Federated Learning Using Three-Operator ADMM

Shashi Kant ⓘ, *Graduate Student Member, IEEE*, José Mairton B. da Silva Jr. ⓘ, *Member, IEEE*,
Gabor Fodor ⓘ, *Senior Member, IEEE*, Bo Göransson ⓘ, *Member, IEEE*, Mats Bengtsson ⓘ, *Senior Member, IEEE*,
and Carlo Fischione ⓘ, *Senior Member, IEEE*

Abstract—**Federated learning (FL) has emerged as an instance of distributed machine learning paradigm that avoids the transmission of data generated on the users' side. Although data are not transmitted, edge devices have to deal with limited communication bandwidths, data heterogeneity, and straggler effects due to the limited computational resources of users' devices. A prominent approach to overcome such difficulties is FedADMM, which is based on the classical two-operator consensus alternating direction method of multipliers (ADMM). The common assumption of FL algorithms, including FedADMM, is that they learn a global model using data only on the users' side and not on the edge server. However, in edge learning, the server is expected to be near the base station and has often direct access to rich datasets. In this paper, we argue that it is much more beneficial to leverage the rich data on the edge server then utilizing only user datasets. Specifically, we show that the mere application of FL with an additional virtual user node representing the data on the edge server is inefficient. We propose FedTOP-ADMM, which generalizes FedADMM and is based on a three-operator ADMM-type technique that exploits a smooth cost function on the edge server to learn a global model in parallel to the edge devices. Our numerical experiments indicate that FedTOP-ADMM has substantial gain up to 33% in communication efficiency to reach a desired test accuracy with respect to FedADMM, including a virtual user on the edge server.**

Index Terms—**Federated learning, three-operator ADMM, distributed machine learning, communication efficiency.**

I. INTRODUCTION

CENTRALIZED training of machine learning models becomes prohibitive for a large number of users, particularly if the users — also known as clients or agents or workers — have to share a large dataset with the central server. Furthermore, sharing a dataset with the central server may not be feasible
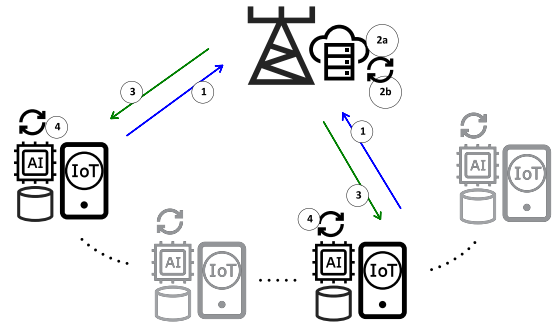
Fig. 1.   Illustration of FL architecture, with the new scenario investigated in this paper of a dataset available on the edge server.

for some users due to privacy concerns. Therefore, training algorithms using distributed and decentralized approaches are preferred. This has led to the concept of federated learning (FL), which results from the synergy between large-scale distributed optimization techniques and machine learning. Consequently, FL has received considerable attention in the last few years since its introduction in [1], [2].

In the Federated learning (FL) framework — illustrated in Fig. 1 — a distributed optimization problem, such as problem (1), is essentially solved considering a central server and other devices by exchanging the parameters/weights of a considered model rather than sharing private data among themselves. The devices desire to achieve a learning model using data from all the other devices for the training. Instead of sending data from the devices to the edge server that computes such a model, these devices execute some local computations and periodically share only their parameters. Specifically, the FL technique intends to minimize a finite sum of (usually assumed) differentiable functions that depend on the data distributions on the various devices. The common solution to such a minimization involves an iterative procedure, wherein at each global communication iteration, the clients transmit their updated local parameters to a central edge server — illustrated as Step ① in Fig. 1. The edge server then updates the global model parameters — shown as Step ② in Fig. 1. However, the processing Steps 2a and 2b are specific to our proposed approach, and leverage the possibility of using a dataset on the edge server to improve the learning model. Thereafter, the server broadcasts the updated parameters or weights to all or the selected nodes—see Step ③ in Fig. 1. Lastly, the clients update their local parameters using their local (private) dataset and the received global model parameters from the central server—Step ④ in Fig. 1—to proceed to the next iteration.

Many state-of-the-art FL techniques, such as `FedAvg` [2] and `FedProx` [3] can be seen as an instance of one-operator proximal splitting techniques [4], i.e., described in (1) with function $g = 0$, which considers learning on the users' side mainly. Differently from these techniques, we propose to go beyond and use a three-operator[1] proximal splitting technique, such as our recently proposed three-operator alternating direction method of multipliers (TOP-ADMM) method [6], to leverage the possibility of learning on the edge server together with the traditional learning on the edge devices.

A. Motivation for Learning Model on the Edge Server

In future (6G) cellular networks, machine learning services will be used both to design the networks and as a service provided by the networks [7]. We expect to leverage the raw data generated not only on the users' side, but also on the edge server's side collocated at the base station or/and radio access networks. More concretely, we foresee scenarios, in which the raw data is available at the base station because data is generated continuously at the physical layer and radio in 5G NR and beyond. Thus, data at the server side together with data on the users' side, enable many distributed signal processing applications such as joint communication and sensing, or edge learning with the Internet of Things (including edge devices). The optimization problem for FL considering learning on the edge server and on the edge devices is naturally described by a sum of three functions or three operators—see (5). A three-operator problem for FL can be solved via traditional block-wise two-operator splitting techniques or treating the learning model on the edge server as a virtual user, e.g., in an existing two-operator-based[2] `FedADMM` [8]. However, we show numerically, see Fig. 5, that such an approach is not necessarily more communication efficient than tackling such a problem fundamentally from the three-operator proximal splitting perspective.

To the best of our knowledge, FL using three-operator techniques have not been addressed in the literature, which tackles the learning on both the server and users independently. This proposal is highly novel and benefits from the richly available datasets on the edge server and edge devices from current fifth generation (5G) and future 6G cellular networks. Hence, these are key motivating reasons to consider a three-operator problem being tackled in an FL fashion.

B. Contributions

We present a new communication-efficient and computationally efficient FL framework, referred to as `FedTOP-ADMM`, using a three-operator alternating direction method of multipliers (ADMM) method. Specifically, our key contributions are:

- We demonstrate the viability of a practical edge learning scenario in which private datasets are available on the devices, and another private dataset is available on the edge server/base station. We model this edge learning scenario using a novel three-operator splitting method that benefits from the private datasets on both edge server and edge devices.
- We propose the `FedTOP-ADMM` method by applying and extending our recently proposed three-operator ADMM

(TOP-ADMM) method [6] to tackle a composite optimization problem (5) comprising a sum of three functions (or three operators). More specifically, we propose two variants of `FedTOP-ADMM`, termed `FedTOP-ADMM I` and `FedTOP-ADMM II`, where `FedTOP-ADMM II` does not learn on the server side when aggregating the parameters from the users and the server itself to generate a common model parameter. However, `FedTOP-ADMM I` learns a model before aggregation of the parameters in addition to learning in parallel with the users. Thus, `FedTOP-ADMM I` has a slightly better performance compared to `FedTOP-ADMM II`.

- We extend the results of [6] by establishing a new theoretical convergence proof of TOP-ADMM under general convex settings. Additionally, extending the convergence results of TOP-ADMM, we prove the optimality conditions of `FedTOP-ADMM`.
- `FedTOP-ADMM` capitalizes on the possible data available on the edge server collocated at the base station in addition to the data available on the users' side. Consequently, our numerical experiments show noticeable communication efficiency gain over the existing state-of-the-art FL schemes using real-world data.
- Our proposed `FedTOP-ADMM` is built on the existing framework of communication- and computationally efficient `FedADMM` [8]. Therefore, `FedTOP-ADMM` inherits all the merits of `FedADMM`. Furthermore, we show that our proposed `FedTOP-ADMM` is up to 33% more efficient in terms of communication rounds to achieve the same target test accuracy as the one achieved by `FedADMM`, where the extra dataset on the edge server is modelled as an additional virtual client collocated at the base station.

II. STATE OF THE ART

In this section, we briefly describe the related works on FL, including two or three operators in proximal splitting techniques useful for FL.

A. Related Works on Federated Learning

Starting from the seminal work [1], several extensions and applications of FL have been proposed, e.g., `FedAvg` [2], `FedProx` [3], `FedADMM` [8], and other variations, [9], [10], [11]. A relatively recent overview can be found in [12].

FL may suffer from two drawbacks: privacy leakage and communication inefficiency. Although FL keeps the data local on the clients and thus inherently has privacy properties, it does not guarantee complete privacy because significant information may still leak through observing the gradients [13]. Moreover, the FL algorithms may have an unsustainable communication cost: the local parameters must be communicated via uplink from the devices to the edge server, and via downlink from the server to the local devices. The local parameters can be vectors of huge sizes whose frequent transmissions and reception may deplete the battery of the devices and consume precious communication resources.

A number of works have addressed the problem of communication efficiency in FL [11], [14], [15], [16], [17], [18], [19]. We can roughly divide them into two classes: 1) data compression in terms of quantization and sparsification of the local parameters before every transmission [14], [15], [16], [17], and 2) reduction of the communication iterations [11], [17],

---

[1]Notice that "operator" terminology is used in the dual of ADMM, i.e., Douglas-Rachford [5]. Therefore, we borrow this terminology in this work.

[2]Although `FedADMM` is built on two-operator ADMM, it actually solves one-operator problem—see discussion in Section IV-B and Table I.

[18], [19]. The works in the second class attempt to reduce some communication rounds between the devices and the edge server, as for example proposed in lazily aggregated gradient (LAG) approach [11], [18]. In LAG, each device transmits its local parameter only if the variation from the last transmission is large enough. However, both classes of approaches assume an underlying iterative algorithm whose iterations are sometimes eliminated or whose carried information (in bits) is processed to consume fewer communication resources. The process of making the underlying algorithm more communication efficient, regardless of the improvements that can be done on top of it, has been less investigated. The state of the art can be found in FedADMM [8], which not only allows the averaging of users' parameters periodically to reduce the communication rounds but also aims to improve the communication efficiency by using one-operator proximal splitting techniques in FL methods. Our paper focuses on this line of research and extends it to three-operator proximal splitting while proposing learning jointly on the edge server and edge devices side.

B. Related Works on Operator/Proximal Splitting

In recent decades, a plethora of proximal or operator splitting techniques, see, e.g., [6], [20], [21], [22], [23], [24], [25], [26], [27], [28], have been proposed in the literature. Although the convergence of many proximal/operator splitting algorithms is proven only for convex settings, these techniques can still be employed to solve many nonconvex problems prevalent in machine learning problems (without convergence or performance guarantees). In the past decade, ADMM-like methods, see, e.g., [21], [28], have enjoyed a renaissance because of their wide applicability in large-scale distributed machine learning problems by breaking down a large-scale problem into easy-to-solve smaller problems. However, these proximal splitting techniques are typically first-order methods, which can be very slow to converge to solutions having high accuracy. Nonetheless, modest accuracy can be sufficient for many practical FL applications.

Operator splitting for two operators, or loosely speaking, optimization problems, in which the objective function is given by the sum of two functions, have recently been employed for FL [4], [8], [29], [30], [31]. However, some FL optimization problems — see problem (5) — can be cast as a composite sum of three functions comprising smooth and non-smooth functions. Unfortunately, operator splitting with more than two composite terms in the objective function are either not straightforward or converges slowly [26], [27]. Recently, some authors have extended the primal-type or dual-type, including ADMM-type algorithms, as well as the primal-dual classes of splitting algorithms from two operators to three operators [6], [26], [27], [28], [32], [33], [34], [35], [36].

C. Notation and Paper Organization

Let the set of complex and real numbers be denoted by $\mathbb{C}$ and $\mathbb{R}$, respectively. $\Re\{x\}$ denotes the real part of a complex number $x \in \mathbb{C}$. The i-th element of a vector $\boldsymbol{a} \in \mathbb{C}^{m \times 1}$ and j-th column vector of a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ are denoted by $a[i] := (\boldsymbol{a})_i \in \mathbb{C}$ and $\boldsymbol{A}[:, j] \in \mathbb{C}^{m \times 1}$, respectively. We form a matrix by stacking the set of vectors $\{\boldsymbol{a}[n] \in \mathbb{C}^{M \times 1}\}_{n=1}^{N}$ and $\{\boldsymbol{b}[m] \in \mathbb{C}^{1 \times N}\}_{m=1}^{M}$ column-wise and row-wise as $\boldsymbol{A} = [\boldsymbol{a}[1], \dots, \boldsymbol{a}[N]] \in \mathbb{C}^{M \times N}$ and $\boldsymbol{B} = [\boldsymbol{b}[1]; \dots; \boldsymbol{b}[M]] \in \mathbb{C}^{M \times N}$, respectively. The transpose and conjugate transpose of a vector or matrix are denoted by $(\cdot)^{\mathrm{T}}$ and $(\cdot)^{\mathrm{H}}$, respectively. The complex conjugate is represented by $(\cdot)^{*}$. The $K \times K$ identity matrix is written as $\boldsymbol{I}_K$. An i-th iterative update reads $(\cdot)^{(i)}$.

The remainder of the paper is organized as follows. In the next section, we introduce the TOP-ADMM technique. In Section IV, we establish our proposed FedTOP-ADMM algorithm. In Section V, we present the numerical results, and in Section VI we conclude with a summary and future work. Appendix A contains some useful definitions and lemmas. Appendix B presents the completely novel convergence proof of our recently proposed TOP-ADMM [6] algorithm.

III. INTRODUCTION TO THE TOP-ADMM ALGORITHM

In this section, we firstly introduce the classical two-operator consensus ADMM. Subsequently, we present our recently proposed TOP-ADMM algorithm [6], [37].

