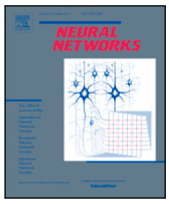
Neural Networks 181 (2025) 106772

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

FedADMM-InSa: An inexact and self-adaptive ADMM for federated learning

Yongcun Song [a], Ziqi Wang [b,*], Enrique Zuazua [b,c,d]

[a] Department of Mathematics, City University of Hong Kong, Kowloon, Hong Kong, China
[b] Chair for Dynamics, Control, Machine Learning and Numerics – Alexander von Humboldt Professorship, Department of Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstrasse 11, Erlangen, 91058, Germany
[c] Departamento de Matemáticas, Universidad Autónoma de Madrid, C. Francisco Tomás y Valiente, 7, Madrid, 28049, Spain
[d] Chair of Computational Mathematics, Fundación Deusto, Avenida de las Universidades, 24, Bilbao, 48007, Basque Country, Spain

ARTICLE INFO

Dataset link: https://github.com/DCN-FAU-AvH/FedADMM-InSa

Keywords:
Federated learning
ADMM
Inexactness criterion
Client heterogeneity

ABSTRACT

Federated learning (FL) is a promising framework for learning from distributed data while maintaining privacy. The development of efficient FL algorithms encounters various challenges, including heterogeneous data and systems, limited communication capacities, and constrained local computational resources. Recently developed FedADMM methods show great resilience to both data and system heterogeneity. However, they still suffer from performance deterioration if the hyperparameters are not carefully tuned. To address this issue, we propose an inexact and self-adaptive FedADMM algorithm, termed FedADMM-InSa. First, we design an inexactness criterion for the clients' local updates to eliminate the need for empirically setting the local training accuracy. This inexactness criterion can be assessed by each client independently based on its unique condition, thereby reducing the local computational cost and mitigating the undesirable straggle effect. The convergence of the resulting inexact ADMM is proved under the assumption of strongly convex loss functions. Additionally, we present a self-adaptive scheme that dynamically adjusts each client's penalty parameter, enhancing algorithm robustness by mitigating the need for empirical penalty parameter choices for each client. Extensive numerical experiments on both synthetic and real-world datasets have been conducted. As validated by some tests, our FedADMM-InSa algorithm improves model accuracy by 7.8% while reducing clients' local workloads by 55.7% compared to benchmark algorithms.

1. Introduction

With the increasing volume of data generated by massively distributed devices and organizations, traditional centralized deep learning paradigms encounter challenges in terms of data collection, privacy concerns, and scalability. To address the above challenges, federated learning (FL) (Li et al., 2020a, 2021; McMahan et al., 2017) has emerged as a promising paradigm and has gained significant attention in recent years.

1.1. Background

In FL, multiple clients (devices) in a distributed environment collaboratively train a neural network model under the coordination of a central server, without centralizing their data. The training process of FL consists of several communication rounds. For example, in the most commonly used FedAvg (McMahan et al., 2017) algorithm, at every communication round, each client computes a model update on its local data using the latest copy of the global model parameters and then sends the model update to the central server. The server aggregates these updates by averaging to update the global model parameters, which are then sent to all clients. Despite the fact that no raw data is transmitted, there is still a risk of privacy breaches, such as data reconstruction attacks (Geng et al., 2023; Wang et al., 2023a; Xiao et al., 2024; Yang et al., 2023).

One of the key properties of FL is that the clients are massively distributed and have limited communication capacity (Kairouz et al., 2021; McMahan et al., 2017; Zhang et al., 2022). To decrease the number of communication rounds needed to train a model, FedAvg suggests increasing local computation on each client. Unlike traditional distributed optimization (Zinkevich et al., 2010) where consensus is performed after every local gradient computation, in FedAvg, the clients perform multiple epochs using full-batch or stochastic gradient descents before these local model parameters are aggregated in order to update the global model parameters.

* Corresponding author.
E-mail addresses: ysong307@gmail.com (Y. Song), ziqi.wang@fau.de (Z. Wang), enrique.zuazua@fau.de (E. Zuazua).

https://doi.org/10.1016/j.neunet.2024.106772
Received 23 February 2024; Received in revised form 31 July 2024; Accepted 28 September 2024
Available online 1 October 2024

1.2. Current challenges

Nevertheless, simply increasing local training epochs as in FedAvg would cause the client drift (Karimireddy et al., 2020) issue due to the inherent data and system heterogeneity in FL (Kairouz et al., 2021; McMahan et al., 2017). The client drift issue comes from the fact that clients in FL normally possess unbalanced and non-independent and identically distributed (non-IID) local datasets, which leads to inconsistent minima of the local and global objective functions (Karimireddy et al., 2020; Wang et al., 2020). As a result, the aggregated global model suffers from deviation and does not converge to its true optimum (Li et al., 2019). To address this, FedProx (Li et al., 2020) was proposed by adding an extra proximal term to the client's loss function to constrain the client's local model to be close to the server's global model. However, the extra proximal term might degrade the training performance unless carefully tuned (Wang et al., 2022). In Karimireddy et al. (2020), a method called Scaffold is proposed to reduce client drift by using variance reduction to correct each client's local updates. However, this method doubles the size of variables that need to be communicated between the server and clients, which is not ideal given the limited communication resources in FL.

Recent findings show that FedADMM methods (Acar et al., 2021; Gong et al., 2022; Wang et al., 2023; Zhang et al., 2021; Zhou & Li, 2023) based on the alternating direction method of multipliers (ADMM) (Boyd et al., 2011; Glowinski & Marroco, 1975) are inherently resilient to heterogeneous clients in FL. FedADMM leverages dual variables to tackle statistical heterogeneity and accommodates system heterogeneity by tolerating variable amounts of work performed by clients. FedADMM maintains identical communication costs per round as FedAvg/Prox and generalizes them using the augmented Lagrangian with dual variables and an extra quadratic penalty term as the client's local loss function. This modification strikes a balance between updating the client's local model and staying consistent with the global model.

1.3. Research motivations

Despite the above-mentioned nice features, current FedADMM methods still suffer from performance deterioration and the straggler effect (Kairouz et al., 2021; Tan et al., 2023) if the hyperparameters are not carefully tuned, especially the amount of local training workload and the choice of the penalty parameter. In particular, FedADMM methods require clients to perform a certain amount of local training workload, which corresponds to solving the ADMM subproblem inexactly. This is implemented empirically by either commanding clients to solve the subproblem to a constant given accuracy (Gong et al., 2022; Zhang et al., 2021; Zhou & Li, 2023) or performing a fixed number of local epochs (Acar et al., 2021; Wang et al., 2023). However, the accuracy of this solution significantly impacts the effectiveness of the algorithm (Glowinski et al., 2022). Meanwhile, empirically assigning the same amount of local training workload overlooks the heterogeneity in clients' data and systems (e.g., non-IID datasets and varying computational resources). This oversight may also cause a severe straggler effect, as waiting for the resource-constrained clients to finish the overload work will slow down the training process, and simply dropping them will lead to a biased global model (Bonawitz et al., 2019; Zhou & Li, 2023).

Another critical issue involves the selection of the penalty parameter of the quadratic term in the augmented Lagrangian functions. It has been shown in ADMM applications (He et al., 2000; Song et al., 2016; Xu et al., 2017) that the efficiency of ADMM heavily depends on the penalty parameter. If the penalty parameter is chosen too small or too large, the solution time can increase significantly. The problem is more complicated in FedADMM as each client can use a different penalty parameter, and an inappropriate choice of the penalty parameter can potentially deteriorate the performance of FedADMM methods.

1.4. Main contributions

To address the aforementioned issues, we propose an inexact and self-adaptive FedADMM algorithm, referred to as FedADMM-InSa. Firstly, we introduce an easy-to-implement and adaptive inexactness criterion to guide the client's local training. Our approach eliminates the need to manually set local epochs or predefine a constant accuracy. This provides each client with the flexibility to solve its subproblem inexactly based on its unique situation, thereby eliminating the potential straggler effect. Furthermore, we design a scheme to dynamically adjust each client's penalty parameter based on the discrepancy between its local model parameters and the global model parameters, avoiding unexpected performance deterioration due to improperly chosen fixed penalty parameters.

Overall, our main contributions are as follows:

1. We propose an inexactness criterion that enables each client to dynamically adjust the precision of local training in each communication round. There is no need to empirically set the number of local training epochs or perceived constant accuracy *a priori*. This flexibility allows our algorithm to better adapt to the heterogeneous clients and datasets, save local computational resources, and mitigate the straggler effect.
2. We develop a self-adaptive scheme for adjusting each client's penalty parameter. The scheme dynamically balances the primal and dual residuals defined by the dissimilarity between the client's local parameters and the server's global parameters between two communication rounds. This adaptive scheme significantly enhances the robustness of our algorithm and eliminates the risk associated with selecting inappropriate penalty parameters for individual clients.
3. The convergence of our proposed algorithm using the inexactness criterion is analyzed. Extensive numerical experiments demonstrate the improved performance of our proposed inexactness criterion and self-adaptive penalty adjusting scheme. As validated by some numerical tests, our proposed algorithm reduces the clients' local computational load by 55.7% while accelerating the learning process when compared to the vanilla FedADMM.

1.5. Organization

The rest of the paper is organized as follows. In Section 2, we first provide a brief background on FL and ADMM, then the vanilla application of ADMM in FL and the resulting FedADMM algorithm. Section 3 presents our proposed FedADMM-InSa method, including the inexactness criterion and the self-adaptive penalty parameter scheme. The experimental setup and simulation results are presented in Section 4. Finally, we conclude the paper in Section 5 and provide more technical details in Appendix.

2. Preliminaries

In this section, we briefly introduce the mathematical formulation of FL and ADMM, which are central subjects of the study in this paper.

2.1. Federated learning

Mathematically, the training process of horizontal FL can be formulated as the following minimization problem:

$$\min_{z\in\mathbb{R}^n} \sum_{i=1}^{m} \alpha_i f_i(z), \tag{2.1}$$

where m is the number of clients, $z \in \mathbb{R}^n$ is the trainable parameters, $f_i : \mathbb{R}^n \to \mathbb{R}$ is the local loss function of each client $i \in [m] := \{1, \dots, m\}$, $\alpha_i > 0$ is the weight coefficient assigned to client i by the server, and $\sum_{i=1}^{m} \alpha_i = 1$. A common choice is $\alpha_i = N_i/N$, where N_i is client i's data volume and $N = \sum_{i=1}^{m} N_i$, i.e., weighting clients proportionally to their data volumes.

2.2. ADMM

Given the following constrained optimization problem:

$$\min_{x\in\mathbb{R}^{n_1},y\in\mathbb{R}^{n_2}} \theta_1(x)+\theta_2(y),\ \text{s.t. } Ax+By=b, \tag{2.2}$$

where $\theta_1:\mathbb{R}^{n_1}\to\mathbb{R}, \theta_2:\mathbb{R}^{n_2}\to\mathbb{R}$, $A\in\mathbb{R}^{n_3\times n_1}$, $B\in\mathbb{R}^{n_3\times n_2}$, and $b\in\mathbb{R}^{n_3}$, its augmented Lagrangian function $L_\beta:\mathbb{R}^{n_1}\times\mathbb{R}^{n_2}\times\mathbb{R}^{n_3}\to\mathbb{R}$ is defined as

$$L_\beta(x,y,\lambda)=\theta_1(x)+\theta_2(y)-\lambda^T(Ax+By-b)+\frac{\beta}{2}\|Ax+By-b\|^2, \tag{2.3}$$

where $\lambda\in\mathbb{R}^{n_3}$ is the Lagrangian multiplier associated with the equality constraint in (2.2), and $\beta>0$ is a penalty parameter. Here and henceforth, we denote by $\|\cdot\|$ the Euclidean (or ℓ_2-) norm.

Then, starting with an initial point $\{y^0,\lambda^0\}$, for $k\geq 0$, the ADMM (Glowinski & Marroco, 1975) iteratively updates the variables $\{x,y,\lambda\}$ in the following way:

$$\begin{cases} x^{k+1}=\arg\min_{x\in\mathbb{R}^{n_1}} L_\beta\left(x,y^k,\lambda^k\right), & \text{(a)}\\ y^{k+1}=\arg\min_{y\in\mathbb{R}^{n_2}} L_\beta\left(x^{k+1},y,\lambda^k\right), & \text{(b)}\\ \lambda^{k+1}=\lambda^k-\beta\left(Ax^{k+1}+By^{k+1}-b\right). & \text{(c)}\end{cases} \tag{2.4}$$

The ADMM is a variant of the classic augmented Lagrangian method (ALM) (Hestenes, 1969; Powell, 1969), where the subproblem of each ALM iteration is decomposed into two parts and then solved in a Gauss–Seidel manner. The ADMM possesses a distinct advantage as the decomposed subproblems are typically much easier than those encountered in the ALM, allowing for the exploitation of inherent properties and structures within the model under investigation. Moreover, the ADMM usually exhibits satisfactory numerical performance without the need for specific initial iterates. These characteristics make the ADMM a benchmark algorithm across various domains, including image processing (He & Yuan, 2018), statistical learning (Goldstein et al., 2015; Yue et al., 2018), and optimal control problems (Glowinski et al., 2020, 2022; Song et al., 2023; Zhang et al., 2017), etc. For a more comprehensive understanding of the ADMM, one can refer to Boyd et al. (2011), Glowinski (2014).

2.3. Vanilla FedADMM

In this subsection, we present the application of ADMM in the context of FL, leading to the derivation of the vanilla FedADMM (Zhou & Li, 2023). Subsequently, we systematically analyze the challenges and limitations associated with each iterative step of the vanilla FedADMM.

To apply the ADMM to address the FL problem (2.1), we first introduce auxiliary variables $u_i=z$, $i\in[\mathrm{m}]$, and rewrite problem (2.1) into the following consensus setting:

$$\min_{u_i,z\in\mathbb{R}^n}\sum_{i=1}^m \alpha_i f_i(u_i),\ \text{s.t. } u_i=z,\forall i\in[m]. \tag{2.5}$$

It is clear that problem (2.5) is equivalent to (2.1) in the sense that their optimal solutions coincide. The formulation of problem (2.5) naturally fits into the FL setting, where $u_i\in\mathbb{R}^n$ can be interpreted as the local model parameters held by client $i\in[m]$, and $z\in\mathbb{R}^n$ as the global model parameters held by the server. Therefore, in the following discussions, we concentrate on solving the optimization problem (2.5) instead of (2.1), as they are equivalent.

