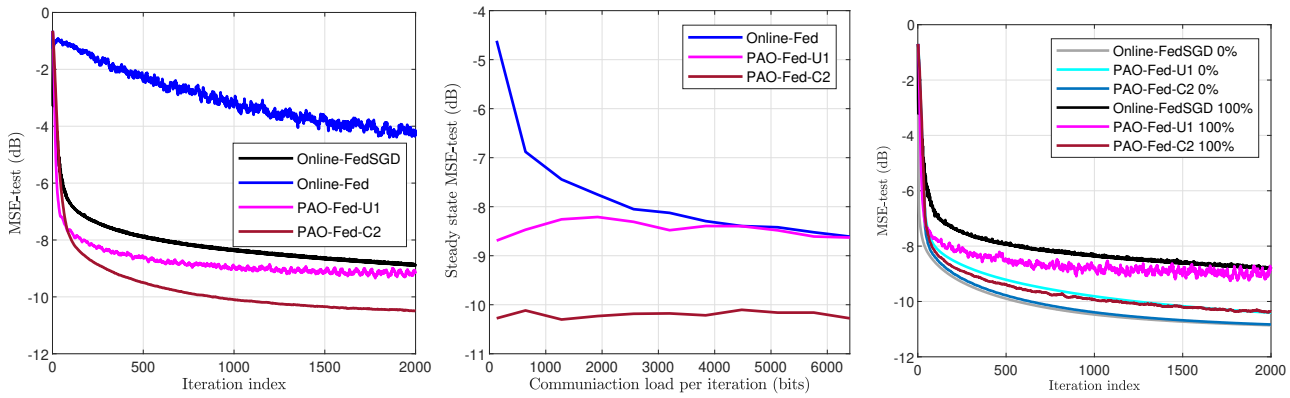
Second, we studied the impact of the number of model parameters m shared by participating clients and the server during the learning process. Fig. 2 (b) shows the performance of the PAO-Fed-U1 algorithm (uncoordinated, making use of local updates) for different values of m, namely $m = 1$, $m = 4$, and $m = 32$. Although sharing more model parameters increases the initial convergence speed, we observed that it decreases the final accuracy for larger m values. This contradicts previous results in the literature about the behavior of partial-sharing in ideal settings [27]. In fact, sharing more model parameters increases the potential negative impact of one single delayed update carrying outdated information, decreasing the overall accuracy. Sharing a small number of model parameters limits the impact of a given update, providing some level of protection against outdated information, and ensuring better model fitting [51]. We chose to set $m = 4$ as a baseline, as it presents a good compromise between initial convergence speed, steady-state accuracy, and communication reduction.

Finally, to reduce the harmful effect of delayed updates on the convergence properties of the algorithm, we introduce the weight-decreasing mechanism for delayed updates proposed in (15) in the versions PAO-Fed-C2 and PAO-Fed-U2. We set $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\max}$. In Fig. 2 (c), we display the performance of these methods alongside PAO-Fed-C1 and PAO-Fed-U1. We observe that decreasing the weight of the delayed updates significantly improves the performance of the PAO-Fed algorithm on the considered asynchronous settings. The proposed mechanism considers the relevance of delayed and potentially outdated updates by effectively reducing their impact on the server model, especially for substantial delays. By doing so, the negative effect of delayed updates is mitigated; in particular, when using the aforementioned weight-decreasing mechanism, PAO-Fed-C2 using coordinated partial sharing and PAO-Fed-U2 using uncoordinated partial sharing exhibit the same performance.

C. Comparison of PAO-Fed with Existing Algorithms

In the following experiments, we compare the performance of the PAO-Fed algorithm with existing online FL methods in the literature. Figs. 3 (a) and (c) display the MSE-test in dB versus the iteration index, and Fig. 3 (b) displays accuracy variation versus communication savings.

First, we compared PAO-Fed-U1 and PAO-Fed-U2 with Online-Fed [18], and Online-FedSGD. Fig. 3 (a) displays the corresponding learning curves. First, we observe that Online-Fed performs poorly; sub-sampling the already reduced pool of available clients is not a viable solution to reduce communication in asynchronous settings. Then, we observe that both



(a) (b) (c)

Fig. 3: Comparison of PAO-Fed with existing methods. (a) Learning curves, (b) Steady-state MSE vs. communication load, (c) Impact of straggler clients.