A. Classical Consensus ADMM

The two-operator consensus ADMM [21], [22] is a popular method in the optimization and machine learning communities to solve problems of the form

$$\underset{\{\boldsymbol{x}_m \in \mathbb{C}^n\}, \boldsymbol{z} \in \mathbb{C}^n}{\text{minimize}} \quad \sum_{m=1}^{M} f_m(\boldsymbol{x}_m) + g(\boldsymbol{z})$$

$$\text{subject to} \quad \boldsymbol{x}_m - \boldsymbol{z} = \boldsymbol{0}, \quad \forall m = 1, \dots, M, \quad (1)$$

where $\{f_m\}$ and $g(\cdot)$ are closed, convex, and proper functions. The classical ADMM algorithm that solves problem (1) can be summarized, following [21, Chapter 7], as follows

$$3\boldsymbol{x}_m^{(i+1)} := \arg\min_{\boldsymbol{x}_m} f_m(\boldsymbol{x}_m) + \rho \left\| \boldsymbol{x}_m - \boldsymbol{z}^{(i)} + \frac{\boldsymbol{y}_m^{(i)}}{\rho} \right\|_2^2, \ \forall m, \tag{2a}$$

$$\boldsymbol{z}^{(i+1)} := \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \sum_{m=1}^{M} \rho \left\| \boldsymbol{x}_m^{(i+1)} - \boldsymbol{z} + \frac{\boldsymbol{y}_m^{(i)}}{\rho} \right\|_2^2, \tag{2b}$$

$$\boldsymbol{y}_m^{(i+1)} := \boldsymbol{y}_m^{(i)} + \rho \left(\boldsymbol{x}_m^{(i+1)} - \boldsymbol{z}^{(i+1)} \right), \ \forall m = 1, \dots, M, \tag{2c}$$

where $\boldsymbol{y}_m^{(i)} \in \mathbb{C}^n$ is the Lagrange multiplier and $\rho \in \mathbb{R}_{>0}$ is a penalty parameter.

If the functions f_m and g contain a computationally inconvenient quadratic term, then one could employ proximal linearized ADMM; see, e.g., [28] and references therein to cancel out such a quadratic term. More specifically, in linearized ADMM, one adds a so-called proximal term $\|\boldsymbol{x}_m - \boldsymbol{x}_m^{(i)}\|_{\boldsymbol{Q}_x}^2$ and $\|\boldsymbol{z} - \boldsymbol{z}^{(i)}\|_{\boldsymbol{Q}_z}^2$ with positive definite matrices $\boldsymbol{Q}_x$ and $\boldsymbol{Q}_z$ to (2a) and (2b), respectively.

B. Consensus TOP-ADMM

The recently proposed three-operator-based[3] TOP-ADMM [6] is one of the generalized algorithms for the classical

---

[3] As briefly mentioned in Section II-B, operator splitting with more than two composite terms in the objective has been an open research problem without resorting to problem reformulations or product space reformulations [38], [39]. The reason is that it has been found empirically that such product space formulations and problem reformulations may not be straightforward and generally may be slow to converge or not be feasible for some problems [26], [27], [40].

consensus ADMM. The TOP-ADMM can be employed to solve many centralized and distributed optimization problems in signal processing, which are difficult to solve by traditional ADMM methods. Specifically, the TOP-ADMM algorithm has been used to solve spectrum shaping via spectral precoding and peak-to-average power ratio reduction in multiple carriers-based wireless communication systems, such as 5G cellular networks [6], [37], which were difficult to solve with the classical two-operator ADMM methods. More precisely, we can refer to [6, Fig. 4 and Fig. 5] as a concrete and realistic example within wireless communications that shows numerically that two-operator ADMM-like algorithm solving three-operator problem exhibit slow convergence compared to TOP-ADMM.

First, let us understand the shortcomings of classical (linearized) ADMM and subsequently present TOP-ADMM to deal with the demerits of ADMM. To this end, let us add a convex L-smooth ($L > 0$) function h to the cost function of (1) with some scaling $\beta \in \mathbb{R}_{>0}$ such that the problem formulation becomes

$$\underset{\{\boldsymbol{x}_m \in \mathbb{C}^n\}, \boldsymbol{z} \in \mathbb{C}^n}{\text{minimize}} \quad \sum_{m=1}^{M} f_m(\boldsymbol{x}_m) + g(\boldsymbol{z}) + \beta h(\boldsymbol{z})$$

$$\text{subject to} \quad \boldsymbol{x}_m - \boldsymbol{z} = \boldsymbol{0}, \quad \forall m = 1, \dots, M. \quad (3)$$

There are at least two possibilities to tackle problem (3) using the classic two-operator ADMM: 1) product space or problem reformulation [38], [39], and 2) consider $\overline{g}(\boldsymbol{z}) := g(\boldsymbol{z}) + \beta h(\boldsymbol{z})$ or $\overline{f}_m(\boldsymbol{x}_m) := f_m(\boldsymbol{x}_m) + (\beta/M)h(\boldsymbol{x}_m)$. The first approach is either not straightforward or the resulting algorithm converges slowly [26], [27]. The second approach yields a subproblem, i.e., $\boldsymbol{z}^{(i+1)}$ or $\boldsymbol{x}_m^{(i+1)}$ update in (2), which may not have a computationally efficient solution in general. Furthermore, the classical two-operator ADMM does not exploit the smoothness of h. Recall that if h is a quadratic function, then clubbing it with either update (2a) or (2b) and employing linearized ADMM can solve the respective subproblem. However, this approach does not necessarily yield a better performance than the TOP-ADMM.

In essence, the TOP-ADMM algorithm can also be classified as a divide-and-conquer method, which decomposes a large optimization problem — difficult to solve in a composite form (3) — into smaller subproblems that are easy to solve.

In the following, we lay out the general definition of our TOP-ADMM method, which was introduced in [6]. For completeness, we present the convergence proof of TOP-ADMM, which is novel and was not present in [6].

Theorem 1 (TOP-ADMM): Consider a problem given in (3) with at least one solution and a suitable step-size $\tau \in \mathbb{R}_{\geq 0}$. Assume subproblems (4a) and (4b) have solutions, and consider a relaxation/penalty parameter $\rho \in \mathbb{R}_{>0}$ and some arbitrary initial values $(\{\boldsymbol{x}_m^{(0)}\}, \boldsymbol{z}^{(0)}, \{\boldsymbol{y}_m^{(0)}\})$. Then, the generated sequences $(\{\boldsymbol{x}_m^{(i)}\}, \boldsymbol{z}^{(i)}, \{\boldsymbol{y}_m^{(i)}\})$ by the following iterative scheme

$$\boldsymbol{x}_m^{(i+1)} = \underset{\boldsymbol{x}_m}{\arg \min} \, f_m(\boldsymbol{x}_m) + \rho \left\| \boldsymbol{x}_m - \boldsymbol{z}^{(i)} + \frac{\boldsymbol{y}_m^{(i)}}{\rho} \right\|_2^2, \forall m,$$
$$(4a)$$

$$\boldsymbol{z}^{(i+1)} = \underset{\boldsymbol{z}}{\arg \min} \, g(\boldsymbol{z})$$
$$+ \sum_{m=1}^{M} \rho \left\| \boldsymbol{x}_m^{(i+1)} - \boldsymbol{z} - \tau \nabla h\left(\boldsymbol{z}^{(i)}\right) + \frac{\boldsymbol{y}_m^{(i)}}{\rho} \right\|_2^2,$$
$$(4b)$$

$$\boldsymbol{y}_m^{(i+1)} = \boldsymbol{y}_m^{(i)} + \rho \left(\boldsymbol{x}_m^{(i+1)} - \boldsymbol{z}^{(i+1)}\right), \; \forall m = 1, \dots, M,$$
$$(4c)$$

at any limit point, converges to a Karush-Kuhn-Tucker (KKT) stationary point of (3).

Proof: See Appendix B.

Note that the classical consensus ADMM algorithm is a special case of our proposed TOP-ADMM algorithm when $h = 0$ in (3) or $\nabla h = 0$ in (4). Although classical consensus ADMM can solve many problems in machine learning, it does not necessarily yield an implementation-friendly algorithm, particularly, if the proximal operator of the L-smooth function h is inefficient to compute.

IV. FEDERATED LEARNING USING TOP-ADMM

FL can used to solve the distributed consensus problem (3). Furthermore, we envision learning on the edge server, say with a loss function h in (3) with some constraint or regularizer expressed by g in (3). Therefore, we pose the generic distributed problem (3) for FL using new variables as given below:

$$\underset{\{\boldsymbol{w}_m \in \mathbb{R}^n\}, \boldsymbol{w} \in \mathbb{R}^n}{\text{minimize}} \sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{w}_m; \mathcal{D}_m) + g(\boldsymbol{w}) + \beta h(\boldsymbol{w}; \mathcal{D})$$

$$\text{subject to} \; \boldsymbol{w}_m - \boldsymbol{w} = \boldsymbol{0}, \quad \forall m = 1, \dots, M, \quad (5)$$

where the global weight vector of the considered learning model is given by $\boldsymbol{w} \in \mathbb{R}^n$ and the weight vector of user m corresponds to $\boldsymbol{w}_m \in \mathbb{R}^n$. Furthermore, the loss function at user m and the server are denoted by f_m and h with training dataset $\mathcal{D}_m$ and $\mathcal{D}$, respectively. The loss function f_m of user m is weighted by $\alpha_m \geq 0$ satisfying $\sum_m \alpha_m = 1$, and the server's loss function h is weighted by some nonnegative $\beta \geq 0$. The edge server is expected to be collocated at the base station such that the server learns the global model using the data $\mathcal{D}$ generated or stored on the server side together with the learning from the data generated/available on the users' side. Observe that in many existing federated learning frameworks, such as FedADMM [8], $g = 0$ and $h = 0$ in (5).

We are now ready to apply the TOP-ADMM algorithm to (5) for the FL purpose with some modifications. Firstly, we swap the update order of TOP-ADMM, i.e., (4a) and (4b). As suggested in [24, Chapter 5], the sequence updates of ADMM for $\boldsymbol{z}^{(i+1)}$ in (2b) and $\boldsymbol{x}_m^{(i+1)}$ in (2a) are performed in Gauss-Seidel fashion. Specifically, one can interchange these updates without penalizing the convergence guarantee but the generated sequences over iterations may be different in general. Since TOP-ADMM generalizes ADMM, we can also interchange the updates. Secondly, we add a proximal term scaled by the parameter $\zeta^{(i)}$ to the subproblem in (2b). Lastly, we employ a so-called Glowinski's relaxation factor $\gamma \in (0, 2)$ to the dual update, see, e.g., [28]. Hence, the TOP-ADMM algorithm tackling (5) can be summarized as follows:

$$\boldsymbol{w}^{(i+1)} = \underset{\boldsymbol{w}}{\arg \min} \, g(\boldsymbol{w})$$

$$+ \sum_{m=1}^{M} \frac{\rho_m}{2} \left\| \boldsymbol{w}_m^{(i)} - \boldsymbol{w} - \tau^{(i)} \nabla h\left(\boldsymbol{w}^{(i)}\right) + \frac{\boldsymbol{\lambda}_m^{(i)}}{\rho_m} \right\|_2^2$$
$$(6a)$$

$$+ \frac{\zeta^{(i)}}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^{(i)} \right\|_2^2,$$

$$w_m^{(i+1)} = \arg\min_{w_m} \alpha_m f_m(w_m) + \frac{\rho_m}{2} \left\| w_m - w^{(i)} + \frac{\lambda_m^{(i)}}{\rho_m} \right\|_2^2, \forall m,$$
(6b)

$$\lambda_m^{(i+1)} = \lambda_m^{(i)}$$
$$+ \gamma \rho_m \left(w_m^{(i+1)} - w^{(i+1)} \right), \forall m = 1, \ldots, M,$$
(6c)

where step size $\tau^{(i)} \in \mathbb{R}_{\geq 0}$ and proximity parameter $\zeta^{(i)} \in \mathbb{R}_{\geq 0}$ are adaptive over iterations, i.e., more concretely, $\tau^{(i+1)} \leq \tau^{(i)}$, and $\zeta^{(i+1)} \leq \zeta^{(i)}$ for all $i = 0, 1, 2, \ldots$.

Unfortunately, a direct application of the enhanced TOP-ADMM algorithm (6) in the FL context would incur high communication costs due to the exchange of parameters among the server and the selected clients in every global iteration. Hence, TOP-ADMM may negatively affect the communication efficiency. Additionally, at a given global iteration in the TOP-ADMM algorithm (6), each client is expected to send and receive the updates to the server synchronously. Unfortunately, not all of the involved clients can transmit and receive the updated parameters in FL due to the limited communication bandwidth and their computational capabilities. Considering the aforementioned limitations of the direct application of TOP-ADMM for the FL, we extend the TOP-ADMM algorithm using the FL framework introduced in [8]. Therefore, we establish a novel algorithm named FedTOP-ADMM described in the next section catering to the FL purpose.

We would like to accentuate that it is unclear how to extend the existing FedADMM [8] framework to support learning on the server, i.e., described by a loss function h in (5) while supporting a nonsmooth regularizer/function g for the distributed learning. Ignoring g for the moment in (5), one could argue to artificially add yet another parallel client on the server in the existing FedADMM framework, i.e., $h \equiv f_{M+1}$. However, the additional virtual client does not necessarily yield better convergence performance. Therefore, we evolve the FedADMM [8] framework using our TOP-ADMM algorithm. Subsequently, we show in the numerical Section V that FedTOP-ADMM renders superior performance over FedADMM with additional virtual client—see, e.g., Fig. 5. Recall that the framework proposed in [8] is based on the classical two-operator consensus ADMM—cf. (1) with $g = 0$. Therefore, the proposed FL using TOP-ADMM, i.e., FedTOP-ADMM, not only inherits all the properties of classical two-operator ADMM, i.e., FedADMM [8], but also additionally exploits the L-smooth function on the server. In other words, FedADMM is a special case of FedTOP-ADMM.

In the sequel, we establish the FedTOP-ADMM algorithm that is built on the FedADMM [8] framework utilizing the TOP-ADMM algorithm.

A. FedTOP-ADMM: Communication-Efficient Algorithm

We present in Algorithm 1 our novel FedTOP-ADMM using the TOP-ADMM (6) and FedADMM [8] framework for communication-efficient FL. Specifically, we propose two variants of FedTOP-ADMM algorithms, which are referred to as FedTOP-ADMM I and FedTOP-ADMM II. In FedTOP-ADMM I, we learn the model on the server side continuously in every global iteration. Conversely, in FedTOP-ADMM II we learn the considered global model when the server is not

---

Algorithm 1: FedTOP-ADMM I/II.