For problem (2.5), we define its augmented Lagrangian function as

$$\begin{aligned} &L_{\alpha,\beta}(u,\lambda,z)=\sum_{i=1}^m \alpha_i L_{\beta_i}(u_i,\lambda_i,z),\ \text{with}\\ &L_{\beta_i}(u_i,\lambda_i,z)=f_i(u_i)-\lambda_i^\top\left(u_i-z\right)+\frac{\beta_i}{2}\left\|u_i-z\right\|^2. \end{aligned} \tag{2.6}$$

Here, $\lambda_i\in\mathbb{R}^n$ and $\beta_i>0$ are the Lagrangian multiplier and the penalty parameter for client $i\in[m]$, respectively. For simplicity, we also denote

$$\begin{aligned} &u=\left(u_1,\ldots,u_m\right)^\top\in\mathbb{R}^{mn},\ \lambda=\left(\lambda_1,\ldots,\lambda_m\right)^\top\in\mathbb{R}^{mn},\\ &\alpha=\left(\alpha_1,\ldots,\alpha_m\right)^\top\in\mathbb{R}^m,\ \beta=\left(\beta_1,\ldots,\beta_m\right)^\top\in\mathbb{R}^m. \end{aligned} \tag{2.7}$$

Then, given an initial point $\{z^0,\lambda^0\}$, for $k\geq 0$, we can have the vanilla FedADMM that iteratively updates the variables $\{u,\lambda,z\}$ in the following steps:

$$\begin{cases} u_i^{k+1}=\arg\min_{u_i\in\mathbb{R}^n} L_{\beta_i}\left(u_i,\lambda_i^k,z^k\right),\forall i\in[m], & \text{(a)}\\ \lambda_i^{k+1}=\lambda_i^k-\beta_i\left(u_i^{k+1}-z^k\right),\forall i\in[m], & \text{(b)}\\ z^{k+1}=\arg\min_{z\in\mathbb{R}^n} L_{\alpha,\beta}\left(u^{k+1},\lambda^{k+1},z\right). & \text{(c)}\end{cases} \tag{2.8}$$

As seen above, these iterative updates naturally suit the FL setting. The updates of u_i^{k+1} in (2.8a) and λ_i^{k+1} in (2.8b) represent client i's local update process, and can be performed in parallel by each client $i\in[m]$. Meanwhile, the update of z^{k+1} in (2.8c) is the server aggregation process after receiving u^{k+1}, λ^{k+1}, and β from the clients. We then take a closer look at the solutions of the subproblems (2.8a)–(2.8c) and analyze the associated difficulties and limitations.

2.3.1. Client's local update: Subproblems (2.8) and (2.8)

The u-subproblem (2.8a) and the λ-subproblem (2.8b) correspond to the client's local update process and can be performed in parallel. The λ-subproblem (2.8b) is simple and has an analytical form. However, the u-subproblem (2.8a) is challenging due to the use of high-dimensional and nonlinear neural networks (such as the ResNets (He et al., 2016)) in FL. It generally lacks a closed-form solution. As a result, the subproblem (2.8a) should be solved iteratively and inexactly, and the implementation of the FedADMM (2.8) must be embedded by an internal iterative process for the subproblem (2.8a)

In the vanilla FedADMM method, the clients normally use gradient-based methods for solving (2.8a) to a certain precision. For example, client i first sets $\hat{u}_i^k:=z^k$, then implements a prescribed number of local epochs consisting of multiple gradient descent steps:

$$\hat{u}_i^k:=\hat{u}_i^k-\eta_i\nabla_{u_i}L_{\beta_i}(\hat{u}_i^k,\lambda_i^k,z^k), \tag{2.9}$$

where $\eta_i>0$ is the learning rate (also known as the step size). Then, client i sets $u_i^{k+1}:=\hat{u}_i^k$ and uses it to update λ_i^{k+1} by (2.8b). While other optimization schemes like stochastic gradient descent and quasi-Newton methods such as L-BFGS (Liu & Nocedal, 1989) are plausible, the accuracy of this inexact solution significantly influences the algorithm's effectiveness (Glowinski et al., 2022). A notable mathematical problem arises concerning the determination of an appropriate inexactness criterion for solving the subproblem (2.8a) inexactly.

Current FedADMM methods often require an empirically perceived precision in advance. For instance, clients may be instructed to perform a fixed number of training epochs (Acar et al., 2021; Wang et al., 2023) or solve the subproblem up to a constant predefined accuracy (Gong et al., 2022; Zhang et al., 2021; Zhou & Li, 2023). However, empirically assigning the same amount of local training workload is not a good strategy for heterogeneous clients with non-IID data and varying computational resources. For instance, assigning too much workload may inefficiently utilize local computational resources and cause a severe straggler effect, especially among clients with limited computational resources. Moreover, there is no necessity to pursue excessively accurate solutions for (2.8a), particularly when the iterates are still far from the solution point. Additionally, current FedADMM methods employ a fixed penalty parameter β throughout the training process, without considering the heterogeneity of clients in FL. If the penalty parameter is inappropriately chosen at the beginning, the method's efficiency and performance would deteriorate without remedy.

To tackle the above-mentioned issues, we design a readily implementable and appropriately accurate inexactness criterion for solving

subproblem (2.8a) and a self-adaptive scheme to adjust each client's penalty parameter β_i in Section 3. As a result, an inexact and self-adaptive FedADMM is proposed, enabling clients to dynamically adjust local training precision each round, eliminating the need to preset accuracy. This flexibility enhances the adaptability of our algorithm to diverse clients and datasets, saves local computational resources, and mitigates the straggler effect. In addition, the self-adaptive penalty parameter scheme improves the robustness of our algorithm by eliminating the risk associated with inappropriate pre-selection of clients' penalty parameters.

2.3.2. Server's aggregation: Subproblem (2.8c)

The z-subproblem (2.8c) is addressed by the server to update the global parameter z^{k+1}. It follows from (2.6) that

$$L_{\alpha,\beta}\left(u^{k+1},\lambda^{k+1},z\right)=\sum_{i=1}^{m}\alpha_i\left(f_i(u_i^{k+1})-(\lambda_i^{k+1})^\top(u_i^{k+1}-z)+\frac{\beta_i}{2}\|u_i^{k+1}-z\|^2\right), \tag{2.10}$$

which implies that

$$z^{k+1}=\arg\min_{z\in\mathbb{R}^n}\sum_{i=1}^{m}\alpha_i\left((\lambda_i^{k+1})^\top z+\frac{\beta_i}{2}\|u_i^{k+1}-z\|^2\right). \tag{2.11}$$

Hence, z^{k+1} has an analytical form given by

$$z^{k+1}=\frac{1}{\sum_{i=1}^{m}\alpha_i\beta_i}\sum_{i=1}^{m}\alpha_i\left(\beta_i u_i^{k+1}-\lambda_i^{k+1}\right). \tag{2.12}$$

The server can also incorporate the partial client participation strategy by selecting a subset of clients $\mathcal{M}^k\subseteq[m]$ to calculate updated parameters u_i^{k+1} and λ_i^{k+1} using (2.8) and (2.8) at the $(k+1)$th communication round. Note from (2.12) that the clients can provide the server with calculated $(\beta_i u_i^{k+1}-\lambda_i^{k+1})$ instead of u_i^{k+1} and λ_i^{k+1}, thereby reducing the communication cost. Meanwhile, for the unselected clients $i\notin\mathcal{M}^k$, the server can simply use their previously communicated local model parameters by setting $u_i^{k+1}:=u_i^k$ and $\lambda_i^{k+1}:=\lambda_i^k$. Finally, after receiving updated parameters $(\beta_i u_i^{k+1}-\lambda_i^{k+1})$ and β_i from the clients, the server can get the updated global model parameters z^{k+1} by (2.12).

With the above analysis of the client update and the server aggregation processes, we summarize the vanilla FedADMM algorithm based on (2.8) in Algorithm 1.

Algorithm 1 Vanilla FedADMM.

Inputs: Initialize $z^0,\lambda_i^0,u_i^0,\beta_i,\alpha_i,\eta_i,i\in[m]$.
for each communication round $k=0,1,\dots,K-1$ **do**
  Server side: Select a subset $\mathcal{M}^k\subseteq[m]$ of clients and send them z^k.
  Client side:
  for each client $i\in\mathcal{M}^k$ in parallel **do**
    Set $\hat{u}_i^k:=z^k$.
    for each epoch $e=1,\dots,E$ **do**
      Update $\hat{u}_i^k:=\hat{u}_i^k-\eta_i\nabla_{u_i}L_{\beta_i}(\hat{u}_i^k,\lambda_i^k,z^k)$ for one epoch.
    end for
    Set $u_i^{k+1}:=\hat{u}_i^k$.
    Update $\lambda_i^{k+1}=\lambda_i^k-\beta_i(u_i^{k+1}-z^k)$.
    Send $(\beta_i u_i^{k+1}-\lambda_i^{k+1})$ and β_i to the server.
  end for
  for each client $i\notin\mathcal{M}^k$ in parallel **do**
    Set $u_i^{k+1}:=u_i^k$, $\lambda_i^{k+1}:=\lambda_i^k$.
  end for
  Server side:
  Update $z^{k+1}=\frac{1}{\sum_{i=1}^m\alpha_i\beta_i}\sum_{i=1}^m\alpha_i(\beta_i u_i^{k+1}-\lambda_i^{k+1})$.
end for

3. An inexact and self-adaptive FedADMM

In this section, we present the design of our inexact and self-adaptive FedADMM algorithm. We first present a refined server update approach with improved stability. Then, we design an inexactness criterion for solving (2.8a) and a self-adaptive penalty parameter scheme to update β. We further show that the inexactness criterion and the self-adaptive scheme can be combined with each other.

3.1. Server's aggregation with memory

From (2.12), it is evident that in the vanilla FedADMM, the update of the server's global model parameters z^{k+1} depends solely on the local parameters updated by clients in the $(k+1)$th communication round. Due to the uncertainty arising from heterogeneous local updates, the global model parameters may undergo significant fluctuations. To address this challenge, we draw inspiration from exponential moving average methods used in several fields (see e.g., Awheda and Schwartz (2016), Cai et al. (2021), Dinh et al. (2020)) and propose an improved method for updating global model parameters z^{k+1} as follows:

$$\hat{z}^{k+1}=\frac{1}{\sum_{i=1}^m\alpha_i\beta_i}\sum_{i=1}^{m}\alpha_i\left(\beta_i u_i^{k+1}-\lambda_i^{k+1}\right), \tag{3.1a}$$

$$z^{k+1}=\frac{1}{1+\delta}\hat{z}^{k+1}+\frac{\delta}{1+\delta}z^k, \tag{3.1b}$$

where $\delta>0$ controls the trade-off between stability and responsiveness in the server update.

The proposed server update strategy (3.1) improves the stability of aggregation by balancing the current and previous global model parameters. This ensures the robustness of the global model in the face of variable clients' local updates. On the one hand, it allows for the memorization of z^k, thereby minimizing the sensitivity to outliers and leading to a more stable global model. On the other hand, the strategy maintains adaptability to the changing $\hat{z}^{k+1}$, ensuring that the global model responds appropriately to evolving local updates.

Meanwhile, the proposed server update strategy (3.1) can be interpreted by augmenting the vanilla z-subproblem (2.8c) with an additional proximal term, given by

$$z^{k+1}=\arg\min_{z\in\mathbb{R}^n}\ L_{\alpha,\beta}\left(u^{k+1},\lambda^{k+1},z\right)+\frac{\delta}{2}\sum_{i=1}^{m}\alpha_i\beta_i\|z-z^k\|^2. \tag{3.2}$$

3.2. An inexact version of FedADMM

In this subsection, we propose an inexactness criterion for solving (2.8a) to address the challenges of presetting the client's local update precision.

Inexactness criterion of (2.8a). We first start with analyzing the optimality condition of the u-subproblem (2.8a). From (2.6), we have

$$L_{\beta_i}\left(u_i,\lambda_i^k,z^k\right)=f_i(u_i)-(\lambda_i^k)^\top(u_i-z^k)+\frac{\beta_i}{2}\|u_i-z^k\|^2, \tag{3.3}$$

and accordingly

$$\nabla_{u_i}L_{\beta_i}\left(u_i,\lambda_i^k,z^k\right)=\nabla f_i(u_i)-\lambda_i^k+\beta_i\left(u_i-z^k\right). \tag{3.4}$$

Let u_i^{k+1} be a solution of the subproblem (2.8a), then, it satisfies

$$\nabla f_i(u_i^{k+1})-\lambda_i^k+\beta_i\left(u_i^{k+1}-z^k\right)=0. \tag{3.5}$$

Based on the optimality condition (3.5), for each client $i\in[m]$, we define $e_i^k(u_i)$ as

$$e_i^k(u_i)=\nabla f_i(u_i)-\lambda_i^k+\beta_i\left(u_i-z^k\right). \tag{3.6}$$

It is clear that a solution u_i^{k+1} of the u-subproblem (2.8a) at the $(k+1)$th iteration satisfies $e_i^k(u_i^{k+1})=0$. Hence, we can use $e_i^k(u_i)$ as the residual for the u-subproblem. With the help of $e_i^k(u_i)$, we propose the following

inexactness criterion. For each client $i \in [m]$, at the $(k+1)$th iteration, it computes u_i^{k+1} such that

$$\left\|e_i^k(u_i^{k+1})\right\| \le \sigma_i \left\|e_i^k(u_i^k)\right\|, \tag{3.7}$$

where σ_i is a given constant satisfying

$$0 < \sigma_i < \frac{\sqrt{2c_i}}{\sqrt{2c_i} + \sqrt{\beta_i}} < 1, \tag{3.8}$$

and $c_i > 0$ is a parameter associated with the strong convexity constant of client i's loss function f_i, i.e., $(\nabla f_i(x) - \nabla f_i(y))^\top (x - y) \ge c_i \|x - y\|^2, \forall x, y \in \mathbb{R}^n$. Equivalently, the condition (3.8) can be written as

$$0 < \sigma_i < \frac{\sqrt{2}}{\sqrt{2} + \sqrt{\tilde{\beta}_i}} < 1, \tag{3.9}$$

where $\tilde{\beta}_i = \beta_i / c_i$. It is evident that a smaller $\tilde{\beta}_i$ leads to a higher permissible value of σ_i, indicating that the subproblem can be solved with less precision. In practical scenarios, each client can (locally) convexify its local loss function by incorporating a regularization term. Our inexactness criterion can also be applied to the non-convex problems by using a $\tilde{\beta}_i > 0$ in (3.9). Subsequent numerical experiments in Section 4, covering both strongly convex and non-convex problems, also underscore the feasibility of our inexactness criterion (3.7).