PAO-Fed-U1 and PAO-Fed-U2 outperform Online-FedSGD while using 98% less communication. The reason for this very good performance is twofold. First, using the local and autonomous local updates in the PAO-Fed algorithm allows it to extract more information from the sparsely participating clients. Second, partial-sharing-based communication provides the PAO-Fed algorithm with an innate resilience to the negative impact of delayed updates; this resilience is further increased in the PAO-Fed-U2 algorithm with the weight-decreasing mechanism, hence its better performance.

Second, we study the relationship between communication load and accuracy. Figure 3 (b) shows the steady-state mean squared error on the test dataset versus the average communication load per iteration when the clients employ either PAO-Fed-U1, PAO-Fed-C2, or Online-Fed algorithms. The communication load is obtained by multiplying the average number of model parameters shared by a client during a given iteration, corresponding to m for the PAO-Fed algorithms, by 32, which is the number of bits on which a model parameter is stored. We find the MSE reached after 2000 iterations in the previous figure by the three algorithms in this figure for a communication load of 128 bits. Similarly, we find the MSE reached after 2000 iterations in the previous figure by Online-FedSGD in this figure for the Online-Fed algorithm with a communication load of 6400 bits. Further, we observe that the higher the communication load is, the better the performance of Online-Fed is. However, the performances of the algorithms using partial-sharing-based communication vary very little with the communication load, as the lower amount of communication is compensated by the use of local updates and the resilience to delayed communications.

Finally, to observe the impact of the straggler clients on the convergence properties of the algorithms, we compare the performance of the algorithms in the proposed settings (100% of clients are potential stragglers) to their performance in an ideal setting where clients are always available when they receive new data and their communication channels do not suffer from delays (0% of clients are potential stragglers). The learning curves are shown in Fig. 3 (c). We observe that, in the absence of straggler clients, the methods using coordinated partial-sharing achieve greater accuracy, almost identical to methods with no communication reduction, while the methods using uncoordinated partial-sharing have slightly worse performance, this corresponds to the results obtained in [27]. Furthermore, we see that the PAO-Fed-C2 algorithm used on straggler clients has convergence properties almost similar to the ones of algorithms in a perfect setting.

D. Performance on a Real-world Dataset

Fig. 4 shows the performance of the proposed PAO-Fed algorithm on the real-world California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset [52]. This dataset comprises oceanographic data from seawater samples collected at various stations and contains more than 800,000 samples. Each sample contains parameters such as temperature, salinity, O_2 saturation, etc. The salinity of the water is linked in a nonlinear manner to the other available parameters, and we employed the proposed method to learn this nonlinear model relating the salinity level in a decentralized manner. For the purpose of the experiment, we consider only 80,000 samples that we distribute progressively and unevenly to the 256 clients throughout the learning process (to ensure non-IID and imbalanced data settings). Further, we simulated the straggler-like behavior of the clients as mentioned above (availability groups are 0.25, 0.1, 0.025, and 0.005; each communication to the server will be delayed by more than l iterations with probability $\delta^l, 0 < l < l_{\max}$, with $\delta = 0.2$ and $l_{\max} = 10$). We observe similar performance for the PAO-Fed, Online-Fed, and Online-FedSGD algorithms to the experiments on synthetic datasets. The PAO-Fed-U1 algorithm is able to achieve the same accuracy as Online-FedSGD while using 98% less communications, and the PAO-Fed-C2 algorithm, also using 98% less communications, is able to outperform all other methods.

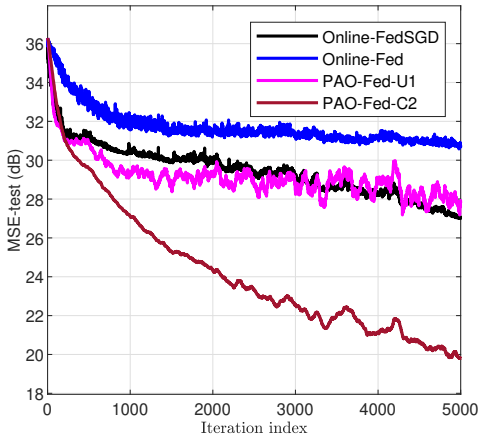


Fig. 4: Learning curves on the CalCOFI dataset.