1: **Input and Initialization**: Choose $\tau^{(0)} \geq 0$, $\zeta^{(0)} \geq 0$, $\{\tau_m^{(0)}\}$, $\{\zeta_m^{(0)}\}$; $\mathcal{U}^{(0)} := [M]$; $w^{(0)}$; $\{w_m^{(0)}\}$, $\{\lambda_m^{(0)}\}$; $\{Q_m\}$; $\{\rho_m\}$; $\{\alpha_m\}$; and $u_m^{(0)} := \rho_m w_m^{(0)} + \lambda_m^{(0)} \forall m$; $\nu^{(0)} := 1/(\sum_{m \in [M]} \rho_m + \zeta^{(0)})$; $\mathcal{D}, \{\mathcal{D}_m\}$; $\gamma \in (0, 2)$

2: **for** $i = 0, \ldots, I - I$ **do**

3:    [**Server weight updates**]

4:    **if** FedTOP-ADMM I OR (FedTOP-ADMM II AND $i \notin \mathcal{P}$) **then**

$$y^{(i+1)} := -\tau^{(i)} \nabla h \left(w^{(i)}; \mathcal{D} \right) + \zeta^{(i)} w^{(i)}$$

5:    **else** $y^{(i+1)} \leftarrow y^{(i)}$

6:    **end if**

   Uplink communications and global parameter updates:

7:    **if** $i \in \mathcal{P}$ **then** % *Uplink communications with users' selection*

8:      [**Server receives updated weights**] $\{v_m^{(i+1)} \leftarrow u_m^{(i)}\}$

9:      [**Client selection**] $\mathcal{U}^{(i+1)} \subseteq [M]$ for downlink communications

10:    **else**

11:      [**Server utilizes previous weights**] $\{v_m^{(i+1)} \leftarrow v_m^{(i)}\}$

12:    **end if**

13:    [**Global parameter aggregation/update**]

$$w^{(i+1)} = \mathbf{prox}_{\nu^{(i)} g} \left(\frac{1}{\nu^{(i)}} \left[\sum_{m=1}^{M} v_m^{(i)} + y^{(i)} \right] \right)$$

   Downlink communications and local parameter updates: %Parallel

14:    [**Server multicasts**]

15:    **for** every $m \in \mathcal{U}^{(i)}$ **do** % *update sequences*

16:      **if** $i \in \mathcal{P}$ **then** %*Selected users receive the updated weights*

$$v^{(i+1)} := w^{(i+1)}$$

17:      **else** $v^{(i+1)} \leftarrow v^{(i)}$

18:      **end if**

19:      [**User weight updates**]

$$\Delta z_m := \rho_m \left(w_m^{(i)} - v^{(i+1)} \right)$$
$$+ \alpha_m \nabla f_m \left(w_m^{(i)}; \mathcal{D}_m \right) + \lambda_m^{(i)}$$
$$w_m^{(i+1)} \approx w_m^{(i)} - (\alpha_m Q_m + \rho_m I)^{-1} \Delta z_m$$
$$\lambda_m^{(i+1)} = \lambda_m^{(i)} + \gamma \rho_m \left(w_m^{(i+1)} - v^{(i+1)} \right)$$
$$u_m^{(i+1)} := \rho_m w_m^{(i+1)} + \lambda_m^{(i+1)}$$

21:    **end for**

22:    **for** every $m \notin \mathcal{U}^{(i)}$ **do**

23:      $u_m^{(i+1)} = u_m^{(i)}$

24:    **end for**

25: **end for**

---

communicating and aggregating the parameters of the model, i.e., the server learns in parallel to the selected users. Notice that

we refer to FedTOP-ADMM I and II as common FedTOP-ADMM, where the performance difference between I and II is non-noticeable. Hence, FedTOP-ADMM corresponds to both FedTOP-ADMM I and FedTOP-ADMM II depending on the context.

As our proposed FedTOP-ADMM generalizes the FedADMM algorithm, it inherits all the properties of the FedADMM algorithm, including its communication efficiency when using only data available on the edge devices. In the FedTOP-ADMM, there are three additional hyperparameters compared to FedADMM, namely $\tau^{(i)}$, $\zeta^{(i)}$, and γ. Meanwhile, note that FedADMM requires tuning of $\{\rho_m\}$ parameters—see [8] and also discussion in Section V.

Remark 1: Let $g = 0$, $\gamma = 1$, and $\tau^{(i)} = 0$ or equivalently $h = 0$ and $\zeta^{(i)} = 0$ for all $i = 0, 1, 2, \ldots$. Then, FedTOP-ADMM I boils down to FedADMM and all the convergence results of FedADMM [8, Section 4.3] hold for any $J \geq 1$.

Remark 2: Suppose $J = 1$, $\tau^{(i)} = \tau \in \mathbb{R}_{\geq 0}$ and $\zeta^{(i)} = 0$ for all i, and $\rho_m = \rho$ for all m. Then, FedTOP-ADMM II becomes TOP-ADMM; and consequently, the global convergence Theorem 1 holds for FedTOP-ADMM II.

To establish the convergence of FedTOP-ADMM Algorithm 1, we extend the vanishing property of residual errors proved in Theorem 1. Using this result, we establish the global convergence of FedTOP-ADMM in the following theorem.

Theorem 2 (Global convergence of FedTOP-ADMM algorithm): Consider a problem given in (5) under general convex settings with at least one solution. Let $\{\boldsymbol{w}^{(i)}\}$ be a sequence generated by FedTOP-ADMM iterative scheme in Algorithm 1. Assume all the subproblems of (6) or Algorithm 1 have solutions. Also, in Algorithm 1, for each given period T, assume all users should be active at least once within the users active sets $\{\mathcal{U}^{(i+1)}, \mathcal{U}^{(i+2)}, \ldots, \mathcal{U}^{(i+T)}\}$ for all $i \geq 0$. Consider suitable non-increasing step-size $\{\tau^{(i)} \in \mathbb{R}_{\geq 0}\}$ and $\{\zeta^{(i)} \in \mathbb{R}_{\geq 0}\}$ for $i \geq 0$, and relaxation/penalty parameter $\{\rho_m \in \mathbb{R}_{>0}\}$ with some arbitrary initial $\{\boldsymbol{w}^{(0)}, \boldsymbol{w}_m^{(0)}, \boldsymbol{\lambda}_m^{(0)}\}$. Assume that for any $J \geq 1$, the so-called dual residual $\lim_{i \to +\infty}(\boldsymbol{w}^{(i+1)} - \boldsymbol{w}^{(i)}) = 0$ and primal residual $\lim_{i \to +\infty}(\boldsymbol{w}_m^{(i+1)} - \boldsymbol{w}^{(i+1)}) = 0$, $\forall m = 1, \ldots, M$. Then, the generated sequence $\{\boldsymbol{w}^{(i)}\}$ globally converges to a KKT point of (5) at any limit point.

Proof: See Appendix C. ∎

We describe the necessary processing steps of our proposed Algorithm 1 as follows. In Algorithm 1, the total number of users participating in the FL process is denoted by M. We specify the maximum number of global iterations as I. However, note that many heuristics-based early stopping techniques can potentially be employed on the server, e.g., when reaching the required test accuracy, but are not considered herein.

In Step-1 of Algorithm 1, we provide the required inputs to the algorithm with appropriate initialization of vectors and (iterative) parameters, including the set $\mathcal{U}^{(0)}$ of selected users for the communication with the server. We denote communication events with users by $\mathcal{P} := \{0, J, 2J, \ldots\}$, which shows periodic events[4] at every J iterations. Therefore, J represents the number of local iterations on the users' side. Consequently,

the communication rounds in Algorithm 1 is given by [8]

$$\text{Communication rounds} := \lfloor i/J \rfloor, \tag{7}$$

where $\lfloor \cdot \rfloor$ denotes flooring to the nearest integer.

The iterative FedTOP-ADMM algorithm starts at Step-2. Note that FedTOP-ADMM stops when the total number of global iterations I is exhausted, or one employs an early stopping criteria within the loop, e.g., using test accuracy criteria as the server is expected to have some test dataset.

If FedTOP-ADMM I is employed, then at Step-3 and Step-4 for any global iteration i, the server performs some intermediate processing reminiscent of gradient descent-like step. Conversely, if FedTOP-ADMM II is employed, then when there is no communication event between the server and any users for a given iteration i, i.e., $i \notin \mathcal{P}$, the server performs the same processing in Step-3 and Step-4 as FedTOP-ADMM I. This intermediate step processing at server, i.e., Step-4, is part of the global weight vector $\boldsymbol{w}$ update—cf. (8) and Step-4 of Algorithm 1. More specifically, assuming g is closed, convex, and proper function (possibly nonsmooth), the solution corresponding to the subproblem (6a), i.e., the $\boldsymbol{w}$ update, reads

$$\boldsymbol{w}^{(i+1)}$$

$$= \mathbf{prox}_{\nu^{(i)}g}\left(\frac{1}{\nu^{(i)}}\left[\sum_{m=1}^{M} \boldsymbol{u}_m^{(i)} - \tau^{(i)}\nabla h\left(\boldsymbol{w}^{(i)}\right) + \zeta^{(i)}\boldsymbol{w}^{(i)}\right]\right), \tag{8}$$

where the definition of proximal operator is given in Definition 3 of Appendix A, $\nu^{(i)} := 1/(\sum_{m \in [M]} \rho_m + \zeta^{(i)})$, and $\boldsymbol{u}_m^{(i+1)} := \rho_m \boldsymbol{w}_m^{(i+1)} + \boldsymbol{\lambda}_m^{(i+1)}$. In the FedTOP-ADMM algorithm, instead of directly using $\{\boldsymbol{u}_m^{(i)}\}$ in (8), the server utilizes $\boldsymbol{v}_m^{(i)}$, which is updated as described from Step-7 to Step-12 of Algorithm 1. When communication events $i \in \mathcal{P}$ occur, the server receives parameter vector $\boldsymbol{u}_m^{(i)}$ from selected user $m \in \mathcal{U}^{(i)}$ and consequently updates $\boldsymbol{v}_m^{(i)}$. Note the difference between the update in (8) with the Step-13 of Algorithm 1, which aggregates all the weights of users and server appropriately. Consequently, the global updated weight vector $\boldsymbol{w}^{(i+1)}$ is generated at Step-13.

The second subproblem of TOP-ADMM (6b), corresponding to the primal update of weight vector $\boldsymbol{w}_m^{(i+1)}$ for each user m, is equivalent to the subproblem in the classical ADMM or FedADMM [8], [31]. We use the inexact solution to this subproblem, i.e., a linear approximation of the function f_m, proposed in [8], [31], in which the recipe is given in Step-19 of Algorithm 1. We refer the interested readers to [8], [31] for the detailed analysis of the inexact solution. Step-19 is repeated J-times before the server receives the updated parameter $\boldsymbol{u}_m^{(i+1)}$ from the selected user $m \in \mathcal{U}^{(i)}$. In Step-23, if the user m was not selected for the communication with the server, i.e., $m \notin \mathcal{U}^{(i)}$, the server essentially utilizes the previously received parameter from the nonselected user m.

B. Comparison Among ADMM, FedADMM, TOP-ADMM, and FedTOP-ADMM

To this end, we would like to accentuate the prowess of three-operator algorithms, such as our proposed TOP-ADMM/FedTOP-ADMM, compared to existing two-operator ADMM (2) or one-operator FedADMM [8]. There are many problems of interest where one-operator such as existing

---

[4]It is straightforward to consider aperiodic communication events $\mathcal{P}$ in the same framework, which would resemble an asynchronous setup. However, study of aperiodic communications is deferred to the future work.

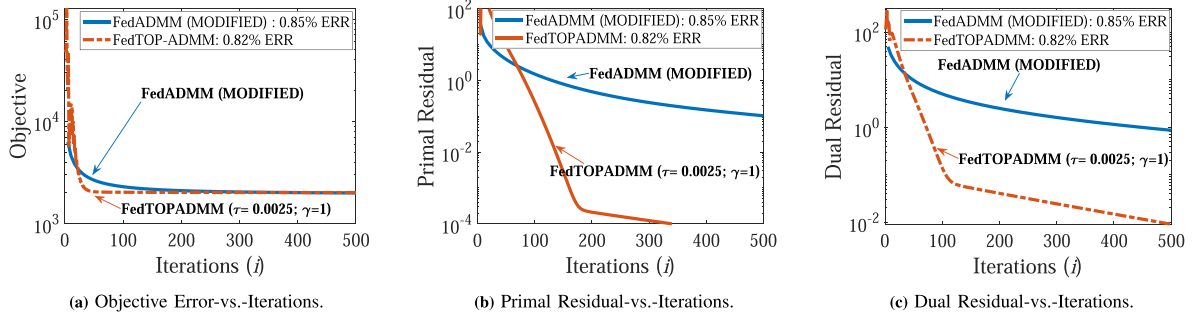| One-operator-based FedADMM [8]	Two-operator ADMM (1)	Three-operator-based TOP-ADMM (4)/(6) or FedTOP-ADMM Algorithm 1
$\min \sum_{m=1}^{M} f_m(\boldsymbol{w}_m)$ s.t. $\boldsymbol{w}_m = \boldsymbol{w}$ $\forall m$	$\min \sum_{m=1}^{M} f_m(\boldsymbol{w}_m) + g(\boldsymbol{w})$ s.t. $\boldsymbol{w}_m = \boldsymbol{w}$ $\forall m$	$\min \sum_{m=1}^{M} f_m(\boldsymbol{w}_m) + g(\boldsymbol{w}) + \beta h(\boldsymbol{w})$ s.t. $\boldsymbol{w}_m = \boldsymbol{w}$ $\forall m$

Fig. 2. Comparison of convergence behaviour between `FedADMM` and `FedTOP-ADMM` for the distributed *sparse* logistic regression problem (9).

`FedADMM` [8] is not capable to tackle the problem having non-differentiable or non-smooth regularizer—see, e.g., sparse logistic regression problem (9a) with non-differentiable ℓ_1 norm. More concretely, see Table I that compares which algorithm can solve what form of distributed optimization problems. Recall, `FedTOP-ADMM` is built on the extended version of (three-operator) TOP-ADMM (6), where the extended TOP-ADMM (6) include variable "proximity" convex regularizers to the primal $\boldsymbol{w}$-update and Glowinski's relaxation parameter to the dual $\boldsymbol{\lambda}$-update in contrast to the vanilla TOP-ADMM (4). Moreover, `FedTOP-ADMM` generalizes (one-operator) `FedADMM` and essentially two-operator ADMM. However, this existing one-operator `FedADMM` utilizes (two-operator) ADMM. Nonetheless, clearly, the `FedTOP-ADMM` in Algorithm 1 becomes `FedADMM` when $\tau^{(i)} = 0$ and $\zeta^{(i)} = 0$ for all $i = 0, 1, \ldots, I-1$ iterations.