It is noteworthy that our inexactness criterion can be assessed by each client autonomously based on its present model and penalty parameter, and it can be seamlessly executed during iterations. There is no requirement to predefine any empirically perceived constant accuracy. This gives each client the flexibility to solve its subproblem inexactly, aligning with its distinct non-IID data and computational resources. Consequently, this approach effectively mitigates the potential risk of the straggler effect that may arise with resource-constrained clients. Overall, these attributes make the inexactness criterion (3.7) straightforward to implement and more likely to result in local computational savings.

FedADMM-In: An inexact fedadmm algorithm. Based on the discussions above, the iterative three steps of FedADMM with the inexactness criterion (3.7) are given by

$$\begin{cases} u_i^{k+1} \approx \arg\min\limits_{u_i \in \mathbb{R}^n} L_{\beta_i}\left(u_i, \lambda_i^k, z^k\right), \forall i \in [m], & \text{(a)} \\ \lambda_i^{k+1} = \lambda_i^k - \beta_i\left(u_i^{k+1} - z^k\right), \forall i \in [m], & \text{(b)} \\ z^{k+1} = \arg\min\limits_{z \in \mathbb{R}^n} L_{\alpha,\beta}\left(u^{k+1}, \lambda^{k+1}, z\right) + \frac{\delta}{2}\sum\limits_{i=1}^m \alpha_i\beta_i \|z - z^k\|^2. & \text{(c)} \end{cases} \tag{3.10}$$

We denote by FedADMM-In the FedADMM algorithm with the inexactness criterion (3.7) and summarize it in Algorithm 2.

Convergence analysis. Let us analyze the convergence of our proposed FedADMM-In algorithm under the following assumptions:

Assumption 3.1. For all $i \in [m]$, the gradient of f_i is s_i-Lipschitz, that is,

$$\left\|\nabla f_i(x) - \nabla f_i(y)\right\| \le s_i \|x - y\|, s_i > 0, \forall x, y \in \mathbb{R}^n. \tag{3.11}$$

Assumption 3.2. For all $i \in [m]$, f_i is c_i-strongly convex, that is,

$$(\nabla f_i(x) - \nabla f_i(y))^\top (x - y) \ge c_i \|x - y\|^2, c_i > 0, \forall x, y \in \mathbb{R}^n. \tag{3.12}$$

From the strong convexity assumption, it is easy to deduce that problem (2.1) has a unique solution, denoted by z^*. As a consequence, problem (2.5), which is equivalent to (2.1), also has a unique solution (u^*, z^*), where $u_i^* = z^*, \forall i \in [m]$. On the other hand, from the above assumptions, the dual problem of problem (2.5) has a unique solution λ^* and the strong duality holds. In the following Theorem 3.3, we show that the sequence generated by iterative scheme (3.10) (i.e., Algorithm 2 with full client participation) converges to (u^*, λ^*, z^*).

Algorithm 2 FedADMM-In: FedADMM with the inexactness criterion (3.7).

Inputs: Initialize $z^0, \lambda_i^0, u_i^0, \beta_i, \sigma_i, \alpha_i, \eta_i, i \in [m]$.
for each communication round $k = 0, 1, \ldots, K-1$ **do**
  Server side: Select a subset $\mathcal{M}^k \subseteq [m]$ of clients and send them z^k.
  Client side:
  for each client $i \in \mathcal{M}^k$ in parallel **do**
    Set $\hat{u}_i^k := z^k$.
    while $\|e_i^k(\hat{u}_i^k)\| > \sigma_i \|e_i^k(u_i^k)\|$ **do**
      Update $\hat{u}_i^k := \hat{u}_i^k - \eta_i \nabla_{u_i} L_{\beta_i}(\hat{u}_i^k, \lambda_i^k, z^k)$ for one epoch.
    end while
    Set $u_i^{k+1} := \hat{u}_i^k$.
    Update $\lambda_i^{k+1} = \lambda_i^k - \beta_i(u_i^{k+1} - z^k)$.
    Send $(\beta_i u_i^{k+1} - \lambda_i^{k+1})$ and β_i to the server.
  end for
  for each client $i \notin \mathcal{M}^k$ in parallel **do**
    Set $u_i^{k+1} := u_i^k$, $\lambda_i^{k+1} := \lambda_i^k$.
  end for
  Server side:
  Obtain $\hat{z}^{k+1} = \frac{1}{\sum_{i=1}^m \alpha_i\beta_i} \sum_{i=1}^m \alpha_i(\beta_i u_i^{k+1} - \lambda_i^{k+1})$.
  Update $z^{k+1} = \frac{1}{1+\delta}\hat{z}^{k+1} + \frac{\delta}{1+\delta} z^k$.
end for

Theorem 3.3. *Let $\{w^k\} = \{(u^k, \lambda^k, z^k)^\top\}$ be the sequence generated by iterative scheme (3.10) (i.e., Algorithm 2 with full client participation). Then, we have the following assertions:*

$$\|e_i^k(u_i^{k+1})\| \xrightarrow{k\to\infty} 0, \quad u_i^k \xrightarrow{k\to\infty} u_i^*, \quad \lambda_i^k \xrightarrow{k\to\infty} \lambda_i^*, \quad z^k \xrightarrow{k\to\infty} z^*, \quad \forall i \in [m]. \tag{3.13}$$

The detailed proof is presented in Appendix A.1, which is inspired by Glowinski et al. (2022, Sec. 3.3). Although the convergence proof is conducted under the strong convexity assumption, our algorithm with the proposed inexactness criterion can also be applied to the non-convex cases by using a $\tilde{\beta}_i > 0$ in (3.9). Moreover, the effectiveness of the algorithm with partial client participation in both strongly convex and non-convex cases is validated by the numerical experiments in Section 4.

Remark 3.4. In practical applications, one can also use z^k instead of u_i^k on the right-hand side of the inexactness criterion (3.7). We empirically demonstrate in Section 4 that such a substitution improves the FL training performance in terms of improved model accuracy and reduced computational load. This substitution alleviates the unnecessary pursuit of the client's local minima, which is normally inconsistent with the global minima due to the non-IID datasets. The convergence analysis of the algorithm employing this modified inexactness criterion and partial client participation serves as an interesting future study.

3.3. Self-adaptive penalty parameter β_i^k

In this subsection, we present a self-adaptive scheme for adjusting the penalty parameter β_i^k for client i at communication round k to further improve the performance of our FedADMM-In algorithm. This also makes the algorithm more robust to different initial choices of the penalty parameter.

Self-adaptive scheme. As shown in (3.10), the FedADMM-In algorithm is an iterative algorithm, and an intuitive stopping criterion of the iterative updates can be $u_i^{k+1} - u_i^k = 0$ and $\lambda_i^{k+1} - \lambda_i^k = 0$. It follows from (3.10b) that $\lambda_i^{k+1} - \lambda_i^k = 0$ implies that $u_i^{k+1} - z^k = 0$. Inspired by the stopping criterion, we define the primal residual p_i^k and the dual

residual d_i^k as follows:

$$p_i^k = \beta_i^k \|u_i^{k+1} - u_i^k\|, \ i \in [m], \tag{3.14a}$$

$$d_i^k = \|u_i^{k+1} - z^k\|, \ i \in [m]. \tag{3.14b}$$

To adaptively update β_i^k, we use the following scheme:

$$\beta_i^{k+1} = \begin{cases} \beta_i^k \tau, & \text{if } d_i^k > \mu p_i^k, \\ \beta_i^k / \tau, & \text{if } p_i^k > \mu d_i^k, \\ \beta_i^k, & \text{otherwise}, \end{cases} \tag{3.15}$$

where $\mu, \tau > 1$ are parameters to choose.

The essence of the self-adaptive penalty parameter scheme (3.15) lies in dynamically balancing the primal residual p_i^k and the dual residual d_i^k. For instance, according to the definition of d_i^k in (3.14b), $d_i^k > \mu p_i^k$ suggests that client i's updated local model parameters u_i^{k+1} deviate significantly from the current global model parameters z^k. Consequently, client i is prompted to increase its penalty parameter β_i^k to impose a more substantial penalty on the constraint. Conversely, a reduction in β_i^k is recommended for client i to mitigate the primal residual.

Finally, the iterative steps of FedADMM with an adaptive β_i^k are given by

$$\begin{cases} u_i^{k+1} \approx \arg\min_{u_i \in \mathbb{R}^n} L_{\beta_i^k}\left(u_i, \lambda_i^k, z^k\right), \forall i \in [m], & \text{(a)} \\ \lambda_i^{k+1} = \lambda_i^k - \beta_i^k \left(u_i^{k+1} - z^k\right), \forall i \in [m], & \text{(b)} \\ z^{k+1} = \arg\min_{z \in \mathbb{R}^n} L_{\alpha, \beta^k}\left(u^{k+1}, \lambda^{k+1}, z\right) + \frac{\delta}{2} \sum_{i=1}^m \alpha_i \beta_i^k \|z - z^k\|^2. & \text{(c)} \end{cases} \tag{3.16}$$

Compatibility with the inexactness criterion. We can easily extend our inexactness criterion (3.7) to make it compatible with the self-adaptive penalty parameter scheme (3.15). To accomplish this, we simply replace the fixed β_i with the adaptive β_i^k in (3.8). Then, for each client $i \in [m]$, at the $(k+1)$th communication round, it computes u_i^{k+1} such that

$$\left\| e_i^k(u_i^{k+1}) \right\| \le \sigma_i^k \left\| e_i^k(u_i^k) \right\|, \tag{3.17}$$

where σ_i^k is a given constant satisfying

$$0 < \sigma_i^k < \frac{\sqrt{2}}{\sqrt{2} + \sqrt{\tilde{\beta}_i^k}} < 1, \tag{3.18}$$

and $\tilde{\beta}_i^k = \beta_i^k / c_i$ can be independently evaluated by each client i based on its unique situation. Hence, the inexactness criterion (3.17) with the self-adaptive penalty parameter scheme can also be executed autonomously by each client during iterations. Moreover, this allows clients to perform personalized local update steps based on their distinct data and computational resources, effectively mitigating the risk of the straggler effect.

FedADMM-InSa: An inexact and self-adaptive fedadmm. Based on the discussions above, we propose the FedADMM-InSa algorithm using an adaptive penalty parameter β_i^k and present it in Algorithm 3. It brings more difficulty to prove the convergence of Algorithm 3 with a varying β_i^k in each iteration, however, the numerical experiments in Section 4 empirically validate its robustness and improved performance.

Algorithm 3 FedADMM-InSa: FedADMM with the inexactness criterion (3.17) and the self-adaptive penalty parameter scheme (3.15).

Inputs: Initialize $z^0, \lambda_i^0, u_i^0, \beta_i^0, \sigma_i^0, c_i, \alpha_i, \eta_i, i \in [m]$, $\mu > 1, \tau > 1$.
for each communication round $k = 0, 1, \ldots, K-1$ **do**
  Server side: Select a subset $\mathcal{M}^k \subseteq [m]$ of clients and send them z^k.
  Client side:
  for each client $i \in \mathcal{M}^k$ in parallel **do**
    Set $\hat{u}_i^k := z^k$.
    while $\|e_i^k(\hat{u}_i^k)\| > \sigma_i^k \|e_i^k(u_i^k)\|$ **do**
      Update $\hat{u}_i^k := \hat{u}_i^k - \eta_i \nabla_{u_i} L_{\beta_i}(\hat{u}_i^k, \lambda_i^k, z^k)$ for one epoch.
    end while
    Set $u_i^{k+1} := \hat{u}_i^k$.
    Update $\lambda_i^{k+1} = \lambda_i^k - \beta_i^k(u_i^{k+1} - z^k)$.
    Send $(\beta_i^k u_i^{k+1} - \lambda_i^{k+1})$ and β_i^k to the server.
    Calculate $p_i^k = \beta_i^k \|u_i^{k+1} - u_i^k\|$ and $d_i^k = \|u_i^{k+1} - z^k\|$.
    if $d_i^k > \mu p_i^k$ **then**
      Update $\beta_i^{k+1} = \beta_i^k \tau$.
    else if $p_i^k > \mu d_i^k$ **then**
      Update $\beta_i^{k+1} = \beta_i^k / \tau$.
    else
      Update $\beta_i^{k+1} = \beta_i^k$.
    end if
    Update $\sigma_i^{k+1} = \frac{\sqrt{2}}{\sqrt{2} + \sqrt{\beta_i^{k+1}/c_i}}$.
  end for
  for each client $i \notin \mathcal{M}^k$ in parallel **do**
    Set $u_i^{k+1} := u_i^k$, $\lambda_i^{k+1} := \lambda_i^k$, $\beta_i^{k+1} := \beta_i^k$, $\sigma_i^{k+1} := \sigma_i^k$.
  end for
  Server side:
  Obtain $\hat{z}^{k+1} = \frac{1}{\sum_{i=1}^m \alpha_i \beta_i^k} \sum_{i=1}^m \alpha_i (\beta_i^k u_i^{k+1} - \lambda_i^{k+1})$.
  Update $z^{k+1} = \frac{1}{1+\delta} \hat{z}^{k+1} + \frac{\delta}{1+\delta} z^k$.
end for

4. Experimental results

In this section, we conduct extensive numerical tests to demonstrate the improved performance of our proposed FedADMM-In (Algorithm 2) and FedADMM-InSa (Algorithm 3) in comparison to the vanilla FedADMM (Zhou & Li, 2023) and FedAvg (McMahan et al., 2017). We first introduce the experimental settings and the implementation details used in our experiments. Then, we present the simulation results and analysis.

