How a Small Amount of Data Sharing Benefits Distributed Optimization and Learning

Mingxi Zhu
Scheller College of Business, Georgia Institute of Technology, Atlanta, GA 30332 mingxiz@stanford.edu

Yinyu Ye
Management Science & Engineering, Stanford University, Stanford, CA 94305, yyye@stanford.edu

Distributed optimization algorithms have been widely used in machine learning. While those algorithms have the merits in parallel processing and protecting data security, they often suffer from slow convergence. This paper focuses on how a small amount of data sharing could benefit distributed optimization and learning. Specifically, we examine higher-order optimization algorithms including distributed multi-block alternating direction method of multipliers (ADMM) and preconditioned conjugate gradient method (PCG).

The contribution of this paper is three-folded. First, in theory, we answer when and why distributed optimization algorithms are slow by identifying the worst data structure. Surprisingly, while PCG algorithm converges slowly under heterogeneous data structure, for distributed ADMM, data homogeneity leads to the worst performance. This result challenges the common belief that data heterogeneity hurts convergence, highlighting the need for a universal approach on altering data structure for different algorithms. Second, in practice, we propose a meta-algorithm of data sharing, with its tailored applications in multi-block ADMM and PCG methods. By only sharing a small amount of prefixed data (e.g. 1%), our algorithms provide good quality estimators in different machine learning tasks within much fewer iterations, while purely distributed optimization algorithms may take hundreds more times of iterations to converge. Finally, in philosophy, we argue that even minimal collaboration can have huge synergy, which is a concept that extends beyond the realm of optimization analysis. We hope that the discovery resulting from this paper would encourage even a small amount of data sharing among different regions to combat difficult global learning problems.

Key words: Distributed Learning; Distributed Optimization; Data Sharing

1. Introduction

Distributed optimization algorithms have been widely used in large scale machine learning problems [Eckstein and Bertsekas (1992a), Boyd et al. (2011), Nishihara et al. (2015)]. The benefit of distributed algorithms is two-folded. Firstly, by distributing the workload to local data centers,
distributed optimization algorithms could potentially reduce the required computation time from parallel computing and processing. A salient feature of most machine learning problems is that the objectives are separable, and distributed optimization algorithms could exploit the benefit of having separable objectives. Secondly, distributed optimization algorithms are preferred when data communication between agents is costly. However, in practice, distributed optimization algorithms often suffer from slow convergence to the target tolerance level when applied to large scale machine learning problems [Ghadimi et al. (2014), Nilsson et al. (2018)].

This work aims at providing a solution to boost the convergence speed for applying distributed optimization algorithms in machine learning problems, from both theoretical and practical aspects. We first provide a theoretical answer to the following questions: why distributed optimization algorithms can sometimes have unsatisfactory performance in terms of convergence speed, and what kind of data distribution leads to such slow convergence. Surprisingly, we found that, contradicting to the well-established results that data heterogeneity hurts convergence speed, for certain algorithms, data homogeneity leads to worst convergence speed. This result sheds light on the importance for the need of a universal approach on altering data distribution across agents for different distributed optimization algorithms. With theoretical guidance, this paper provides a meta data-sharing algorithm that only requires sharing a small amount of pre-fixed data to build a global data pool. And we numerically show that only a small amount of data share is sufficient to improve the convergence speed for different distributed optimization algorithms. With tailored design on utilizing the shared data to different distributed algorithms, we are able to enjoy both the benefit from faster convergence from data sharing, and from parallel computing by keeping the main structure of the optimization algorithms closer to a distributed manner. Sharing a small amount of data is also of great practical importance, as most of the time, sharing all the data from local agents and building a centralized optimization algorithm may not be a feasible solution due to either high cost of data communication, or constraints on communication regarding privacy and security concerns. However, building a global data pool from a one-time communication on a small amount of pre-fixed data is way less costly compared to centralized optimization by sharing all data to central server. Besides, when the local agents process a hybrid of private data and synthetic data, sharing synthetic data across agents is feasible and would not violate privacy or security constraints [Giuffrè and Shung (2023)].

More importantly, philosophically, the results provided in this paper confirm the synergy of even minimal degree of collaboration, which is a concept that goes beyond the regime of optimization analysis. Many studies have examined the benefit of data sharing, in aspects of data-driven decision making, supply chain management and revenue management to name a few [Lee et al. (2000),
Shang et al. (2016), Feasey and de Streel (2020), Choe et al. (2023). In this work, we show that data sharing not only increases the social surplus and individual utility, but also improves the efficiency in optimization algorithms. And such benefit could be achieved with even 5% of sharing pre-fixed data. And the synergy of collaboration across agents is amplified in data-driven decision making, through both the knowledge spillover effect in utility analysis, and the improved efficiency when applying optimization algorithms in data-driven problems. Hence, we hope that the discovery resulted from this paper would encourage even a small amount of data sharing among different regions to combat difficult global learning problems.

2. Literature Review

Our paper examines how a small amount of data sharing benefits distributed optimization algorithms when applying to machine learning problems, with focus on multi-block ADMM algorithm and PCG methods. In this section, we first review literature related to multi-block ADMM and PCG algorithms. One stream of work, federated learning, is also closely related to applying distributed optimization in machine learning. We review the literature in federated learning and point out the differences between our work and federated learning, in hope that the results presented in this paper would also shed lights in federated learning algorithm design. Lastly, we then review literature related in distributed optimization and federated learning, with focus on the negative impacts of data heterogeneity. Contrary to those study, our result shows that data heterogeneity can sometimes benefit higher-order optimization algorithms, which suggests that a simple solution to mitigate the challenge of data heterogeneity is to “order-up”.

ADMM algorithms have been widely viewed as an optimization method that lies in between first order methods and second order methods, and many researchers have found that ADMM algorithms are well-suited for large scale statistical inference and machine learning problems [Boyd et al. (2011), Mohan et al. (2014), Huang and Sidiropoulos (2016), Taylor et al. (2016)]. The traditional two-block ADMM algorithm and its convergence analysis have been excessively explored [Gabay and Mercier (1976), Eckstein and Bertsekas (1992a), He et al. (2012), Monteiro and Svaiter (2013), Glowinski (2014), Deng and Yin (2016)]. However, the two-block ADMM is not well-suited for solving large scale machine learning problems, as it requires processing and factorization for a large matrix, which is both time and space consuming [Stellato et al. (2018), Zarepisheh et al. (2018), Lin et al. (2017)]. Instead, variants of multi-block ADMM emerges, several prevailing multi-block ADMM algorithms include distributed ADMM [Eckstein and Bertsekas (1992b), He and Yuan (2012), Ouyang et al. (2013)], cyclic ADMM [Hestenes (1969), Powell (1978)], and randomized multi-block ADMM [Sun et al. (2020), Mihić et al. (2020)]. While distributed ADMM is known to be robust for convergence under any choice of step size parameter, many studies have also found
that the convergence rate of primal distributed ADMM is highly sensitive to the choice of step size, and may suffer from slow convergence once the step-size is not well-tuned [Ghadimi et al. (2014), Nishihara et al. (2015), Iutzeler et al. (2015)]. On the other hand, while cyclic ADMM can boost up the convergence rate sometimes compared with distributed ADMM, it is not guaranteed to converge for all instances in theory [Chen et al. (2017)]. Mihić et al. (2020) first introduced data-share across different blocks in multi-block ADMM algorithm. However, their algorithm requires a full randomized data exchange across blocks at each iteration, which may not be feasible in practice if there is data security concerns. Moreover, it might require more computation and communication cost under full randomization scheme.

Similar as ADMM algorithms, conjugate gradient methods have also been considered as lying between first order gradient methods and second order Newton’s method [Luenberger et al. (1984), Shewchuk et al. (1994)]. Conjugate gradient methods have been recently applied to not only solving large scale linear system, but also machine learning tasks with parallel implementation [Le et al. (2011), Helfenstein and Koko (2012), Hsia et al. (2018)]. The convergence of conjugate gradient method has been well-studied, and many studies assure on the point that preconditioning is indispensable to algorithm performance improvement, as ill-conditioning of matrix often leads to divergence, which leads to the prevailing preconditioned conjugate gradient method (PCG) [Axelsson and Lindskog (1986), Kaasschieter (1988), Kaporin (1994), Herzog and Sachs (2010)]. Recently, Drineas and Mahoney (2016) introduce a randomized sampling process to improve the computational efficiency in preconditioning of large-scale matrix. However, their algorithm is designed to solve large scale linear algebra problems, and cannot be directly applied to distributed learning problems.

Federated learning is also related to our work. Federated learning is a machine learning technique that trains the data/observations in decentralized local data sets [Konečný et al. (2015), Bonawitz et al. (2019)]. In the setup of federated learning, most literature assumes that the center only has access to the intermediate training weights of local agents without exchanging raw data [McMahan et al. (2017), Stich (2018), Khaled et al. (2020), Yuan and Ma (2020)]. And the common prevailing algorithms employed in federated learning are first-order based (e.g. FedAvg, FSVRG [Mangasarian (1995), McMahan et al. (2017), Zhou and Cong (2017), Nilsson et al. (2018)]), while several studies “order-up” the gradient based algorithm [Agafonov et al. (2022)]. Although the regime of our study in distributed optimization with data sharing does not directly belong to the setup in federated learning, where the raw data remains private. We still believe that in practice, when local agents possess a hybrid of data sets which contains both privacy-sensitive and less privacy-sensitive data, exchanging a small amount of less privacy-sensitive raw data across agents could benefit algorithm
performance. And we hope our results in distributed optimization could shed lights in federated learning algorithm design.

In all previous streams of literature related to distributed optimization and federated learning, data heterogeneity across local agents typically leads to adverse effects. Specifically, for first-order algorithms, data heterogeneity is often viewed as a major challenge in speeding up training methods [Wang et al. (2020), Woodworth et al. (2020), Karimireddy et al. (2020)]. Several survey papers [Gao et al. (2022), Ye et al. (2023)] have also summarized the negative impact of data heterogeneity in federated learning framework. And currently, many first-order based distributed learning algorithms provide unsatisfying training results with heterogeneous data sources [Li et al. (2022), Mendieta et al. (2022)]. In our paper, we show that contrary to those results, when applying higher-order optimization algorithms, data heterogeneity may have positive impacts on boosting up the convergence speed. This suggests that one way to mitigate the challenge of data heterogeneity may simply be order-up the optimization algorithms, and shift from first-order based training methods to higher order optimization methods in machine learning.

Our main theoretical contribution to the previous literature in multi-block ADMM, PCG methods, and impact of data heterogeneity to distributed optimization, is on discovering when and why the distributed optimization algorithms suffer from slow convergence. To answer those questions, we apply new technical method with matrix Jensen inequality to show that for multi-block distributed ADMM, data homogeneity hurts convergence. The high level intuition lies in that the ADMM algorithm updates the dual variable by averaging over the local auxiliaries/estimators which are essentially the product of inverse of local matrices. And the dual updating has the least momentum when the local matrices are similar by matrix Jensen inequality. Hence, any data sharing that increases the difference between local matrices would boost up the momentum for dual updating. While our proof techniques depend on the linear mapping of the algorithm which is only valid when the objective is quadratic under least square regression, in the numerical experiments, we found that data-sharing also helps logistic regression. As whether multi-block ADMM would still have linear convergence rate for general objectives is still an open problem (Han et al. (2018), Liu et al. (2018)), we decide to focus on the regime of direct comparing the linear convergence rate under quadratic objectives in this paper, and put the analysis with general convex objectives in future works. Numerically, we contribute to previous streams of literature by presenting a novel idea of creating a data pool with a small amount of prefixed shared data across local agents, and utilizing this data pool in each iteration later. We find that such approach is effective for both least square and logistic machine learning tasks.
This paper is organized as follows. Section 3 answers when and why multi-block ADMM and PCG methods suffer from slow convergence. We show that while data heterogeneity hurts PCG methods, it benefits multi-block ADMM methods under certain regime. Section 4 provides the design of a meta-algorithm of data sharing, with tailored application to multi-block ADMM and PCG methods. We further apply the tailored algorithms to least square and logistic regressions with both synthetic and public benchmark data sets (UCI machine learning regression repository [Dua and Graff (2017)]). Numerically, we show that with 5% of data share, our algorithms converge ten or hundreds more times faster than those without data sharing. Section 5 concludes our work, provides future research directions, and echoes on encouraging even small amount of data sharing among different regions to combat difficult global learning problems.

3. Theory

In this paper, we mainly focus on multi-block distributed Alternating Direction Method of Multipliers (ADMM) and preconditioned conjugate gradient methods (PCG), which are both known as the algorithms in between the first order gradient descent method and the second order Newton’s methods. Specifically, we consider the following distributed statistical learning problem. Let \((x_j \in \mathbb{R}^{1 \times p}, y_j \in \mathbb{R})\) be the model data and dependent variable pair. The data pair \((x_j, y_j)\) locates at different local agents. We assume there are \(b\) agents, each of the agent possesses \(s_i\) numbers of observations \((X_i = [x_{1i}; \ldots; x_{si}] \in \mathbb{R}^{s_i \times p}, y_i = [y_{1i}; \ldots; y_{si}] \in \mathbb{R}^{s_i \times 1})\), with \(i \in \{1, \ldots, b\}\), with \((x_{i,j}, y_{i,j})\) being the \(j^{th}\) data pair associated with \(i^{th}\) agent. Let the total number of observations \(n = \sum_{i=1}^{b} s_i\).

We denote \(X = [X_1; \ldots; X_b] \in \mathbb{R}^{n \times p}\), and \(y = [y_1; \ldots; y_b]\) as the aggregate model data matrix and dependent variable vector. In this paper, we consider the case where centralized optimizing by uploading all local data to one machine is not feasible, which implies that one cannot directly operates with \((X, y)\). This is common in practice when either the data communication across agents is costly, or there is privacy/security constraints associated with certain data sets.

A decision maker tries to find \(\beta \in \mathbb{R}^p\) that minimizes the following global loss function

\[
F((X, y); \beta) = \sum_{i=1}^{b} \sum_{j=1}^{s_i} f((x_{i,j}, y_{i,j}); \beta)
\]  

where \(f((x, y); \beta)\) is the regression model choice of the decision maker. Several commonly used regression models include least squares regression \(f((x, y); \beta) = \|x\beta - y\|_2^2\), ridge \(f((x, y); \beta) = \|x\beta - y\|_2^2 + \alpha \|\beta\|_2^2\), lasso \(f((x, y); \beta) = \|x\beta - y\|_2^2 + \alpha \|\beta\|_1\), elastic net \(f((x, y) = \|x\beta - y\|_2^2 + \alpha \|\beta\|_1 + (1 - \alpha) \|\beta\|_2^2\), and logistic regression \(f((x, y); \beta) = log(1 + exp(-yx\beta))\).
3.1. Application in distributed multi-block distributed ADMM method

Consider the following classic formulation to solve problem (1) with distributed ADMM by introducing auxiliary $\beta_i$ to the optimization problem for each local agent.

$$\min_b \sum_{i=1}^{b} \sum_{j=1}^{s_i} f((x_{i,j}, y_{i,j}); \beta_i)$$
$$s.t. \quad \beta_i - \beta = 0 \quad \forall \ i = 1, \ldots, b$$

(2)

To apply the primal distributed ADMM algorithm, the decision maker solves the relaxed augmented Lagrangian problem. Let $\lambda_{t,i}$ be the dual with respect to the constraint $\beta_i - \beta = 0$, and $\rho_p$ the step size to the primal distributed ADMM. The augmented Lagrangian is thus given by

$$L(\beta_i, \beta, \lambda_t) = \sum_{i=1}^{b} \sum_{j=1}^{s_i} f((x_{i,j}, y_{i,j}); \beta_i) + \sum_{i=1}^{b} \lambda_{t,i}^T (\beta_i - \beta) + \sum_{i=1}^{b} \frac{\rho_p}{2} (\beta_i - \beta)^T (\beta_i - \beta)$$

(3)

The algorithm of primal distributed ADMM is as follows.