C. Connections With Existing Works on FL

The other benchmarking algorithms besides `FedADMM` [8] are `FedProx` [3] and `FedAvg` [2], which can easily coexist with the Algorithm 1 framework as highlighted in [8]. Specifically, in Step-19 of Algorithm 1, the `FedTOP-ADMM` becomes `FedProx`/`FedAvg` by setting the following TOP-ADMM parameters to zero, i.e., $\tau^{(i)} = 0, \zeta^{(i)} = 0$, and $\gamma = 0$ and replacing the weight update for each user with

$$
\boldsymbol{u}_m^{(i+1)} := \boldsymbol{w}_m^{(i+1)} \approx \boldsymbol{w}_m^{(i)}
$$
$$
- \eta \left[\nabla f_m\left(\boldsymbol{w}_m^{(i)}\right) + \mu\left(\boldsymbol{w}_m^{(i)} - \boldsymbol{v}^{(i+1)}\right) \right],
$$

where η is a step size and μ is a scaling parameter for the proximal term, in which $\mu = 0$ for `FedAvg`. Moreover, note that `FedADMM-VC` represents `FedADMM` with a virtual client collocated at the edge server.

V. NUMERICAL RESULTS

In this section, we conduct the experiments using the distributed (sparse) logistic regression to benchmark the performance of these existing algorithms against our proposed

`FedTOP-ADMM` algorithms.

$$
\underset{\{\boldsymbol{w}_m \in \mathbb{R}^n\}, \boldsymbol{w} \in \mathbb{R}^n}{\text{minimize}} \quad \sum_{m=1}^{M} \frac{1}{d_m} \sum_{j=1}^{d_m}
$$

$$
\left[\log\left(1 + \exp\left\{(\boldsymbol{A}_m[:,j])^{\mathrm{T}} \boldsymbol{w}_m\right\}\right) \right.
$$
$$
\left. -\boldsymbol{t}_m[j]\left\{(\boldsymbol{A}_m[:,j])^{\mathrm{T}} \boldsymbol{w}_m\right\} + \frac{\kappa}{2}\|\boldsymbol{w}_m\|_2^2 \right]
$$
$$\tag{9a}$$

$$
+ \upsilon \|\boldsymbol{w}\|_1 \tag{9b}
$$

$$
\text{subject to} \quad \boldsymbol{w}_m - \boldsymbol{w} = \boldsymbol{0}, \quad \forall m = 1, \ldots, M, \tag{9c}
$$

where the training dataset on the user m is $\mathcal{D}_m := \{\boldsymbol{t}_m, \boldsymbol{A}_m\}_{m=1}^{M}$, i.e., the binary output is $\boldsymbol{t}_m \in \{0, +1\}^{d_m}$, the input feature matrix is $\boldsymbol{A}_m \in \mathbb{R}^{n \times d_m}$, and the regression weight vector is $\boldsymbol{w} \equiv \boldsymbol{w}_m \in \mathbb{R}^n$. The scaling factor to the regularizer is $\kappa = 0.001$ in the experiments, unless otherwise mentioned. In case of non-sparse logistic regression problem, one can ignore the ℓ_1-norm regularizer (9b) by setting zero to $\upsilon \in \mathbb{R}_{\leq 0}$. Notice that the non-sparse problem is also used in [8].

For all the considered methods including our proposed method, the loss or objective function at each user m or at the server for `FedTOP-ADMM` in (9) reads

$$
h(\boldsymbol{w}) \equiv f_m(\boldsymbol{w}_m) := \frac{1}{d_m} \sum_{j=1}^{d_m}
$$

$$
\times \left[\log\left(1 + \exp\left\{(\boldsymbol{A}_m[:,j])^{\mathrm{T}} \boldsymbol{w}_m\right\}\right) \right.
$$
$$
\left. -\boldsymbol{t}_m[j]\left\{(\boldsymbol{A}_m[:,j])^{\mathrm{T}} \boldsymbol{w}_m\right\} + \frac{\kappa}{2}\|\boldsymbol{w}_m\|_2^2 \right]. \tag{10}
$$

A. Experimental Settings

In this section, we present numerical results to illustrate the performance of our proposed `FedTOP-ADMM I` and `FedTOP-ADMM II` algorithms compared to the state-of-the-art algorithms `FedADMM` [8], `FedProx` [3], and `FedAvg` [2].

Before we proceed further with more realistic dataset and comparison of our proposed `FedTOP-ADMM` with the above-mentioned benchmarking methods, we want to highlight the strength of our proposed `FedTOP-ADMM` in contrast to these

(a) Without any scaling   **(b)** With a scaling approach 1   **(c)** With a proposed scaling approach 2
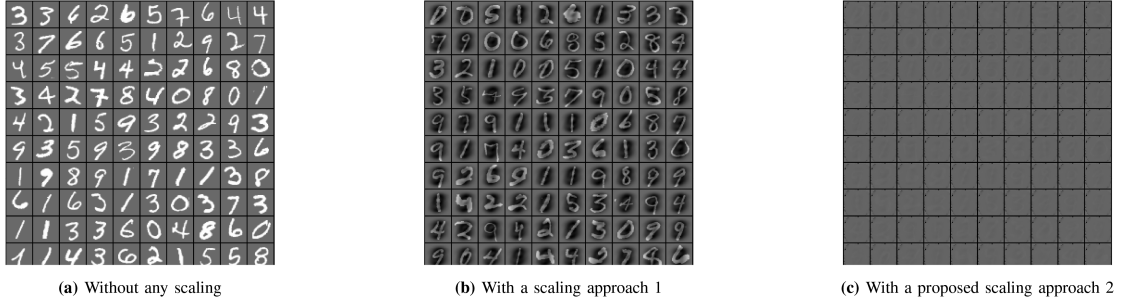
Fig. 3. Examples of MNIST handwritten digits without scaling and with two different scaling approaches.
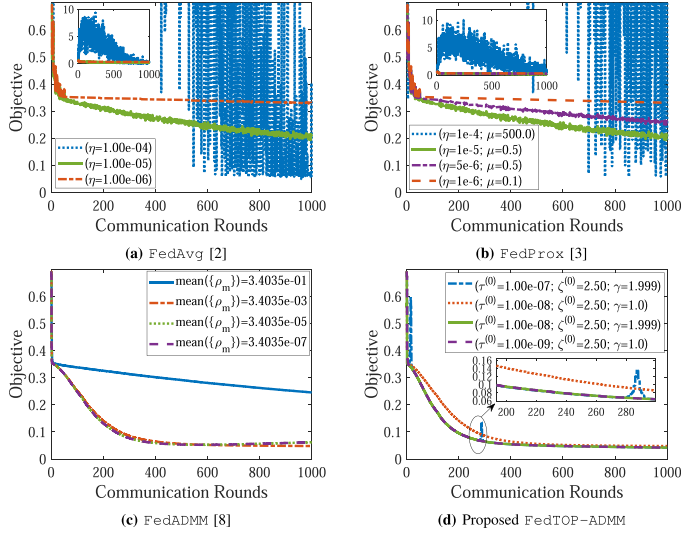

Fig. 4. Convergence analysis of existing, `FedAvg`, `FedProx`, and `FedADMM`, and our proposed `FedTOP-ADMMI/II` algorithms for various hyperparameters under $J = 10$ local iterations.

Fig. 5. Comparison of `FedTOP-ADMM I/II` with `FedADMM` and `FedADMM-VC` under $J = 10$.

existing algorithms. In other words, we show the potency of three-operator structure of `FedTOP-ADMM` in contrast to these existing one-operator, i.e., corresponding to sum of separable f_m functions, `FedAvg`, `FedProx`, `FedADMM`. More concretely, we show how easily our proposed three-operator `FedTOP-ADMM` can exploit the non-differentiable regularizer (9b) that can easily be handled by setting, say, function g as a scaled ℓ_1-norm, whose proximal operator is well known in the literature, see, e.g., [25], [28]. Although it is unclear how to employ ℓ_1-norm regularizer with existing one-operator methods `FedAvg` and `FedProx` because of non-differentiablity of ℓ_1-norm, we can only modify `FedADMM` to incorporate the second operator corresponding to function g since `FedADMM` is built on the classical consensus two-operator ADMM (1). Thus, in the *modified* `FedADMM`, after the global aggregation one can include the second operator, i.e., proximal operator corresponding to the function g, without the third operator corresponding to the gradient of h. More precisely, the Step-13 of Algorithm 1 for the modified `FedADMM` without the third operator (corresponding to the gradient of function h) can be expressed as $\boldsymbol{w}^{(i+1)} = \mathbf{prox}_{\nu^{(i)}g}\big(\frac{1}{\nu^{(i)}}[\sum_{m=1}^{M} \boldsymbol{v}_m^{(i)}]\big)$ with $\gamma = 1$. The numerical convergence behaviour of `FedTOP-ADMM` and modified `FedADMM`, in terms of objective, primal residual, and dual residual against iterations are depicted in Fig. 2. In this simple but illustrative example, the random test setup is similar to [21, Section 11.2]. Additionally, in this toy example, we have
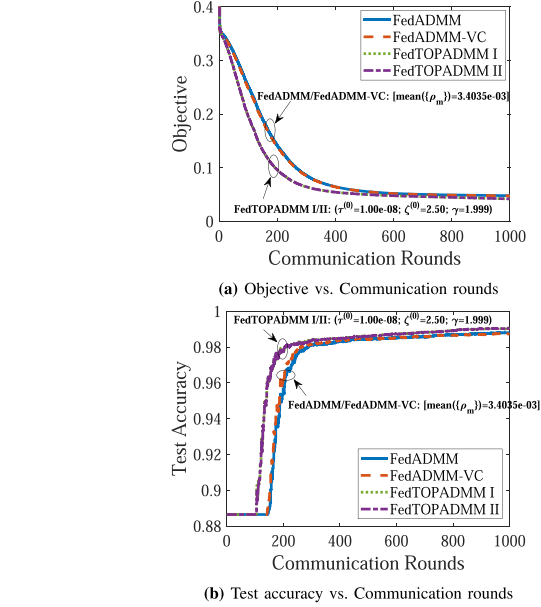
generated the synthetic sparse training dataset with a total of 20000 examples having a feature vector length $n = 100$. We have employed $M = 100$ users that are distributed, and each user has $d_M = 200$ training examples, where all the users are active and $J = 1$. Clearly, `FedTOP-ADMM` shows faster convergence than `FedADMM`, while delivering similar/better training error (0.82%) than that of `FedADMM` (0.85%).

In our next set of experiments, we have ignored the sparse parameter in the logistic regression such that we can compare `FedTOP-ADMM` with not only (modified) `FedADMM` but also `FedAvg` and `FedProx`. Additionally, in all the subsequent considered simulations, the total number of users is fixed to 200, i.e., $M = 200$. However, 10 users are selected uniformly at random during each communication event of the global iteration, i.e., $i \in \mathcal{P}$.

We have conducted experiments using one of the most popular real-world datasets, namely MNIST [41]. Specifically, we have scaled/normalized the MNIST input data. There are many ways to scale the input data matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, where $d = \sum_{m=1}^{M} d_m$ and n represent the total number of data samples for training and testing, and feature vector length, respectively. Moreover, we have analyzed two scaling approaches: 1) $\boldsymbol{a} \in \mathbb{R}^{n \times 1}$, $\boldsymbol{a} := \mathtt{mean}(\boldsymbol{A}, 2) \oslash \mathtt{std}(\boldsymbol{A}, [], 2)$, and 2) $\boldsymbol{a} := \mathtt{mean}(\boldsymbol{A}, 2) \oslash \mathtt{var}(\boldsymbol{A}, [], 2) \in \mathbb{R}^{n \times 1}$, where $\oslash$
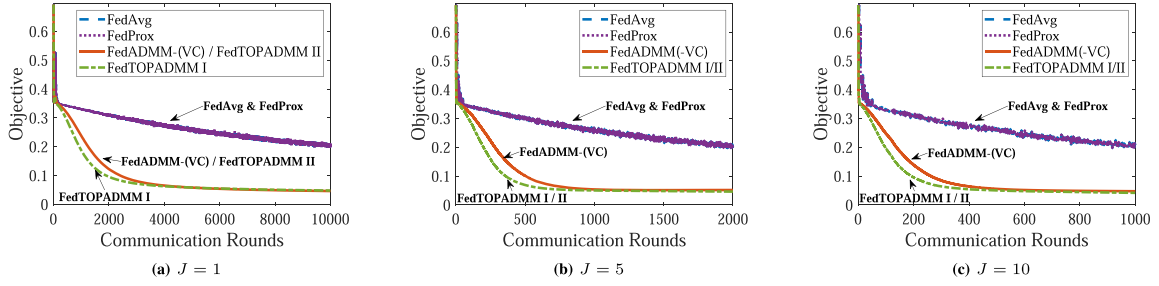
Fig. 6. Performance comparison, in terms of objective against communication rounds, of proposed `FedTOP-ADMM I`/`FedTOP-ADMM II` with `FedADMM`/`FedADMM-VC`, `FedProx`, and `FedAvg`, considering three cases of local iterations on the users' side, i.e., $J \in \{1, 5, 10\}$.

corresponds to elementwise division[5], and `mean`, `std`, and `var` represent the mean, standard deviation, and variance along the second dimension of matrix $\boldsymbol{A}$ as used in MATLAB expressions. Finally, the scaled version of the MNIST dataset is expressed as $\boldsymbol{A} \leftarrow \boldsymbol{A} - \mathbf{1}_{1 \times d} \otimes \boldsymbol{a}$, where $\mathbf{1}_{1 \times d}$ denotes a row vector of all ones with dimension $1 \times d$, and $\otimes$ represents a Kronecker product. Additionally, we use other popular datasets, such as CIFAR-10 and CIFAR-100 [42].

We evaluate the performance using two data partitioning: 1) independent and identically distributed (i.i.d.), where the data is randomly shuffled and the corresponding labels are shuffled accordingly, and 2) non-i.i.d., where the training labels are sorted in ascending order and the corresponding input data is ordered accordingly. Thus, in case of MNIST dataset, this non-i.i.d. data split is one of the pathological cases because each user or base station would have at most two class labels. In Fig. 3, we illustrate the examples of MNIST digits without, see Fig. 3(a) and with these two aforementioned scaling approaches, see Figs. 3(b)–3(c). Based on our exhaustive experiments, we have found the second approach in Fig. 3(c) more challenging to learn than the first scaling approach in Fig. 3(b) and the unscaled original version in Fig. 3(a). Consequently, we have employed the second scaling approach for our further numerical analysis. The data distribution among the users and the server is i.i.d. unless otherwise stated. Moreover, we have considered a binary classifier by simply employing digit 1 as the true label and other digits as false labels.