4.1. Setups

We conducted experiments on three combinations of datasets and models, covering both cross-device (many clients with few data points per client) and cross-silo (few clients with many data points per client) scenarios (Kairouz et al., 2021). The details of the experiments are elaborated below and also summarized in Table 1.

Example 1: Linear regression with a synthetic dataset. In this example, we set each client's local loss function to be

$$f_i(u_i) = \frac{1}{2N_i} \sum_{j=1}^{N_i} \left(u_i^\top a_i^j - b_i^j\right)^2 + \frac{\gamma_i}{2} \|u_i\|^2, \quad i \in [m]. \tag{4.1}$$

Here, $a_i^j \in \mathbb{R}^{n_a}$ and $b_i^j \in \mathbb{R}$ are the jth data of client $i \in [m]$, N_i is the volume of data of client i, and we denote $N = \sum_{i=1}^m N_i$. Let $\lceil \cdot \rceil$ be the ceiling function, i.e., $\lceil x \rceil$ is the smallest integer not smaller than x, e.g., $\lceil 1.5 \rceil = 2$. Then, following the setup in Zhou and Li (2023), we generate $\lceil N/3 \rceil$ samples from the standard normal distribution, $\lceil N/3 \rceil$ samples from the Student's t distribution with degree 5, and $N - 2\lceil N/3 \rceil$ samples from the uniform distribution in $[-5, 5]$. In the tests of Example 1, we set $n_a = 5,000$, $N = 50{,}000$, and $\gamma_i = 0.01$ for all $i \in [m]$.

Example 2: Image classification with the MNIST dataset. In the second example, we address the image classification problem using convolutional neural networks (CNN) with the MNIST dataset (LeCun et al., 1998). The MNIST dataset contains images of handwritten digits. It has 60,000 training data and 10,000 test data. The samples in this dataset are 28×28 grayscale images with handwritten digits from 0 to 9 in the center. In this example, each client uses a CNN with two convolutional

Table 1
Datasets, models, and parameters of three examples.

	Example 1 Linear regression	Example 2 Image classification	Example 3 Image classification
Dataset	Synthetic	MNIST	CIFAR-10
Model	Linear	CNN	ResNet
Training set size	50 000	60 000	50 000
Test set size	–	10 000	10 000
Data dimension	5000	$28 \times 28 \times 1$	$32 \times 32 \times 3$
Number of clients	200	200	10
Data per client	250	300	5000
Active clients per round	20%	20%	20%
Client learning rate	0.001	0.01	0.001
Batch size	50	50	500
Epochs per round	20	20	10
Communication rounds	300	300	300

layers, comprising 32 and 64 channels respectively, as the same to that in McMahan et al. (2017).

Example 3: Image classification with the CIFAR-10 dataset. In the third example, we test the image classification of the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). This dataset contains color images of size 32×32 with 10 categories. There are 50,000 training data and 10,000 test data. For the neural networks, each client uses a ResNet-20 (He et al., 2016), which consists of 20 stacked weighted layers.

Non-IID data separation. To test the performance of the algorithms on non-IID data, we separate datasets in the following way. In Example 1, we first shuffle the training samples and then distribute them evenly among all clients. In Example 2 and 3, we first sorted the data by labels and then divided them into several shards, each shard containing images with the same label. Then, we assign the data to make sure that each client has at most two kinds of labels in Example 2 and five kinds of labels in Example 3, corresponding to pathological non-IID scenarios.

FL training parameters. In each round, the server uniformly samples 20% of the clients to perform the local training. For the vanilla FedADMM and FedAvg algorithms, the clients perform fixed epochs of training using stochastic gradient descent. For the FedADMM-In and FedADMM-InSa algorithms, the number of epochs is controlled by the inexactness criterion, which is evaluated after each epoch. We constrain the maximum number of epochs per round to be the same as the fixed epochs in the vanilla FedADMM. For the inexactness criterion, we set $c_i = 0.01$ in Example 1 and 2, and $c_i = 0.001$ in Example 3. In all examples, we replace u_i^k with z^k and set $\delta = 0.01$ for the server aggregation. For FedADMM-InSa, the adaptive penalty scheme uses $\mu = 5$ and $\tau = 2$ in all examples. Finally, important parameters used in the tests are summarized in Table 1.

4.2. Results analysis

In this subsection, we present the comparison results of our FedADMM-In and FedADMM-InSa algorithms with benchmark FedADMM and FedAvg algorithms. For ADMM-based algorithms, we show results with different values of $\beta_i \in \{0.1, 1, 2, 5, 10\}$. The detailed results are summarized in Table 2 and elaborated below.

4.2.1. Evaluation of FedADMM-In

In this subsection, we compare our FedADMM-In algorithm with the vanilla FedADMM and FedAvg algorithms. The results with the penalty parameter $\beta_i = 1$ are plotted in Fig. 1, while other results are summarized in Table 2.

As shown in Figs. 1(a)–1(c), FedADMM-In achieves superior results compared to FedAvg and comparable results to FedADMM across all three examples. Furthermore, Figs. 1(d)–1(f) demonstrate that FedADMM-In allows clients to execute fewer local epochs compared to the other two algorithms while maintaining training performance. As detailed in Table 2, FedADMM-In achieves an average local epoch reduction of 35.4%, 49.0%, and 16.6% in Examples 1–3, respectively, leading to a substantial improvement in computational savings. These results indicate that our proposed FedADMM-In algorithm can achieve comparable or superior training outcomes while reducing the computational load on clients, significantly mitigating the potential waste of valuable computational resources in FL.

4.2.2. Evaluation of FedADMM-InSa

In this subsection, we present comparison results of our FedADMM-InSa algorithm with the vanilla FedADMM and FedAvg algorithms. The results for the three examples are presented in Figs. 2–4 and are also summarized in Table 2.

Results of Example 1 (Linear regression). Fig. 2 shows the comparison results of our FedADMM-InSa with the vanilla FedADMM and FedAvg algorithms. We plot scenarios with $\beta_i \in \{0.1, 1\}$, and other results can be seen in Table 2. It can be seen from Fig. 2(a) that our FedADMM-InSa consistently achieves the lowest training loss. For FedADMM, it performs better than FedAvg when $\beta_i = 1$. However, for $\beta_i = 0.1$, FedADMM only decreases to a loss value similar to that of FedAvg, while our FedADMM-InSa reaches the lowest loss value. The average β_i of clients are plotted in Fig. 2(b). For FedADMM-InSa, despite different initial β_i values, they converge to a similar value around three. Additionally, Fig. 2(c) presents the number of average local epochs of active clients in each round. It is evident that FedADMM-InSa requires fewer local epochs than FedADMM and FedAvg. In the later training rounds, the local epochs saturate probably because the algorithm has approached the local minimum.

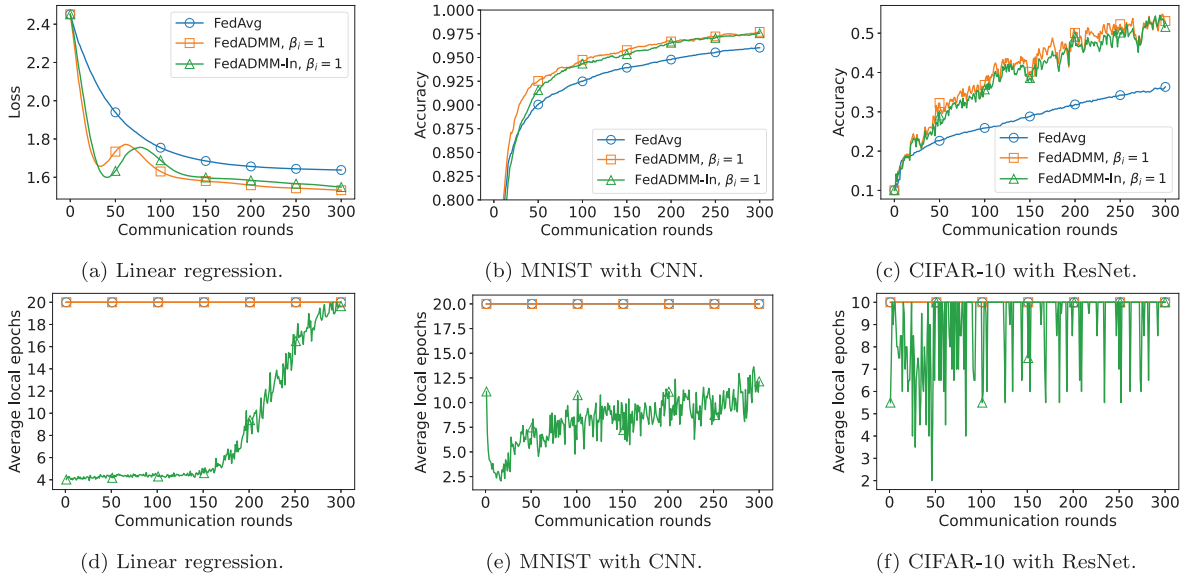
Results of Example 2 (MNIST with CNN). Fig. 3 shows the comparison results on the MNIST dataset using a CNN model. Similar to Example 1, the performance of FedADMM varies significantly with different β_i values. In the two cases plotted with $\beta_i \in \{5, 10\}$, FedADMM performs even worse compared to FedAvg, highlighting its sensitivity to the choice of β_i. Conversely, our proposed FedADMM-InSa consistently achieves superior performance, demonstrating lower training loss and higher test accuracy across all algorithms.

Fig. 3(c) illustrates the change in average penalty parameters of all clients. It can be seen that FedADMM-InSa maintains an effective penalty parameter adaptation process, contributing to its robust performance. Meanwhile, Fig. 3(d) shows the average local epochs. FedADMM-InSa requires much fewer local epochs on average compared to FedADMM, indicating a more efficient training process. This efficiency is critical in FL scenarios, as the client can complete local training faster without compromising the global model accuracy.

Results of Example 3 (CIFAR-10 with ResNet-20). In Fig. 4, the performance of different algorithms on the CIFAR-10 dataset using a ResNet-20 model is presented. Similar to the previous two examples, FedADMM-InSa achieves the best performance, as evidenced by the lowest training loss in Fig. 4(a) and the highest test accuracy in Fig. 4(b). The average local epochs required by FedADMM-InSa, as

Table 2
Comparison results of different examples. The columns showing epoch reduction use the vanilla FedADMM and FedAvg as the baseline.

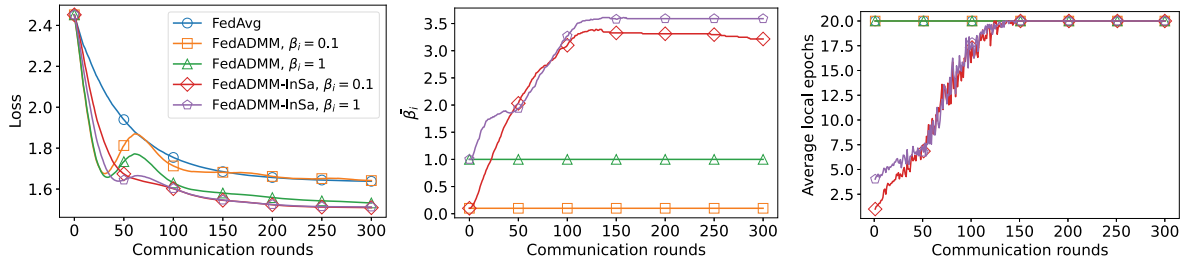
β_i	Algorithm	Example 1 linear regression		Example 2 MNIST, CNN		Example 3 CIFAR-10, ResNet-20	
		Loss	Epoch reduction	Accuracy	Epoch reduction	Accuracy	Epoch reduction
–	Fedavg	1.64	–	96.0%	–	36.3%	–
0.1	FedADMM	1.64	–	**98.6%**	–	51.1%	–
	FedADMM-In	1.63	94.3%	98.5%	92.4%	41.0%	71.5%
	FedADMM-InSa	**1.51**	20.3%	97.8%	66.9%	**53.1%**	14.4%
1	FedADMM	1.53	–	97.7%	–	**52.9%**	–
	FedADMM-In	1.55	58.5%	97.6%	58.7%	51.6%	9.4%
	FedADMM-InSa	**1.51**	18.8%	**97.9%**	61.0%	52.1%	9.4%
2	FedADMM	1.51	–	96.2%	–	46.7%	–
	FedADMM-In	1.51	19.3%	96.7%	41.7%	45.7%	2.1%
	FedADMM-InSa	**1.51**	16.2%	**97.7%**	57.9%	**50.4%**	6.9%
5	FedADMM	1.51	–	92.5%	–	39.5%	–
	FedADMM-In	1.51	3.9%	93.4%	31.1%	39.3%	0.0%
	FedADMM-InSa	**1.51**	12.5%	**97.9%**	55.9%	**51.5%**	4.0%
10	FedADMM	1.51	–	89.7%	–	34.0%	–
	FedADMM-In	1.51	0.9%	90.2%	21.1%	37.3%	0.0%
	FedADMM-InSa	**1.51**	6.8%	**97.5%**	55.7%	**49.8%**	4.8%



(a) Linear regression. (b) MNIST with CNN. (c) CIFAR-10 with ResNet.

(d) Linear regression. (e) MNIST with CNN. (f) CIFAR-10 with ResNet.

Fig. 1. Comparison results of FedADMM-In in three examples.