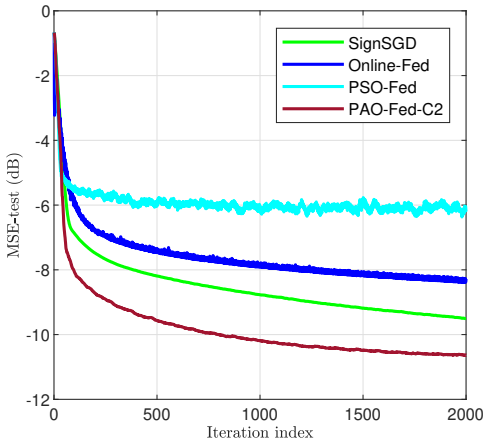


Fig. 5: Learning curves of PAO-Fed, PSO-Fed, Online-Fed, and SignSGD.

E. Comparison of Various Communication Reduction Methods in Asynchronous Settings

In this simulation, we compare the performance of the proposed method with the PSO-Fed [27], Online-Fed [18], and SignSGD [53] algorithms. The PSO-Fed algorithm combines client scheduling and partial-sharing-based communications. For a fair comparison, it has been tailored to reduce the overall communication load by 98%, similar to the proposed PAO-Fed-C2 algorithms. By design, the SignSGD drastically reduces the communication load from clients to server but does not reduce the communication load from server to clients. Its communication load reduction is, therefore, less than 50%. For this reason, the Online-Fed algorithm has been tailored to reduce the communication load by only 50%. The learning curves are displayed in Fig. 5. We observe that reducing the communication load via a combination of client scheduling and partial-sharing-based communication, as in PSO-Fed, is not desirable in asynchronous settings. Furthermore, we see that the SignSGD achieves significantly better performance than Online-Fed for a similar communication load reduction, making it a viable alternative to partial-sharing-based communication in asynchronous settings. However, it would need to be complemented by server-to-client communication reduction and a weight-decreasing mechanism to achieve the same accuracy and communication load reduction as the proposed PAO-Fed-C2.

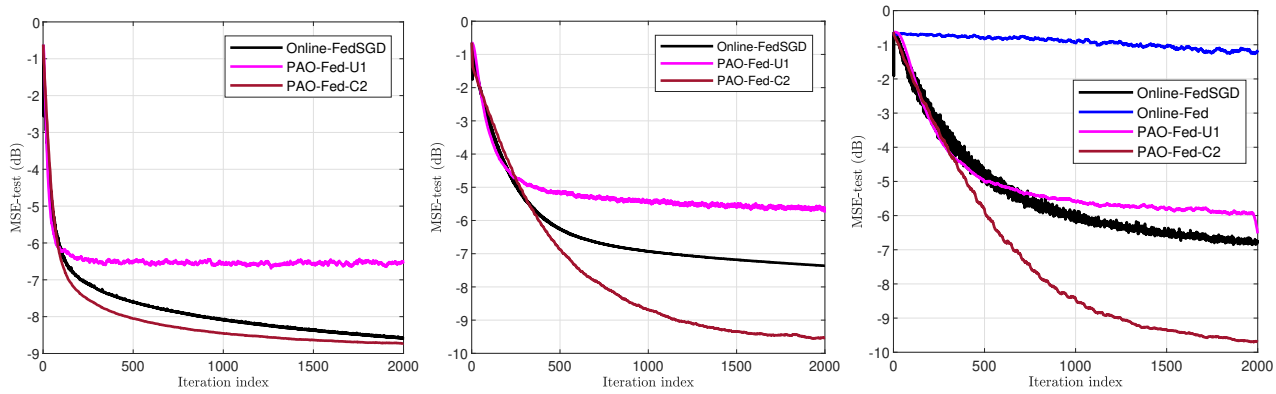
F. Impact of the Environment on Convergence Properties

In these last experiments, we study the impact that a change in the external environment can have on the convergence properties of the proposed algorithms and existing methods. The corresponding learning curves are shown in Fig. 6.