We perform grid search to tune the hyperparameters of the benchmarking algorithms including our proposed algorithms. The step size of the gradient descent in `FedAvg` is chosen from the candidate set, i.e., $\eta \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}\}$. For `FedProx`, in addition to the step size η, we need to tune the hyperparameter μ for the proximal term, which is appropriately selected from the set, i.e., $\mu \in \{0.01, 0.1, 0.5, 500\}$. The hyperparameters of `FedADMM` are chosen as suggested in [8], [31]. More specifically, the inexact version of `FedADMM` sets $\boldsymbol{Q}_m = r_m \boldsymbol{I}$ with $r_m = \text{eig}_{\max}(\boldsymbol{A}_m^{\mathrm{T}} \boldsymbol{A}_m)/(4 + \kappa)$ that utilizes the maximum eigenvalue of $\boldsymbol{A}_m^{\mathrm{T}} \boldsymbol{A}_m$ Gram matrix of input data, such that the hyperparameter $\rho_m(a) = [a \log(M d_m) \alpha_m r_m]/(\log(2 + J))$ with $\alpha_m = 1/(M d_m)$ and a user defined parameter a. We have considered the following mean values of the hyperparameters $\text{mean}(\{\rho_m\}) \in \{3.4035\text{e-}1, \ 3.4035\text{e-}2, \ 3.4035\text{e-}3, \ 3.4035\text{e-}4, \ 3.4035\text{e-}5, 3.4035\text{e-}6, \ 3.4035\text{e-}7\}$ using appropriate values of a and the

maximum eigenvalues of the input data matrices. Subsequently, the tunable parameters in `FedTOP-ADMM` are chosen in common with `FedADMM`. Furthermore, the extra tunable parameters of `FedTOP-ADMM`, particularly, the parameters on the server side $\tau^{(i)}$ and $\zeta^{(i)}$, are selected to be monotonically decreasing with increasing iterations. Specifically, we have employed the following recipe for both $\tau^{(i)}$ and $\zeta^{(i)}$ to decrease monotonically over iterations: $\beta^{(i+1)} = \beta^{(0)}/(1 + [i\mu'\beta^{(i)}])$, where $\beta := \tau$ and $\beta := \zeta$, appropriately, and $\mu' = 10$. The candidate set of additional tunable parameters of `FedTOP-ADMM` are $\tau^{(0)} \in \{10^{-1}, 10^{-2}, 10^{-3}, \dots, 10^{-9}\}$, $\zeta^{(0)} \in \{5, 2.5, 1, 0.5, 0.025, 0.005, 0.00025\}$, and $\gamma \in \{0.1, 0.5, 1, 1.5, 1.999\}$.

B. Experimental Results

Fig. 4 illustrates the convergence behaviour of `FedAvg` [2], `FedProx` [3], `FedADMM` [8], and our proposed `FedTOP-ADMM` for some chosen set of parameters from the candidate set. Based on these numerical results, we have selected the following parameters of the respective methods: 1) `FedAvg` :- $\eta = 10^{-5}$, 2) `FedProx`:- $\eta = 10^{-5}; \mu = 0.5$, 3) `FedADMM`/`FedADMM-VC` :- $\text{mean}(\{\rho_m\}) = 3.4035e - 3$, and 4) `FedTOP-ADMM I`/`FedTOP-ADMM II` :- $\text{mean}(\{\rho_m\}) = 3.4035e - 3$, $\tau^{(0)} = 1e - 8$, $\zeta^{(0)} = 2.5$, and $\gamma = 1.999$ unless otherwise mentioned.

Fig. 5 compares the performance in terms of both loss function or objective (10) and test accuracy, among `FedTOP-ADMM I`, `FedTOP-ADMM II`, `FedADMM`, and `FedADMM-VC` with the aforementioned chosen parameters. Additionally, Fig. 6 compares the performance for $J \in \{1, 5, 10\}$. Noticeably, these results substantiate our argument that exploiting the data knowledge on the edge server using our proposed `FedTOP-ADMM` schemes outperform `FedADMM-VC`, i.e., with a virtual client. Moreover, these results indicate that `FedTOP-ADMM II` has non-noticeable performance loss compared to `FedTOP-ADMM I` when $J > 1$. For instance, with $J = 10$, `FedTOP-ADMM` has a gain of up to 33% in the communication efficiency with respect to `FedADMM` to reach a test accuracy of 98%. Furthermore, as mentioned before, `FedTOP-ADMM II` boils down to `FedADMM` when $J = 1$.

In Fig. 7, we compare the performance of `FedTOP-ADMM` with `FedADMM` and `FedProx` under non-i.i.d. distribution of the MNIST dataset for $J = 10$. Recall that this non-i.i.d. data split is one of the pathological cases because each user or base station would have at most two class labels. Nevertheless, `FedProx` performs slightly better than `FedAvg`. However, both `FedTOP-ADMM` and `FedADMM` outperform `FedProx` and

---

[5]Because of numerical division, if any element is not a number, we replace the element by zero.

(a) Objective vs. communication rounds   (b) Test accuracy vs. communication rounds
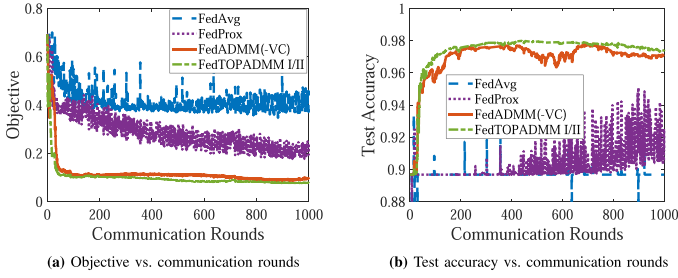
Fig. 7. Performance comparison of proposed `FedTOP-ADMMI/II` with existing `FedADMM(-VC)`, `FedProx`, and `FedAvg` under non-i.i.d. data without scaling MNIST dataset and $J = 10$.
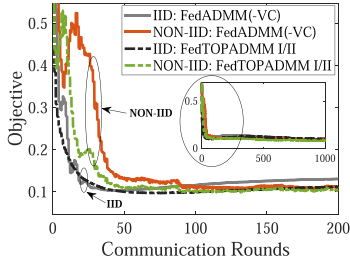

Fig. 8. Performance comparison among i.i.d. and non-i.i.d. (without scaling of) MNIST dataset.
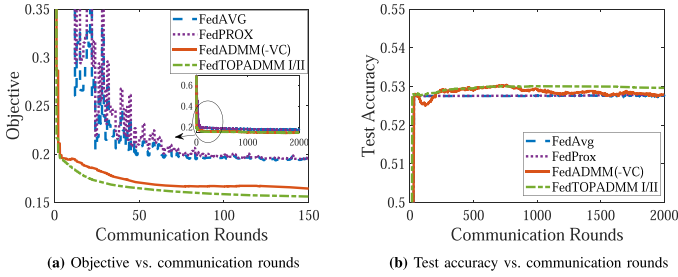

(a) Objective vs. communication rounds   (b) Test accuracy vs. communication rounds

Fig. 9. Performance comparison of proposed `FedTOP-ADMMI/II` with considered existing methods for CIFAR-100 i.i.d. dataset (coarse labels) and $J = 10$.

`FedAvg`. Further, `FedTOP-ADMM` has a gain of up to 27% in the communication efficiency with respect to `FedADMM` to reach a test accuracy of 97% under non-i.i.d. distribution of MNIST dataset. Observe that we have changed the tunable parameters of all the methods compared to the previous results of MNIST. In particular, we have chosen the following parameters of respective methods: a) `FedTOP-ADMMI/II` $(\text{mean}(\{\rho_m\}) = 6.5731e - 6, \tau^{(0)} = 1e - 7, \zeta^{(0)} = 1.5)$, b) `FedADMM(-VC)` $(\text{mean}(\{\rho_m\}) = 6.5731e - 6)$, c) `FedProx` $(\eta = 1e - 3; \mu = 0.5)$, and d) `FedAvg` $(\eta = 0.5e - 3)$. For completeness, in Fig. 8, we compare the performance of non-i.i.d. with i.i.d. data considering the same tunable parameters used for Fig. 7, where the performance under i.i.d. data is unsurprisingly slightly better than the performance under non-i.i.d. data.

Lastly, in Fig. 9 and Fig. 10, we present the performance for CIFAR-100 and CIFAR-10 dataset, respectively. We can construe the similar performance trend as observed in MNIST.

VI. CONCLUSION

In this paper, we proposed a novel `FedTOP-ADMM` algorithmic framework for communication-efficient FL utilizing our
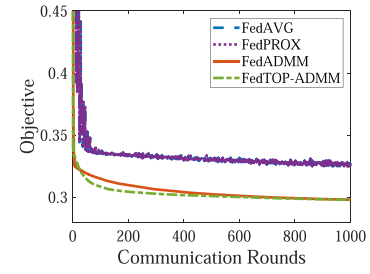

Fig. 10. Objective vs. iterations for $J = 10$ and CIFAR-10 i.i.d. dataset.

recently proposed consensus TOP-ADMM algorithm, which can tackle the sum of three composite functions in a distributed manner. Specifically, we developed two variants of `FedTOP-ADMM`, namely `FedTOP-ADMM I` and `FedTOP-ADMM II` that learn a considered global machine learning model using data on both the edge server and the users. Our experiments showed that FedTOP-ADMM has a significant gain of up to 33% in the communication efficiency with respect to `FedADMM` to reach a desired test accuracy of 98% using the proposed scaling of the MNIST dataset. For future works, we intend to establish the convergence analysis of FedTOP-ADMM for $J > 1$ and enhanced TOP-ADMM. Moreover, we intend to investigate the scheduling of edge devices to participate in the FL using FedTOP-ADMM, as well as the power allocation of the selected devices.

APPENDIX A
SOME USEFUL LEMMAS AND DEFINITIONS

We present herein some useful definitions, propositions and lemmas that are important to ADMM methods.

Definition 1 (L-smooth function [5], [25]): A differentiable function $f : \mathbb{C}^n \to \mathbb{R}$ is L-smooth, i.e., has L-Lipschitz continuous gradient (for $L > 0$) if $\|\nabla f(z_1) - \nabla f(z_2)\| \leq L\|z_1 - z_2\|, \forall z_1, z_2 \in \mathbb{C}^n$.

Definition 2 (Subgradient [5, Definition 16.1]): Given a proper function $f : \mathbb{C}^n \to \mathbb{R}$, a vector $g \in \mathbb{C}^n$ is denoted as a subgradient of $f(z)$ at some point x if

$$f(z) \geq f(x) + 2\Re\left\{g^{\mathrm{H}}(z - x)\right\}, \quad \forall z \in \mathbb{C}^n. \tag{11}$$

Definition 3 (Proximal mapping [22] [25]): Let us consider a proper closed convex function $f : \mathrm{dom}_f \mapsto (-\infty, +\infty]$, where dom_f corresponds to the domain of a function f. Then, the proximal mapping of f is the operator given by:

$$\mathbf{prox}_{\lambda f}(x) := (I + \lambda \partial f)^{-1}(x),$$
$$= \arg \min_{z \in \mathrm{dom}_f}\left\{f(z) + \frac{1}{\beta\lambda}\|x - z\|_2^2\right\},$$

for any $x \in \mathrm{dom}_f$, where ∂f is a subdifferential of f [5], [43], and $\lambda > 0$. If z is complex-valued or real-valued, $\beta = 1$ or $\beta = 2$, respectively. Note that the proximal operator to an indicator function becomes an orthogonal projection, i.e., $\mathbf{prox}_{\lambda \delta_C}(z) = \mathbf{proj}_C(z)$.

APPENDIX B
CONVERGENCE ANALYSIS OF TOP-ADMM

To establish the convergence of TOP-ADMM algorithm 4, we first present two standard assumptions from ADMM proofs in the literature followed by five lemmas in the sequel. Subsequently, these assumptions and five lemmas are required to prove Proposition 1, which guarantees that the primal and dual

residual errors vanish asymptotically. Finally, we establish the global convergence of TOP-ADMM, i.e., proof of Theorem 1. Although the proof structure is inspired by the convergence results from the classic ADMM [21], our convergence analysis results are new from the sum of three functions with consensus constraints perspective, i.e., for TOP-ADMM.

Towards the convergence analysis goal, we define the augmented Lagrangian to problem (3) as

$$
\mathcal{L}_\rho \left(\{\boldsymbol{x}_m\}_{m=1}^M, \boldsymbol{z}, \{\boldsymbol{y}_m\}_{m=1}^M \right)
$$

$$
:= \sum_{m=1}^M f_m\left(\boldsymbol{x}_m\right) + g\left(\boldsymbol{z}\right) + \beta h\left(\boldsymbol{z}\right)
$$

$$
+ \sum_{m=1}^M 2\Re\left\{ \boldsymbol{y}_m^{\mathrm{H}}\left(\boldsymbol{x}_m - \boldsymbol{z}\right) \right\} + \sum_{m=1}^M \rho \left\| \boldsymbol{x}_m - \boldsymbol{z} \right\|_2^2, \quad (12)
$$

and, for brevity, we define the objective value at iteration i as

$$
p^{(i)} := \sum_{m=1}^M f_m\left(\boldsymbol{x}_m^{(i)}\right) + g\left(\boldsymbol{z}^{(i)}\right) + \beta h\left(\boldsymbol{z}^{(i)}\right). \quad (13)
$$

Then, let us consider two assumptions that are standard in ADMM literature [21], [24].

Assumption 1: Let $(\{\boldsymbol{x}_m^\star\}_{m=1}^M, \boldsymbol{z}^\star, \{\boldsymbol{y}_m^\star\}_{m=1}^M)$ be a saddle point for the (unaugmented) Lagrangian $\mathcal{L}_0$ in (12). Specifically, the following holds for all $\{\boldsymbol{x}_m\}_{m=1}^M, \boldsymbol{z}, \{\boldsymbol{y}_m\}_{m=1}^M$:

$$
\mathcal{L}_0 \left(\{\boldsymbol{x}_m^\star\}_{m=1}^M, \boldsymbol{z}^\star, \{\boldsymbol{y}_m\}_{m=1}^M \right)
$$

$$
\leq \mathcal{L}_0 \left(\{\boldsymbol{x}_m^\star\}_{m=1}^M, \boldsymbol{z}^\star, \{\boldsymbol{y}_m^\star\}_{m=1}^M \right),
$$

$$
\leq \mathcal{L}_0 \left(\{\boldsymbol{x}_m\}_{m=1}^M, \boldsymbol{z}, \{\boldsymbol{y}_m^\star\}_{m=1}^M \right).
$$

Assumption 2: Consider subproblems (4a) and (4b). We assume that each subproblem has at least one solution.

Note that Assumption 2 does not require the uniqueness of the solution.