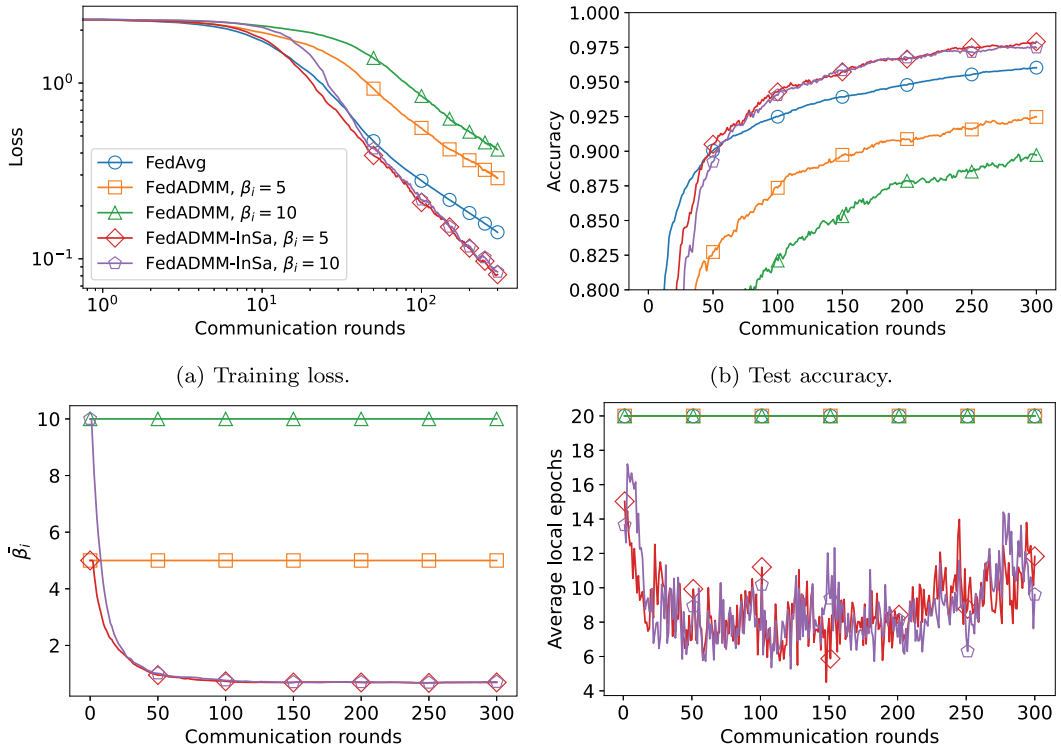
(a) Training loss. (b) Average penalty parameters. (c) Average local epochs.

Fig. 2. Comparison results of FedADMM-InSa in Example 1 (Linear regression).

shown in Fig. 4(d), are still slightly lower than those conducted by FedADMM and FedAvg. The penalty parameter updates in Fig. 4(c) illustrate the adaptive capability of FedADMM-InSa, contributing to its overall effectiveness. In summary, the comparison results across all three examples consistently demonstrate that our FedADMM-InSa outperforms both FedAvg and FedADMM in terms of test accuracy, workload reduction, and robustness across various penalty parameters.

4.2.3. *Evaluation of the self-adaptive penalty parameter scheme* (3.15)

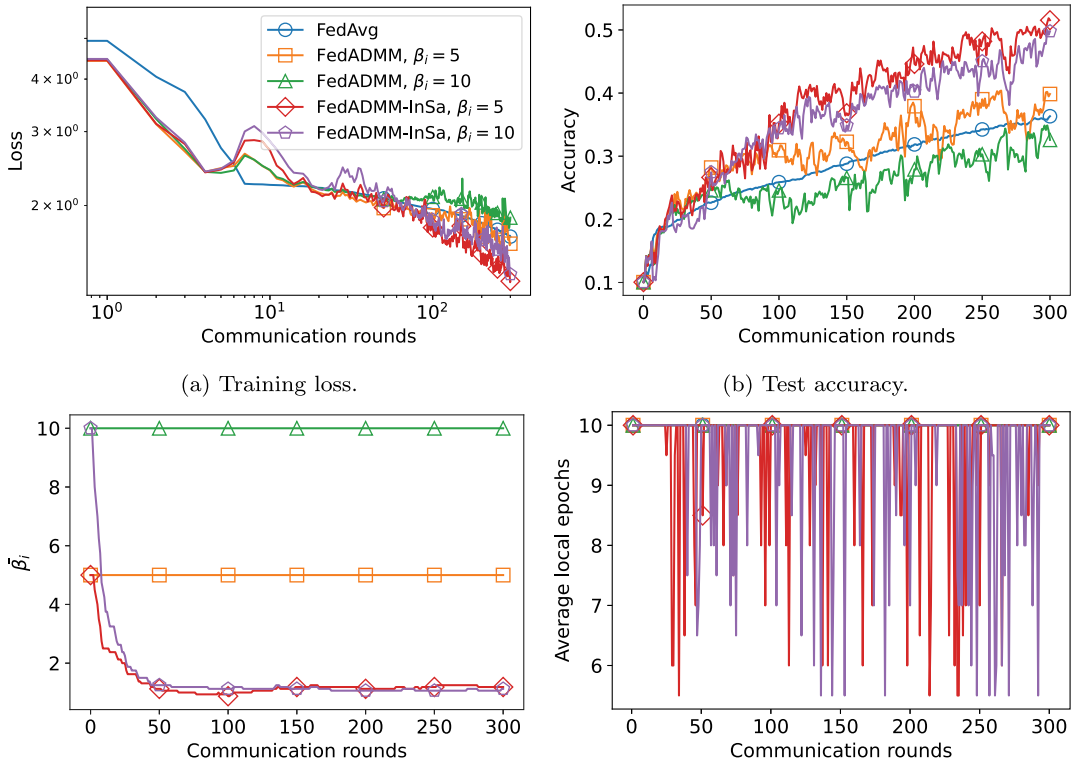
As observed in the three examples above, the pre-selected penalty parameter β_i plays a crucial role in the performance of the FedADMM



(a) Training loss. (b) Test accuracy.

(c) Average penalty parameters. (d) Average local epochs.

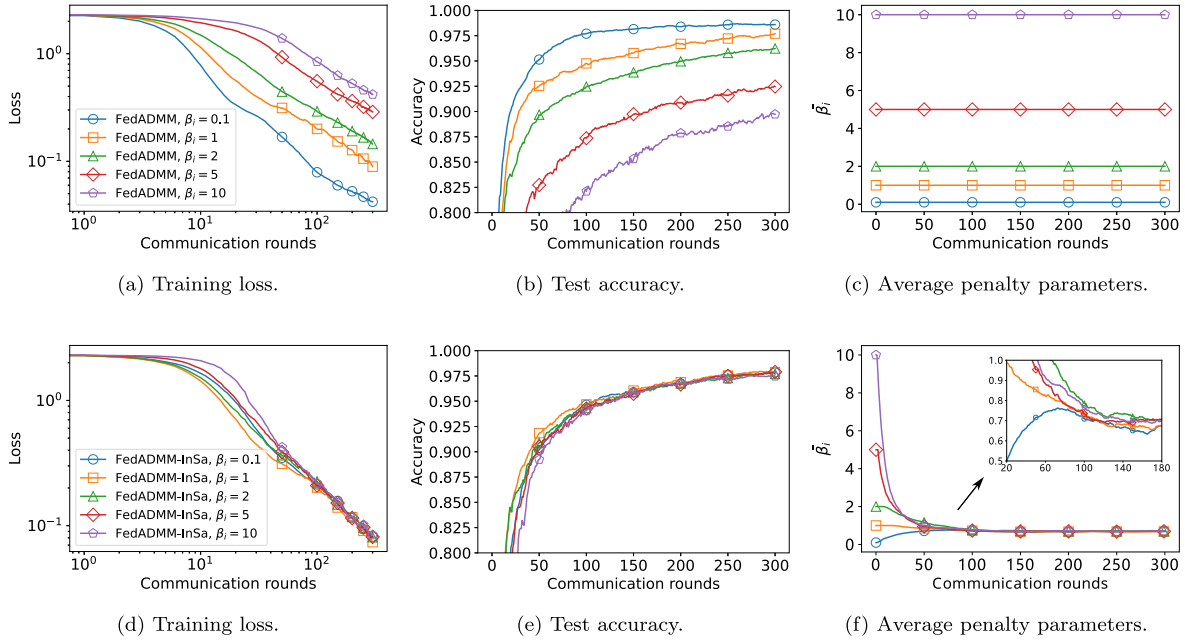
Fig. 3. Comparison results of FedADMM-InSa in Example 2 (MNIST with CNN).



(a) Training loss. (b) Test accuracy.

(c) Average penalty parameters. (d) Average local epochs.

Fig. 4. Comparison results of FedADMM-InSa in Example 3 (CIFAR-10 with ResNet).



(a) Training loss. (b) Test accuracy. (c) Average penalty parameters.

(d) Training loss. (e) Test accuracy. (f) Average penalty parameters.

Fig. 5. Comparison of FedADMM and FedADMM-InSa with different values of β_i in Example 2. FedADMM (top row) uses fixed penalty parameters. Our FedADMM-InSa (bottom row) utilizes adaptive penalty parameter scheme (3.15).

algorithm. For example, it performs better with a larger β_i in Example 1, but with a smaller β_i in Examples 2 and 3. In this subsection, we compare the results of FedADMM and FedADMM-InSa under different values of $\beta_i \in \{0.1, 1, 2, 5, 10\}$ using Example 2. The results of the other two examples are similar and omitted to save space.

It is clear from Fig. 5 that our proposed FedADMM-InSa (bottom row) demonstrates robustness across different values of β_i, while FedADMM (top row) only performs well under certain values. As shown in Fig. 5(f), our proposed adaptive scheme dynamically adjusts the penalty parameter and ensures it converges to stable values, allowing our algorithm to maintain robust performance across different initial penalty parameter values. This adaptive mechanism prevents over-penalization, which can hinder convergence, and under-penalization, which can lead to insufficient coordination among clients. As a result, as shown in Figs. 5(d) and 5(e), FedADMM-InSa achieves lower training loss and higher test accuracy. Moreover, FedADMM-InSa also requires fewer local epochs compared to FedADMM, as detailed in Table 2. Overall, our adaptive penalty parameter scheme (3.15) enhances the flexibility and robustness of our FedADMM-InSa algorithm, making it a superior choice for FL applications where training efficiency and computational resources are critical considerations.

5. Conclusions

In this paper, we introduce the FedADMM-InSa algorithm, a novel approach that leverages the alternating direction method of multipliers (ADMM) to address the challenges of federated learning (FL) in the presence of data and system heterogeneity. Distinguished from current FedADMM methods, our inexact and self-adaptive algorithm mitigates the need for intricate empirical hyperparameter settings. The introduced inexactness criterion improves computational efficiency by removing the requirement to determine local training accuracy in advance. Additionally, the self-adaptive scheme dynamically adjusts each client's penalty parameter, enhancing the robustness of our algorithm. Numerical tests demonstrate the reduction in clients' local computational load and accelerated learning performance on both synthetic and real-world datasets, highlighting the practical advancements our approach brings to FL systems. In addition to the demonstrated benefits of our proposed algorithm, the integration of privacy-preserving techniques and the investigation of the algorithm's performance in large-scale FL applications are interesting and challenging future directions.

CRediT authorship contribution statement

Yongcun Song: Writing – review & editing, Validation, Methodology, Formal analysis, Conceptualization. **Ziqi Wang:** Writing – original draft, Visualization, Software, Methodology, Formal analysis, Investigation, Data curation, Conceptualization. **Enrique Zuazua:** Writing – review & editing, Supervision, Resources, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The co-author Enrique Zuazua is an Acton Editor for Neural Networks and was not involved in the editorial review or the decision to publish this article. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Codes used in this paper can be found at: https://github.com/DCN-FAU-AvH/FedADMM-InSa.

Acknowledgments

Authors' names are listed in alphabetical order by family name to signify equal contributions. The authors are grateful to anonymous referees for their valuable comments which have helped improve the paper substantially. This work has been funded by the Alexander von Humboldt-Professorship program, the Humboldt Research Fellowship for postdoctoral researchers, the European Union's Horizon

Europe MSCA project ModConFlex (grant number 101073558), the COST Action MAT-DYN-NET, the Transregio 154 Project of the DFG, grants PID2020-112617GB-C22 and TED2021-131390B-I00 of MINECO (Spain). Madrid Government - UAM Agreement for the Excellence of the University Research Staff in the context of the V PRICIT (Regional Programme of Research and Technological Innovation).

Appendix

A.1. Convergence analysis of the sequence generated by (3.10)

A.1.1. Notations

To present our convergence analysis in a compact form, we first rewrite the augmented Lagrangian function (2.6) as

$$\begin{aligned} L_{\alpha,\beta}(u,\lambda,z) &= \sum_{i=1}^{m} \alpha_i \left(f_i(u_i) - \lambda_i^\top (u_i - z) + \frac{\beta_i}{2} \|u_i - z\|^2 \right) \\ &= \alpha^\top f(u) - \lambda^\top I_\alpha (u - Bz) + \frac{1}{2}\|u - Bz\|^2_{I_\alpha I_\beta}. \end{aligned} \tag{A.1}$$

Here, $\lambda_i \in \mathbb{R}^n$ is the Lagrange multiplier associated with the equality constraint $u_i = z$, $\beta_i > 0$ is a penalty parameter,

$$\begin{aligned} &u = (u_1, \dots, u_m)^\top \in \mathbb{R}^{mn}, \ \lambda = (\lambda_1, \dots, \lambda_m)^\top \in \mathbb{R}^{mn}, \\ &\alpha = (\alpha_1, \dots, \alpha_m)^\top \in \mathbb{R}^m, \ \beta = (\beta_1, \dots, \beta_m)^\top \in \mathbb{R}^m, \\ &f(u) := (f_1(u_1), \dots, f_m(u_m))^\top \in \mathbb{R}^m, \\ &\nabla f(u) := (\nabla f_1(u_1), \dots, \nabla f_m(u_m))^\top \in \mathbb{R}^{mn}, \\ &e^k(u) := (e_1^k(u_1), \dots, e_m^k(u_m))^\top \in \mathbb{R}^{mn}, \\ &B = (\underbrace{I_n, \dots, I_n}_{m})^\top \in \mathbb{R}^{mn \times n}, \end{aligned} \tag{A.2}$$

and diagonal matrices $I_\alpha, I_\beta \in \mathbb{R}^{mn \times mn}$ are

$$I_\alpha = \begin{pmatrix} \alpha_1 I_n & & \\ & \ddots & \\ & & \alpha_m I_n \end{pmatrix}, \ I_\beta = \begin{pmatrix} \beta_1 I_n & & \\ & \ddots & \\ & & \beta_m I_n \end{pmatrix}. \tag{A.3}$$

We define the H-norm with a symmetric and positive-definite matrix $H \in \mathbb{R}^{n \times n}$ as

$$\|x\|_H = (x^\top H x)^{1/2}, \quad \forall x \in \mathbb{R}^n. \tag{A.4}$$

We also denote $w \in W := \mathbb{R}^{mn} \times \mathbb{R}^{mn} \times \mathbb{R}^n$, $v \in V := \mathbb{R}^{mn} \times \mathbb{R}^n$ and the function $F(w)$ as follows:

$$w = \begin{pmatrix} u \\ \lambda \\ z \end{pmatrix}, \ v = \begin{pmatrix} \lambda \\ z \end{pmatrix}, \ F(w) = \begin{pmatrix} I_\alpha(\nabla f(u) - \lambda) \\ I_\alpha(u - Bz) \\ B^\top I_\alpha \lambda \end{pmatrix}. \tag{A.5}$$

Recall that (u^*, z^*) is the unique solution of problem (2.5) and λ^* is the solution of its dual problem. Let $w^* = (u^*, \lambda^*, z^*)^\top$, then the first-order optimality conditions of problem (2.5) and its dual problem can be expressed as:

$$F(w^*) = 0. \tag{A.6}$$

From Assumption 3.2 and the formula of F, it is easy to deduce that the solution w^* is unique.