**Algorithm 1** Primal Distributed ADMM for solving (2)

**Initialization**: $t = 0$, step size $\rho_p \in \mathbb{R}^+$, $\beta_t \in \mathbb{R}^p$, $\lambda_{t,i} \in \mathbb{R}^p$, $\beta_{t,i} \in \mathbb{R}^p$ for all $i \in \{1, \ldots, b\}$, and stopping rule $\tau$

**while** $t \leq \tau$ **do**

Each local agent $i$ updates $\beta_{t+1,i}$ in parallel by

$$\beta_{t+1,i} = \arg \min_{\beta_i \in \mathbb{R}^p} \sum_{j=1}^{s_i} f((x_{i,j}, y_{i,j}); \beta_i) + \lambda_{t,i}^T (\beta_i - \beta') + \frac{\rho_p}{2} (\beta_i - \beta)^T (\beta_i - \beta')$$

Decision maker updates

$$\beta_{t+1} = \frac{1}{b} \sum_{i=1}^{b} \beta_{t+1,i} + \frac{1}{b \rho_p} \sum_{i=1}^{b} \lambda_{t,i} \lambda_{t+1,i} - \lambda_{t,i} + \rho_p (\beta_{t+1,i} - \beta_{t+1})$$

**end while**

**Output**: $\beta_{\tau}$ as global estimator

Specifically, when applying to least square regression, the problem becomes a quadratic optimization with linear constraints. In this section, we focus on providing the theory for the least square regression.

$$\min_{\beta_i, \beta} \sum_{i=1}^{b} \frac{1}{2} (x_i \beta_i - y_i)^T (x_i \beta_i - y_i)$$
$$s.t. \quad \beta_i - \beta = 0 \quad \forall \ i$$

(4)

While many studies have already studied the impact of global matrix conditioning [Eckstein and Bertsekas (1992b), Iutzeler et al. (2015), Makhdoumi and Ozdaglar (2017)], our work is the first to study how local data structure impacts the convergence speed for distributed ADMM. In order to focus on the impact of local data structure, we first introduce the following assumption on the
global data matrix $\mathbf{X}$. We assume the global covariance matrix $\mathbf{X}^T \mathbf{X}$ has fixed conditioning number. Without loss of generality, we further assume that the global covariance matrix is normalized by its Frobenius norm, hence the largest eigenvalue of the global covariance matrix is smaller than 1\(^1\). Fixing the global matrix conditioning allows us to purely focus on the effect on how data structure influences convergence. Also it’s worth noticing that any data share across local centers would not influence the conditioning of the global covariance matrix. However, data sharing still significantly changes the local data structure, hence, influencing the convergence speed.

**Assumption 3.1.** The regressor matrix $\mathbf{X}$ is normalized by its Frobenius norm $\|\mathbf{X}\|_F$, and the smallest and largest eigenvalue of $\mathbf{X}^T \mathbf{X}$, $\underline{q}$ and $\bar{q}$ are fixed, with $\mathbf{X}_i^T \mathbf{X}_i > 0$ for all $i \in \{1, \ldots, b\}$.

Before introducing the main theorem, we provides an illustrating example on showing how data structure influences the convergence rate under the case matrix conditioning $\underline{q}$ and $\bar{q}$ fixed. Consider a simple case with feature dimensionality $p = 1$, number of agents $b = 2$ and step size $\rho_p = 1$. In the first scenario, the original model matrix $\tilde{\mathbf{X}}_i$ and the model matrix after normalizing by Frobenius norm, $\mathbf{X}_i$ are given by

$$
\begin{align*}
\tilde{\mathbf{X}}_1 &= \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix}, & \tilde{\mathbf{X}}_2 &= \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, & \mathbf{X}_1 &= \begin{bmatrix} 0.7379 \\ 0.0075 \end{bmatrix}, & \mathbf{X}_2 &= \begin{bmatrix} 0.6708 \\ 0.0745 \end{bmatrix} \quad (5)
\end{align*}
$$

Here, as feature dimensionality $p = 1$, $\bar{q} = \underline{q} = 1$. If we directly applying Algorithm 1, the linear convergence rate, which is also the spectrum of the linear mapping matrix, is given by 0.6661. However, consider the second scenario, where we just swap the distribution of data without changing model matrix condition $\underline{q}$ and $\bar{q}$.

$$
\begin{align*}
\tilde{\mathbf{X}}_1 &= \begin{bmatrix} 0.99 \\ 0.9 \end{bmatrix}, & \tilde{\mathbf{X}}_2 &= \begin{bmatrix} 0.01 \\ 0.1 \end{bmatrix}, & \mathbf{X}_1 &= \begin{bmatrix} 0.7379 \\ 0.6708 \end{bmatrix}, & \mathbf{X}_2 &= \begin{bmatrix} 0.0075 \\ 0.0745 \end{bmatrix} \quad (6)
\end{align*}
$$

The convergence rate of applying Algorithm 1 is given by 0.5264. In fact, one could show that, for primal distributed ADMM the worst case convergence rate is 0.6667 and the best convergence rate one could achieve is 0.5 with $b = 2$ and feature dimensionality $p = 1$. Here, data structure significantly influences on the convergence rate of primal distributed ADMM from near worst convergence rate to near optimal convergence rate. From this illustrating example, we notice that if different agent shares similar model covariance matrix $\mathbf{D}_i = \mathbf{X}_i^T \mathbf{X}_i$, primal distributed ADMM suffers from convergence.

\(^1\)In the proof provided in appendix, one would notice that normalizing the global covariance matrix is similar as adjusting the step size $\rho_p$. Since we allow step size to vary across $(0, \infty)$, normalizing is without loss of generality.
We further generalize the previous example. Let \( D_i = X_i^T X_i \), and \( c_i = -X_i^T y_i \), following algorithm 1, the updating process becomes

\[
(\rho P + D_i)\beta_{t+1}^i = \rho_p \beta^i - \lambda^i - c_i
\]

\[
\beta_{t+1}^i = \frac{1}{b} \sum_i \beta_{t+1}^i + \frac{1}{bp_p} \sum_i \lambda^i
\]

\[
\lambda_{t+1}^i = \lambda^i + \rho_p (\beta_{t+1}^i - \beta_{t}^i)
\]

(7)

The process in (7) could be represented as a linear mapping system. Following Iutzeler et al. (2015), \( \xi_{k+1} = \rho_p \bar{\beta}_{k+1} + \bar{\lambda}^k \) is sufficient to represent the system dynamics, where \( \bar{\beta}^{k+1} = [\beta_{1}^{k+1}; \ldots; \beta_{i}^{k+1}; \ldots; \beta_{b}^{k+1}] \in \mathbb{R}^{bp \times 1} \), and \( \bar{\lambda}^k = [\lambda_{1}^k; \ldots; \lambda_{i}^k; \ldots; \lambda_{b}^k] \in \mathbb{R}^{bp \times 1} \). The mapping matrix of the primal distributed ADMM is given by

\[
M_p = \left( I - P - \left( I + \frac{D}{\rho_p} \right)^{-1} \right) (I - 2P)
\]

(8)

where \( P = b^{-1}1_b 1_b^T \otimes I_p \) is the projection matrix. Here \( 1_b \) is size \( b \times 1 \) all one vector; \( I_p \) is size \( p \times p \) identity matrix; \( D = \text{diag}([D_1, \ldots, D_i, \ldots, D_b]) \) is the block diagonal matrix of \( D_i \); and \( \otimes \) represents the tensor product of \( b^{-1}1_b 1_b^T \) and \( I_p \).

As shown before, for least square regression the primal distributed ADMM uses \( M_p \) for linear updating. Since we are comparing algorithms under linear convergence regime, we define the convergence rate of primal distributed ADMM as the spectral radius of linear mapping matrix \( M_p \), \( \rho(M_p) \). From previous literature, primal distributed ADMM converges for any constant step size and \( \rho(M_p) < 1 \) (Eckstein and Bertsekas (1992b)). The data structure that leads to worst convergence rate depends on the choice of step size \( \rho_p \). In the following theorem, we provides the results when step size is relatively large.

**Theorem 3.2.** For \( \rho_p > \bar{q} \), the linear convergence rate of distributed ADMM is upper bounded by \( \frac{bp_p^{bp}}{bp_p + \bar{q}} \), and the upper bound is achieved when \( X_i^T X_i = X_j^T X_j \) for all \( i, j \in \{1, \ldots, b\} \).

Theorem 3.2 provides the tight upper bound on the convergence rate of distributed ADMM. To our knowledge, this is the first result on providing how data structure influences the convergence rate, and provides tight upper bound on the convergence rate of primal distributed ADMM with fixed model matrix conditioning \( \bar{q} \) and \( q \). Surprisingly, data homogeneity leads to worst data structure for higher order optimization algorithms, which challenges the conventional belief on data heterogeneity hurts convergence for distributed learning algorithms. The detailed proof is provided in appendix A.1. We present the sketch of proof here for high level intuition. To show that such convergence rate attains upper bound, we first show that the eigenvalue of \( M_p \) is real, which is
not trivial as \( M_p \) is non-symmetric. With this, one could further apply matrix Jensen equality to prove Theorem 3.2. The key proof lies in the intrinsic updating rules of distributed ADMM that involves taking average of local auxiliary variables when updating the global variable \( \beta \) and the dual variables \( \lambda_i \). And updating each local auxiliary variable is based on taking the inverse of the local covariance matrix. When \( X_i^T X_i \) and \( X_j^T X_j \) are closer, averaging the inverse of local covariance matrices provides smaller momentum on pushing the dual variables to converge to the KKT point.

The following corollary shows the benefit of data sharing under data homogeneity.

**Corollary 3.2.1.** When \( X_i^T X_i = X_j^T X_j \) for all \( i, j \in \{1, \ldots, b\} \), any data share that changes the local covariance matrix strictly improves the linear convergence rate of distributed ADMM.

It’s worth to have some further discussions on Theorem 3.2. Firstly, notice that the upper bound on convergence speed is increasing with respect to number of data centers \( b \), and decreasing with \( q \). These two effects are more intuitively direct, as more number of centers hurts convergence, and ill conditioning of model matrix also hurts convergence. More importantly, from Theorem 3.2 we know data homogeneity hurts convergence for higher order optimization algorithms as distributed ADMM. However, one popular technique in machine learning is data augmentation, where the decision maker creates synthetic agents by augmenting existing data through scaling, flipping, or rotation, to name a few. Such data augmentation technique tends to increase data homogeneity across agents. While data homogeneity may benefits the first order algorithms [Wang et al. (2020), Li et al. (2022)], it hurts higher order distributed primal-dual optimization algorithms. And one needs to carefully apply data augmentation together with higher order optimization algorithms.

Theorem 3.2 provides bounds on convergence rate and worst case data structure when step size is relatively large. We introduce the following theorem when step size is relatively small and \( b = 2 \). Let \( \lambda(D_i) \) be the smallest eigenvalue of \( D_i \), and \( q_1 \) be the smallest eigenvalue among all \( D_i \). From Assumption 3.1, \( q_1 > 0 \). The following theorem on worst case data structure and upper bound on convergence rate holds for \( b = 2 \).

**Theorem 3.3.** For \( \rho_p < q_1 \) and \( b = 2 \), the convergence rate of distributed ADMM is upper bounded by \( \frac{q_1}{2 \rho_p + \bar{q}} \), and the upper bound is achieved when \( X_1^T X_1 = X_2^T X_2 \).

Theorem 3.3 provides the worst case data structure when \( b = 2 \). The detailed proof is provided in appendix A.2. The worst case data structure is similar as in Theorem 3.2. When \( b = 2 \), data homogeneity still hurts convergence. However, for \( b > 2 \), with relative small step size \( \rho_p < q_1 \), the
system becomes more complex. Firstly, the data structure with $D_i = D_j$ for all $i, j \in \{1, \ldots, b\}$ does not provide an upper bound for the convergence rate. In appendix, we show that under the structure where $D_i = D_j$ for all $i, j \in \{1, \ldots, b\}$ and $\rho_p < q_1$, the convergence rate is given by $\frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}$, which decreases as number of blocks increases. This suggests that the data structure of $D_i = D_j$ may not be the worst case data structure when $b > 2$. In the following proposition, we provide an upper bound on the convergence rate for smaller step size $\rho_p$ and $b > 2$.

**Proposition 3.4.** For $\rho_p < q_1$, the convergence rate of distributed ADMM is upper bounded by $\frac{\bar{q}}{\rho_p + \bar{q}}$, and there exists a data structure that provides convergence rate of $\frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}$.

The proving technique used in Proposition 3.4 is different from Theorem 3.2. Instead, we use the classic bound on $\rho(M) \leq \rho(\frac{1}{2}M^T + \frac{1}{2}M)$ for non-symmetric matrix. Hence, the bound is not tight. The data structure that provides convergence rate of $\frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}$ tries to mimic the worst case data structure under two block distributed ADMM by having two agents possessing the majority of data with similar structures, and the rest of agents possessing minimum data to ensure that local covariance matrices have strictly positive eigenvalues. And the convergence rate of the data structure provided in Proposition 3.4 is strictly higher compared with the data structure of $D_i = D_j$ for all $i, j \in \{1, \ldots, b\}$.

Lastly, we compare our results with other common optimization method that is also widely used in learning across local data centers, first-order method of gradient descent. Here, we assume the decision maker could perform a synchronized aggregation scheme with local full gradient information, which differs from the prevailing federated learning algorithms. Nevertheless, updating with synchronized full gradients provides an upper bound for federated learning algorithms, where federated learning uses sampled gradient with asynchronous updating. While distributed ADMM converges for any constant step size, gradient descent method converges only when step size is within some specific range. For least square regression, the convergence rate is given by the spectrum of matrix $M_{GD} = (I - \rho_p X^T X)$, where $\rho_p$ is the step size. We provide the following corollary on comparing the convergence rate between gradient descent and distributed ADMM for least square regression under same choice of step size. From proposition 3.5, the convergence rate of gradient descent method is better than the convergence rate of distributed ADMM for only a specific sweet range of step size.

**Proposition 3.5.** For $\rho_p \in (0, s_1) \cup (s_2, \infty)$, $\rho(M_p) < \rho(M_{GD})$, where

$$s_1 = \min \left( \frac{1}{q} - \bar{q}, q_1 \right), \quad s_2 = \frac{2b - \bar{q}q + \sqrt{4b^2 + (\bar{q}q)^2}}{2b\bar{q}}.$$
From proposition 3.5, firstly, we are convinced that higher order optimization algorithms like distributed ADMM have its merits in faster and robust convergence. As gradient descent method is not guaranteed to converge for any choice of step size. Secondly, it’s not surprising that choosing a good step size is crucial for faster convergence, for all algorithms. However, our result shows that distributed ADMM method outperforms gradient method for a large range of step size choices. This implies the first order method is more sensitive to the parameter selection, and the first order method could only perform well within a small range of “sweet-spot” step size.

3.2. Application in PCG method

In this section, we consider using PCG method for solving problem (4), which is equivalent as solving the following linear system with distributed conjugate gradient method.

\[ \sum_{i=1}^{b} X_i^T X_i = \sum_{i=1}^{b} X_i^T y_i \]  

The algorithm of conjugate gradient method with distributed preconditioners is as follows.