First, we studied in Fig. 6 (a) the importance of using partial-sharing-based communications both at the server and at the clients. The algorithms using partial-sharing-based communications have been altered in this simulation with $\mathbf{M}_{k,n} = \mathbf{I}, \forall k, n$; that is, the server sends its entire model to the participating clients at each iteration. This modification can be appealing if the server is not subject to power constraints. The clients behave normally and only send a portion of their local model; however, unlike in the other simulations, the received global model replaces the local model at each participant, see (10). In such a case, we observe that the performance of the partial-sharing-based methods is drastically reduced. It is the information kept by the clients in the not-yet-shared portions of their local models that allows partial-sharing-based methods to outperform Online-FedSGD. We note that clients may choose to ignore part of the received model to avoid this downfall.

Second, we studied the algorithm behaviors in an environment where most communications are delayed, but delays cannot be too lengthy. To this aim, the delay probability has been significantly increased, and the maximum possible delay reduced ($\delta = 0.8$ and $l_{\max} = 5$). We observe in Fig. 6 (b) that the limited maximum delay allows Online-FedSGD to outperform PAO-Fed-U1, as the benefit of partial-sharing against data of poor quality does not out-weight the smaller amount of communication available to PAO-Fed-U1. To compensate for the fact that most incoming information is weighted down by the weight-decreasing mechanism of PAO-Fed-C2, its learning rate has been increased to near its maximum value obtained in **Theorem 2**. Despite this, the PAO-Fed-C2 algorithm reaches very low steady-state error and significantly outperforms Online-FedSGD.

Finally, we modeled an environment where availability groups are given the probabilities 0.025, 0.01, 0.0025, and 0.0005; communications to the server have a probability $\delta = 0.4$ to be delayed. Further, delays last for more than l iterations, l taking the values $10i, 0 \leqslant i \leqslant 6$, with probability $\delta^{\frac{l}{10}}$; $l_{\max}$ is set to 60. This notably implies that, in this environment, delayed updates have a greater probability of arriving after a non-delayed update coming from the same client. Such an environment where clients are less likely to be available to participate, communications are more likely to be delayed, and delays last for more iterations, is less favorable to learning. An application relying on edge devices that are poorly available and unreliable would evolve in an environment similar to this. Fig. 6 (c) presents the learning curves of Online-Fed, Online-FedSGD, and the PAO-Fed algorithm in this new environment



(a) (b) (c)

Fig. 6: Learning curves in different environments. (a) Full server communication, (b) Common delays, (c) Increased straggler behavior.

to see how it may impact the convergence properties of the algorithms. We observe that, in this environment, reducing the weight given to the delayed updates gains importance as the accuracy difference between PAO-Fed-C2 and PAO-Fed-U1 increases. In fact, delayed updates may carry information that is significantly outdated and, therefore, prevent the algorithms not using a weight-decreasing mechanism for delayed updates to reach satisfactory steady-state error. For this reason, the PAO-Fed-C2 algorithm achieves significantly better accuracy than Online-FedSGD in this environment.

VI. Conclusions

This paper proposed a communication-efficient FL algorithm adapted to a realistic environment. The proposed FL algorithm operates with significantly reduced communication requirements and can cope with an unevenly distributed system with poor client availability, potential failures, and communication delays. The proposed partial sharing mechanism reduces the communication overhead and diminishes the negative impact of delayed updates on accuracy. We further proposed a weight-decreasing aggregation mechanism that emphasizes more recent updates to improve performance in environments suffering from substantial delays, poor participation, and straggler devices. Our numerical results showed that the proposed algorithm outperforms standard FL methods in an asynchronous environment while reducing the communication overhead by 98 percent. The proposed approach is ideal for extracting information in real-time from diverse geographically dispersed devices without overloading the system, making it highly desirable in IoT applications in particular. Future works include expanding the proposed algorithm to a multi-server or networked architecture to alleviate the strain on the single server in applications with many clients.

Appendix A
Evaluation of $\mathbb{E}[\mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{B}_{e,n}]$

The matrix $\mathbf{A}_{e,n}$ is composed of $D\times D$-sized blocks $\mathbf{A}_{i,j,n}$, given by:

$$\mathbf{A}_{i,j,n} = \begin{cases} \mathbf{I}_D & \text{if } i=j \wedge (i=1 \vee i > K+1), \\ a_{k,n}\mathbf{M}_{k,n} & \text{if } i \in [|2,\dots,K+1|] \wedge j=1, \\ \mathbf{I}_D - a_{k,n}\mathbf{M}_{k,n} & \text{if } i \in [2,\dots,K+1] \wedge i=j, \\ \mathbf{0}_D & \text{otherwise,} \end{cases}$$

where $k = i-1$.