A.1.2. Optimality conditions

Using the notations in Appendix A.1.1, we can rewrite scheme (3.10) as

$$\begin{cases} u^{k+1} \approx \arg\min\limits_{u \in \mathbb{R}^{mn}} L_{\alpha,\beta}(u, \lambda^k, z^k), & \text{(a)} \\ \lambda^{k+1} = \lambda^k - I_\beta (u^{k+1} - Bz^k), & \text{(b)} \\ z^{k+1} = \arg\min\limits_{z \in \mathbb{R}^n} L_{\alpha,\beta}(u^{k+1}, \lambda^{k+1}, z) + \frac{\delta}{2} \|B(z - z^k)\|^2_{I_\alpha I_\beta}. & \text{(c)} \end{cases} \tag{A.7}$$

For the point $w^{k+1} = (u^{k+1}, z^{k+1}, \lambda^{k+1})^\top$ generated by (A.7), utilizing the definitions of $e_i^k(u_i)$ in (3.6) and $e^k(u)$ in (A.2), and combining the first-order optimality condition of (A.7c), we have:

$$\begin{cases} \nabla_u L_{\alpha,\beta}(u^{k+1}, \lambda^k, z^k) = I_\alpha (e^k(u^{k+1})), & \text{(a)} \\ \lambda^{k+1} = \lambda^k - I_\beta (u^{k+1} - Bz^k), & \text{(b)} \\ (z - z^{k+1})^\top \left(B^\top I_\alpha \lambda^{k+1} - B^\top I_\alpha I_\beta (u^{k+1} - Bz^{k+1}) + \delta B^\top I_\alpha I_\beta B(z^{k+1} - z^k) \right) \geq 0. & \text{(c)} \end{cases} \tag{A.8}$$

Moreover, note that the inexactness condition (3.8) implies that

$$0 < \frac{\sigma_i^2 \beta_i}{2c_i(1 - \sigma_i)^2} = \left(\frac{\sigma_i}{2c_i(1 - \sigma_i)} \right) \left(\frac{\sigma_i \beta_i}{1 - \sigma_i} \right) < 1, \quad \forall i \in [m]. \tag{A.9}$$

Then, there exists a constant $\mu_i > 0$ such that

$$\left(c_i - \frac{\mu_i}{2} \frac{\sigma_i}{1 - \sigma_i} \right) > 0 \quad \text{and} \quad \left(1 - \frac{1}{\mu_i} \frac{\sigma_i \beta_i}{1 - \sigma_i} \right) > 0, \quad \forall i \in [m]. \tag{A.10}$$

The above inequalities will be used later in the proof.

A.1.3. Convergence proof

With the above preparations, we start to prove the convergence of sequence $\{w^k\}$ generated by (3.10). We first prove two lemmas which will be useful in the following discussion. First of all, we analyze how different the point w^k generated by our algorithm is away from the solution w^* of (A.6).

Lemma A.1. *Let $\{w^k\} = \{(u^k, \lambda^k, z^k)^\top\}$ be the sequence generated by scheme (3.10). Then, for all $w \in W$, one has*

$$\begin{aligned} (w^{k+1} - w)^\top F(w^{k+1}) \leq\ & (u^{k+1} - u)^\top \nabla_u L_{\alpha,\beta}(u^{k+1}, \lambda^k, z^k) \\ & + \frac{1}{2} \left(\|v^k - v\|^2_{H_1} - \|v^{k+1} - v\|^2_{H_1} - \|v^k - v^{k+1}\|^2_{H_1} \right), \end{aligned} \tag{A.11}$$

where

$$v = \begin{pmatrix} \lambda \\ z \end{pmatrix}, \ H_1 = \begin{pmatrix} I_\alpha I_\beta^{-1} I_\alpha B \\ B^\top I_\alpha (1 + \delta) B^\top I_\alpha I_\beta B \end{pmatrix} \succ 0. \tag{A.12}$$

Proof. For all $w \in W$, we have

$$\begin{aligned} (w^{k+1} - w)^\top F(w^{k+1}) \overset{(A.5)}{=}\ & (u^{k+1} - u)^\top I_\alpha (\nabla f(u^{k+1}) - \lambda^{k+1}) \\ & + (\lambda^{k+1} - \lambda)^\top I_\alpha (u^{k+1} - Bz^{k+1}) \\ & + (z^{k+1} - z)^\top B^\top I_\alpha \lambda^{k+1} \end{aligned} \tag{A.13}$$

It follows from (A.1) that

$$\begin{aligned} \nabla_u L_{\alpha,\beta}(u^{k+1}, \lambda^k, z^k) &= I_\alpha (\nabla f(u^{k+1}) - \lambda^k + I_\beta (u^{k+1} - Bz^k)) \\ &\overset{(A.8b)}{=} I_\alpha (\nabla f(u^{k+1}) - \lambda^{k+1}). \end{aligned} \tag{A.14}$$

For the first term on the right-hand side of (A.13), it follows from (A.14) that

$$(u^{k+1} - u)^\top I_\alpha (\nabla f(u^{k+1}) - \lambda^{k+1}) \overset{(A.14)}{=} (u^{k+1} - u)^\top \nabla_u L_{\alpha,\beta}(u^{k+1}, \lambda^k, z^k) \tag{A.15}$$

For the second term on the right-hand side of (A.13), we have

$$\begin{aligned} & (\lambda^{k+1} - \lambda)^\top I_\alpha (u^{k+1} - Bz^{k+1}) \\ &= (\lambda^{k+1} - \lambda)^\top I_\alpha (u^{k+1} - Bz^k + Bz^k - Bz^{k+1}) \\ &= (\lambda^{k+1} - \lambda)^\top I_\alpha (u^{k+1} - Bz^k) + (\lambda^{k+1} - \lambda)^\top I_\alpha B (z^k - z^{k+1}) \\ &\overset{(A.8b)}{=} (\lambda^{k+1} - \lambda)^\top I_\alpha I_\beta^{-1} (\lambda^k - \lambda^{k+1}) + (\lambda^{k+1} - \lambda)^\top I_\alpha B (z^k - z^{k+1}) \end{aligned} \tag{A.16}$$

For the third term on the right-hand side of (A.13), we have

$$\begin{aligned}
&\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} \lambda^{k+1} \\
\overset{(A.8c)}{\leq}& \left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta}\left(u^{k+1}-B z^{k+1}\right)+\delta\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right) \\
=& \left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta}\left(u^{k+1}-B z^{k}+B z^{k}-B z^{k+1}\right) \\
&+\left(z^{k+1}-z\right)^{\top} \delta B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right) \\
=& \left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta}\left(u^{k+1}-B z^{k}\right)+\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right) \\
&+\delta\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right) \\
\overset{(A.8b)}{=}& \left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha}\left(\lambda^{k}-\lambda^{k+1}\right)+(1+\delta)\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right).
\end{aligned} \tag{A.17}$$

Substituting (A.15)–(A.17) back into (A.13) yields

$$\begin{aligned}
&\left(w^{k+1}-w\right)^{\top} F\left(w^{k+1}\right) \\
\leq& \left(u^{k+1}-u\right)^{\top} \nabla_{u} L_{\alpha,\beta}\left(u^{k+1}, \lambda^{k}, z^{k}\right) \\
&+\left(\lambda^{k+1}-\lambda\right)^{\top} I_{\alpha} I_{\beta}^{-1}\left(\lambda^{k}-\lambda^{k+1}\right)+\left(\lambda^{k+1}-\lambda\right)^{\top} I_{\alpha} B\left(z^{k}-z^{k+1}\right) \\
&+\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha}\left(\lambda^{k}-\lambda^{k+1}\right)+(1+\delta)\left(z^{k+1}-z\right)^{\top} B^{\top} I_{\alpha} I_{\beta} B\left(z^{k}-z^{k+1}\right).
\end{aligned} \tag{A.18}$$

Using v and H_1 defined by (A.12), we can rewrite (A.18) in a more compact form:

$$\begin{aligned}
\left(w^{k+1}-w\right)^{\top} F\left(w^{k+1}\right) \leq& \left(u^{k+1}-u\right)^{\top} \nabla_{u} L_{\alpha,\beta}\left(u^{k+1}, \lambda^{k}, z^{k}\right) \\
&+\left(v^{k+1}-v\right)^{\top} H_{1}\left(v^{k}-v^{k+1}\right).
\end{aligned} \tag{A.19}$$

Applying the identity

$$(c-a)^{\top} M(b-c)=\frac{1}{2}\left(\|b-a\|_{M}^{2}-\|c-a\|_{M}^{2}-\|b-c\|_{M}^{2}\right) \tag{A.20}$$

to the second term on the right-hand side of (A.19), we have

$$\left(v^{k+1}-v\right)^{\top} H_{1}\left(v^{k}-v^{k+1}\right)=\frac{1}{2}\left(\|v^{k}-v\|_{H_1}^{2}-\|v^{k+1}-v\|_{H_1}^{2}-\|v^{k}-v^{k+1}\|_{H_1}^{2}\right). \tag{A.21}$$

Substituting (A.21) back into (A.19) yields

$$\begin{aligned}
\left(w^{k+1}-w\right)^{\top} F\left(w^{k+1}\right) \leq& \left(u^{k+1}-u\right)^{\top} \nabla_{u} L_{\alpha,\beta}\left(u^{k+1}, \lambda^{k}, z^{k}\right) \\
&+\frac{1}{2}\left(\|v^{k}-v\|_{H_1}^{2}-\|v^{k+1}-v\|_{H_1}^{2}-\|v^{k}-v^{k+1}\|_{H_1}^{2}\right).
\end{aligned} \tag{A.22}$$

We thus complete the proof. □

On the right-hand side of (A.11), the three quadratic terms are easily amenable to manipulation across various indicators through algebraic operations. However, it is less apparent how the cross-term can be controlled to demonstrate the convergence of the sequence $\{w^k\}$. Therefore, we study this term and demonstrate that the sum of these cross-terms over $K \geq 1$ iterations can be bounded by certain quadratic terms. This result is shown in the following Lemma A.2.

Lemma A.2. *Let $\{w^k\}=\{(u^k,\lambda^k,z^k)^{\top}\}$ be the sequence generated by scheme (3.10). For any integer $K \geq 1$ and μ_i satisfying (A.10), one has*

$$\begin{aligned}
&\sum_{k=1}^{K}\left(u^{k+1}-u\right)^{\top} \nabla_{u} L_{\alpha,\beta}\left(u^{k+1}, \lambda^{k}, z^{k}\right) \\
\leq& \sum_{k=1}^{K}\sum_{i=1}^{m} \alpha_i \frac{\mu_i}{2} \frac{\sigma_i}{1-\sigma_i}\left\|u_i^{k+1}-u_i\right\|^{2}+\sum_{k=1}^{K-1}\sum_{i=1}^{m} \alpha_i \frac{1}{2\mu_i} \frac{\sigma_i}{1-\sigma_i} \beta_i\left\|v_i^{k}-v_i^{k+1}\right\|_{H_{\beta_i}}^{2} \\
&+\sum_{i=1}^{m} \alpha_i \frac{1}{2\mu_i} \frac{\sigma_i}{1-\sigma_i}\left(\left\|e_i^{0}\left(u_i^{1}\right)\right\|+\sqrt{\beta_i}\left\|v_i^{0}-v_i^{1}\right\|_{H_{\beta_i}}\right)^{2}, \quad \forall u \in \mathbb{R}^{mn},
\end{aligned} \tag{A.23}$$

where

$$v_i=\begin{pmatrix}\lambda_i\\ z\end{pmatrix}, \quad H_{\beta_i}=\begin{pmatrix}\frac{1}{\beta_i} I_n & I_n\\ I_n & (1+\delta)\beta_i I_n\end{pmatrix} \succ 0. \tag{A.24}$$

Proof. For the residual $e_i^k(u_i), i \in [m]$, it follows from (3.6) that

$$e_i^{k}\left(u_i^{k}\right)=\nabla f_i\left(u_i^{k}\right)-\lambda_i^{k}+\beta_i\left(u_i^{k}-z^{k}\right), \tag{A.25a}$$

$$e_i^{k-1}\left(u_i^{k}\right)=\nabla f_i\left(u_i^{k}\right)-\lambda_i^{k-1}+\beta_i\left(u_i^{k}-z^{k-1}\right). \tag{A.25b}$$

Combining (A.25a) and (A.25b) we have

$$e_i^{k}(u_i^{k})=e_i^{k-1}(u_i^{k})+\beta_i(z^{k-1}-z^{k})+\lambda_i^{k-1}-\lambda_i^{k}. \tag{A.26}$$