**Algorithm 2** Conjugate gradient method with local preconditioning

**Initialization:** \( t = 0 \), \( \beta_0 = r_0^T y \); local pre-conditioner matrices \( H_i \), \( p_0 = H_i r_0^T \) stopping rule \( \tau \)

Decision maker receives \( r_0^T, p_0^T \), calculates \( r_0 = \sum_i r_0^T, p_0 = \sum_i p_0^T \) and broadcasts it to each agent

\[ \text{while } t \leq \tau \text{ do} \]

\[ \text{Each agent calculates } r_k^T H_i r_k \text{ and } p_k^T X_i^T X_i p_k \text{ and sends to decision maker} \]

Decision maker updates \( \alpha_k = \frac{\sum_i r_k^T H_i r_k}{\sum_i p_k^T X_i^T X_i p_k} \) and sends to each agent

Each agent calculates \( \alpha_k X_i^T X_i p_k \) and sends to decision maker

Decision maker updates \( \beta_{k+1} = \beta_k + \alpha_k p_k, \ r_{k+1} = r_k - \sum_i \alpha_k X_i^T X_i p_k \) and sends to each agent

Each agent calculates \( H_i r_{k+1} \), \( r_{k+1}^T H_i r_{k+1} \) and sends to decision maker

Decision maker updates \( t_k = \frac{\sum_i r_{k+1}^T H_i r_{k+1}}{\sum_i r_k^T H_i r_k} \), \( p_{k+1} = \sum_i H_i r_{k+1} + t_k p_k \) and sends to each agent

\[ \text{end while} \]

**Output:** \( \beta_\tau \) as global estimator

From the algorithm description, conjugate gradient with local preconditioning takes more communications between agents and the decision maker compared with distributed ADMM algorithm, which may not be ideal, if the cost of communication between agents and decision maker is high. In the meantime, local preconditioning performs good if the sum of the local preconditioning matrix \( H = \sum_i H_i \) is a good approximation of \( (X^T X)^{-1} \), where \( X^T X = \sum_i X_i^T X_i \). However, without data
sharing, each of the agent could only determine $H_i$ based on local information of $X_i^T X_i$, and without data sharing, approximating $X^T X$ with only local information is hard. However, if one can share sampled data that is representative to the local agent’s data distribution, the local preconditioning with data sharing could perform much better.

We first introduce an approach on constructing local pre-conditioners with shared data, and provide a theoretical improvement on convergence rate for preconditioned conjugate gradient (PCG) with data sharing. We further show that preconditioning is helpful for faster convergence with numerical experiments. Let $X_{s_i} \in \mathbb{R}^{r_i \times p}$ be the selected shared data from agent $i$ with $X_i \in \mathbb{R}^{s \times p}$. We construct local preconditioning matrix with global data share as $H_i^g = \left( \frac{1}{b_i} X_i^T X_i + \sum_{j \neq i} \frac{1}{b_j} X_j^T X_j \right)^{-1}$. Intuitively, $H_i$ tries to create an estimate of $(X^T X)^{-1}$ with the information from the sampled data. We compare the convergence rate of local preconditioning with shared data to conjugate gradient without preconditioning, and conjugate gradient with local preconditioning of $H_i = (X_i^T X_i)^{-1}$. The following example shows that data sharing is essential for preconditioning.

Proposition 3.6. When solving the linear system $Ax = b$, where $A = \sum_{i=1}^b A_i$, there exists data structure such that the condition number of $HA$ could be arbitrarily large if we construct local preconditioning matrix $H$ without data share; while with data share, the condition number of $HA$ decreases and converges to 1.

Consider the case where we have two agents, both of them possess $s$ observations with feature dimension $p = 2$. Agent 1 possess $(X_1, Y_1)$, and $X_1 = [x_1^1; \ldots; x_1^1; \ldots; x_1^s]$ with $x_1^i = \frac{1}{\sqrt{s}} (1, \xi_i)$. Agent 2 possess $(X_2, Y_2)$, and $X_2 = [x_2^1; \ldots; x_2^1; \ldots; x_2^s]$ with $x_2^i = \frac{1}{\sqrt{s}} (1, \xi_i)$. $\xi_i$ and $\xi_j$ are i.i.d. Gaussian random variables $\epsilon_1 N(0,1)$ and $\epsilon_2 N(0,1)$. In order to perform least square regression, one needs to solve the linear system of $\sum_{i=1}^b A_i = b$, where

$$A_1 = X_1^T X_1 = \begin{bmatrix} 1 & a_1 \\ a_1 & b_1 \end{bmatrix} \quad A_2 = X_2^T X_1 = \begin{bmatrix} 1 & a_2 \\ a_2 & b_2 \end{bmatrix} \quad b = \sum_{i=1}^b X_i^T y_i$$

(10)

with $a_1, a_2$ following gaussian distribution $\frac{1}{\sqrt{b_1}} N(0,1)$, $\frac{1}{\sqrt{b_2}} N(0,1)$ respectively, and $b_1$, $b_2$ following chi-squared distribution $\frac{1}{2} \chi_b$ and $\frac{1}{2} \chi_b$ respectively. As the number of observations $s$ increases, $A_1$ and $A_2$ converges to

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon_1^2 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon_2^2 \end{bmatrix}$$

(11)

Let $\epsilon_2 = \frac{1}{\epsilon_1}$, and take $\epsilon_1$ to be small enough, without data sharing, the local preconditioning matrix at agent 1 is given by $H_1 = (X_1^T X_1)^{-1}$, and the conditioning number of $H_1 A$ is $\frac{2+1}{2\epsilon_1^2}$. The local preconditioning matrix at agent 2 is given by $H_2 = (X_2^T X_2)^{-1}$, and the conditioning number of $H_2 A$ is $\frac{2}{1+\epsilon_1^2}$. Simply apply local preconditioning provides the aggregated preconditioning matrix
of $H_{local} = H_1 + H_2$, and the conditioning number of $H_{local}A$ is $\frac{(\epsilon^2 + 1)^2}{4\epsilon^2}$. However, with any linear fraction amount of data share, if we construct local preconditioning matrix with global data share as $H^g_i = (\frac{1}{b}X_i^T X_i + \sum_{j \neq i \neq r} \frac{1}{\sigma_j} X_{\sigma_j}^T X_{\sigma_j})^{-1}$, with $H_{global} = \sum_{i=1}^b H^g_i$, one could show that as $s$ increases, the conditioning number of $H_{global}A$ converges to 1. In the numerical results, we assume local agents have heterogeneous data sources. We implement PCG method with local preconditioners and global preconditioners with data sharing, and we compared the result with conjugate gradient method without preconditioning. Under PCG method, data heterogeneity hurts convergence. However, a small amount of data sharing helps providing a much better preconditioner, which mitigates the challenge of data heterogeneity.

To summarize the theory section, we find that data heterogeneity has complex effect on algorithm convergence rate. While under PCG method, data heterogeneity hurts convergence, under multi-block distributed ADMM method, data heterogeneity benefits convergence. This result highlights the need for a universal approach on altering data distribution for different algorithms. In the following section, we introduce a meta data-sharing algorithm, and provide numerical evidence on how data sharing could significantly improve convergence speed for different higher order algorithms.

4. Algorithms Design and Numerical Results

4.1. Data sharing algorithm

In this section, we describe the sampling procedure to enable data sharing across local agents. The meta-algorithm of data sharing is simple and easy to implement – it samples $\alpha b\%$ of data uniform randomly from each agents, and builds a global data pool with the total $\alpha b\%$ prefixed sampled data. The benefit of having a global data pool is two-folded – (a) it allows the decision maker to have the freedom on changing the local data structure; (b) it allows the decision maker to have an unbiased sketch of the global higher order information of the objective function, e.g., the sketch of Hessian information.

**Algorithm 3** Meta-algorithm of data sharing

**Sampling Procedure**: Randomly sample $\frac{\alpha}{b}\%$ of data pairs from each agent one-time

Let $r \in \mathbb{Z}^{m \times 1}$ be the index of selected data ($m = \lfloor \alpha \% n \rfloor$). Let $r_i$ be the index of selected data at agent $i$, and $l_i$ be the index of data remains local at agent $i$.

**Output**: global data pool $(X_r, y_r) = ([X_{r_1}; \ldots; X_{r_b}]; [y_{r_1}; \ldots; y_{r_b}])$ with shared access to all local agents.

With the meta-algorithm of data sharing and the global data pool, now the distributed optimization algorithms have the access to a sketch of the global data. From numerical evidence, we are
convinced that we only need a small amount of data share to improve the convergence speed. Setting \( \alpha \) at a low level also allows us to enjoy the benefits from distributed optimization with parallel computation. As the majority of data still remains at local, the algorithm could take advantages from such structure. For example, the algorithm could pre-factorize the local data observation matrix for faster computation. Hence, without specification, we fix \( \alpha \) to be 5% across numerical experiments. After we decide on the desirable level of data share, for each distributed optimization algorithm, we still need to carefully design how the algorithm should utilize the global data pool efficiently with theory guidance. In the next section, we implement the algorithms with careful and tailored design on utilizing data sharing for multi-block ADMM and PCG method.

The numerical result section is organized as follows. Section 4.2 provides the algorithm design for multi-block ADMM with data-sharing. We further show the numerical results for both the least square regression and the logistic regression, and compare our performance with distributed multi-block ADMM without data sharing, together with other variants of multi-block ADMM algorithms. Section 4.3 provides the algorithm design for PCG method with data sharing.

### 4.2. Apply data sharing in multi-block ADMM methods

In this section, we present results on both the least square regression and logistic regression. For least square regression, we test the algorithms on the benchmark of UCI machine learning repository regression data (Dua and Graff (2017)). And we compare the absolute loss among different algorithms, with absolute loss \( AL = \| \beta^* - \hat{\beta} \|_2 \), where \( \beta^* \) is the optimal estimator and \( \hat{\beta} \) is the estimator produced by each algorithm. We further add a L2 regularization for all regression problems in order to guarantee the uniqueness of \( \beta^* \). For logistic regression, we generate the synthetic data with Gaussian noise and the ground truth estimator \( \beta^* \). We further compare the absolute loss \( AL = \| \beta^* - \hat{\beta} \|_2 \), where \( \beta^* \) is the optimal estimator and \( \hat{\beta} \) is the estimator produced by each algorithm.

Firstly, from previous result, we know that the worst case data structure for distributed ADMM depends on the relations between the step size and the local data matrix conditioning. And making the local data structure differ from each other would improve the convergence speed. Hence, a simple and direct way to improve the performance of distributed ADMM is to allocate all the global data pool to one existing agent/block, in order to make that block have different data structure from others. We test the modified distributed ADMM with global data, and compare it with classic distributed ADMM in UCI machine learning repository regression data. With fixed number of iteration equals to 200, block number equals to 4, and percentage of sample \( \alpha = 5\% \), the accuracy

\(^2\) All the codes in Section 4 are public available at https://github.com/mingxiz/data_sharing_matlab.
of estimator $\beta$ improves for 13 out of 14 problem instances. Besides, compared with the distributed ADMM without data sharing, in average distributed ADMM with data sharing decreases the absolute loss by 20%.

However, 20% decrease in absolute loss is not satisfying. Here, we carefully design a randomized multi-block ADMM algorithm to better utilize the global data pool. We further introduce a tailored multi-block ADMM algorithm – the Dual Randomly Assembled and Permuted ADMM (DRAP-ADMM). We use the least square regression as an example to illustrate the idea of DRAP-ADMM. One could design DRAP-ADMM for different loss functions by introducing the conjugate functions.

Introducing the auxiliary $\zeta$, the primal problem could also be formulated as

$$\min_{\zeta} \frac{1}{2} \zeta^T \zeta \quad s.t. \quad X\beta - y = \zeta \quad (12)$$

And let $t$ be the dual variables with respect to the primal constraints $X\beta - y = \zeta$. Taking the dual with respect to problem (12), we have

$$\min_t \frac{1}{2} t^T t + y^T t \quad s.t. \quad X^T t = 0 \quad (13)$$

The augmented Lagrangian is thus given by

$$L(t, \beta) = \frac{1}{2} t^T t + y^T t - \beta^T X t + \frac{\rho_d}{2} t^T X X^T t \quad (14)$$

The global estimator $\beta$ is the dual variable with respect to the constraint $X^T t = 0$ and $\rho_d$ be the step-size of dual problem. The reason we take dual is that, the dual variables $t$ serves as a label for each (potentially) exchanged data pair, and the randomization is more effective in the dual space. We show that simply taking the dual does not improve the convergence speed. The following proposition guarantees that, the primal distributed algorithm and dual distributed algorithm are exactly the same in terms of computation and convergence rate. The proof is provided in appendix A.5.

**Proposition 4.1.** The primal distributed ADMM algorithm and the dual distributed ADMM algorithm have exactly the same linear convergence rate if the step size for primal and dual algorithms satisfies $\rho_p \rho_d = 1$ when applied to the least square regression under the partition of blocks with $X = [X_1; \ldots; X_b]$ and $y = [y_1; \ldots; y_b]$. 

Proposition 4.1 states that by only taking the dual of the problem would not impact the convergence speed, and more delicate design on the algorithms is required to enjoy the benefit of data sharing. Hence, we design the Dual Randomly Assembled and Permuted ADMM (DRAP-ADMM) under the inspiration of Mihić et al. (2020). DRAP-ADMM updates the auxiliary variables following a random permuted order across each agent. We present the high level idea on how we utilize the global data pool as follows. Firstly, the meta-algorithm of data sharing randomly selects a subset of data $(X_{r_i}, y_{r_i})$ from agent $i$, and builds the global data pool $(X_{\cdot r}, y_{\cdot r})$. When designing the algorithm, each agent $i$ may first pre-compute and pre-factorize $X_l^T X_l$ in order to enjoy the benefit of the distributed structure, where $X_l$ is the local data at agent $i$. Then, at each iteration, each agent $i$ receives a random sample (without replacement) from the global data pool. The size of received data from global data pool is the same as the size that the agent $i$ contributes to the pool initially. Finally, we random permute the update order across agents at each iteration. The general algorithm of DRAP-ADMM is provided in Algorithm 4.

**Algorithm 4 DRAP-ADMM**

**Initialization**: $t = 0$, global data pool $(X_{r}, y_{r})$, step size $\rho_d \in \mathbb{R}^+$ $t_i \in \mathbb{R}^n$, $\beta_t \in \mathbb{R}^p$, and stopping rule $\tau$

**while** $t \leq \tau$ **do**

Random permute $r$ to $\sigma_t(r)$, partition $\sigma_t(r) = [\sigma^1_t(r); \ldots; \sigma^b_t(r)]$ according to $|r_i|$ (size of $r_i$).

Random permute the block-wise update order $\xi_t(b) = [\xi^1_t, \ldots, \xi^b_t]$

For $i = \xi^1_t, \ldots, \xi^b_t$

Let $\sigma_i^t = l_i \cup \sigma(r)$. Agent $i$ updates $t_i^{\sigma_i^t} = \arg \min_{B_{r_i} \in [r_i]} L(t_i^{\sigma_i^t}, \ldots, t_i^{\sigma_i^{i-1}}, t_i, t_i^{\sigma_i^{i+1}}, \ldots, t_i^{\sigma_i^b}, \beta_t)$

Decision maker updates $\beta_{t+1} = \beta_t - \rho_d X_t^T t_{t+1}$

**end while**

**Output**: $\beta_\tau$ as global estimator

As the majority of data still remains local, DRAP-ADMM could both enjoy the benefit of pre-factorization for solving local training tasks, and utilize the shared data to change the local data structure constantly, in order to avoid an unfavorable data structure that leads to slow convergence. In order to better utilize the global data pool, we use random permuted updating order instead of distributed updating order. In appendix A.6, we show the benefit of having a random permute updating – it improves the convergence rate under the worst case data structure compared with distributed updating scheme. Further, a random assemble of local blocks with global data pool would further help improve the convergence speed, and following a similar proof in Sun et al. (2015)
and Milić et al. (2020), one could show that DRAP-ADMM converges in expectation for linearly constrained quadratic optimization problems.