Lemma 1: Consider the optimal objective, $p^\star$, and the objective at iteration $i + 1$, $p^{(i+1)}$ defined in (13). Then, the difference between the optimal objective and the objective at iteration $i + 1$ is as follows:

$$
p^\star - p^{(i+1)} \leq \sum_{m=1}^M 2\Re\left\{ (\boldsymbol{y}_m^\star)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}.
$$

Proof: Using the primal feasibility $\boldsymbol{x}_m^\star - \boldsymbol{z}^\star = \boldsymbol{0} \ \forall m = 1, \ldots, M$, and Assumption 1 (or duality theory [43]), let us write

$$
p^\star := \mathcal{L}_0 \left(\{\boldsymbol{x}_m^\star\}_{m=1}^M, \boldsymbol{z}^\star, \{\boldsymbol{y}_m^\star\}_{m=1}^M \right)
$$

such that

$$
p^\star \leq \mathcal{L}_0 \left(\left\{\boldsymbol{x}_m^{(i+1)}\right\}_{m=1}^M, \boldsymbol{z}^{(i+1)}, \{\boldsymbol{y}_m^\star\}_{m=1}^M \right).
$$

Then, we can write

$$
p^\star \leq \sum_{m=1}^M \left[f_m\left(\boldsymbol{x}_m^{(i+1)}\right) + g\left(\boldsymbol{z}^{(i+1)}\right) + \beta h\left(\boldsymbol{z}^{(i+1)}\right) \right]
$$

$$
+ \sum_{m=1}^M 2\Re\left\{ (\boldsymbol{y}_m^\star)^{\mathrm{H}} \left(\boldsymbol{x}_m^{(i+1)} - \boldsymbol{z}^{(i+1)}\right) \right\},
$$

which can be rewritten as $p^\star \leq p^{(i+1)} + \sum_{m=1}^M 2\Re\{(\boldsymbol{y}_m^\star)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)}\}$, where $\Delta \boldsymbol{r}_m^{(i+1)} = (\boldsymbol{x}_m^{(i+1)} - \boldsymbol{z}^{(i+1)})$, as in Proposition 1. ∎

The following lemma will be useful in Lemma 3.

Lemma 2 (Three-point inequality): Let the convex and differentiable[6] function $h : \mathbb{C}^n \to \mathbb{R}$ have L-Lipschitz continuous gradient (for $L \geq 0$), where $\boldsymbol{z} \in \mathbb{C}^n$, then $\forall \boldsymbol{z}^{(i+1)}, \boldsymbol{z}^{(i)}$, the following inequality holds:

$$
2\Re\left\{ \nabla h\left(\boldsymbol{z}^{(i)}\right)^{\mathrm{H}} \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}\right) \right\} \geq h\left(\boldsymbol{z}^{(i+1)}\right) - h\left(\boldsymbol{z}\right)
$$

$$
- L \left\| \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right\|^2. \quad (14)
$$

Proof: It follows by applying the descent lemma in [5], and the convexity of h. ∎

In the subsequent lemma, we will use the dual residual error definition: $\Delta \boldsymbol{x}_m^{(i+1)} := (\boldsymbol{x}_m^{(i+1)} - \boldsymbol{x}_m^{(i)})$. Observe that we will use Lemma 1 and the following Lemma 3 in Lemma 4.

Lemma 3: The difference between the achieved objective at iteration $i + 1$, i.e., $p^{(i+1)}$, and the optimal objective, $p^\star$, is

$$
p^{(i+1)} - p^\star \leq - \sum_{m=1}^M 2\Re\left\{ \left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right.
$$

$$
+ \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}\right)^{\mathrm{H}} \left(\boldsymbol{x}_m^{(i+1)} - \boldsymbol{x}_m^\star\right) \Big\}
$$

$$
+ \rho M \tau L \left\| \Delta \boldsymbol{z}^{(i+1)} \right\|_2^2. \quad (15)
$$

Proof: We know that $\boldsymbol{x}_m^{(i+1)}$ minimizes the sequence update defined in (4a) such that

$$
\boldsymbol{0} \in \partial f_m\left(\boldsymbol{x}_m^{(i+1)}\right) + \rho \left(\boldsymbol{x}_m^{(i+1)} - \boldsymbol{z}^{(i)} + \frac{\boldsymbol{y}_m^{(i)}}{\rho} \right),
$$

$$
\iff \partial f_m\left(\boldsymbol{x}_m^{(i+1)}\right) \ni -\boldsymbol{y}_m^{(i+1)} - \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}\right), \quad (16)
$$

where we use dual update (4c), the notation ∂ denotes subdifferential [5], [43], and $\iff$ means if and only if and also overloaded as an equivalent operator.

Using the subgradient Definition 2, we then have

$$
f_m\left(\boldsymbol{x}_m\right) \geq f_m\left(\boldsymbol{x}_m^{(i+1)}\right)
$$

$$
+ 2\Re\left\{ \left(-\boldsymbol{y}_m^{(i+1)} - \rho\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}\right) \right)^{\mathrm{H}} \left(\boldsymbol{x}_m - \boldsymbol{x}_m^{(i+1)}\right) \right\},
$$

such that

$$
f_m\left(\boldsymbol{x}_m\right) + 2\Re\left\{ \left(\boldsymbol{y}_m^{(i+1)} + \rho\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}\right) \right)^{\mathrm{H}} \boldsymbol{x}_m \right\}
$$

$$
\geq f_m\left(\boldsymbol{x}_m^{(i+1)}\right)
$$

$$
+ 2\Re\left\{ \left(\boldsymbol{y}_m^{(i+1)} + \rho\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}\right) \right)^{\mathrm{H}} \boldsymbol{x}_m^{(i+1)} \right\}. \quad (17)
$$

Similarly, we also know that $\boldsymbol{z}^{(i+1)}$ minimizes (4b). Then, let us consider $\beta := \rho M \tau$ such that

$$
\boldsymbol{0} \in \partial g\left(\boldsymbol{z}^{(i+1)}\right) + \underbrace{\rho M \tau}_{:=\beta} \nabla h\left(\boldsymbol{z}^{(i)}\right)
$$

---

[6] By differentiability, we mean Wirtinger complex gradient exist for functions $f : \mathbb{C}^n \to \mathbb{R}$, see, e.g., [44].

$$-\sum_{m=1}^{M}\left(\underbrace{\rho\left(\boldsymbol{x}_m^{(i+1)}-\boldsymbol{z}^{(i+1)}\right)+\boldsymbol{y}_m^{(i)}}_{=\boldsymbol{y}_m^{(i+1)}}\right),$$

$$\Longleftrightarrow \partial g\left(\boldsymbol{z}^{(i+1)}\right)\ni -\beta\nabla h\left(\boldsymbol{z}^{(i)}\right)+\sum_{m=1}^{M}\boldsymbol{y}_m^{(i+1)}. \quad (18)$$

Hence, using (18) and the subgradient Definition 2, note that $g(\boldsymbol{z}^\star)\geq g(\boldsymbol{z}^{(i+1)})+2\Re\{(-\beta\nabla h(\boldsymbol{z}^{(i)})+\sum_{m=1}^{M}\boldsymbol{y}_m^{(i+1)})^{\mathrm{H}}(\boldsymbol{z}^\star-\boldsymbol{z}^{(i+1)})\}$. Thus, the following inequalities hold:

$$g\left(\boldsymbol{z}^\star\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^\star\right\}$$

$$\geq g\left(\boldsymbol{z}^{(i+1)}\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^{(i+1)}\right\}$$

$$+\beta 2\Re\left\{\left(\nabla h\left(\boldsymbol{z}^{(i)}\right)\right)^{\mathrm{H}}\left(-\boldsymbol{z}^\star+\boldsymbol{z}^{(i+1)}\right)\right\},$$

$$\overset{(b)}{\geq} g\left(\boldsymbol{z}^{(i+1)}\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^{(i+1)}\right\}$$

$$+\beta h\left(\boldsymbol{z}^{(i+1)}\right)-\beta h(\boldsymbol{z}^\star)-\rho M\tau L\left\|\underbrace{\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}}_{=\Delta\boldsymbol{z}^{(i+1)}}\right\|_2^2, \quad (19)$$

where in (b) we used inequality (14). Then, inequality (19) can be rearranged as

$$g\left(\boldsymbol{z}^{(i+1)}\right)+\beta h\left(\boldsymbol{z}^{(i+1)}\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^{(i+1)}\right\}$$

$$\leq g\left(\boldsymbol{z}^\star\right)+\beta h\left(\boldsymbol{z}^\star\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^\star\right\}$$

$$+\rho M\tau L\left\|\Delta\boldsymbol{z}^{(i+1)}\right\|_2^2. \quad (20)$$

Now, we add (17) for all $m=1,\ldots,M$ into (20) such that

$$\sum_{m=1}^{M}f_m\left(\boldsymbol{x}_m^{(i+1)}\right)$$

$$+2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}+\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)\right)^{\mathrm{H}}\boldsymbol{x}_m^{(i+1)}\right\}$$

$$+g\left(\boldsymbol{z}^{(i+1)}\right)+\beta h\left(\boldsymbol{z}^{(i+1)}\right)$$

$$-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^{(i+1)}\right\},$$

$$\leq \sum_{m=1}^{M}f_m\left(\boldsymbol{x}_m^\star\right)$$

$$+2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}+\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)\right)^{\mathrm{H}}\boldsymbol{x}_m^\star\right\}$$

$$+g\left(\boldsymbol{z}^\star\right)+\beta h\left(\boldsymbol{z}^\star\right)-\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\boldsymbol{z}^\star\right\}$$

$$+\rho M\tau L\left\|\Delta\boldsymbol{z}^{(i+1)}\right\|_2^2,$$

$$\Longleftrightarrow p^{(i+1)}+\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\left(\boldsymbol{x}_m^{(i+1)}-\boldsymbol{z}^{(i+1)}\right)\right.$$

$$\left.+\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)^{\mathrm{H}}\boldsymbol{x}_m^{(i+1)}\right\},$$

$$\leq p^\star+\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\left(\boldsymbol{x}_m^\star-\boldsymbol{z}^\star\right)\right.$$

$$\left.+\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)^{\mathrm{H}}\boldsymbol{x}_m^\star\right\}+\rho M\tau L\left\|\Delta\boldsymbol{z}^{(i+1)}\right\|_2^2. \quad (21)$$

Considering the primal feasibility result, $\boldsymbol{x}_m^\star-\boldsymbol{z}^\star=\boldsymbol{0}$, and the primal residual error definition, $\Delta\boldsymbol{r}_m^{(i+1)}=\boldsymbol{x}_m^{(i+1)}-\boldsymbol{z}^{(i+1)}$, in the above inequality results in

$$p^{(i+1)}-p^\star\leq -\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\Delta\boldsymbol{r}_m^{(i+1)}\right.$$

$$\left.+\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)^{\mathrm{H}}\left(\boldsymbol{x}_m^{(i+1)}-\boldsymbol{x}_m^\star\right)\right\}$$

$$+\rho M\tau L\left\|\Delta\boldsymbol{z}^{(i+1)}\right\|_2^2.$$

We define the following function for the subsequent lemma.

Definition 4: Let a Lyapunov candidate function for the TOP-ADMM algorithm at given iteration i be defined as

$$V^{(i)}:=\sum_{m=1}^{M}\frac{1}{\rho}\left\|\boldsymbol{y}_m^{(i)}-\boldsymbol{y}_m^\star\right\|_2^2+\rho\left\|\boldsymbol{z}^{(i)}-\boldsymbol{z}^\star\right\|_2^2. \quad (22)$$

∎

Lemma 4: The difference between the Lyapunov function (22) at every iteration $i+1$ and the previous iteration i fulfils the following inequality:

$$V^{(i+1)}-V^{(i)}$$

$$\leq \sum_{m=1}^{M}-\rho\left[\left\|\Delta\boldsymbol{r}_m^{(i+1)}\right\|_2^2+\left\|\Delta\boldsymbol{z}^{(i+1)}\right\|_2^2+\tau L\left\|\Delta\boldsymbol{z}^{(i)}\right\|_2^2\right]. \quad (23)$$

Proof: We add the inequalities of Lemma 1 and Lemma 3, and then rearrange the terms such that

$$\underbrace{\left(p^{(i+1)}-p^\star\right)+\left(p^\star-p^{(i+1)}\right)}_{=0}$$

$$\leq -\sum_{m=1}^{M}2\Re\left\{\left(\boldsymbol{y}_m^{(i+1)}\right)^{\mathrm{H}}\Delta\boldsymbol{r}_m^{(i+1)}\right\}$$

$$-\sum_{m=1}^{M}2\Re\left\{\rho\left(\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^{(i)}\right)^{\mathrm{H}}\underbrace{\left(\boldsymbol{x}_m^{(i+1)}-\boldsymbol{x}_m^\star\right)}_{=\Delta\boldsymbol{r}_m^{(i+1)}+\boldsymbol{z}^{(i+1)}-\boldsymbol{z}^\star}\right\}$$

$$+ \sum_{m=1}^{M} 2\Re \left\{ (\boldsymbol{y}_m^\star)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\} + \rho M \tau L \left\| \Delta \boldsymbol{z}^{(i+1)} \right\|_2^2,$$

$$\iff 0 \geq \sum_{m=1}^{M} \Bigg[\underbrace{\Re \left\{ 2 \left(\boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^\star \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}}_{(a)}$$

$$+ 2\Re \left\{ \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}$$

$$+ 2\Re \left\{ \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right) \right\} \Bigg]$$

$$- \rho M \tau L \left\| \Delta \boldsymbol{z}^{(i+1)} \right\|_2^2. \tag{24}$$

We rewrite part (a) of (24) using the dual update (4c), $\boldsymbol{y}_m^{(i+1)} = \boldsymbol{y}_m^{(i)} + \rho \Delta \boldsymbol{r}_m^{(i+1)}$, as follows:

$$\Re \left\{ 2 \left(\boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^\star \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}$$

$$= \Re \left\{ 2 \left(\boldsymbol{y}_m^{(i)} - \boldsymbol{y}_m^\star \right)^{\mathrm{H}} \underbrace{\Delta \boldsymbol{r}_m^{(i+1)}}_{=1/\rho \left(\boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^{(i)} \right)} + \underbrace{\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2}_{=1/\rho \left\| \boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^{(i)} \right\|_2^2} \right\}$$

$$+ \Re \left\{ \rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2 \right\},$$

$$= \frac{1}{\rho} \left(2\Re \left\{ \left(\boldsymbol{y}_m^{(i)} - \boldsymbol{y}_m^\star \right)^{\mathrm{H}} \left(\boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^{(i)} \right) \right\} \right.$$

$$\left. + \left\| \boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^{(i)} \right\|_2^2 \right)$$

$$+ \rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2,$$

$$= \frac{1}{\rho} \left(\left\| \boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^\star \right\|_2^2 - \left\| \boldsymbol{y}_m^{(i)} - \boldsymbol{y}_m^\star \right\|_2^2 \right) + \rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2. \tag{25}$$