We note that $\mathbb{E}[a_{k,n}\mathbf{M}_{k,n}] = p_{k,n}p_m\mathbf{I}_D$, with $p_{k,n}$ being the probability that client k participates at iteration n, and p_m being the probability that a given model parameter is selected by the selection matrix (i.e., the density of the selection: $\frac{m}{D}$). Since $0 \leqslant p_{k,n}p_m \leqslant 1$, and given the above decomposition, matrix $\mathbb{E}[\mathbf{A}_{e,n}]$ is right stochastic.

Further, we note that by construction, $(a_{k,n}\mathbf{M}_{k,n})^2 = a_{k,n}\mathbf{M}_{k,n}$; therefore, under **Assumption 3**, we have

$$\mathbb{E}[a_{k,n}\mathbf{M}_{k,n}a_{k',n'}\mathbf{M}_{k',n'}] = \begin{cases} p_{k,n}p_m\mathbf{I}_D & \text{if } k=k' \wedge n=n', \\ p_{k,n}p_{k',n'}p_m^2\mathbf{I}_D & \text{otherwise.} \end{cases}$$

Similarly, we decompose the matrix $\mathbf{B}_{e,n}$ in $D\times D$-sized blocks $\mathbf{B}_{i,j,n}$ as follows:

$$\mathbf{B}_{i,j,n} = \begin{cases} \mathbf{B}_n & \text{if } i=j=1, \\ \mathbf{B}_{0,n}^{(j-1)} & \text{if } i=1 \wedge j \in [2,K+1], \\ \mathbf{B}_{\lceil \frac{j-1}{K} \rceil - 3,n}^{(j-1 \mod K)} & \text{if } i=1 \wedge j \in \\ & [|3K+2,\dots,(l_{\max}+3)K+1|], \\ \mathbf{I}_D & \text{if } i \in [|1,2|] \wedge j=2, \\ \mathbf{I}_D & \text{if } i>3 \wedge j>2 \wedge i=j+1, \\ \mathbf{0}_D & \text{otherwise.} \end{cases}$$

The blocks are given by:

$$\begin{aligned}\mathbf{B}_n &= \mathbf{I} - \sum_{l=0}^{l_{\max}} \alpha_l \sum_{k\in\mathcal{K}_{n,l}} \frac{b_{k,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{k,n-l},\\ \mathbf{B}_{l,n}^{(k)} &= \mathbf{B}_{l,n}[k],\\ \mathbf{B}_{l,n} &= \left[\frac{\alpha_l b_{1,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{1,n-l}, \cdots, \frac{\alpha_l b_{K,n,1}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{K,n-l}\right].\end{aligned}$$

We note that by construction,

$$\mathbf{B}_n + \sum_{l=1}^{l_{\max}}\sum_{k=1}^{K}\mathbf{B}_{l,n}^{(k)} = \mathbf{I},$$

hence, the matrix $\mathbb{E}[\mathbf{B}_{e,n}]$ is right stochastic as well.

Appendix B
Evaluation of $\mathcal{Q}_\mathbf{A}$ and $\mathcal{Q}_\mathbf{B}$

We decompose matrix $\mathcal{Q}_\mathbf{A}$ into $D \times D$-sized blocks and prove the property by computing the Kronecker product $\mathbf{A}_{e,n} \otimes_b \mathbf{A}_{e,n}$ before taking the expectation. In particular, we have

$$\mathcal{Q}_\mathbf{A} = [\mathbb{E}[\mathbf{A}_{i,j,n} \otimes_b \mathbf{A}_{e,n}], (i,j) \in [|1,\dots,K(l_{\max}+1)+1|]^2],$$

and we note that $\mathcal{Q}_\mathbf{A}$ can be proven to be right stochastic one block-row at a time, considering sets of D rows indexed by i in the above equation.

The property is easy to prove on the block-rows $i = 1$ and $i > K + 1$. On those block-rows, we have

$$\mathbf{A}_{i,j,n} = \begin{cases}\mathbf{I}_D & \text{if } i = j\\ \mathbf{0}_D & \text{otherwise}\end{cases},$$

therefore, since $\mathbb{E}[\mathbf{A}_{e,n}]$ satisfies the property, it is satisfied on those block-rows.