There are several other variants of multi-block ADMM algorithms, including the symmetric Gauss-Seidel multi-block ADMM (double-sweep ADMM) [Floudas and Pardalos (2008), Xiao et al. (2019)] and the random-permuted ADMM [Sun et al. (2020)]. In the following numerical experiments provided in Table 1, we use UCI machine learning regression data Dua and Graff (2017) to first compare the performance of DRAP-ADMM with (1) primal distributed ADMM, (2) double-sweep ADMM, (3) cyclic-ADMM and (4) RP-ADMM. Besides, since the meta-algorithm of data sharing could also be applied to primal-distributed ADMM, double-sweep ADMM, and RP-ADMM in a similar way as in DRAP-ADMM, we also compare the DRAP-ADMM with (5) primal distributed ADMM with data-share, (6) double-sweep ADMM with data-share, and (7) RP-ADMM with data-share. We fix step-size to be $\rho = 1$ for both the primal algorithms and the dual algorithms in order to eliminate the effect of step-size choices. And we set the percentage of shared data $\alpha = 5\%$. The data set has dimensionality of $n = 463,715$ and $p = 90$. From this set of experiments, we are convinced that, firstly, the performance of multi-block ADMM algorithm significantly improves with only a small amount of data share. Secondly, the random permute updating order seems to be the most compatible algorithm to a small amount of data-sharing, compared with other multi-block updating orders.

| Algorithm                           | Fix run time = 100 s | Fix number of iteration = 200 |
|-------------------------------------|----------------------|-------------------------------|
| Primal Distributed ADMM            | $2.98 \times 10^{-3}$| $4.10 \times 10^{-2}$        |
| Double-Sweep ADMM                  | $5.92 \times 10^{-3}$| $3.44 \times 10^{1}$        |
| Cyclic ADMM                        | $5.66 \times 10^{-6}$| $3.44 \times 10^{4}$        |
| Random Permuted ADMM               | $6.62 \times 10^{-6}$| $3.44 \times 10^{4}$        |
| Primal Distributed ADMM with data sharing | $2.41 \times 10^{-3}$| $4.01 \times 10^{-2}$        |
| Double-Sweep ADMM with data sharing | $3.80 \times 10^{-9}$| $1.44 \times 10^{-5}$        |
| Cyclic ADMM with data sharing       | $3.07 \times 10^{-9}$| $1.13 \times 10^{-5}$        |
| DRAP-ADMM                          | $1.12 \times 10^{-9}$| $9.25 \times 10^{-6}$        |

Table 1 Absolute Loss of different multi-block ADMM algorithms for L2 regression estimation on data set Year Prediction MSD

From previous experiments, we are convinced that DRAP-ADMM performs better than the other variants of multi-block ADMM methods. In the following numerical results, we compare DRAP-ADMM with primal distributed ADMM without data sharing. Firstly, we examine the sensitivity of convergence speed to percentage of data shared. We find that, $\alpha$ does not need to be very large for significant efficiency improvement, and a small amount of data sharing is sufficient to boost convergence speed. The following figure (Figure 1) reports the sensitivity of convergence speed
to the percentage of data shared. We fix the same convergence criteria to be $\|\beta_t - \beta_{t-1}\| \leq 10^{-4}$ for all different values of $\alpha$. It’s worth mentioning that the required time for convergence with no data shared ($\alpha = 0$) is 2403.72 seconds. And in Figure 1 (a), by sharing even 1% of data, the required time to converge decreases to 221.60 seconds. Similarly, the required number of iterations for converging to the same target tolerance level with no data shared is 3952. And in Figure 1 (b), by sharing 1% of data, the required number of iterations to converge decreases to 307.

![Figure 1](image)

**Figure 1** Comparison between distributed multi-block ADMM and multi-block ADMM with data sharing.

$\alpha\% = 0$ implies sharing no data, and $\alpha\% = 1$, implies sharing all data.

As we only need a small amount of data sharing, in the following experiments, we fix $\alpha\% = 5\%$. The following table shows more numerical results we performed on UCI machine learning repository. We set number of agents to be 4. We fix the step-size $\rho_p = \rho_d = 1$ for primal distributed ADMM and DRAP-ADMM. Note setting step-size equals to 1 does not favor the primal ADMM nor the dual ADMM, as we show that the primal distributed ADMM and dual distributed ADMM share the same convergence rate when $\rho_p \rho_d = 1$. We consider two stopping rules, fixing the same number or iteration, or the same run time.

From Table 2, we observe that compared with primal distributed ADMM, a small amount of data sharing with DRAP-ADMM algorithm could attain a good quality predictor within much fewer number of iterations or time. Specifically, with 200 iterations, DRAP-ADMM significantly outperforms primal distributed ADMM. This suggests that DRAP-ADMM algorithms with a small amount of data sharing could be a good solution when the cost per iterations is high. For example, in practice, when conducting regression prediction with healthcare trial data, the decision maker (researcher) would have to present physically to each hospitals in order to perform optimization with local data. Hence, minimizing number of iteration required would be a major objective for
decision maker when performing estimation across agents. Nonetheless, we observe that DRAP-ADMM still enjoys some benefit even when we fix the run time.

For general regression analysis including logistic regression, DRAP-ADMM would still apply. The logistic regression minimizes the following objective

$$
\min \sum_{i=1}^{b} \sum_{j=1}^{n_i} \log(1 + \exp(-y_{i,j}x_{i,j}\beta))
$$

with $y_{i,j} \in \{-1, 1\}$. Similarly one could apply distributed ADMM to solve logistic regression following Algorithm 1. One need to take the conjugate function of the primal objective, and solves a different optimization problem for each agent. When designing the ADMM method for logistic regression, one could introduce the auxiliary variables in order to further improve the efficiency of the algorithm. For DRAP-ADMM, let $\mathcal{X} = y \cdot X$, we solve the following dual problem

$$
\min \sum_{i=1}^{b} \sum_{j=1}^{n_i} t_{i,j} \log(t_{i,j}) + (1 - t_{i,j}) \log(1 - t_{i,j})
\text{s.t.} \quad \mathcal{X}^T z = 0 \quad \cdots \beta
\quad t - z = 0 \quad \cdots \xi
$$

In Table 3, we present the result on comparing the performances across gradient descent method (with backtracking step-size), primal distributed ADMM and DRAP-ADMM (with step-size equals to 1). A widely used algorithm for solving logistic problem is via Newton’s method. To further compare the algorithms, we select the benchmark algorithm to be the Newton’s method. We need to point out here that the classic Newton’s method requires centralized learning and optimization,
which is not the focus of this paper. Nonetheless, we use centralized Newton’s method as the benchmark, and show that distributed optimization with data sharing could outperform centralized optimization method in aspect of convergence rate. Similarly, we fix $\alpha = 5\%$. We report the relative ratio in the absolute loss with benchmark of centralized Newton’s method. The relative ratio in the absolute loss is defined as $r_{AL} = \frac{AL_{AL} - AL_{newton}}{AL_{newton}}$. We fix block numbers equals to 4 and the number of iterations to be 10 for all the different algorithms. We expect the Newton’s method to perform well and a positive relative ratio of $r_{AL}$ is not surprising, as we allow the Newton’s method to perform centralized optimization. However, notice that $r_{AL} < 0$ implies that under fixed iteration, the algorithm outperforms centralized Newton’s method with smaller absolute value. We report the average of relative ratio in the absolute loss for each size of problem instances with 20 sample of experiments.

| $n$  | $p$  | Gradient Descent | Primal Distributed | DRAP-ADMM |
|------|------|------------------|--------------------|-----------|
| 500  | 20   | $8.18 \times 10^{-3}$ | $9.55 \times 10^{-3}$ | $-6.73 \times 10^{-3}$ |
| 800  | 40   | $2.53 \times 10^{-3}$ | $2.97 \times 10^{-3}$ | $-5.89 \times 10^{-3}$ |
| 1000 | 100  | $4.38 \times 10^{-4}$ | $5.18 \times 10^{-4}$ | $-2.23 \times 10^{-3}$ |

Table 3 Relative ratio of absolute loss on logistic regression

From Table 3, we observe that primal distributed ADMM method performs similarly as gradient method. Several previous studies have already shown that ADMM method may not be suitable for logistic regression (e.g. Gopal and Yang (2013)). However, it’s worth mentioning that with 5% of data sharing, multi-block ADMM could even out-perform centralized optimization method in terms of convergence speed. These results shed light on the importance of data sharing in the design of multi-block ADMM method. With a small amount of data sharing, one could hugely boost the convergence speed for multi-block ADMM method.

4.3. Apply data sharing in PCG methods

In this section, we apply data sharing algorithm in PCG method. We use the global data pool to construct more efficient local preconditioning matrices. With global data pool $X^g$, we could construct the local preconditioning matrix

$$H_i = \frac{1}{b} [X_i^T X_i + \sum_{j \neq i} \frac{s_j}{|r_j|} X_{r_j}^T X_{r_j}]^{-1}$$

Intuitively, we use the sampled data to sketch the data structure from other agents in order to estimate the global Hessian. The following numerical experiment considers the synthetic data with 2 agents. Each agent has number of observations $n = 1000$ with feature dimension $p = 500$. Data possessed in agent 1 follow independent and identical standard Gaussian distributed. Data
possessed in agent 2 follow independent and identical uniform distribution. We report the relative residuals in L2 norm, $\|b - Ax_t\|_2^2 / \|b\|_2^2$, at each iteration $t$. From Figure (2), we are convinced that the preconditioning with global data pool performs better than all the preconditioning methods without data share.

5. Conclusions

In this paper, we focus on how a small amount of data sharing benefits distributed optimization and learning, with focus on higher order optimization algorithms including distributed multi-block alternating direction method of multipliers (ADMM) method and preconditioned conjugate gradient (PCG) method. Theoretically, we answer the questions on when and why distributed optimization algorithms may sometimes suffer from slow convergence. Surprisingly, while PCG method suffers under data heterogeneity, for distributed ADMM algorithm, data heterogeneity may lead to faster convergence, which is contrary to conventional result that data heterogeneity hurts convergence. To answer the question of why this counter-intuitive phenomenon happens in distributed ADMM, we show that the key lies in the mechanism for updating dual variables by averaging across local estimators. And data homogeneity leads to weakening the momentum for dual updating. This result not only suggests that a simple approach to mitigate the challenge of data heterogeneity is to “order-up” the optimization algorithms, but also highlights the need for a universal approach, which is easy to implement in practice, on altering the local data structures to boost convergence rate for different optimization algorithms. With the guidance from theory works, we further develop a meta-algorithm of data sharing that builds a global data pool from pulling a small amount of prefixed data from each agent. We observe that sharing a small amount of data across agents may
have huge synergy in improving algorithm convergence speed, and 1% of data sharing leads to around 10 times speed up for higher order optimization algorithms. We hope that the results provided in this paper would encourage collaborative efforts from different regions to combat difficult global learning problems.

For future work, firstly, in theory part, providing analysis on how the convergence speed depends on the percentage of data shared is an exciting on-going work. We believe that the convergence rates of different algorithms may have different dependence on the percentage of data shared, and more theoretical work could be done in order to further guide the decision on how much data to be shared. Secondly, while in this paper we only focused on higher-order optimization algorithms including multi-block ADMM and PCG methods, one could also apply data sharing to other higher-order distributed optimization algorithms. Another exciting direction is to consider data-masking the global data pool for privacy concerns. Several exciting future research directions are to study how data-masking may impact the convergence speed, and to examine the trade-off between privacy and efficiency.
References

Jonathan Eckstein and Dimitri P Bertsekas. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992a.

Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

Robert Nishihara, Laurent Lessard, Ben Recht, Andrew Packard, and Michael Jordan. A general analysis of the convergence of admm. In *International Conference on Machine Learning*, pages 343–352. PMLR, 2015.

Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the second workshop on distributed infrastructures for deep learning*, pages 1–8, 2018.

Mauro Giuffrè and Dennis L Shung. Harnessing the power of synthetic data in healthcare: innovation, application, and privacy. *NPJ Digital Medicine*, 6(1):186, 2023.

Hau L Lee, Kut C So, and Christopher S Tang. The value of information sharing in a two-level supply chain. *Management science*, 46(5):626–643, 2000.

Weixin Shang, Albert Y Ha, and Shilu Tong. Information sharing in a supply chain with a common retailer. *Management Science*, 62(1):245–263, 2016.

Richard Feasey and Alexandre de Streele. *Data sharing for digital markets contestability: Towards a governance framework*. Centre on Regulation in Europe asbl (CERRE), 2020.

Chongwoo Choe, Jiajia Cong, and Chengsi Wang. Softening competition through unilateral sharing of customer data. *Management Science*, 2023.

Karthik Mohan, Palma London, Maryam Fazel, Daniela Witten, and Su-In Lee. Node-based learning of multiple gaussian graphical models. *The Journal of Machine Learning Research*, 15(1):445–488, 2014.

K. Huang and N. D. Sidiropoulos. Consensus-admm for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing*, 64(20):5297–5310, Oct 2016.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731, 2016.

Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.
Bingsheng He, Min Tao, and Xiaoming Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, 22(2):313–340, 2012.

Renato DC Monteiro and Benar F Svaiter. Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. *SIAM Journal on Optimization*, 23(1):475–507, 2013.

Roland Glowinski. On alternating direction methods of multipliers: a historical perspective. In *Modeling, simulation and optimization for science and technology*, pages 59–82. Springer, 2014.

Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pages 339–339. IEEE, 2018.

Masoud Zarepisheh, Lei Xing, and Yinyu Ye. A computation study on an integrated alternating direction method of multipliers for large scale optimization. *Optimization Letters*, 12(1):3–15, 2018.

T. Lin, S. Ma, Y. Ye, and S. Zhang. An ADMM-Based Interior-Point Method for Large-Scale Linear Programming. *ArXiv e-prints*, 2017.

Jonathan Eckstein and Dimitri P Bertsekas. On the douglas–rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992b.

Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pages 80–88, 2013.

Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4 (5):303–320, 1969.

M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14, 1978.

Ruoyu Sun, Zhi-Quan Luo, and Yinyu Ye. On the efficiency of random permutation for admm and coordinate descent. *Mathematics of Operations Research*, 45(1):233–271, 2020.

Krešimir Mihić, Mingxi Zhu, and Yinyu Ye. Managing randomization in the multi-block alternating direction method of multipliers for quadratic optimization. *Mathematical Programming Computation*, pages 1–75, 2020.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2015.
Caihua Chen, Min Li, Xin Liu, and Yinyu Ye. Extended admm and bcd for nonseparable convex minimization models with quadratic coupling terms: convergence analysis and insights. *Mathematical Programming*, Nov 2017. ISSN 1436-4646. doi: 10.1007/s10107-017-1205-9. URL https://doi.org/10.1007/s10107-017-1205-9.

David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *ICML*, 2011.

Rudi Helfenstein and Jonas Koko. Parallel preconditioned conjugate gradient algorithm on gpu. *Journal of Computational and Applied Mathematics*, 236(15):3584–3590, 2012.

Chih-Yang Hsia, Wei-Lin Chiang, and Chih-Jen Lin. Preconditioned conjugate gradient methods in truncated newton frameworks for large-scale linear classification. In *Asian Conference on Machine Learning*, pages 312–326. PMLR, 2018.

Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

Erik F Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied Mathematics*, 24(1-2):265–275, 1988.

Igor E Kaporin. New convergence results and preconditioning strategies for the conjugate gradient method. *Numerical linear algebra with applications*, 1(2):179–210, 1994.

Roland Herzog and Ekkehard Sachs. Preconditioned conjugate gradient method for optimal control problems with control and state constraints. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2291–2317, 2010.

Petros Drineas and Michael W Mahoney. Randnla: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90, 2016.

Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.05575*, 2015.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.
Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. *Advances in Neural Information Processing Systems*, 33:5332–5344, 2020.

LO Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995.

Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. *arXiv preprint arXiv:1708.01012*, 2017.

Artem Agafonov, Brahim Erraji, and Martin Takáč. Flecs-cgd: A federated learning second-order framework via compression and sketching with compressed gradient differences. *arXiv preprint arXiv:2210.09626*, 2022.

Jianyu Wang, Qingshu Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

Blake E Woodworth, Kumar Krithij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

Dashan Gao, Xin Yao, and Qiang Yang. A survey on heterogeneous federated learning. *arXiv preprint arXiv:2210.04505*, 2022.

Mang Ye, Xiuyun Fang, Bo Du, Pong C Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

Matias Mendieta, Taojiangnan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8397–8406, 2022.

Deren Han, Defeng Sun, and Liwei Zhang. Linear rate convergence of the alternating direction method of multipliers for convex composite programming. *Mathematics of Operations Research*, 43(2):622–637, 2018.

Yongchao Liu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. Partial error bound conditions and the linear convergence rate of the alternating direction method of multipliers. *SIAM Journal on Numerical Analysis*, 56(4):2095–2123, 2018.
Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Ali Makhdoumi and Asuman Ozdaglar. Convergence rate of distributed admm over networks. IEEE Transactions on Automatic Control, 62(10):5082–5095, 2017.

Ruoyu Sun, Zhi-Quan Luo, and Yinyu Ye. On the expected convergence of randomly permuted admm. Optimization for Machine Learning, OPT2015, 2015.

Christodoulos A Floudas and Panos M Pardalos. Encyclopedia of optimization. Springer Science & Business Media, 2008.

Peijun Xiao, Zhisheng Xiao, and Ruoyu Sun. Understanding limitation of two symmetrized orders by worst-case complexity. arXiv preprint arXiv:1910.04366, 2019.

Siddharth Gopal and Yiming Yang. Distributed training of large-scale logistic models. In International Conference on Machine Learning, pages 289–297. PMLR, 2013.

Zheng Xu, Gavin Taylor, Hao Li, Mario Figueiredo, Xiaoming Yuan, and Tom Goldstein. Adaptive consensus admm for distributed optimization. arXiv preprint arXiv:1706.02869, 2017.

Robert J Plemmons. M-matrix characterizations. i—nonsingular m-matrices. Linear Algebra and its Applications, 18(2):175–188, 1977.

Jan Brinkhuis, Zhi-Quan Luo, and Shuzhong Zhang. Matrix convex functions with applications to weighted centers for semidefinite programming. Technical report, Report / Econometric Institute, Erasmus University Rotterdam., 2005.
Appendix

A. Proofs on Section 3
A.1. Proof on Theorem 3.2

To prove Theorem 3.2, notice that the mapping matrix of primal distributed ADMM is given by $M_p = I - P - \Phi + 2\Phi P$, where $\Phi = (I + \frac{1}{\rho_p}D)^{-1}$. Let $\lambda \in \text{eig}(M_p)$, and $v$ be the eigenvector associated with $\lambda$, we have $\lambda$ and $v$ satisfies

$$(I - \Phi + (2\Phi - I)P)v = \lambda v. \tag{18}$$

For all block $i \in \{1, \ldots, b\}$, equation (18) becomes

$$(I - \Phi_i)v_i + (2\Phi_i - I)\bar{v} = \lambda v_i \quad \forall i, \tag{19}$$

where $\Phi_i = (I + \frac{1}{\rho_p}D_i)^{-1}$, $v_i \in \mathbb{R}^{s \times 1}$ is the $i^{th}$ block of $v$ (the $\{s(i-1) + 1, s(i-1) + 2, \ldots, si\}^{th}$ row of $v$), and $\bar{v} = \frac{1}{b}\sum_{i=1}^{b}v_i$ is the average of $v_i$.

We first give the proof of a special case where $D_i = D_j$ for all $i,j \in \{1, \ldots, b\}$. Later we show that, such data structure is indeed the worst data structure that leads to highest spectrum of mapping matrix $M_p$ for the distributed ADMM with fixed $\bar{q}$ and $q$ when $\rho_p > \bar{q}$. Let $\tilde{X}$ be the model matrix under the worst case data structure where $D_i = D_j = \frac{1}{b}\tilde{X}^T\tilde{X}$, equation (19) becomes

$$\left(I - \left(I + \frac{1}{b\rho_p}\tilde{X}^T\tilde{X}\right)^{-1}\right)v_i + \left(2\left(I + \frac{1}{b\rho_p}\tilde{X}^T\tilde{X}\right)^{-1} - I\right)\bar{v} = \lambda v_i \quad \forall i. \tag{20}$$

Let $M_s$ be the primal distributed mapping matrix under $\tilde{X}$, and let $\lambda$ and $v$ be the eigenvalue and eigenvector pairs of $M_s$, the following lemma holds.

Lemma A.1.

$$\lambda \neq 0 \in \text{eig}(M_s) \iff \frac{b\rho_p(1-\lambda)}{\lambda} \in \text{eig}(\tilde{X}^T\tilde{X}) \quad \text{or} \quad \frac{b\rho_p\lambda}{1-\lambda} \in \text{eig}(\tilde{X}^T\tilde{X})$$

Proof. Suppose $\lambda \neq 0 \in \text{eig}(M_s)$, let $\lambda$ and $v$ be the eigenvalue and eigenvector pairs of $M_s$. Consider the following two cases:

Case 1. $\bar{v} = \frac{1}{b}\sum_{i=1}^{b}v_i \neq 0$.

We first show $\lambda \neq 0 \in \text{eig}(M_s) \Rightarrow \frac{b\rho_p(1-\lambda)}{\lambda} \in \text{eig}(\tilde{X}^T\tilde{X})$. Sum over equation (20) across centers $i$ and take average,

$$\left(I + \frac{1}{b\rho_p}\tilde{X}^T\tilde{X}\right)^{-1}\bar{v} = \lambda\bar{v}. \tag{21}$$
With some algebra
\[ \tilde{X}^T \tilde{X} \mathbf{v} = \frac{bp_p(1 - \lambda)}{\lambda} \mathbf{v}. \]  
(22)

Since \( \mathbf{v} \neq 0 \) and \( \lambda \neq 0 \), we conclude that \( \frac{bp_p(1 - \lambda)}{\lambda} \) is an eigenvalue of \( \tilde{X}^T \tilde{X} \).

We then show \( \lambda \neq 0 \in eig(\mathbf{M}_s) \) if \( \frac{bp_p(1 - \lambda)}{\lambda} \in eig(\tilde{X}^T \tilde{X}) \). Suppose \( \frac{bp_p(1 - \lambda)}{\lambda} \in eig(\tilde{X}^T \tilde{X}) \), let \( \frac{bp_p(1 - \lambda)}{\lambda} \) and \( \mathbf{v} \) be the eigenvalue eigenvector pair of \( \tilde{X}^T \tilde{X} \), we have \( \lambda \neq 0 \) and \( \mathbf{v} \neq 0 \). And it’s easy to verify that \( \lambda \) and \( \mathbf{v} = [\tilde{v}; \ldots; \tilde{v}] \) satisfies equation (20) for all \( i \). Hence, \( \lambda \in eig(\mathbf{M}_s) \).

**Case 2.** \( \mathbf{v} = \frac{1}{b} \sum_{i=1}^{b} \mathbf{v}_i = 0 \).

To prove \( \lambda \neq 0 \in eig(\mathbf{M}_s) \Rightarrow \frac{bp_p \lambda}{1 - \lambda} \in eig(\tilde{X}^T \tilde{X}) \), we first claim that if \( \lambda, \mathbf{v} \) are eigenvalue eigenvector pair associated with \( \mathbf{M}_s \) and \( \mathbf{v} = \frac{1}{b} \sum_{i=1}^{b} \mathbf{v}_i = 0 \), then \( \lambda \neq 1 \), so \( \frac{bp_p \lambda}{1 - \lambda} \) is well-defined. To see this, suppose \( \lambda = 1 \in eig(\mathbf{M}_s) \) and the associated eigenvector pair \( \mathbf{v} \) satisfies \( \mathbf{v} = 0 \), (20) becomes
\[ \left( 2 \left( I + \frac{1}{bp_p} \tilde{X}^T \tilde{X} \right)^{-1} - I \right) \mathbf{v} = \left( I + \frac{1}{bp_p} \tilde{X}^T \tilde{X} \right)^{-1} \mathbf{v}_i \quad \forall i. \]  
(23)

With some algebra
\[ \mathbf{v}_i = \left( I - \frac{1}{bp_p} \tilde{X}^T \tilde{X} \right) \mathbf{v} \quad \forall i. \]  
(24)

As \( \mathbf{v} = 0 \), \( \mathbf{v}_i = 0 \) for all \( i \), which contradicts to the fact that \( \mathbf{v} \) is the eigenvector of \( \mathbf{M}_s \). Hence when \( \mathbf{v} = 0 \), if \( \lambda \in eig(\mathbf{M}_s) \), \( \lambda \neq 1 \). And \( \frac{bp_p \lambda}{1 - \lambda} \) is well-defined.

Take any non-zero \( \mathbf{v}_i \) (which exists as \( \mathbf{v} \neq 0 \)), equation (20) becomes
\[ \left( I - \left( I + \frac{1}{bp_p} \tilde{X}^T \tilde{X} \right)^{-1} \right) \mathbf{v}_i = \lambda \mathbf{v}_i. \]  
(25)

With some algebra
\[ \tilde{X}^T \tilde{X} \mathbf{v}_i = \frac{bp_p \lambda}{1 - \lambda} \mathbf{v}_i. \]  
(26)

Since \( \mathbf{v}_i \neq 0 \) and \( \lambda \neq 1 \), we conclude that \( \frac{bp_p \lambda}{1 - \lambda} \in eig(\tilde{X}^T \tilde{X}) \).

To prove \( \lambda \neq 0 \in eig(\mathbf{M}_s) \Rightarrow \frac{bp_p \lambda}{1 - \lambda} \in eig(\tilde{X}^T \tilde{X}) \), suppose \( \frac{bp_p \lambda}{1 - \lambda} \in eig(\tilde{X}^T \tilde{X}) \), let \( \mathbf{v} \) be the associated eigenvector pair, \( \lambda \neq 1 \). Let \( \mathbf{v} \) be such that \( \mathbf{v}_i = \tilde{v} \) and \( \mathbf{v}_j = -\frac{1}{b - 1} \tilde{v} \) for all \( j \neq i \), we have \( \mathbf{v} = \frac{1}{b} \sum_{i=1}^{b} \mathbf{v}_i = 0 \), and it’s easy to verify that \( \lambda \) and \( \mathbf{v} \) satisfies equation (20) for all \( i \).

With Lemma A.1, let \( \tilde{q} \in eig(\tilde{X}^T \tilde{X}) \) and \( \lambda \in eig(\mathbf{M}_s) \), we have \( \lambda = \frac{bp_p}{bp_p + \tilde{q}} \) or \( \lambda = -\frac{\tilde{q}}{bp_p + \tilde{q}} \). As \( \tilde{X}^T \tilde{X} \) has fixed largest eigenvalue \( \tilde{q} \), \( \rho_p > \tilde{q} \) and \( b \geq 2 \), \( \frac{bp_p}{bp_p + \tilde{q}} > \frac{-\tilde{q}}{bp_p + \tilde{q}} \). As \( \tilde{X}^T \tilde{X} \) has fixed smallest eigenvalue \( \tilde{q} \), the spectral radius of \( \mathbf{M}_s \) is given by
\[ \rho(\mathbf{M}_s) = \frac{bp_p}{bp_p + \tilde{q}}. \]  
(27)
To prove Theorem 3.2, we first introduce the following lemma to guarantee that the eigenvalues of mapping matrix $M_p$ is in the real space for $\rho_p > \bar{q}$.

**Lemma A.2.** Let $\lambda(M_p)$ be the eigenvalue of mapping matrix $M_p$ with $\rho_p > \bar{q}$. $\lambda(M_p) \in \mathbb{R}$.

**Proof.** Note $M_p = (I - \Phi) + (2\Phi - I)P$, let $S = 2\Phi - I$, $S$ is a block diagonal matrix with each diagonal block $i$ given by $S_i = 2\left(I + \frac{1}{\rho_p}D_i\right)^{-1} - I$. For $\rho_p > \bar{q}$, $S_i > 0$ for all blocks $i$. To prove this, let $q_i \in eig(D_i)$, since $\rho_p > \bar{q}$, we have the spectral radius of $\rho\left(\frac{1}{\rho_p}D_i\right) < 1$, and the Neumann series exist, with $S_i = 2\sum_{k=0}^{\infty}(-1)^k \left(\frac{1}{\rho_p}D_i\right)^k - I$, so $S_i$ is a polynomial function of $D_i$, and the eigenvalue of $S_i$ is $\frac{2}{1 + \bar{q}/\rho_p} - 1 > 0$.

Since $S_i > 0$ for all $i$ and $S$ is a block diagonal matrix with $S_i > 0$, $S > 0$, and there exists an invertible matrix $B \in \mathbb{R}^{bp \times bp}$ such that $S = B^T B$. Note that $M_p S = (I - \Phi)S + (2\Phi - I)P(2\Phi - I)$ and $SM_p^T = S(I - \Phi) + (2\Phi - I)P(2\Phi - I)$. Since $S(I - \Phi) = -I + 3\Phi - 2\Phi^2$, $\Phi$ is symmetric, $S(I - \Phi)$ symmetric, and $M_p S = SM_p^T$. Equivalently, $B^{-1}M_p B = (B^{-1}M_p B)^T$. Let $\hat{\lambda}$ and $\hat{v}$ be the eigenvalue eigenvector pair of $B^{-1}M_p B$. Since $B^{-1}M_p B$ is symmetric, $\hat{\lambda} \in \mathbb{R}$, and $B^{-1}M_p B\hat{v} = \hat{\lambda}\hat{v}$. Hence $\hat{\lambda}$ and $B\hat{v} \neq 0$ are the eigenvalue eigenvector pair of $M_p$, and $\lambda(M_p) \in \mathbb{R}$. \qed

With Lemma A.2, we could transfer the spectrum of $M_p$ to the eigenvalues of $M_p$, and we prove theorem 3.2 by contradiction. From convergence of distributed ADMM (e.g. Eckstein and Bertsekas (1992b), He and Yuan (2012), Ouyang et al. (2013), Xu et al. (2017)), $\rho(M_p) < 1$.

Suppose that $\rho(M_p) > \frac{b\rho_p}{b\rho_p + 2}$. This implies there exists $\lambda \in eig(M_p)$ and $\lambda \in \left(\frac{b\rho_p}{b\rho_p + 2}, 1\right)$ or $\lambda \in \left(-1, -\frac{b\rho_p}{b\rho_p + 2}\right)$. We start to prove by contradiction for the two different cases.