Now, replace part (a) of (24) with (25) such that (24) becomes

$$0 \geq \sum_{m=1}^{M} \Bigg[\underbrace{\frac{1}{\rho} \left(\left\| \boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^\star \right\|_2^2 - \left\| \boldsymbol{y}_m^{(i)} - \boldsymbol{y}_m^\star \right\|_2^2 \right)}_{(a)}$$

$$+ \underbrace{\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2}_{(b)} + \underbrace{2\Re \left\{ \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}}_{(c)}$$

$$+ \underbrace{2\Re \left\{ \rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right) \right\}}_{(d)} \Bigg]$$

$$- \rho M \tau L \left\| \Delta \boldsymbol{z}^{(i+1)} \right\|_2^2. \tag{26}$$

We rewrite the sum of parts (b)–(d) of (26) such that

$$\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2 + \Re \left\{ 2\rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}$$

$$+ \Re \left\{ 2\rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \left(\underbrace{\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star}_{=\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right) + \left(\boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right)} \right) \right\},$$

$$= \underbrace{\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} \right\|_2^2 + \Re \left\{ 2\rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \Delta \boldsymbol{r}_m^{(i+1)} \right\}}_{(a)}$$

$$+ \underbrace{\rho \left\| \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right\|_2^2}_{(b)} + \rho \left\| \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right\|_2^2$$

$$+ \Re \left\{ 2\rho \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right)^{\mathrm{H}} \left(\boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right) \right\}. \tag{27}$$

Note that the sum of part (a) and (b) of (27) equals $\rho \| \Delta \boldsymbol{r}_m^{(i+1)} + (\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}) \|_2^2$. Thus, (27) can be expressed as follows:

$$\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} + \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right) \right\|_2^2$$

$$+ \rho \left\| \underbrace{\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}}_{=\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right) - \left(\boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right)} \right\|_2^2$$

$$+ \Re \left\{ 2\rho \left(\underbrace{\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)}}_{=\left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right) - \left(\boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right)} \right)^{\mathrm{H}} \left(\boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right) \right\},$$

$$= \rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} + \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right) \right\|_2^2 + \rho \left\| \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right\|_2^2$$

$$- \rho \left\| \boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right\|_2^2. \tag{28}$$

Now, we substitute the sum of parts (b)–(d) of (26) with (28) such that (26) becomes

$$0 \geq \sum_{m=1}^{M} \Bigg[\left(\frac{1}{\rho} \left(\left\| \boldsymbol{y}_m^{(i+1)} - \boldsymbol{y}_m^\star \right\|_2^2 \right) + \rho \left\| \boldsymbol{z}^{(i+1)} - \boldsymbol{z}^\star \right\|_2^2 \right)$$

$$- \left(\frac{1}{\rho} \left(\left\| \boldsymbol{y}_m^{(i)} - \boldsymbol{y}_m^\star \right\|_2^2 \right) + \rho \left\| \boldsymbol{z}^{(i)} - \boldsymbol{z}^\star \right\|_2^2 \right) \Bigg]$$

$$+ \sum_{m=1}^{M} \rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} + \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right) \right\|_2^2$$

$$- \rho M \tau L \left\| \Delta \boldsymbol{z}^{(i+1)} \right\|_2^2. \tag{29}$$

Using the Lyapunov definition—see (22)—in (29), the inequality can be rearranged as

$$V^{(i+1)} - V^{(i)}$$

$$\leq \sum_{m=1}^{M} -\rho \left\| \Delta \boldsymbol{r}_m^{(i+1)} + \left(\boldsymbol{z}^{(i+1)} - \boldsymbol{z}^{(i)} \right) \right\|_2^2$$

$$+ \rho M \tau L \left\| \Delta z^{(i+1)} \right\|_2^2,$$

$$= \sum_{m=1}^{M} -\rho \left(\left\| \Delta r_m^{(i+1)} \right\|_2^2 + \left\| \Delta z^{(i+1)} \right\|_2^2 - \tau L \left\| \Delta z^{(i+1)} \right\|_2^2 \right)$$

$$- \sum_{m=1}^{M} \rho 2 \Re \left\{ \left(\Delta r_m^{(i+1)} \right)^{\mathrm{H}} \left(z^{(i+1)} - z^{(i)} \right) \right\}. \tag{30}$$

We will now bound the last component in (30). Utilizing the result in (20) by recalling that $z^{(i+1)}$ minimizes (4b), subsequently applying the result of three-point inequality Lemma 2, and replacing $z^\star$ with $z^{(i)}$, the following inequality holds at iteration $i + 1$ with $y_m^{(i+1)}$ at hand:

$$g \left(z^{(i+1)} \right) + \beta h \left(z^{(i+1)} \right) - \sum_{m=1}^{M} 2 \Re \left\{ \left(y_m^{(i+1)} \right)^{\mathrm{H}} z^{(i+1)} \right\}$$

$$\leq g \left(z^{(i)} \right) + \beta h \left(z^{(i)} \right) - \sum_{m=1}^{M} 2 \Re \left\{ \left(y_m^{(i+1)} \right)^{\mathrm{H}} z^{(i)} \right\}$$

$$+ \rho M \tau L \left\| \Delta z^{(i+1)} \right\|_2^2. \tag{31}$$

Similarly, recalling that $z^{(i)}$ minimizes (4b), subsequently applying the result of the three-point inequality in Lemma 2, and replacing $z^\star$ with $z^{(i)}$ at the i-th iteration with $y_m^{(i)}$ at hand, the following inequality holds:

$$g \left(z^{(i)} \right) + \beta h \left(z^{(i)} \right) - \sum_{m=1}^{M} 2 \Re \left\{ \left(y_m^{(i)} \right)^{\mathrm{H}} z^{(i)} \right\}$$

$$\leq g \left(z^{(i+1)} \right) + \beta h \left(z^{(i+1)} \right) - \sum_{m=1}^{M} 2 \Re \left\{ \left(y_m^{(i)} \right)^{\mathrm{H}} z^{(i+1)} \right\}$$

$$+ \rho M \tau L \left\| \Delta z^{(i)} \right\|_2^2. \tag{32}$$

Now, we add (31) and (32), and rearrange such that

$$- \sum_{m=1}^{M} 2 \Re \left\{ \left(\underbrace{y_m^{(i+1)} - y_m^{(i)}}_{= \rho \Delta r_m^{(i+1)} \text{ using (4c)}} \right)^{\mathrm{H}} \left(z^{(i+1)} - z^{(i)} \right) \right\},$$

$$\leq \rho M \tau L \left(\left\| \Delta z^{(i+1)} \right\|_2^2 + \left\| \Delta z^{(i)} \right\|_2^2 \right)$$

$$\Longleftrightarrow - \sum_{m=1}^{M} \rho 2 \Re \left\{ \left(\Delta r_m^{(i+1)} \right)^{\mathrm{H}} \left(z^{(i+1)} - z^{(i)} \right) \right\}$$

$$\leq \rho M \tau L \left(\left\| \Delta z^{(i+1)} \right\|_2^2 + \left\| \Delta z^{(i)} \right\|_2^2 \right). \tag{33}$$

We use (33) in (30), which finally yields (23). ∎

In the subsequent lemma, we show that the Lyapunov function is non-increasing for $\tau \geq 0$.

Lemma 5: Consider the Lyapunov function $V^{(i)}$ from Definition 4. Then, $V^{(i)}$ is non-increasing over the iterations when $\tau \geq 0$.

Proof: The proof follows when $\tau \geq 0$ is considered in the right hand side of (23) (with $\rho > 0$), i.e., $-\rho[\|\Delta r_m^{(i+1)}\|^2 + \|\Delta z^{(i+1)}\|^2 + \tau L \|\Delta z^{(i)}\|_2^2] \leq 0 \ \forall i$.

We are now prepared to present and prove Proposition 1, which establishes the convergence to zero of the residual error, the objective residual error, and the primal residual error.

Proposition 1: The TOP-ADMM iterative scheme in (4) ensures that the residual error, the objective residual error, and the primal residual error converge to zero, asymptotically:

$$\lim_{i \to +\infty} \Delta z^{(i+1)} = 0, \tag{34}$$

$$\lim_{i \to +\infty} \left(p^{(i+1)} - p^\star \right) = 0, \tag{35}$$

$$\lim_{i \to +\infty} \Delta r_m^{(i+1)} = 0, \quad \forall m = 1, \dots, M. \tag{36}$$

Proof: Using the result of Lemma 4 and Lemma 5, i.e., (23), and $V^{(i)} \geq 0$ for every iteration i, we conclude $V^0 \geq \sum_{i=1}^{\infty} (V^{(i)} - V^{(i+1)}) \geq \sum_{i=1}^{\infty} \sum_{m=1}^{M} \rho[\|\Delta r_m^{(i+1)}\|_2^2 + \|\Delta z^{(i+1)}\|_2^2 + \tau L \|\Delta z^{(i)}\|_2^2]$ where the left hand series is absolutely convergent as $V^0 < \infty$. This absolute convergence implies that $\lim_{i \to +\infty} \Delta z^{(i+1)} := \lim_{i \to +\infty} (z^{(i+1)} - z^{(i)}) = 0$ and for all $m = 1, \dots, M$, $\lim_{i \to +\infty} \Delta r_m^{(i+1)} := \lim_{i \to +\infty} (x_m^{(i+1)} - z^{(i+1)}) = 0$. Furthermore, Lemma 1 results in $\lim_{i \to +\infty} p^{(i+1)} - p^\star = 0$, since $\lim_{i \to +\infty} \Delta r_m^{(i+1)} = 0$ for all $m = 1, \dots, M$. Therefore, the proof is completed. ∎

Corollary 1: For every m $\lim_{i \to +\infty} \Delta x_m^{(i+1)} := \lim_{i \to +\infty} (x_m^{(i+1)} - x_m^{(i)}) = 0$.

Proof: This is a direct consequence of Proposition 1, specifically on the results on the asymptotic dual and primal residual error. ∎

In addition to Proposition 1 and Corollary 1, the following lemma is used for the convergence proof of Theorem 1 of TOP-ADMM.

Lemma 6: Given the results from Proposition 1 and Corollary 1, and applying Definition 1, we have

$$\lim_{i \to +\infty} \left(\nabla h \left(z^{(i+1)} \right) - \nabla h \left(z^{(i)} \right) \right) = 0; \tag{37a}$$

$$\lim_{i \to +\infty} \left(\nabla h \left(x_m^{(i+1)} \right) - \nabla h \left(x_m^{(i)} \right) \right) = 0, \ \forall m. \tag{37b}$$

To this end, we are now ready to establish the convergence of TOP-ADMM.

Global convergence proof of Theorem 1: According to the KKT optimality conditions—see, e.g., [43], the necessary and sufficient optimality conditions for the considered general problem (3) are the dual feasibility, i.e.,

$$\mathbf{0} \in \frac{\partial}{\partial (x_m^\star)^*} \mathcal{L}_0 \left(\{ x_m^\star \}_{m=1}^{M}, z^\star, \{ y_m^\star \}_{m=1}^{M} \right),$$

$$\Longleftrightarrow \mathbf{0} \in \partial f_m (x_m^\star) + y_m^\star \ \forall m = 1, \dots, M, \tag{38}$$

$$\mathbf{0} \in \frac{\partial}{\partial (z^\star)^*} \mathcal{L}_0 \left(\{ x_m^\star \}_{m=1}^{M}, z^\star, \{ y_m^\star \}_{m=1}^{M} \right),$$

$$\Longleftrightarrow \mathbf{0} \in \partial g (z^\star) + \beta \nabla h (z^\star) - \sum_{m=1}^{M} y_m^\star, \tag{39}$$

and the primal feasibility, i.e.,

$$x_m^\star - z^\star = \mathbf{0} \quad \forall m = 1, \dots, M. \tag{40}$$

Our goal is to show that (39)–(40) are satisfied. To do so, we now analyze the iterative TOP-ADMM updates (4). In the first

step (4a), each $x_m^{(i+1)}$ minimizes the update over x_m, i.e.,

$$0 \in \partial f_m\left(x_m^{(i+1)}\right) + \rho\left(x_m^{(i+1)} - z^{(i+1)} + \frac{y_m^{(i)}}{\rho}\right).$$

When $i \to \infty$, we apply the results from Proposition 1 such that

$$0 \in \partial f_m\left(x_m^{(i+1)}\right) + y_m^{(i+1)},$$

which clearly satisfies the stationarity condition (38).

Similarly, in the second step (4b), $z^{(i+1)}$ minimizes that update over z, i.e.,

$$0 \in \partial g\left(z^{(i+1)}\right)$$
$$- \sum_{m=1}^{M} \rho\left(x_m^{(i)} - z^{(i+1)} - \tau\nabla h\left(z^{(i)}\right) + \frac{y_m^{(i)}}{\rho}\right), \quad (41)$$

which can be rewritten as

$$0 \in \partial g\left(z^{(i+1)}\right)$$
$$+ \rho M\tau\left(\nabla h\left(z^{(i)}\right) + \nabla h\left(z^{(i+1)}\right) - \nabla h\left(z^{(i+1)}\right)\right)$$
$$- \sum_{m=1}^{M}\left(\rho\left(x_m^{(i+1)} - z^{(i+1)}\right) + y_m^{(i)}\right) + \sum_{m=1}^{M} \rho\left(x_m^{(i+1)} - x_m^{(i)}\right).$$

Plugging in the dual variable update (4c) into the above expression and then rearranging terms yield

$$0 \in \partial g\left(z^{(i+1)}\right)$$
$$+ \rho M\tau\left[\nabla h\left(z^{(i+1)}\right) - \left(\nabla h\left(z^{(i+1)}\right) - \nabla h\left(z^{(i)}\right)\right)\right]$$
$$- \sum_{m=1}^{M} y_m^{(i+1)} + \sum_{m=1}^{M} \rho\Delta x_m^{(i+1)}.$$

Considering that TOP-ADMM generates sequences as $i \to \infty$, together with the results from Corollary 1 and Lemma 6, we have that $0 \in \partial g(z^{(i+1)}) + \beta\nabla h(z^{(i+1)}) - \sum_{m=1}^{M} y_m^{(i+1)}$ satisfies the stationarity condition (39).

Finally, primal feasibility (40) is directly satisfied by Proposition 1 when $i \to \infty$. ∎

Therefore, the above result ensures the convergence of TOP-ADMM to a KKT stationary point for a suitable choice of step-size τ. Clearly, given the convexity assumptions, this implies that such a stationary point is also the optimal solution to problem (3).