We now consider the remaining block-rows. For this purpose, let $i \in [|2,\dots,K+1|]$. According to the decomposition of the left-hand side $\mathbf{A}_{e,n}$, the block-row i of $\mathcal{Q}_\mathbf{A}$ reduces to only two non-zero elements, $\mathbb{E}[\mathbf{A}_{i,1,n} \otimes_b \mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{A}_{i,i,n} \otimes_b \mathbf{A}_{e,n}]$. Hence we can compute:

$$\begin{aligned}&\mathbb{E}[\mathbf{A}_{i,1,n} \otimes_b \mathbf{A}_{e,n}] + \mathbb{E}[\mathbf{A}_{i,i,n} \otimes_b \mathbf{A}_{e,n}]\\ &\quad= \mathbb{E}[a_{i-1,n}\mathbf{M}_{i-1,n} \otimes_b \mathbf{A}_{e,n}]\\ &\quad+ \mathbb{E}[(\mathbf{I}_D - a_{i-1,n}\mathbf{M}_{i-1,n}) \otimes_b \mathbf{A}_{e,n}]\\ &\quad= \mathbb{E}[\mathbf{I}_D \otimes_b \mathbf{A}_{e,n}],\end{aligned}$$

and conclude that the block-row i satisfies the property.

Similarly, we decompose the matrix $\mathcal{Q}_\mathbf{B}$ into $D \times D$-sized blocks and prove that it is right stochastic by computing the Kronecker product $\mathbf{B}_{e,n} \otimes_b \mathbf{B}_{e,n}$ before taking the expectation.

$$\mathcal{Q}_\mathbf{B} = [\mathbb{E}[\mathbf{B}_{i,j,n} \otimes_b \mathbf{B}_{e,n}], (i,j) \in [1,\dots,K(l_{\max}+1)+1]^2].$$

The evaluation is trivial for the block-rows $i \in [2,\dots,K(l_{\max}+1)+1]$, where the decomposition of the left-hand side $\mathbf{B}_{e,n}$ reduces to only one non-zero element: $\mathbf{I}$. Therefore, since $\mathbf{B}_{e,n}$ satisfies the property, it is satisfied on those block-rows.

We now consider the block-row $i = 1$ and compute the sum of the elements as:

$$\begin{aligned}&\mathbb{E}[\mathbf{B}_n \otimes_b \mathbf{B}_{e,n} + \sum_{l=1}^{l_{\max}}\sum_{k=1}^{K}\mathbf{B}_{l,n}^{(k)} \otimes_b \mathbf{B}_{e,n}]\\ &\quad= \mathbb{E}[\mathbf{I}_D \otimes_b \mathbf{B}_{e,n}],\end{aligned}$$

by construction of the $\mathbf{B}_{l,n}^{(k)}$ matrices. We conclude that the block-row $i = 1$ satisfies the property as well.

We have proven that both $\mathcal{Q}_\mathbf{A}$ and $\mathcal{Q}_\mathbf{B}$ are right stochastic matrices.

References

[1] 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.

[2] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, Oct. 2016.

[3] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Int. Conf. Comput. Learn. Theory*, 2006, pp. 453–467.

[4] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Ind. Eng.*, vol. 149, p. 106854, 2020.

[5] S. Boll and J. Meyer, "Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services," *IEEE MultiMedia*, vol. 29, no. 1, pp. 136–140, May 2022.

[6] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[7] 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, May 2022.

[8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[9] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, Mar. 2021.

[10] 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.

[11] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: accelerated federated learning with drift control," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2021, pp. 467–472.

[12] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, Jul. 2019.

[13] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[14] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Nov. 2020.

[15] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-efficient and optimized federated learning with local differential privacy," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[16] Y. Lu, Z. Liu, and Y. Huang, "Parameters compressed mechanism in federated learning for edge computing," in *Proc. IEEE Int. Conf. Cyber Secur. Cloud Comput.*, Jun. 2021, pp. 161–166.

[17] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Communication-efficient federated learning through 1-bit compressive sensing and analog aggregation," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2021, pp. 1–6.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial Intel. Statis.*, pp. 1273–1282, Apr. 2017.

[19] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," *arXiv preprint arXiv:2109.00151*, Sep. 2021.

[20] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, Mar. 2019.