**Case 1.** Suppose $\lambda \in eig(M_p)$ and $\lambda \in \left(\frac{b\rho_p}{b\rho_p + 2}, 1\right)$.

**Proof.** Suppose there exists $\lambda \in \left(\frac{b}{b + \lambda_{min}(X^TX)}, 1\right)$ and $\lambda$ is the eigenvalue of $M_p$. Let $v = [v_1; \ldots; v_i; \ldots; v_b]$ be the eigenvector associated with $\lambda$. $\lambda$ and $v$ satisfy equation (19). Summing over all the $b$ equations and taking the average on both side, we have

$$
-\frac{1}{b} \sum_{i=1}^{b} \Phi_i v_i + \frac{2}{b} \sum_{i=1}^{b} \Phi_i \bar{v} = \lambda \bar{v}.
$$

(28)

Besides, from equation (19), if $\lambda \in eig(M_p)$, with some algebra

$$
((1 - \lambda)I - \Phi_i)v_i = (I - 2\Phi_i)\bar{v}.
$$

(29)
Following the assumption that $\lambda \in \left(\frac{bp_p}{bp_p + q}, 1\right)$, $(1 - \lambda)I - \Phi_i$ exists. To see this, notice

$$(1 - \lambda)I - \Phi_i)^{-1} = -\lambda^{-1}\Phi_i^{-1}\left(I - \frac{1 - \lambda}{\lambda p_p}D_i\right)^{-1}. \quad (30)$$

As $\frac{bp_p}{bp_p + q} > \frac{1}{2}$, suppose $\lambda \in \left(\frac{bp_p}{bp_p + q}, 1\right)$, $\lambda \in (\frac{1}{2}, 1)$, and $\frac{1 - \lambda}{\lambda p_p} \in (0, 1)$. And since $X$ is normalized by its Frobenius norm, $\text{eig}(X^TX) \in (0, 1)$, so $\text{eig}(D_i) \in (0, 1)$, $(I - \frac{1 - \lambda}{\lambda p_p}D_i) > 0$, and the inverse of $(I - \frac{1 - \lambda}{\lambda p_p}D_i)$ exists. In fact, following the notation of Plemmons (1977), $(I - \frac{1 - \lambda}{\lambda p_p}D_i)$ is a $M$-matrix, so its inverse exists with $(I - \frac{1 - \lambda}{\lambda p_p}D_i)^{-1} > 0$ and $(I - \frac{1 - \lambda}{\lambda p_p}D_i)$ is inverse positive. Hence $(1 - \lambda)I - \Phi_i)^{-1}$ exists, and

$$v_i = ((1 - \lambda)I - \Phi_i)^{-1}(1 - 2\Phi_i)v. \quad (31)$$

Notice that $\bar{v} \neq 0$. If $\bar{v} = 0$, $v_i = 0$ for all $i$ and this contradicts with the assumption that $v$ is the eigenvector of $M$. Plugging equation (31) into (28),

$$\frac{1}{b}\sum_{i=1}^{b}[2\Phi_i - \Phi_i((1 - \lambda) - \Phi_i)^{-1}(I - 2\Phi_i)]\bar{v} = \lambda\bar{v}. \quad (32)$$

With some algebra,

$$2\Phi_i - \Phi_i((1 - \lambda) - \Phi_i)^{-1}(I - 2\Phi_i) = -(2\lambda - 1)\Phi_i((1 - \lambda)I - \Phi_i)^{-1}$$

$$= -(2\lambda - 1)\Phi_i(((1 - \lambda)\Phi_i^{-1} - I)\Phi_i)^{-1} = (2\lambda - 1)\left(\lambda I - \frac{1 - \lambda}{\rho_p}D_i\right)^{-1}. \quad (33)$$

Equation (32) becomes

$$\frac{1}{b}\sum_{i=1}^{b}\left[(2\lambda - 1)\left(\lambda I - \frac{1 - \lambda}{\rho_p}D_i\right)^{-1}\right]v = \lambda\bar{v}. \quad (34)$$

Let

$$f(t|X, v) = \bar{v}^T\left(\frac{1}{b}\sum_{i=1}^{b}\left[(2t - 1)\left(tI - \frac{1 - t}{\rho_p}D_i\right)^{-1}\right]\right)\bar{v} - tv^Tv, \quad (35)$$

where $\bar{v}^T$ is the transpose of $\bar{v}$ and $\bar{v} \in \mathbb{R}^{p \times 1}$. Let $X$ be the associated model matrix of $M_p$, the following relation holds

$$\lambda \in \text{eig}(M_p) \Rightarrow \text{there exists } \bar{v} \in \mathbb{R}^{p \times 1} \neq 0 \text{ such that } f(\lambda|X, \bar{v}) = 0. \quad (36)$$

To see this, let $\lambda$ and $v$ be the eigenvalue eigenvector pair of $M_p$, since $\lambda \in \mathbb{R}$ and $M_p \in \mathbb{R}^{bp \times bp}$, so $v \in \mathbb{R}^{bp \times 1}$. Let $\bar{v} = \sum_{i=1}^{b} v_i$, from equation 31, $\bar{v} \in \mathbb{R}^{p \times 1} \neq 0$, and it’s easy to verify that $f(\lambda|X, \bar{v}) = 0$ for $\lambda \in \text{eig}(M_p)$ and $\bar{v}$.

We further prove $\lambda \notin \left(\frac{\rho_p}{\rho_p + q}, 1\right)$ by showing that for all $t \in \left(\frac{\rho_p}{\rho_p + q}, 1\right)$ and any $\bar{v} \neq 0$, $f(t|X, \bar{v}) > 0$, which contradicts to (36), hence if $\lambda \in \text{eig}(M_p)$, $\lambda \leq \frac{bp_p}{bp_p + q}$. 
Let $\tilde{X}$ be the model matrix such that $D_i = D_j$ for all $\{i, j\} \in \{1, \ldots, b\}$ and let $\bar{t} = \frac{b \rho_{p}}{b \rho_{p} + q^2}$. By Lemma A.1, $\bar{t} \in \text{eig}(M_\bar{t})$, and $f(\bar{t})\tilde{X}, \bar{v}) = 0$, with $\bar{v} = \frac{1}{b} \sum_{i=1}^{b} v_i$ and $v = [v_1; \ldots; v_i; \ldots; v_b]$ the associated eigenvector to $\bar{t}$. We propose the following claims to show that for all $t \in (\bar{t}, 1)$, $f(t|X, \bar{v}) > 0$ for all $X$ satisfies assumption 3.1 and $\bar{v} \neq 0$.

**Claim 1:** $f(\bar{t})\tilde{X}, \bar{v}) \geq 0$ for all $\bar{v} \neq 0$.

**Proof.** When $D_i = D_j$ for all $\{i, j\} \in \{1, \ldots, b\}$, we have

$$f(\bar{t})\tilde{X}, \bar{v}) = \bar{v}^T \left( \frac{2\bar{t} - 1}{\bar{t}} \left( I - \frac{1 - \bar{t}}{t \rho_{p}} \tilde{X}^T \tilde{X} \right)^{-1} - \bar{t} I \right) \bar{v}. \tag{37}$$

It’s sufficient to show that

$$\frac{2\bar{t} - 1}{\bar{t}} \left( I - \frac{1 - \bar{t}}{t \rho_{p}} \tilde{X}^T \tilde{X} \right)^{-1} - \bar{t} I \succeq 0. \tag{38}$$

Note $\bar{t} = \frac{b \rho_{p}}{b \rho_{p} + q^2}$, (38) is equivalent as

$$\left( I - \frac{q}{b^2 \rho_{p}^2} \tilde{X}^T \tilde{X} \right)^{-1} \succeq \frac{b^2 \rho_{p}^2}{b^2 \rho_{p}^2 - q^2} I. \tag{39}$$

Note that the spectral radius of $\frac{q}{b^2 \rho_{p}^2} \tilde{X}^T \tilde{X}$, $\rho \left( \frac{q}{b^2 \rho_{p}^2} \tilde{X}^T \tilde{X} \right) = \frac{q}{b^2 \rho_{p}^2} < 1$. Hence the Neumann series exists and $\left( I - \frac{q}{b^2 \rho_{p}^2} \tilde{X}^T \tilde{X} \right)^{-1}$ could be written as the polynomial of $\tilde{X}^T \tilde{X}$. And let $\bar{q}$ be the eigenvalue of $\tilde{X}^T \tilde{X}$, the eigenvalue of $\left( I - \frac{q}{b^2 \rho_{p}^2} \tilde{X}^T \tilde{X} \right)^{-1}$ is given by $\frac{b^2 \rho_{p}^2}{b^2 \rho_{p}^2 - q^2}$, which is lowerbounded by $\frac{b^2 \rho_{p}^2}{b^2 \rho_{p}^2 - q^2}$. So (39) holds and $f(\bar{t})\tilde{X}, \bar{v}) \geq 0$. □

**Claim 2:** $f(\bar{t})|X, \bar{v}) \geq 0$, with strict inequality holds when $D_i - D_j$ is non-singular for all $i$ and $j$.

**Proof.** Note for any $X = [X_1, \ldots, X_i, \ldots, X_b]$ and $\bar{v} \neq 0$,

$$f(\bar{t})|X, \bar{v}) = \bar{v}^T \left( \frac{2\bar{t} - 1}{\bar{t}} \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1 - \bar{t})}{t \rho_{p}} D_i \right)^{-1} \right] - \bar{t} I \right) \bar{v}. \tag{40}$$

We first show that for any $X = [X_1, \ldots, X_i, \ldots, X_b]$ and $\bar{v} \neq 0$,

$$f(\bar{t})|X, \bar{v}) - \bar{v}^T \left( \frac{2\bar{t} - 1}{\bar{t}} \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1 - \bar{t})}{t \rho_{p}} D_i \right)^{-1} \right] - \bar{t} I \right) \bar{v} \tag{41}$$

$$= \frac{2\bar{t} - 1}{\bar{t}} \bar{v}^T \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1 - \bar{t})}{t \rho_{p}} D_i \right)^{-1} \right] - \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1 - \bar{t})}{t \rho_{p}} D_i \right) \bar{v} \geq 0. \tag{42}$$
And when $D_i - D_j$ non singular
\[
    f(\bar{t}|X, \bar{v}) - \bar{v}^T \left( \frac{2\bar{t} - 1}{t} \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1-\bar{t})}{\bar{t}\rho_p} D_i \right) \right]^{-1} - \bar{t} I \right) \bar{v} > 0. \tag{43}
\]

Since $\bar{t} > \frac{1}{2}$, $\frac{2\bar{t} - 1}{t} > 0$. It’s sufficient to show that
\[
    \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{1-\bar{t}}{\bar{t}\rho_p} D_i \right)^{-1} \geq \left( \frac{1}{b} \sum_{i=1}^{b} I - \frac{1-\bar{t}}{\bar{t}\rho_p} D_i \right)^{-1}, \tag{44}
\]
and when $D_i - D_j$ is non-singular for all $i, j \in \{1, \ldots, b\}$,
\[
    \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{1-\bar{t}}{\bar{t}\rho_p} D_i \right)^{-1} > \left( \frac{1}{b} \sum_{i=1}^{b} I - \frac{1-\bar{t}}{\bar{t}\rho_p} D_i \right)^{-1}. \tag{45}
\]

To prove this, we notice that matrix inverse is a (strictly) convex operation. Specifically note that $\left( I - \frac{1-\bar{t}}{\bar{t}\rho_p} D_i \right)^{-1} > 0$ for all $i$. Following the Brinkhuis et al. (2005), for positive definite matrix $X$ and $Y$, with $\alpha \in [0, 1]$, the following identity holds
\[
    \alpha X^{-1} + (1-\alpha) Y^{-1} - [\alpha X + (1-\alpha) Y]^{-1} = \alpha(1-\alpha) X^{-1}(Y - X) Y^{-1}[\alpha Y^{-1} + (1-\alpha) X^{-1}] Y^{-1}(Y - X) X^{-1}. \tag{46}
\]

So for $\alpha \in [0, 1]$, $X, Y \succ 0$
\[
    \alpha X^{-1} + (1-\alpha) Y^{-1} \succeq [\alpha X + (1-\alpha) Y]^{-1}, \tag{47}
\]
with the following equation holds when $\alpha \in (0, 1)$ and $X - Y$ non singular
\[
    \alpha X^{-1} + (1-\alpha) Y^{-1} \succ [\alpha X + (1-\alpha) Y]^{-1}. \tag{48}
\]

By induction on applying (47) and (48), we prove (44) and (45). We further show that,
\[
    \bar{v}^T \left( \frac{2\bar{t} - 1}{t} \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1-\bar{t})}{\bar{t}\rho_p} D_i \right) \right]^{-1} - \bar{t} I \right) \bar{v} \geq 0. \tag{49}
\]

Note
\[
    \bar{v}^T \left( \frac{2\bar{t} - 1}{t} \left[ \frac{1}{b} \sum_{i=1}^{b} \left( I - \frac{(1-\bar{t})}{\bar{t}\rho_p} D_i \right) \right]^{-1} - \bar{t} I \right) \bar{v} = \bar{v}^T \left( \frac{2\bar{t} - 1}{t} \left[ I - \frac{(1-\bar{t})}{\bar{t}\rho_p} \frac{X^T X}{b} \right] \right) \bar{v}, \tag{50}
\]

it’s sufficient to show
\[
    \frac{2\bar{t} - 1}{t} \left[ I - \frac{(1-\bar{t})}{\bar{t}\rho_p} \frac{X^T X}{b} \right] \succ - \bar{t} I \succ 0. \tag{51}
\]

Following (39), this is equivalent as showing
\[
    \left( I - \frac{q}{b^2 \rho_p^2} X^T X \right)^{-1} \succeq \frac{b^2 \rho_p^2}{b^2 \rho_p^2 - q^2} I. \tag{52}
\]
Similarly, let \( q = eig(X^TX) \), the eigenvalue of \( \left( I - \frac{q}{\mu_p^2}X^TX \right)^{-1} \) is given by \( \frac{\mu_p^2}{\mu_p^2 - q} \), which is lowerbounded by \( \frac{\mu_p^2}{\mu_p^2 - q} \). So (52) holds, hence we prove (49) holds. And we finish the proof on Claim 2.

\[ \square \]

**Claim 3:** \( f(t|X,\bar{v}) > 0 \) for all \( t \in (\bar{t},1) \), \( X \) and \( \bar{v} \neq 0 \).

**Proof.** Firstly, notice that given \( X \), \( f(t|X,\bar{v}) \) is a twice differentiable continuous function on \( t \) for \( t \in (\bar{x},1) \), and \( f(1|X,\bar{v}) = 0 \). Furthermore,

\[
\frac{\partial f(t|X,\bar{v})}{\partial t} = \bar{v}^T \left[ \frac{1}{b} \sum_{i=1}^{b} 2 \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} - \frac{1}{b} \sum_{i=1}^{b} (2t-1) \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \left( I + \frac{1}{\rho_p} D_i \right) \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} - I \right] \bar{v}
\]

(53)

When \( t = 1 \),

\[
\frac{\partial f(t|X,\bar{v})}{\partial t} \bigg|_{t=1} = -\bar{v}^T \left[ \frac{1}{b\rho_p} X^TX \right] \bar{v} < 0.
\]

(54)

And with some algebra, the second order derivative with respect to \( t \) is given by

\[
\frac{\partial^2 f(t|X,\bar{v})}{\partial t^2} = \bar{v}^T \left[ \frac{2}{b} \sum_{i=1}^{b} \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} M'_i \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \right] \bar{v},
\]