For the residual $e_i^k(u_i^{k+1}), i \in [m]$, it follows from (3.7) that

$$\begin{aligned}
\left\|e_i^{k}(u_i^{k+1})\right\| \overset{(3.7)}{\leq}& \sigma_i\left\|e_i^{k}(u_i^{k})\right\| \\
\overset{(A.26)}{=}& \sigma_i\left\|e_i^{k-1}(u_i^{k})+\beta_i(z^{k-1}-z^{k})+\lambda_i^{k-1}-\lambda_i^{k}\right\| \\
\leq& \sigma_i\left\|e_i^{k-1}(u_i^{k})\right\|+\sigma_i\left\|\beta_i(z^{k}-z^{k-1})+\lambda_i^{k}-\lambda_i^{k-1}\right\|.
\end{aligned} \tag{A.27}$$

For the second term on the right-hand side of (A.27), with any $\delta \geq 0$, we have

$$\begin{aligned}
&\sigma_i\left\|\beta_i(z^{k}-z^{k-1})+\lambda_i^{k}-\lambda_i^{k-1}\right\| \\
=&\sigma_i\left(\beta_i^{2}\left\|z^{k}-z^{k-1}\right\|^{2}+\left\|\lambda_i^{k}-\lambda_i^{k-1}\right\|^{2}+2\beta_i\left(z^{k}-z^{k-1}\right)^{\top}\left(\lambda_i^{k}-\lambda_i^{k-1}\right)\right)^{\frac{1}{2}} \\
=&\sigma_i\sqrt{\beta_i}\left(\beta_i\left\|z^{k}-z^{k-1}\right\|^{2}+\frac{1}{\beta_i}\left\|\lambda_i^{k}-\lambda_i^{k-1}\right\|^{2}+2\left(z^{k}-z^{k-1}\right)^{\top}\left(\lambda_i^{k}-\lambda_i^{k-1}\right)\right)^{\frac{1}{2}} \\
\leq&\sigma_i\sqrt{\beta_i}\left((1+\delta)\beta_i\left\|z^{k}-z^{k-1}\right\|^{2}+\frac{1}{\beta_i}\left\|\lambda_i^{k}-\lambda_i^{k-1}\right\|^{2}+2\left(z^{k}-z^{k-1}\right)^{\top}\left(\lambda_i^{k}-\lambda_i^{k-1}\right)\right)^{\frac{1}{2}}.
\end{aligned} \tag{A.28}$$

Using the H-norm notation in (A.4) with v_i and H_{β_i} defined by (A.24), we have

$$\begin{aligned}
\left\|v_i^{k}-v_i^{k-1}\right\|_{H_{\beta_i}}^{2}=&(1+\delta)\beta_i\left\|z^{k}-z^{k-1}\right\|^{2}+\frac{1}{\beta_i}\left\|\lambda_i^{k}-\lambda_i^{k-1}\right\|^{2} \\
&+2\left(z^{k}-z^{k-1}\right)^{\top}\left(\lambda_i^{k}-\lambda_i^{k-1}\right).
\end{aligned} \tag{A.29}$$

Substituting (A.28) back into (A.27) and using (A.29), we have

$$\begin{aligned}
\left\|e_i^{k}(u_i^{k+1})\right\| \leq& \sigma_i\left\|e_i^{k-1}(u_i^{k})\right\|+\sigma_i\sqrt{\beta_i}\left\|v_i^{k}-v_i^{k-1}\right\|_{H_{\beta_i}} \\
\leq& \sum_{j=0}^{k-1}\sigma_i^{k-j}\sqrt{\beta_i}\left\|v_i^{j}-v_i^{j+1}\right\|_{H_{\beta_i}}+\sigma_i^{k}\left\|e_i^{0}(u_i^{1})\right\|.
\end{aligned} \tag{A.30}$$

By using (A.8) and (A.30), for any $\mu_i > 0$ satisfying (A.10) and $u \in \mathbb{R}^{mn}$, we have

$$\begin{aligned}
&\sum_{k=1}^{K}\left(u^{k+1}-u\right)^{\top} \nabla_{u} L_{\alpha,\beta}\left(u^{k+1}, z^{k}, \lambda^{k}\right) \overset{(A.8a)}{=} \sum_{k=1}^{K}\left(u^{k+1}-u\right)^{\top} I_{\alpha} e^{k}(u^{k+1}) \\
=&\sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\left(u_i^{k+1}-u_i\right)^{\top} e_i^{k}(u_i^{k+1}) \leq \sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\left\|u_i^{k+1}-u_i\right\|\left\|e_i^{k}(u_i^{k+1})\right\| \\
\overset{(A.30)}{\leq}&\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\alpha_i\sigma_i^{k-j}\sqrt{\beta_i}\left\|u_i^{k+1}-u_i\right\|\left\|v_i^{j}-v_i^{j+1}\right\|_{H_{\beta_i}}+\sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\sigma_i^{k}\left\|u_i^{k+1}-u_i\right\|\left\|e_i^{0}(u_i^{1})\right\| \\
\leq&\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{k-1}\alpha_i\sigma_i^{k-j}\sqrt{\beta_i}\left\|u_i^{k+1}-u_i\right\|\left\|v_i^{j}-v_i^{j+1}\right\|_{H_{\beta_i}} \\
&+\sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\sigma_i^{k}\left\|u_i^{k+1}-u_i\right\|\left(\left\|e_i^{0}\left(u_i^{1}\right)\right\|+\sqrt{\beta_i}\left\|v_i^{0}-v_i^{1}\right\|_{H_{\beta_i}}\right) \\
\leq&\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{k-1}\frac{\mu_i}{2}\alpha_i\sigma_i^{k-j}\left\|u_i^{k+1}-u_i\right\|^{2}+\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{k-1}\frac{1}{2\mu_i}\alpha_i\sigma_i^{k-j}\beta_i\left\|v_i^{j}-v_i^{j+1}\right\|_{H_{\beta_i}}^{2} \\
&+\sum_{k=1}^{K}\sum_{i=1}^{m}\frac{\mu_i}{2}\alpha_i\sigma_i^{k}\left\|u_i^{k+1}-u_i\right\|^{2}+\sum_{k=1}^{K}\sum_{i=1}^{m}\frac{1}{2\mu_i}\alpha_i\sigma_i^{k}\left(\left\|e_i^{0}\left(u_i^{1}\right)\right\|+\sqrt{\beta_i}\left\|v_i^{0}-v_i^{1}\right\|_{H_{\beta_i}}\right)^{2} \\
=&\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=0}^{k-1}\frac{\mu_i}{2}\alpha_i\sigma_i^{k-j}\left\|u_i^{k+1}-u_i\right\|^{2}+\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{k-1}\frac{1}{2\mu_i}\alpha_i\sigma_i^{k-j}\beta_i\left\|v_i^{j}-v_i^{j+1}\right\|_{H_{\beta_i}}^{2} \\
&+\sum_{k=1}^{K}\sum_{i=1}^{m}\frac{1}{2\mu_i}\alpha_i\sigma_i^{k}\left(\left\|e_i^{0}\left(u_i^{1}\right)\right\|+\sqrt{\beta_i}\left\|v_i^{0}-v_i^{1}\right\|_{H_{\beta_i}}\right)^{2}.
\end{aligned} \tag{A.31}$$

It follows from (3.8) that $\sigma_i \in (0,1), \forall i \in [m]$. Then, by using the property of the geometric series, we can have

$$\begin{aligned}
&\sum_{k=1}^{K} \left(u^{k+1}-u\right)^{\top} \nabla_u L_{\alpha,\beta}\left(u^{k+1}, z^k, \lambda^k\right) \\
&\leq \sum_{k=1}^{K}\sum_{i=1}^{m} \alpha_i \frac{\mu_i}{2}\frac{\sigma_i-\sigma_i^{k+1}}{1-\sigma_i}\left\|u_i^{k+1}-u_i\right\|^2 + \sum_{i=1}^{m}\sum_{j=1}^{K-1}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i-\sigma_i^{K-j+1}}{1-\sigma_i}\beta_i\left\|v_i^j-v_i^{j+1}\right\|_{H_{\beta_i}}^2 \\
&\quad + \sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i-\sigma_i^{K+1}}{1-\sigma_i}\left(\left\|e_i^0\left(u_i^1\right)\right\| + \sqrt{\beta_i}\left\|v_i^0-v_i^1\right\|_{H_{\beta_i}}\right)^2 \qquad \text{(A.32)} \\
&\overset{(3.8)}{\leq} \sum_{k=1}^{K}\sum_{i=1}^{m} \alpha_i \frac{\mu_i}{2}\frac{\sigma_i}{1-\sigma_i}\left\|u_i^{k+1}-u_i\right\|^2 + \sum_{k=1}^{K-1}\sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i}{1-\sigma_i}\beta_i\left\|v_i^k-v_i^{k+1}\right\|_{H_{\beta_i}}^2 \\
&\quad + \sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i}{1-\sigma_i}\left(\left\|e_i^0\left(u_i^1\right)\right\| + \sqrt{\beta_i}\left\|v_i^0-v_i^1\right\|_{H_{\beta_i}}\right)^2, \quad \forall u \in \mathbb{R}^{mn}.
\end{aligned}$$

This completes the proof. □

Now, with the help of Lemmas A.1 and A.2, we can show the convergence of our algorithm in the subsequent proof.

Proof of Theorem 3.3. First, from (3.12) and (A.5), we can have

$$\begin{aligned}
\left(w^{k+1}-w\right)^{\top}\left(F(w)-F(w^{k+1})\right) &\overset{(A.5)}{=} \left(u^{k+1}-u\right)^{\top} I_\alpha\left(\nabla f(u)-\nabla f(u^{k+1})\right) \\
&= \sum_{i=1}^{m}\alpha_i\left(u_i^{k+1}-u_i\right)^{\top}\left(\nabla f_i\left(u_i\right)-\nabla f_i(u_i^{k+1})\right) \\
&\overset{(3.12)}{\leq} -\sum_{i=1}^{m}\alpha_i c_i\left\|u_i-u_i^{k+1}\right\|^2.
\end{aligned} \qquad \text{(A.33)}$$

Then, using (A.11) and (A.23) established in Lemmas A.1 and A.2, respectively, we obtain

$$\begin{aligned}
&\sum_{k=1}^{K}\left(w^{k+1}-w\right)^{\top} F(w) \\
&= \sum_{k=1}^{K}\left(w^{k+1}-w\right)^{\top} F(w^{k+1}) + \sum_{k=1}^{K}\left(w^{k+1}-w\right)^{\top}\left(F(w)-F(w^{k+1})\right) \\
&\overset{(A.11),\ (A.33)}{\leq} \sum_{k=1}^{K}\left(u^{k+1}-u\right)^{\top}\nabla_u L_{\alpha,\beta}\left(u^{k+1},\lambda^k,z^k\right) + \frac{1}{2}\left(\left\|v^1-v\right\|_{H_1}^2 - \left\|v^{K+1}-v\right\|_{H_1}^2\right) \\
&\quad - \sum_{k=1}^{K}\frac{1}{2}\left\|v^k-v^{k+1}\right\|_{H_1}^2 - \sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i c_i\left\|u_i-u_i^{k+1}\right\|^2 \\
&\overset{(A.23)}{\leq} \sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\frac{\mu_i}{2}\frac{\sigma_i}{1-\sigma_i}\left\|u_i^{k+1}-u_i\right\|^2 - \sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i c_i\left\|u_i-u_i^{k+1}\right\|^2 \\
&\quad + \sum_{k=1}^{K-1}\sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i\beta_i}{1-\sigma_i}\left\|v_i^k-v_i^{k+1}\right\|_{H_{\beta_i}}^2 - \sum_{k=1}^{K}\sum_{i=1}^{m}\frac{\alpha_i}{2}\left\|v_i^k-v_i^{k+1}\right\|_{H_{\beta_i}}^2 \qquad \text{(A.34)} \\
&\quad + \sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i}{1-\sigma_i}\left(\left\|e_i^0\left(u_i^1\right)\right\| + \sqrt{\beta_i}\left\|v_i^0-v_i^1\right\|_{H_{\beta_i}}\right)^2 \\
&\quad + \frac{1}{2}\left(\left\|v^1-v\right\|_{H_1}^2 - \left\|v^{K+1}-v\right\|_{H_1}^2\right) \\
&= \sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\left(\frac{\mu_i}{2}\frac{\sigma_i}{1-\sigma_i}-c_i\right)\left\|u_i^{k+1}-u_i\right\|^2 \\
&\quad + \sum_{k=1}^{K-1}\sum_{i=1}^{m}\frac{\alpha_i}{2}\left(\frac{1}{\mu_i}\frac{\sigma_i\beta_i}{1-\sigma_i}-1\right)\left\|v_i^k-v_i^{k+1}\right\|_{H_{\beta_i}}^2 \\
&\quad + \sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i}{1-\sigma_i}\left(\left\|e_i^0\left(u_i^1\right)\right\| + \sqrt{\beta_i}\left\|v_i^0-v_i^1\right\|_{H_{\beta_i}}\right)^2 \\
&\quad + \frac{1}{2}\left(\left\|v^1-v\right\|_{H_1}^2 - \left\|v^{K+1}-v\right\|_{H_1}^2 - \left\|v^K-v^{K+1}\right\|_{H_1}^2\right).
\end{aligned}$$

For the solution point w^*, it follows from (A.6) that $F(w^*)=0$. Setting $w=w^*$ in (A.34), together with the above property, for any integer $K>1$, we have

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{i=1}^{m}\alpha_i\left(c_i-\frac{\mu_i}{2}\frac{\sigma_i}{1-\sigma_i}\right)\left\|u_i^{k+1}-u_i^*\right\|^2 \\
&\quad + \sum_{k=1}^{K-1}\sum_{i=1}^{m}\frac{\alpha_i}{2}\left(1-\frac{1}{\mu_i}\frac{\sigma_i\beta_i}{1-\sigma_i}\right)\left\|v_i^k-v_i^{k+1}\right\|_{H_{\beta_i}}^2 \\
&\leq \sum_{i=1}^{m}\alpha_i\frac{1}{2\mu_i}\frac{\sigma_i}{1-\sigma_i}\left(\left\|e_i^0\left(u_i^1\right)\right\| + \sqrt{\beta_i}\left\|v_i^0-v_i^1\right\|_{H_{\beta_i}}\right)^2 \qquad \text{(A.35)} \\
&\quad + \frac{1}{2}\left(\left\|v^1-v^*\right\|_{H_1}^2 - \left\|v^{K+1}-v^*\right\|_{H_1}^2 - \left\|v^K-v^{K+1}\right\|_{H_1}^2\right).
\end{aligned}$$