[21] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2020, pp. 15–24.

[22] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous Federated Learning over Wireless Communication Networks," *IEEE Trans. Wireless Commun.*, Mar. 2022.

[23] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: a communication-efficient federated learning method with asynchronous tiers under non-iid data," *arXiv preprint arXiv:2010.05958*, Oct. 2020.

[24] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Dec. 2019.

[25] Z. Wang, Z. Zhang, and J. Wang, "Asynchronous federated learning over wireless communication networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–7.

[26] X. Qiu, T. Parcollet, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "Can federated learning save the planet?" *arXiv preprint arXiv:2010.06537*, Oct. 2020.

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

[28] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[29] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, Oct. 2016.

[30] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proc. Nat. Acad. Sciences*, vol. 118, no. 17, Apr. 2021.

[31] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," *Proc. Int. Conf. Artificial Intel. Statis.*, pp. 3581–3607, May 2022.

[32] R. Wang and W.-T. Tsai, "Asynchronous federated learning system based on permissioned blockchains," *Sensors*, vol. 22, no. 4, p. 1672, Feb. 2022.

[33] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Mar. 2022.

[34] H. Zhu, Y. Zhou, H. Qian, Y. Shi, X. Chen, and Y. Yang, "Online client selection for asynchronous federated learning with fairness consideration," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2493–2506, Oct. 2022.

[35] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," *Proc. Int. Symp. High-Perform. Parallel Distrib. Comput.*, pp. 125–136, Jun. 2020.

[36] X. Zhang, Y. Liu, J. Liu, A. Argyriou, and Y. Han, "D2D-Assisted Federated Learning in Mobile Edge Computing Networks," *IEEE Wireless Commun. Netw. Conf.*, pp. 1–7, Mar. 2021.

[37] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Computers*, vol. 70, no. 5, pp. 655–668, May 2020.

[38] S. Ko, K. Lee, H. Cho, Y. Hwang, and H. Jang, "Asynchronous federated learning with directed acyclic graph-based blockchain in edge computing: Overview, design, and challenges," *Elsevier Expert Syst. Applications*, p. 119896, Mar. 2023.

[39] L. You, S. Liu, Y. Chang, and C. Yuen, "A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24 199–24 211, Jul. 2022.

[40] 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.

[41] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: a random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, Jun. 2016, pp. 1–5.

[42] A. Rahimi, B. Recht *et al.*, "Random features for large-scale kernel machines." in *Proc. Conf. on Neural Inf. Proc. Syst.*, vol. 3, no. 4, Dec. 2007, pp. 1–5.

[43] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Jan. 2008.

[44] V. C. Gogineni, V. R. Elias, W. A. Martins, and S. Werner, "Graph diffusion kernel LMS using random Fourier features," *54th Asilomar Conf. Signals, Syst., Computers*, pp. 1528–1532, Nov. 2020.

[45] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: an introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, Mar. 2021.

[46] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 2020, pp. 8743–8747.

[47] C.-H. Hu, Z. Chen, and E. G. Larsson, "Scheduling and aggregation design for asynchronous federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 874–886, Jul. 2023.

[48] V. C. Gogineni, S. P. Talebi, and S. Werner, "Performance of clustered multitask diffusion lms suffering from inter-node communication delays," *IEEE Trans. on Circuits and Syst. II: Express Briefs*, vol. 68, no. 7, pp. 2695–2699, 2021.

[49] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

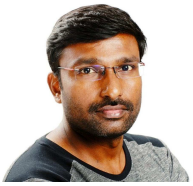
[50] R. H. Koning, H. Neudecker, and T. Wansbeek, "Block Kronecker products and the vecb operator," *Linear algebra and its applications*, vol. 149, pp. 165–184, Apr. 1991.

[51] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, Nov. 2022.

[52] S. Dane, "CalCOFI, Over 60 years of oceanographic data," available at: https://www.kaggle.com/sohier/calcofi?select=bottle.csv.

[53] 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.

Francois Gauthier (Member, IEEE) received both the B.Sc. and M.Sc. degree in mathematics and computer science from École Nationale Supérieure d'Informatiques et de Mathématiques Appliquées de Grenoble. He is pursuing a Ph.D. degree at the Department of Electronic Systems at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His research focuses on federated learning, differential privacy, communication efficiency, personalized learning, and reinforcement learning.