(55)

where

\[
M'_i = -2 \left( I + \frac{D_i}{\rho_p} \right) + (2t-1) \left( I + \frac{D_i}{\rho_p} \right)^2 \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \left( I + \frac{D_i}{\rho_p} \right)
\]

\[
= -2 \left( I + \frac{D_i}{\rho_p} \right) + (2t-1) \left( I + \frac{D_i}{\rho_p} \right)^2 \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1}
\]

\[
= \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \left[ -2 \left( I + \frac{D_i}{\rho_p} \right) \left( tI - \frac{1-t}{\rho_p} D_i \right)^2 + (2t-1) \left( tI - \frac{1-t}{\rho_p} D_i \right) \left( I + \frac{D_i}{\rho_p} \right) \right] \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1}.
\]

(56)

The second and third inequality come from the fact that \( \left( I + \frac{D_i}{\rho_p} \right) \) and \( \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \) commute.

\[
\left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1} \left( I + \frac{D_i}{\rho_p} \right) = \frac{1}{t} \left( I - \frac{1-t}{t\rho_p} D_i \right)^{-1} \left( I + \frac{D_i}{\rho_p} \right)
\]

\[
= \frac{1}{t} \sum_{k=0}^{\infty} \left( \frac{1-t}{t\rho_p} D_i \right)^k \left( I + \frac{D_i}{\rho_p} \right)
\]

\[
= \frac{1}{t} \sum_{k=0}^{\infty} \left( I + \frac{D_i}{\rho_p} \right) \left( \frac{1-t}{t\rho_p} D_i \right)^k
\]

\[
= \left( I + \frac{D_i}{\rho_p} \right) \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-1}.
\]

(57)

And

\[
\frac{\partial^2 f(t|X,\bar{v})}{\partial t^2} = \bar{v}^T \left[ \frac{2}{b} \sum_{i=1}^{b} \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-2} M_i \left( tI - \frac{1-t}{\rho_p} D_i \right)^{-2} \right] \bar{v},
\]

(58)
where
\[
M_i = -2 \left( I + \frac{D_i}{\rho_p} \right) \left( tI - \frac{1 - t}{\rho_p} D_i \right)^2 + (2t - 1) \left( tI - \frac{1 - t}{\rho_p} D_i \right) \left( I + \frac{D_i}{\rho_p} \right)^2.
\] (59)

Note that \(M_i\) is a polynomial function of \(D_i\). Let \(M_i = P(D_i)\), where
\[
P(x) = \left( \frac{x^2}{\rho_p^2} - 1 \right) \left( t - \frac{x(1 - t)}{\rho_p} \right).
\] (60)

Let \(\lambda_i \in eig(D_i)\), we have \(P(\lambda_i) \in eig(M_i)\). And for \(\lambda_i \in (0, 1)\) and \(t \in \left( \frac{1}{2}, 1 \right)\), \(P(\lambda_i) < 0\), hence \(M_i < 0\) for all \(i\), and
\[
\frac{\partial^2 f(t|X, \bar{v})}{\partial t^2} < 0, \quad \text{for } t \in \left( \frac{1}{2}, 1 \right).
\] (61)

Combining the fact that \(\bar{v} > 0\), \(f(\bar{t}|X, \bar{v}) \geq 0\), and \(f(1|X, \bar{v}) = 0\), \(\frac{\partial f(t|X, \bar{v})}{\partial t}|_{t=1} < 0\), \(\frac{\partial^2 f(t|X, \bar{v})}{\partial t^2} < 0\), \(f(t|X, \bar{v}) > 0\) for all \(t \in (\bar{t}, 1)\), and we finish the proof of Claim 3. \(\square\)

Suppose \(\lambda \in eig(M_p)\) and \(\lambda \in \left( \frac{b\rho_p}{\rho_p + b}, 1 \right)\), there must exist \(\bar{v} \neq 0\) such that \(f(\lambda|X, \bar{v}) = 0\). However, for all \(t \in \left( \frac{b\rho_p}{\rho_p + b}, 1 \right)\) and \(\bar{v} \neq 0\), \(f(t|X, \bar{v}) > 0\). Hence if \(\lambda \in eig(M_p)\), \(\lambda \notin \left( \frac{b\rho_p}{\rho_p + b}, 1 \right)\). And we finish the proof by contradiction for Case 1. \(\square\)

**Case 2.** Suppose \(\lambda \in eig(M_p)\) and \(\lambda \in \left( -1, -\frac{b\rho_p}{\rho_p + b} \right)\).

**Proof.** Similarly, \((1 - \lambda)^{-1}\) exists. To see this, notice
\[
((1 - \lambda)I - \Phi_i)^{-1} = -\lambda^{-1} \Phi_i^{-1} \left( I - \frac{1 - \lambda}{\lambda \rho_p} D_i \right)^{-1}.
\] (62)

As \(\left( I - \frac{1 - \lambda}{\lambda \rho_p} D_i \right)\) is positive definite, the inverse exists. And
\[
\mathbf{v}_i = ((1 - \lambda)I - \Phi_i)^{-1}(1 - 2\Phi_i)\bar{v}.
\] (63)

Hence we have \(\bar{v} \neq 0\), as if \(\bar{v} = 0\), \(\mathbf{v}_i = 0\) for all \(i\) which contradicts to \(\mathbf{v}_i\) is a valid eigenvector.

Let
\[
f(t|X, \bar{v}) = \bar{v}^T \left( \frac{1}{b} \sum_{i=1}^{b} \hat{M}_i \right) \bar{v},
\] (64)

where
\[
\hat{M}_i = \frac{2t - 1}{t} \left( I - \frac{1 - t}{tp_p} D_i \right)^{-1} - t I.
\] (65)

The following relation holds
\[
\lambda \in eig(M_p) \Rightarrow \text{there exists } \bar{v} \in \mathbb{R}^{p \times 1} \neq 0 \text{ such that } f(\lambda|X, \bar{v}) = 0.
\] (66)

For \(t \in \left( -1, -\frac{b\rho_p}{\rho_p + b} \right)\), \(I - \frac{1 - t}{tp_p} D_i > 0\), so the inverse is also positive definite, and \(\frac{2t - 1}{t} > 0\), so \(\hat{M}_i > 0\) for all \(i\), and \(f(t|X, \bar{v}) > 0\) for all \(t \in \left( -1, -\frac{b\rho_p}{\rho_p + b} \right)\) and \(\bar{v} \neq 0\).
Hence if \( \lambda \in eig(M_p) \), \( \lambda \notin \left(-1, -\frac{b_p}{b_p + q}\right) \). And we finish the proof by contradiction for Case 2.

\[ \square \]

With the previous proof on contradiction, we conclude that \( \lambda \notin \left(-\frac{b_p}{b_p + q}, 1\right) \) and \( \lambda \notin \left(-1, -\frac{b_p}{b_p + q}\right) \). Hence for \( \rho_p > q \), the convergence rate of distributed ADMM is upper bounded by \( \frac{b_p}{b_p + q} \) and the upperbound is achieved when \( D_i = D_j \) for all \( i, j \in \{1, \ldots, b\} \).

### A.2. Proof on Theorem 3.3

To prove Theorem 3.3, we first introduce the following lemma to guarantee that the eigenvalues of mapping matrix \( M_p \) is in the real space for \( \rho_p < q_1 \).

**Lemma A.3.** Let \( \lambda(M_p) \) be the eigenvalue of mapping matrix \( M_p \) with \( \rho_p < q_1 \). \( \lambda(M_p) \in \mathbb{R} \).

**Proof.** Note \( M_p = (I - \Phi) - (I - 2\Phi)P \), let \( S = I - 2\Phi \), \( S \) is a block diagonal matrix with each diagonal block \( i \) given by \( S_i = I - 2\left(I + \frac{1}{\rho_p} D_i\right)^{-1} \). For \( \rho_p < q_1 \), \( S_i > 0 \) for all blocks \( i \). To prove this, let \( q_i \in eig(D_i) \), \( \frac{\rho_p}{\rho_p + q_i} \in eig\left(I + \frac{1}{\rho_p} D_i\right)^{-1} \), hence \( \frac{q_i - \rho_p}{\rho_p + q_i} \in eig(S_i) \). Since \( \rho_p < q_1 \), \( S_i > 0 \), and \( S > 0 \), following similar proof in Lemma A.2, \( \lambda(M_p) \in \mathbb{R} \).

With Lemma A.3, we are ready to prove Theorem 3.3.

**Proof.** Since \( eig(M_p) = eig(M_p^T) \), consider \( M_p^T = (I - \Phi) - P(I - 2\Phi) = \frac{1}{\rho_p} \Phi D + P \Phi \left(I - \frac{1}{\rho_p} D\right) \). For \( b = 2 \), let \( \lambda \in eig(M_p^T) \) and \( v = [v_1; v_2] \) be the unit eigenvector associated with \( \lambda \). Since \( M_p \in \mathbb{R}^{2p \times 2p} \) and \( \lambda \in \mathbb{R} \), \( v \in \mathbb{R}^{2p \times 1} \), \( v^T v = 1 \). And \( \lambda, v = [v_1; v_2] \) satisfies

\[
\frac{1}{\rho_p} \Phi_1 D_1 v_1 + \frac{1}{2} \sum_{i=1}^{b} \Phi_i \left(I - \frac{1}{\rho_p} D_i\right) v_i = \lambda v_1, \quad (67)
\]

\[
\frac{1}{\rho_p} \Phi_2 D_2 v_2 + \frac{1}{2} \sum_{i=1}^{b} \Phi_i \left(I - \frac{1}{\rho_p} D_i\right) v_i = \lambda v_2. \quad (68)
\]

With some algebra, we have

\[
\frac{1}{2} v_1 + \frac{1}{2} \Phi_2 \left(I - \frac{1}{\rho_p} D_2\right) v_2 = \lambda v_1, \quad (69)
\]

\[
\frac{1}{2} v_2 + \frac{1}{2} \Phi_1 \left(I - \frac{1}{\rho_p} D_1\right) v_1 = \lambda v_2. \quad (70)
\]

Multiply equation (69) by \( v_1^T \) and (70) by \( v_1^T \) on both side,

\[
\frac{1}{2} v_1^T v_1 + \frac{1}{2} v_1^T \Phi_2 \left(I - \frac{1}{\rho_p} D_2\right) v_2 = \lambda v_1^T v_1, \quad (71)
\]
\[
\frac{1}{2}v_2^T v_2 + \frac{1}{2}v_2^T \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) v_1 = \lambda v_2^T v_2. \tag{72}
\]

Sum over equation (71) and (72), as \( v \) is the unit eigenvector

\[
\lambda = \frac{1}{2} + \frac{1}{2}v_1^T \Phi_2 \left( I - \frac{1}{\rho_p} D_2 \right) v_2 + \frac{1}{2}v_2^T \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) v_1. \tag{73}
\]

Since \( \Phi_i \left( I - \frac{1}{\rho_p} D_i \right) = -(I - 2\Phi_i) \prec 0 \) for all \( i = 1, 2 \), there exists \( A \) and \( B \) such that \( \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) = -A^T A \) and \( \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) = -B^T B \), we have

\[
\lambda = \frac{1}{2} + \frac{1}{2}v_1^T \Phi_2 \left( I - \frac{1}{\rho_p} D_2 \right) v_2 + \frac{1}{2}v_2^T \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) v_1
\]

\[
= \frac{1}{2} + \frac{1}{2}v_1^T (A^T A) v_2 + \frac{1}{2}v_2^T (-B^T B) v_1
\]

\[
\leq \frac{1}{2} + \frac{1}{2} |v_1^T A^T A v_2| + \frac{1}{2} |v_2^T B^T B v_1|
\]

\[
\leq \frac{1}{2} + \frac{1}{2} \left[ \frac{(A v_1)^T (A v_1)}{2} + \frac{(A v_2)^T (A v_2)}{2} \right]
\]

\[
= \frac{1}{2} + \frac{1}{2} \left[ \frac{1}{2} (I - 2 \Phi_1) + \frac{1}{2} (I - 2 \Phi_2) \right] v_1 + v_2^T \left[ \frac{1}{2} (I - 2 \Phi_1) + \frac{1}{2} (I - 2 \Phi_2) \right] v_2
\]

\[
= 1 - \frac{1}{2} \left( v_1^T (\Phi_1 + \Phi_2) v_1 + v_2^T (\Phi_1 + \Phi_2) v_2 \right).
\]

The first inequality comes from triangle inequality, the second inequality comes from Cauchy-Schwarz and the third inequality comes from AMGM. By matrix convexity from (47), we have

\[
\frac{1}{2} \Phi_1 + \frac{1}{2} \Phi_2 \succeq \left( I + \frac{X^T X}{2\rho_p} \right)^{-1}
\]

\[
\frac{1}{2}v_1^T (\Phi_1 + \Phi_2) v_1 \succeq v_1^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} v_1, \quad \frac{1}{2}v_2^T (\Phi_1 + \Phi_2) v_2 \succeq v_2^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} \] \( v_2, \tag{76} \)

and

\[
\lambda \leq 1 - v_1^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} v_1 - v_2^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} v_2. \tag{77} \]

Let \( q \in eig(X^T X) \), since \( eigh \left( I + \frac{X^T X}{2\rho_p} \right) = \frac{2\rho_p}{2\rho_p + q} \), by definition of \( \bar{q} \), we have

\[
\left( I + \frac{X^T X}{2\rho_p} \right)^{-1} \succeq \frac{2\rho_p}{2\rho_p + \bar{q}} I
\]

and

\[
v_1^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} v_1 \succeq \frac{2\rho_p}{2\rho_p + \bar{q}} v_1^T v_1, \quad v_2^T \left( I + \frac{X^T X}{2\rho_p} \right)^{-1} v_2 \succeq \frac{2\rho_p}{2\rho_p + \bar{q}} v_2^T v_2. \tag{79} \]
Hence, by the fact that $v^Tv = 1$

$$\lambda \leq 1 - \frac{2\rho_p}{2\rho_p + \bar{q}} = \frac{\bar{q}}{2\rho_p + \bar{q}}. \tag{80}$$

We further show that when $D_1 = D_2$, $\lambda = \frac{\bar{q}}{2\rho_p + \bar{q}}$. By Lemma A.1, let $\bar{q} \in e(i_{g(\hat{X}^T\hat{X})})$, where $\hat{X}$ is the model matrix with $D_1 = D_2$, and $\lambda \in e(i_{g(M_u)})$, we have when $b = 2$, $\lambda = \frac{2\rho_p}{2\rho_p + \bar{q}}$ or $\lambda = \frac{\bar{q}}{2\rho_p + \bar{q}}$. Hence the upper bound is achieved when $D_1 = D_2$. \hfill \Box

### A.3. Proof on Proposition 3.4

We first show that for $\rho_p < q_1$, the convergence rate of distributed ADMM is upper bounded by $\frac{\bar{q}}{\rho_p + q}$. 

**Proof.** Following Lemma A.3, let $\lambda \in e(i_{g(M_p)}) \in \mathbb{R}$. Since $\rho(M_p) \leq \rho(\frac{1}{2}M_p + \frac{1}{2}M_p^T)$, define $\hat{M}_p$ as

$$\hat{M}_p = \frac{1}{2}M_p + \frac{1}{2}M_p^T = I - \Phi + \Phi P + P \Phi - P. \tag{81}$$

Let $\hat{\lambda}$ and $\hat{v} = [v_1; \ldots; v_b]$ be the associated eigenvalue eigenvector pair of $\hat{M}_p$, we have $\hat{\lambda}$ and $\hat{v}$ satisfies

$$(I - \Phi + \Phi P + P \Phi - P)v = \lambda v. \tag{82}$$

Multiply $P$ by both side and let $\bar{v} = \sum_{i=1}^{b} v_i$, $\hat{\lambda}$ and $\hat{v}$ satisfies

$$P \Phi P \hat{v} = \lambda P \hat{v}, \quad \frac{1}{b} \sum_{j} \Phi_j \bar{v} = \hat{\lambda} \bar{v}. \tag{83}$$

We consider two cases, $\bar{v} \neq 0$ or $\bar{v} = 0$.