Note that $\alpha_i>0, \forall i\in[m]$, and it follows from (A.10) that

$$\left(c_i-\frac{\mu_i}{2}\frac{\sigma_i}{1-\sigma_i}\right)>0 \quad \text{and} \quad \left(1-\frac{1}{\mu_i}\frac{\sigma_i\beta_i}{1-\sigma_i}\right)>0, \quad \forall i\in[m]. \qquad \text{(A.36)}$$

Therefore, the inequality (A.35) implies that

$$\left\|u_i^{k+1}-u_i^*\right\| \overset{k\to\infty}{\longrightarrow} 0, \quad \left\|v_i^{k+1}-v_i^k\right\|_{H_{\beta_i}} \overset{k\to\infty}{\longrightarrow} 0, \quad \forall i\in[m]. \qquad \text{(A.37)}$$

Using $\|v_i^{k+1}-v_i^k\|_{H_{\beta_i}} \overset{k\to\infty}{\longrightarrow} 0$ and (A.24), we can infer that

$$\left\|z^{k+1}-z^k\right\| \overset{k\to\infty}{\longrightarrow} 0, \quad \left\|\lambda_i^{k+1}-\lambda_i^k\right\| \overset{k\to\infty}{\longrightarrow} 0, \quad \forall i\in[m]. \qquad \text{(A.38)}$$

Since $\|u_i^{k+1}-z^{k+1}\| = \frac{1}{\beta_i}\|\lambda_i^{k+1}-\lambda_i^k\|$, we can also have

$$\left\|u_i^{k+1}-z^{k+1}\right\| \overset{k\to\infty}{\longrightarrow} 0, \quad \forall i\in[m] \qquad \text{(A.39)}$$

Combining the facts that $u_i^k \overset{k\to\infty}{\longrightarrow} u_i^*$ in (A.37), $u_i^*=z^*$, and $\|u_i^{k+1}-z^{k+1}\| \overset{k\to\infty}{\longrightarrow} 0$ in (A.39), one has

$$z^k \overset{k\to\infty}{\longrightarrow} z^*. \qquad \text{(A.40)}$$

It follows from (A.37) that for any $\epsilon>0$, there exists k_0, such that for all $k\geq k_0$, we have $\|v_i^{k+1}-v_i^k\|_{H_{\beta_i}}<\epsilon$ and $\sigma_i^k<\epsilon$. Then, for all $k\geq k_0$, it follows from (A.30) that

$$\begin{aligned}
&\left\|e_i^k(u_i^{k+1})\right\| \\
&\leq \sum_{j=0}^{k-1}\sigma_i^{k-j}\sqrt{\beta_i}\|v_i^j-v_i^{j+1}\|_{H_{\beta_i}} + \sigma_i^k\left\|e_i^0\left(u_i^1\right)\right\| \\
&= \sum_{j=0}^{k_0-1}\sigma_i^{k-j}\sqrt{\beta_i}\|v_i^j-v_i^{j+1}\|_{H_{\beta_i}} + \sum_{j=k_0}^{k-1}\sigma_i^{k-j}\sqrt{\beta_i}\|v_i^j-v_i^{j+1}\|_{H_{\beta_i}} + \sigma_i^k\left\|e_i^0\left(u_i^1\right)\right\| \\
&\leq \left(\sqrt{\beta_i}\max_{0\leq j\leq k_0-1}\|v_i^j-v_i^{j+1}\|_{H_{\beta_i}}\sum_{j=0}^{k_0-1}\sigma_i^{k-k_0-j}\right)\sigma_i^{k_0} + \epsilon\sqrt{\beta_i}\sum_{j=k_0}^{k-1}\sigma_i^{k-j} + \epsilon\left\|e_i^0\left(u_i^1\right)\right\| \\
&\leq \epsilon\left(\sqrt{\beta_i}\max_{0\leq j\leq k_0-1}\|v_i^j-v_i^{j+1}\|_{H_{\beta_i}}\sum_{j=0}^{k_0-1}\sigma_i^{k-k_0-j} + \sqrt{\beta_i}\sum_{j=k_0}^{k-1}\sigma_i^{k-j} + \left\|e_i^0\left(u_i^1\right)\right\|\right),
\end{aligned}$$

which implies that

$$\left\|e_i^k(u_i^{k+1})\right\| \overset{k\to\infty}{\longrightarrow} 0, \quad \forall i\in[m]. \qquad \text{(A.41)}$$

From (3.6) and (A.6), we can have $\lambda_i^*=\nabla f_i(u_i^*)$ and

$$\lambda_i^k = \nabla f_i(u_i^{k+1}) + \beta_i(u_i^{k+1}-z^k) - e_i^k(u_i^{k+1}). \qquad \text{(A.42)}$$

Then, we have

$$\lambda_i^k-\lambda_i^* = \nabla f_i(u_i^{k+1})-\nabla f_i(u_i^*)+\beta_i(u_i^{k+1}-u_i^k)+\beta_i(u_i^k-z^k)-e_i^k(u_i^{k+1}). \qquad \text{(A.43)}$$

Note that $u_i^k \overset{k\to\infty}{\longrightarrow} u_i^*, u_i^k-z^k \overset{k\to\infty}{\longrightarrow} 0$, $e_i^k\left(u^{k+1}\right) \overset{k\to\infty}{\longrightarrow} 0$, and the gradient of f_i is Lipschitz continuous (see Assumption 3.1), we have

$$\lambda_i^k \overset{k\to\infty}{\longrightarrow} \lambda_i^*, \quad \forall i\in[m]. \qquad \text{(A.44)}$$

We thus complete the proof. □

References

Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., & Saligrama, V. (2021). Federated learning based on dynamic regularization. http://dx.doi.org/10.48550/arXiv.2111.04263, arXiv preprint arXiv:2111.04263.

Awheda, M. D., & Schwartz, H. M. (2016). Exponential moving average based multiagent reinforcement learning algorithms. *Artificial Intelligence Review*, *45*(3), 299–332. http://dx.doi.org/10.1007/s10462-015-9447-5.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., & McMahan, B. (2019). Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, *1*, 374–388.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, *3*(1), 1–122. http://dx.doi.org/10.1561/2200000016.

Cai, Z., Ravichandran, A., Maji, S., Fowlkes, C., Tu, Z., & Soatto, S. (2021). Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 194–203).

Dinh, C. T., Tran, N., & Nguyen, J. (2020). Personalized federated learning with moreau envelopes. *33*, In *Advances in neural information processing systems* (pp. 21394–21405). Curran Associates, Inc..

Geng, J., Mou, Y., Li, Q., Li, F., Beyan, O., Decker, S., & Rong, C. (2023). Improved gradient inversion attacks and defenses in federated learning. *IEEE Transactions on Big Data*, 1–13. http://dx.doi.org/10.1109/TBDATA.2023.3239116.

Glowinski, R. (2014). On alternating direction methods of multipliers: A historical perspective. In W. Fitzgibbon, Y. A. Kuznetsov, P. Neittaanmäki, & O. Pironneau (Eds.), *vol. 34*, *Modeling, simulation and optimization for science and technology* (pp. 59–82). Dordrecht: Springer Netherlands, http://dx.doi.org/10.1007/978-94-017-9054-3_4.

Glowinski, R., & Marroco, A. (1975). Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, *9*(R2), 41–76. http://dx.doi.org/10.1051/m2an/197509R200411.

Glowinski, R., Song, Y., & Yuan, X. (2020). An ADMM numerical approach to linear parabolic state constrained optimal control problems. *Numerische Mathematik*, *144*(4), 931–966. http://dx.doi.org/10.1007/s00211-020-01104-4.

Glowinski, R., Song, Y., Yuan, X., & Yue, H. (2022). Application of the alternating direction method of multipliers to control constrained parabolic optimal control problems and beyond. *Annals of Applied Mathematics*, *38*, http://dx.doi.org/10.4208/aam.OA-2022-0004.

Goldstein, T., Li, M., & Yuan, X. (2015). Adaptive primal-dual splitting methods for statistical learning and image processing. *28*, In *Advances in neural information processing systems*. Curran Associates, Inc..

Gong, Y., Li, Y., & Freris, N. M. (2022). FedADMM: A robust federated deep learning framework with adaptivity to system heterogeneity. In *2022 IEEE 38th international conference on data engineering* (pp. 2575–2587). http://dx.doi.org/10.1109/ICDE53745.2022.00238.

He, B., Yang, H., & Wang, S. (2000). Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, *106*(2), 337–356. http://dx.doi.org/10.1023/A:1004603514434.

He, B., & Yuan, X. (2018). A class of ADMM-based algorithms for three-block separable convex programming. *Computational Optimization and Applications*, *70*(3), 791–826. http://dx.doi.org/10.1007/s10589-018-9994-1.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications*, *4*(5), 303–320. http://dx.doi.org/10.1007/BF00927673.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., ... Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1–210. http://dx.doi.org/10.1561/2200000083.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T. (2020). SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th international conference on machine learning* (pp. 5132–5143). PMLR.

Krizhevsky, A., & Hinton, G. (2009). *Learning Multiple Layers of Features from Tiny Images*: *Technical Report*, University of Toronto: University of Toronto.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791.

Li, X., Huang, K., Yang, W., Wang, S., & Zhang, Z. (2019). On the convergence of FedAvg on non-IID data. In *International conference on learning representations*.

Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, *37*(3), 50–60. http://dx.doi.org/10.1109/MSP.2020.2975749.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, *2*, 429–450.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., & He, B. (2021). A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 1. http://dx.doi.org/10.1109/TKDE.2021.3124599.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1), 503–528. http://dx.doi.org/10.1007/BF01589116.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th international conference on artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Powell, M. J. (1969). A method for nonlinear constraints in minimization problems. *Optimization*, 283–298.

Song, C., Yoon, S., & Pavlovic, V. (2016). Fast ADMM algorithm for distributed optimization with adaptive penalty. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1), http://dx.doi.org/10.1609/aaai.v30i1.10069.

Song, Y., Yuan, X., & Yue, H. (2023). The ADMM-PINNs algorithmic framework for nonsmooth PDE-constrained optimization: A deep learning approach. http://dx.doi.org/10.48550/arXiv.2302.08309, arXiv preprint arXiv:2302.08309.

Tan, A. Z., Yu, H., Cui, L., & Yang, Q. (2023). Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(12), 9587–9603. http://dx.doi.org/10.1109/TNNLS.2022.3160699.

Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. In *NIPS'20*, *Proceedings of the 34th international conference on neural information processing systems* (pp. 7611–7623). Red Hook, NY, USA: Curran Associates Inc..

Wang, H., Marella, S., & Anderson, J. (2022). FedADMM: A federated primal-dual algorithm allowing partial participation. In *2022 IEEE 61st conference on decision and control* (pp. 287–294). http://dx.doi.org/10.1109/CDC51059.2022.9992745.

Wang, Z., Song, Y., & Zuazua, E. (2023). Approximate and weighted data reconstruction attack in federated learning. http://dx.doi.org/10.48550/arXiv.2308.06822, arXiv preprint arXiv:2308.06822.

Wang, S., Xu, Y., Wang, Z., Chang, T.-H., Quek, T. Q. S., & Sun, D. (2023). Beyond ADMM: A unified client-variance-reduced adaptive federated learning framework. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(8), 10175–10183. http://dx.doi.org/10.1609/aaai.v37i8.26212.

Xiao, D., Li, J., & Li, M. (2024). Privacy-preserving federated compressed learning against data reconstruction attacks based on secure data. In B. Luo, L. Cheng, Z.-G. Wu, H. Li, & C. Li (Eds.), *Communications in computer and information science*, *Neural information processing* (pp. 325–339). Singapore: Springer Nature, http://dx.doi.org/10.1007/978-981-99-8184-7_25.

Xu, Y., Liu, M., Lin, Q., & Yang, T. (2017). ADMM without a fixed penalty parameter: Faster convergence with new adaptive penalization. *30*, In *Advances in neural information processing systems*. Curran Associates, Inc..

Yang, Y., Ma, Z., Xiao, B., Liu, Y., Li, T., & Zhang, J. (2023). Reveal your images: Gradient leakage attack against unbiased sampling-based secure aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 1–14. http://dx.doi.org/10.1109/TKDE.2023.3271432.

Yue, H., Yang, Q., Wang, X., & Yuan, X. (2018). Implementing the alternating direction method of multipliers for big datasets: A case study of least absolute shrinkage and selection operator. *SIAM Journal on Scientific Computing*, *40*(5), A3121–A3156. http://dx.doi.org/10.1137/17M1146567.

Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., & Avestimehr, A. S. (2022). Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, *5*(1), 24–29. http://dx.doi.org/10.1109/IOTM.004.2100182.

Zhang, X., Hong, M., Dhople, S., Yin, W., & Liu, Y. (2021). FedPD: A federated learning framework with adaptivity to non-IID data. *IEEE Transactions on Signal Processing*, *69*, 6055–6070. http://dx.doi.org/10.1109/TSP.2021.3115952.

Zhang, K., Li, J., Song, Y., & Wang, X. (2017). An alternating direction method of multipliers for elliptic equation constrained optimization problem. *Science China. Mathematics*, *60*, 361–378. http://dx.doi.org/10.1007/s11425-015-0522-3.

Zhou, S., & Li, G. Y. (2023). Federated learning via inexact ADMM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(8), 9699–9708. http://dx.doi.org/10.1109/TPAMI.2023.3243080.

Zinkevich, M., Weimer, M., Li, L., & Smola, A. (2010). Parallelized stochastic gradient descent. *23*, In *Advances in neural information processing systems*. Curran Associates, Inc..