APPENDIX C
PROOF OF THEOREM 2

We present the basic convergence result of proposed FedTOP-ADMM algorithm by extending the vanishing residual error property results of Theorem 1.

Recall FedTOP-ADMM algorithm solves exactly the same problem as TOP-ADMM—see Table I. Therefore, the augmented Lagrangian is same as (12), i.e.,

$$\mathcal{L}_{(\{\rho_m\})}\left(\{w_m\}_{m=1}^{M}, w, \{\lambda_m\}_{m=1}^{M}\right)$$
$$:= \sum_{m=1}^{M} f_m\left(w_m\right) + g\left(w\right) + \beta h\left(w\right)$$

$$+ \sum_{m=1}^{M}\left\{\lambda_m^{\mathrm{T}}\left(w_m - w\right)\right\} + \sum_{m=1}^{M} \frac{\rho_m}{2}\|w_m - w\|_2^2.$$

Similarly, the necessary and sufficient optimality conditions for (5) consist of 1) dual feasibility, i.e.,

$$0 \in \partial g\left(w^\star\right) + \beta\nabla h\left(w^\star\right) - \sum_{m=1}^{M} \lambda_m^\star, \quad (42)$$

$$0 \in \partial f_m\left(w_m^\star\right) + \lambda_m^\star, \forall m = 1, \ldots, M, \quad (43)$$

and 2) the primal feasibility, i.e.,

$$w_m^\star - w^\star = 0 \quad \forall m = 1, \ldots, M. \quad (44)$$

To show the convergence of the proposed FedTOP-ADMM, we show that the server and client processing satisfy the abovementioned optimality conditions asymptotically, i.e., when $i \to \infty$.

During the processing at the server side of Algorithm 1, $w^{(i+1)}$ essentially minimizes the w-update corresponding to (6a), i.e.,

$$0 \in \partial g\left(w^{(i+1)}\right)$$
$$- \sum_{m=1}^{M} \rho_m\left(w_m^{(i)} - w^{(i+1)} - \tau^{(i)}\nabla h\left(w^{(i)}\right) + \frac{\lambda_m^{(i)}}{\rho_m}\right)$$
$$+ \zeta^{(i)}\left(w^{(i+1)} - w^{(i)}\right), \quad (45)$$

$$= \partial g\left(w^{(i+1)}\right)$$
$$+ \sum_{m=1}^{M} \rho_m\tau^{(i)}\left[\nabla h\left(w^{(i)}\right) + \nabla h\left(w^{(i+1)}\right) - \nabla h\left(w^{(i+1)}\right)\right]$$
$$- \sum_{m=1}^{M} \rho_m\left[w_m^{(i)} - w^{(i+1)} + \frac{\lambda_m^{(i)}}{\rho_m} + w_m^{(i+1)} - w_m^{(i+1)}\right]$$
$$+ \zeta^{(i)}\left(w^{(i+1)} - w^{(i)}\right), \quad (46)$$

$$\overset{(a)}{=} \partial g\left(w^{(i+1)}\right)$$
$$+ \sum_{m=1}^{M} \rho_m\tau^{(i)}\nabla h\left(w^{(i)}\right) - \sum_{m=1}^{M} \lambda_m^{(i+1)}$$
$$- \sum_{m=1}^{M} \rho_m\tau^{(i)}\left[\nabla h\left(w^{(i+1)}\right) - \nabla h\left(w^{(i)}\right)\right]$$
$$+ \sum_{m=1}^{M} \rho_m\left(w_m^{(i+1)} - w_m^{(i)}\right) + \zeta^{(i)}\left(w^{(i+1)} - w^{(i)}\right), \quad (47)$$

where in (a) we have plugged in the dual variable update (6c) assuming $\gamma = 1$.

Now, extending the vanishing residual errors property of TOP-ADMM to FedTOP-ADMM, we have dual residual $(w^{(i+1)} - w^{(i)}) \to 0$ and primal residual $(w_m^{(i+1)} - w^{(i)}) \to 0$, which implies $(w_m^{(i+1)} - w_m^{(i)}) \to 0$ when $i \to \infty$. Because we assume that the gradient of h is L-Lipschitz and $(w^{(i+1)} - w^{(i)}) \to 0$ for sufficiently large iterations, then the residual error $(\nabla h(w^{(i+1)}) - \nabla h(w^{(i)})) \to 0$ by using Definition 1—such that (47) satisfies the stationarity condition (42).

The processing at the user side corresponding to (6b), each $\boldsymbol{w}_m^{(i+1)}$ minimizes the $\boldsymbol{w}_m$-update, i.e.,

$$\boldsymbol{0} \in \partial f_m\left(\boldsymbol{w}_m^{(i+1)}\right) + \rho_m\left(\boldsymbol{w}_m^{(i+1)} - \boldsymbol{w}^{(i+1)} + \frac{\boldsymbol{\lambda}_m^{(i)}}{\rho_m}\right). \quad (48)$$

Now, using dual variable update (6c), (48) is

$$\boldsymbol{0} \in \partial \mathcal{F}_m(\boldsymbol{y}_m^{(i+1)}) + \boldsymbol{z}_m^{(i+1)}, \quad (49)$$

which always satisfies the stationarity condition (43) for sufficiently large iteration number $i \to \infty$.

Finally, primal feasibility (44) is satisfied by extending the primal residual error result in Theorem 1 to `FedTOP-ADMM`, i.e., $\lim_{i \to +\infty}(\boldsymbol{w}_m^{(i+1)} - \boldsymbol{w}^{(i+1)}) = 0$.

REFERENCES

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. Neural Inf. Process. Syst. Private Multi-Party Mach. Learn. Workshop*, 2016.

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

[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.

[4] S. Malekmohammadi, K. Shaloudegi, Z. Hu, and Y. Yu, "Splitting algorithms for federated learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, M. K., Ed., Cham, Switzerland: Springer, 2021, pp. 159–176.

[5] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* (CMS Books in Mathematics Series). New York, NY, USA: Springer, 2011.

[6] S. Kant, M. Bengtsson, B. Göransson, G. Fodor, and C. Fischione, "Efficient optimization for large-scale MIMO-OFDM spectral precoding," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5496–5513, Sep. 2021.

[7] H. Hellström et al., "Wireless for machine learning: A survey," *Found. Trends Signal Process.*, vol. 15, no. 4, pp. 290–399, 2022.

[8] S. Zhou and G. Y. Li, "Federated learning via inexact ADMM," Apr. 2022, *arXiv:2204.10607*.

[9] M. I. Jordan, J. D. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *J. Amer. Stat. Assoc.*, vol. 114, no. 526, pp. 668–681, 2019, doi: 10.1080/01621459.2018.1429274.

[10] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.

[11] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 5050–5060, 2018.

[12] 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.

[13] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15453–15462.

[14] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4447–4458.

[15] S. Di, D. Tao, X. Liang, and F. Cappello, "Efficient lossy compression for scientific data based on pointwise relative error bound," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 2, pp. 331–345, Feb. 2019.

[16] J. Wangni et al., "Gradient sparsification for communication-efficient distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1299–1309.

[17] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2021.

[18] J. Sun et al., "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3370–3380.

[19] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 5693–5700.

[20] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York, NY, USA: Springer, 2011, pp. 185–212.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[22] N. Parikh, S. Boyd, N. Parikh, and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2013.

[23] N. Komodakis and J. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 31–54, Nov. 2015.

[24] R. Glowinski, S. J. Osher, and W. Yin, *Splitting Methods in Communication, Imaging, Science, and Engineering* (Scientific Computation Series). Cham, Switzerland: Springer, 2016.

[25] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017.

[26] D. Davis and W. Yin, "A three-operator splitting scheme and its optimization applications," *Set-Valued Variational Anal.*, vol. 25, no. 4, pp. 829–858, Dec. 2017.

[27] M. Yan, "A new primal-dual algorithm for minimizing the sum of three functions with a linear operator," *J. Sci. Comput.*, vol. 76, pp. 1698–1717, 2018.

[28] E. K. Ryu and W. Yin, *Large-Scale Convex Optimization Via Monotone Operators*. Cambridge, U.K.: Cambridge Univ. Press, 2022.

[29] R. Pathak and M. J. Wainwright, "FedSplit: An algorithmic framework for fast federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 7057–7066.

[30] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen, "FedDR–randomized Douglas-Rachford splitting algorithms for nonconvex federated composite optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 4447–4458.

[31] S. Zhou and G. Y. Li, "Communication-efficient ADMM-based federated learning," Jan. 2022, *arXiv:2110.15318*.

[32] H. Wang and A. Banerjee, "Bregman alternating direction method of multipliers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2816–2824.

[33] S. Banert, R. I. Boţ, and E. R. Csetnek, "Fixing and extending some recent results on the ADMM algorithm," *Numer. Algorithms*, vol. 86, pp. 1303–1325, 2021.

[34] P. L. Combettes and J.-C. Pesquet, "Fixed point strategies in data science," *IEEE Trans. Signal Process.*, vol. 69, pp. 3878–3905, 2021.

[35] L. Condat, D. Kitahara, A. Contreras, and A. Hirabayashi, "Proximal splitting algorithms: A tour of recent advances, with new twists!," Dec. 2021, *arXiv:1912.00137*.

[36] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Math. Prog.*, vol. 162, no. 1/2, pp. 165–199, 2017.

[37] S. Kant, M. Bengtsson, G. Fodor, B. Göransson, and C. Fischione, "EVM mitigation with PAPR and ACLR constraints in large-scale MIMO-OFDM using TOP-ADMM," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9460–9481, Nov. 2022.

[38] G. Pierra, "Decomposition through formalization in a product space," *Math. Program.*, vol. 28, no. 1, pp. 96–115, Jan. 1984.

[39] G. Stathopoulos, H. Shukla, A. Szücs, Y. Pu, and C. N. Jones, "Operator splitting methods in control," *Found. Trends Syst. Control*, vol. 3, no. 3, pp. 249–362, 2016.

[40] S. Kant, C. Fischione, M. Bengtsson, G. Fodor, and B. Göransson, "An introduction to three-operator ADMM for wireless communications and machine learning," in preparation, 2022.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, Ontario, Tech. Rep., 2009.

[43] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[44] A. Hjørungnes, *Complex-Valued Matrix Derivatives: With Applications in Signal Processing and Communications*. Cambridge, U.K: Cambridge Univ. Press, 2011.

Shashi Kant (Graduate Student Member, IEEE) received the M.Sc.E.E. degree from Aalborg University, Aalborg, Denmark, in 2007. He is currently working toward the Industrial Ph.D. degree with the KTH Royal Institute of Technology, Stockholm, Sweden. He is currently a Senior Data Scientist with Ericsson AB, Stockholm. Since September 2007, he has been with many organizations in Denmark and Sweden. He holds more than a dozen patents (issued and pending). His research interests include federated/edge learning, (large-scale) distributed optimization, machine learning, signal processing, and their applications to wireless communications.

José Mairton B. da Silva Jr. (Member, IEEE) received the Ph.D. degree from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2019. He is currently a Marie Skłodowska-Curie Postdoctoral Fellow with Princeton University, Princeton, NJ, USA, and the KTH Royal Institute of Technology, Sweden. From April 2023, he will be an Assistant Professor with the Division of Computer Systems, Uppsala University, Uppsala, Sweden. His research interests include distributed machine learning and optimization over wireless communications. He was the Secretary of the IEEE Communications Society Emerging Technology Initiative on Full Duplex Communications between 2018 and 2021. He has been involved in the organization of many IEEE conferences and workshops, including IEEE SECON 2022 and IEEE GLOBECOM 22 Workshop on Wireless Communications for Distributed Intelligence. He gave several tutorials at many IEEE flagship conferences, including ICASSP, PIMRC, ICC, and GLOBECOM.

Gabor Fodor (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 1998, and the D.Sc. degree (Doctor of MTA) from the Hungarian Academy of Sciences, Budapest, Hungary, in 2019. He is currently a Master Researcher with Ericsson Research and a Docent and an Adjunct Professor with the KTH Royal Institute of Technology, Stockholm, Sweden. He is currently the Editor of IEEE Transactions on Wireless Communications and IEEE Wireless Communications.

Bo Göransson (Member, IEEE) received the M.Sc. degree in applied physics and electrical engineering from Linköping University, Sweden, in 1991, and the Ph.D. degree in array signal processing from the KTH Royal Institute of Technology, Stockholm, Sweden, in 1997. In 1998, he joined Ericsson, Stockholm, where he has been working with research and standardization of 3G, 4G, and 5G physical layer with special interest for MIMO and beamforming technologies. He is currently a Senior Expert in multi antenna systems and architectures with Ericsson, and an Adjunct Professor with the KTH Royal Institute of Technology. He holds more than 150 patents (issued and pending). He was the recipient of the Ericsson Inventor of the Year Award in 2012.

Mats Bengtsson (Senior Member, IEEE) received the M.S. degree in computer science from Linköping University, Linköping, Sweden, in 1991, and the Tech.Lic. and Ph.D. degrees in electrical engineering from the KTH Royal Institute of Technology, Stockholm, Sweden, in 1997 and 2000, respectively. From 1991 to 1995, he was with Ericsson Telecom AB Karlstad. He is currently a Professor in signal processing with the Information Science and Engineering Department, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. His research interests include statistical signal processing and optimization theory and its applications to communications, multiantenna processing, cooperative communication, radio resource management, and propagation channel modeling. He was an Associate Editor for IEEE Transactions on Signal Processing, as a Member of the IEEE SPCOM Technical Committee from 2007 to 2012, and as the Technical Chair for the IEEE SPAWC 2015.

Carlo Fischione (Senior Member, IEEE) received the Laurea degree (*summa cum laude*) in electronic engineering and the Ph.D. degree in electrical and information engineering from the University of L'Aquila, L'Aquila, Italy, in April 2001 and May 2005, respectively. He is currently a Full Professor with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. He is also the Director of the Data Science KTH Micro degree Program. He is also the Co-Founder and Scientific Director of ELK. He has held research positions in 2015 as a Visiting Professor, with the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2015, an Associate with Harvard University, Cambridge, and from 2004 to 2005 and 2007 to 2008, a Visiting Scholar and Research Associate with the University of California at Berkeley, CA, USA. His research interests include optimization with applications to networks, wireless and sensor networks, the Internet of Things, and digital systems. He has coauthored more than 180 publications, including a book, book chapters, international journals, and conferences, and international patents. He was the recipient or corecipient of a number of best paper awards. He is the Editor of IEEE Transactions on Communications and IEEE Journal on Selected Areas on Communications.