Vinay Chakravarthi Gogineni (Senior Member, IEEE) received the Bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Andhra Pradesh, India, in 2005, the Master's degree in communication engineering from VIT University, India, in 2008, and the Ph.D. degree in electronics and electrical communication engineering from Indian Institute of Technology Kharagpur, India in 2019. Currently, he is an Assistant Professor at SDU Applied AI and Data Science, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark. Prior to this, he worked as a postdoctoral research fellow at NTNU and Simula, Norway. From 2008 to 2011, he was with a couple of MNCs in India. His research interests include statistical signal processing, distributed machine learning, geometric deep learning, and their application in healthcare. He was a recipient of the ERCIM Alain Bensoussan Fellowship in 2019 and the Best Paper Award at APSIPA ASC-2021, Tokyo, Japan. He is 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 a D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is a Professor at the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Director of IoT@NTNU, and Adjunct Professor at Aalto University in 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. He held an Academy Research Fellowship, funded by the Academy of Finland, from 2009 to 2014. 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 for the EURASIP Journal of Signal Processing and the IEEE Transactions on Signal and Information Processing over Networks.

Yih-Fang Huang (Life Fellow, IEEE) is Professor of Electrical Engineering and Special Advisor to the Dean of the College of Engineering. He received his B.S.E.E. degree from National Taiwan University, M.S.E.E. degree from University of Notre Dame, M.A. and Ph.D. degrees from Princeton University. He was chair of Notre Dame's Electrical Engineering department from 1998 to 2006, and was Senior Associate Dean for Education and Undergraduate Programs for the College of Engineering from 2013 to 2023. His research lies in the area of statistical and adaptive signal processing and employs principles in mathematical statistics to solve signal detection and estimation problems that arise in various applications, including wireless communications, distributed sensor networks, smart electric power grid, etc. Dr. Huang received the Golden Jubilee Medal of the IEEE Circuits and Systems Society in 1999. He also served as Vice President in 1997-98 and was a Distinguished Lecturer for the same society in 2000-2001. He served as the lead Guest Editor for a Special Issue on Signal Processing in Smart Electric Power Grid of the IEEE Journal of Selected Topics in Signal Processing, December 2014. At the University of Notre Dame, he received Presidential Award in 2003, the Electrical Engineering department's Outstanding Teacher Award in 1994 and in 2011, the Rev. Edmund P. Joyce, CSC Award for Excellence in Undergraduate Teaching in 2011, and the Engineering College's Outstanding Teacher of the Year Award in 2013. In Spring 1993, Dr. Huang received the Toshiba Fellowship and was Toshiba Visiting Professor at Waseda University, Tokyo, Japan. From April to July 2007, he was a visiting professor at the Munich University of Technology, Germany. In Fall, 2007, Dr. Huang was awarded the Fulbright-Nokia scholarship for lectures/research at Helsinki University of Technology in Finland. He was appointed Honorary Professor in the College of Electrical Engineering and Computer Science at National Chiao-Tung University, Hsinchu, Taiwan, in 2014. Dr. Huang is a Life Fellow of the IEEE and a Fellow of the AAAS.

Anthony Kuh (Fellow, IEEE) received his B.S. in Electrical Engineering and Computer Science at the University of California, Berkeley in 1979, an M.S. in Electrical Engineering from Stanford University in 1980, and a Ph.D. in Electrical Engineering from Princeton University in 1987. He previously worked at AT&T Bell Laboratories and has been on the faculty in the Department of Electrical and Computer Engineering at the University of Hawai'i since 1986. He is currently a Professor and previously served as Department Chair. His research is in the area of neural networks and machine learning, adaptive signal processing, sensor networks, and renewable energy and smart grid applications. He won a National Science Foundation (NSF) Presidential Young Investigator Award and is an IEEE Fellow. He is currently serving as a program director for NSF in the Electrical, Communications, and Cyber Systems (ECCS) division working in the Energy, Power, Control, and Network (EPCN) group. He previously served for the IEEE Signal Processing Society on the Board of Governors as a Regional Director-at-Large Regions1-6, as a senior editor for the IEEE Journal of Selected Topics in Signal Processing, and as a member of the Awards Board. He previously also served as President of the Asia Pacific Signal and Information Processing Association (APSIPA).