**Case 1. $\bar{v} \neq 0$.**

Since $\bar{v} \neq 0$, from equation 83, $\hat{\lambda} \in e(i_{g(\hat{M}_p)})$ implies $\hat{\lambda} \in e(i_{g(\frac{1}{b} \sum_{j} \Phi_j)})$, and by Weyl’s theorem,

$$\rho(\frac{1}{b} \sum_{j} \Phi_j) \leq \frac{1}{b} \sum_{j} \rho(\Phi_j). \tag{84}$$

And let $q_1 = \min_{i=1, \ldots, b}$

$$\rho(\Phi_j) \leq \frac{\rho_p}{\rho_p + q_1} \quad \forall j. \tag{85}$$

Hence

$$\rho(M_p) \leq \rho(\hat{M}_p) \leq \frac{\rho_p}{\rho_p + q_1} \leq \frac{\bar{q}}{\rho_p + \bar{q}}. \tag{86}$$

**Case 2. $\bar{v} = 0$.**
Since \( \mathbf{v} = \mathbf{Pv} = 0 \), let \( \hat{\lambda} \in \text{eig}(\hat{M}_p) \) and \( \mathbf{v} \) be the unit eigenvector \( (\mathbf{v}^T \mathbf{v} = 1) \), the following equation holds

\[
(I - \Phi + \Phi \mathbf{P} + \mathbf{P} \Phi - \mathbf{P}) \mathbf{v} = (I - \Phi + \Phi \mathbf{P}) \mathbf{v} = \hat{\lambda} \mathbf{v}.
\]  

(87)

As \( \mathbf{P} = \mathbf{P}^T \), multiply both side by \( \mathbf{v}^T \)

\[
1 - \mathbf{v}^T \Phi \mathbf{v} + (\mathbf{Pv})^T \Phi \mathbf{v} = \hat{\lambda}, \quad \hat{\lambda} = 1 - \mathbf{v}^T \Phi \mathbf{v}.
\]  

(88)

Let \( q_2 = \max_{i=1,\ldots,b} \rho(D_i) \), by the fact that \( \Phi - \frac{\rho_p}{\rho_p + q_2} \mathbf{I} \succ 0 \), \( \hat{\lambda} \) is upperbounded by

\[
\hat{\lambda} = 1 - \mathbf{v}^T \Phi \mathbf{v} \leq 1 - \frac{\rho_p}{\rho_p + q_2}.
\]  

(89)

Since \( \bar{q} = \rho(\mathbf{X}^T \mathbf{X}) \geq q_2 \),

\[
\rho(\mathbf{M}_p) \leq \hat{\lambda} \leq 1 - \frac{\rho_p}{\rho_p + q_2} \leq 1 - \frac{\rho_p}{\rho_p + \bar{q}} = \frac{\bar{q}}{\rho_p + \bar{q}}.
\]  

(90)

We proved that for \( \rho_p < q_1 \), the convergence rate of distributed ADMM is upper bounded by \( \frac{\bar{q}}{\rho_p + \bar{q}} \).

\[ \square \]

We further construct a data structure that provides the convergence rate of \( \frac{\bar{q} - (b-2)p_p}{2\rho_p + \bar{q} - (b-2)p_p} \). Consider the data structure with \( \mathbf{D}_1 = \mathbf{D}_2 = \frac{1}{2}(\mathbf{X}^T \mathbf{X} - (b-2)p_p \mathbf{I}) \), and \( \mathbf{D}_j = \rho_p \mathbf{I} \) for \( j \neq 1,2 \). We first show that under such data structure, the spectrum of mapping matrix is upperbounded by \( \frac{\bar{q} - (b-2)p_p}{2\rho_p + \bar{q} - (b-2)p_p} \), then provide a valid eigenvalue-eigenvector pair that achieves such convergence rate. Note by assuming \( \rho_p < p_1 \), \( D_i > 0 \) for all \( i \in \{1,\ldots,b\} \). Following Lemma A.3, let \( \lambda \in \mathbb{R} \) and \( \mathbf{v} = [\mathbf{v}_1;\ldots;\mathbf{v}_b] \in \mathbb{R}^{b \times 1} \) be the eigenvalue eigenvector pair of \( \mathbf{M}_p^T \) under the specific data structure, the following equation holds for each block \( i \in \{1,\ldots,b\} \).

\[
\frac{1}{\rho_p} \Phi_i \mathbf{D}_i \mathbf{v}_i + \frac{1}{b} \sum_{j=1}^{b} \Phi_j \left( I - \frac{1}{\rho_p} \mathbf{D}_j \right) \mathbf{v}_j = \lambda \mathbf{v}_i, \quad \forall \ i.
\]  

(91)

Multiply both side by \( \mathbf{v}_i^T \), with some algebra,

\[
\mathbf{v}_i^T \frac{1}{\rho_p} \Phi_i \mathbf{D}_i \mathbf{v}_1 + \frac{1}{b} \left( \mathbf{v}_i^T \Phi_i \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_1 + \mathbf{v}_i^T \Phi_1 \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_2 \right) = \lambda \mathbf{v}_i^T \mathbf{v}_1,
\]  

(92)

\[
\mathbf{v}_i^T \frac{1}{\rho_p} \Phi_i \mathbf{D}_i \mathbf{v}_2 + \frac{1}{b} \left( \mathbf{v}_i^T \Phi_i \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_1 + \mathbf{v}_i^T \Phi_1 \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_2 \right) = \lambda \mathbf{v}_i^T \mathbf{v}_2,
\]  

(93)

\[
\frac{1}{2} \mathbf{v}_i^T \mathbf{v}_i + \frac{1}{b} \left( \mathbf{v}_i^T \Phi_i \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_1 + \mathbf{v}_i^T \Phi_1 \left( I - \frac{1}{\rho_p} \mathbf{D}_1 \right) \mathbf{v}_2 \right) = \lambda \mathbf{v}_i^T \mathbf{v}_1, \quad \forall \ i \neq 1,2.
\]  

(94)
Assuming unit eigenvector, we have

\[ |\lambda| \leq |v_1^T \left( \frac{1}{b} \Phi_1 + \frac{b}{b-1} \rho_p \Phi_1 D_1 \right) v_1| + |v_2^T \left( \frac{1}{b} \Phi_1 + \frac{b}{b-1} \rho_p \Phi_1 D_1 \right) v_2| + \frac{2}{b} |v_1^T \Phi_1 \left( I - \frac{1}{\rho_p} D_1 \right) v_2| \]

\[ \leq v_1^T \left( \frac{1}{b} \Phi_1 + \frac{b}{b-1} \rho_p \Phi_1 D_1 \right) v_1 + v_2^T \left( \frac{1}{b} \Phi_1 + \frac{b}{b-1} \rho_p \Phi_1 D_1 \right) v_2 + \frac{1}{b} v_1^T \Phi_1 \left( \frac{1}{\rho_p} D_1 - I \right) v_1 + \frac{1}{b} v_2^T \Phi_1 \left( \frac{1}{\rho_p} D_1 - I \right) v_2 \]

\[ \leq v_1^T \left( \frac{1}{\rho_p} \Phi_1 D_1 \right) v_1 + v_2^T \left( \frac{1}{\rho_p} \Phi_1 D_1 \right) v_2 \leq \frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p} \]

(95)

where the second inequality is similar as in equations (74). We further show there exist an eigenvalue eigenvector pair that achieves such convergence rate. Let \((\frac{1}{2} \bar{q} - (b-2)\rho_p, \hat{v})\) be the associated eigenvalue eigenvector pair of \(D_1\), let \(\lambda = \frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}\), and \(v = [\hat{v}; -\hat{v}; 0; \ldots; 0]\), it’s easy to verify that \((\lambda, v)\) is a valid eigenvalue-eigenvector pair of \(M_p^T\).

We next show that the convergence rate of \(\frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}\) is higher than the previous worst case data structure of \(D_i = D_j\) for all \(i, j\). When \(\rho_p < q_1\), following Lemma A.1, the convergence rate under the data structure of \(D_i = D_j\) is given by \(\frac{\bar{q}}{b\rho_p + \bar{q}}\), and it’s sufficient to show

\[ \frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p} \geq \frac{\bar{q}}{b\rho_p + \bar{q}}. \]  

(96)

with some algebra, one needs to show that

\[ \rho_p(\bar{q} - b\rho_p)(b - 2) \geq 0. \]  

(97)

As \(b \geq 0\), and \(\rho_p < q_1\), by Wely’s theorem,

\[ b\rho_p \leq \sum_{i=1}^{b} \lambda(D_i) \leq \lambda(X^T X) \leq \bar{q}. \]  

(98)

Hence \(\frac{\bar{q} - (b-2)\rho_p}{2\rho_p + \bar{q} - (b-2)\rho_p}\) is higher than the previous worst case data structure of \(D_i = D_j\) for all \(i, j\).

A.4. Proof on Proposition 3.5

As \(M_{GD} = I - \rho_p X^T X\), the convergence rate is given by \(\max\{1 - \rho_p q, \rho_p \bar{q} - 1\}\). First, consider \(\rho_p > \bar{q}\), the upper bound on convergence rate of distributed ADMM is \(\frac{b\rho_p}{b\rho_p + \bar{q}}\). And for \(\rho_p > \frac{2}{\sqrt{\bar{q}} + 2}\),

\[ \max\{1 - \rho_p q, \rho_p \bar{q} - 1\} = \rho \bar{q} - 1. \]

It’s easy to verify that for \(\rho_p > s_2 = \frac{2\bar{q} - \bar{q}^2 + \sqrt{4\bar{q}^2 + (\bar{q})^2}}{2\bar{q}}\), \(\frac{b\rho_p}{b\rho_p + \bar{q}} < \rho \bar{q} - 1\). Also, note that \(s_2 > \frac{2}{\sqrt{\bar{q}} + 2} > \bar{q}\), hence \(\rho(M_p) < \rho(M_{GD})\). This implies for step size \(\rho_p > s_2\), fixing same step-size, primal distributed ADMM converges faster than gradient descent for any data structure.
For \( \rho_p < q_1 \), the upper bound on convergence rate of distributed ADMM is \( \frac{d}{\rho_p + q} \). For \( \rho_p < \frac{2}{q+2} \), \( \max\{1 - \rho_p q, \rho_p q - 1\} = 1 - \rho_p q \). It’s also easy to verify that for \( \rho_p < s_1 \), \( 1 - \rho_p q > \frac{d}{\rho_p + q} \). Hence \( \rho(M_p) < \rho(M_GD) \). This implies for step size \( \rho_p < s_1 \), fixing same step-size, primal distributed ADMM converges faster than gradient descent for any data structure.

### A.5. Proofs on primal distributed ADMM and dual distributed ADMM sharing exactly the same convergence rate

**Proof.** We need to show that the dual parallel algorithm could be represented as a linear system with mapping matrix \( M_d \), such that \( M_d = M_p = (I - P - \rho_p(D + \rho_p I)^{-1})(I - 2P) \).

Introducing the auxiliary variables, the dual distributed ADMM solves the following optimization problem under the same partition of blocks with \( X = [X_1; \ldots; X_b] \) and \( y = [y_1; \ldots; y_b] \).

\[
\min_T \frac{1}{2} \sum_{i=1}^{b} t_i^T y_i + y_i^T t_i \\
\text{s.t.} \quad X_i^T t_i - v_i = 0 \\
\quad \sum_{i=1}^{b} v_i = 0
\]  

Let \( \rho_d \) be the step size with respect to the augmented Lagrangian, the augmented Lagrangian of the dual problem is given by

\[
L(t_i, v_i, \beta) = \frac{1}{2} \sum_{i=1}^{b} t_i^T t_i + y_i^T t_i - \beta^T (X_i^T t_i - v_i) + \frac{b \rho_d}{2} (X_i^T t_i - v_i)^T (X_i^T t_i - v_i). 
\]  

(100)

And the updating follows the rule

\[
(\rho_d X_i X_i^T + I)t_i^{k+1} = \rho_d X_i v_i^k + X_i \beta^k - y_i \\
v_i^{k+1} = X_i t_i^{k+1} - \frac{1}{b} \sum_{i=1}^{b} X_i t_i^{k+1} \\
\beta^{k+1} = \beta^k - \frac{\rho_d}{b} \sum_{i=1}^{b} X_i t_i^{k+1}. 
\]  

(101)

Introducing \( \mu_i = X_i^T t_i \), we have updating \( t_i^{k+1} \) is equivalent as solving the following linear equations

\[
t_i^{k+1} + \rho_d X_i \mu_i^{k+1} = \rho_d X_i v_i^k + X_i \beta^k - y_i, \\
\mu_i^{k+1} = X_i^T t_i^{k+1}
\]  

(102)

. Rearranging

\[
t_i^{k+1} = -\rho_d X_i \mu_i^{k+1} + \rho_d X_i v_i^k + X_i \beta^k - y_i, \\
\mu_i^{k+1} = -\rho_d X_i^T X_i \mu_i^{k+1} + \rho_d X_i^T X_i v_i^k + X_i^T X_i \beta^k - X_i^T y_i.
\]  

(103)
And equation (101) is equivalent as
\[
(\rho_d X^T X_i + I)\mu_i^{k+1} = \rho_d X^T X_i v_i^k + X^T X_i \beta_i^k - X^T y_i,
\]
\[
v_i^{k+1} = \mu_i^{k+1} - \frac{1}{b} \sum_{j=1}^b \mu_j^{k+1},
\]
\[
\beta_i^{k+1} = \beta_i^k - \frac{\rho_d}{b} \sum_{j=1}^b \mu_j^{k+1}.
\]
(104)

Let \(\eta_i^{k+1} = \rho_d \mu_i^{k+1} - \beta_i^k\), where \(\mu_i^{k+1} = [\mu_1; \ldots; \mu_i; \ldots; \mu_b] \in \mathbb{R}^{bp \times 1}\), and \(\beta_i^k = [\beta_1^k; \ldots; \beta_i^k; \ldots; \beta_k^k] \in \mathbb{R}^{bp \times 1}\). \(\eta_i^{k+1}\) is sufficient to capture the dynamic of the system. And the system follows
\[
\tilde{\beta}_i^{k+1} = \beta_i^k - \frac{\rho_d}{b} \sum_{j=1}^b \mu_j^{k+1} = -P \eta_i^{k+1},
\]
\[
v_i^{k+1} = \mu_i^{k+1} - P \mu_i^{k+1} = \frac{1}{\rho_d} (I - P) \eta_i^{k+1},
\]
\[
\eta_i^k = \rho_d \mu_i^k - \tilde{\beta}_i^{k-1} = \rho_d v_i^k - \tilde{\beta}_i^k,
\]
(105)

where \(v_i^{k+1} = [v_1^{k+1}; \ldots; v_i^{k+1}; \ldots; v_b^{k+1}]\). Since the system can be represented by \(\eta_i\), let \(c_i = X_i^T y_i\), and \(c = [c_1; \ldots; c_b]\) the mapping of \(\eta_i\) follows
\[
(\rho_d D + I) \mu_i^{k+1} = \rho_d D v_i^k + D \tilde{\beta}_i^k - c,
\]
\[
\mu_i^{k+1} = (\rho_d D + I)^{-1} D (I - P) \eta_i^k - (\rho_d D + I)^{-1} DP \eta_i^k - (\rho_d D + I)^{-1} c,
\]
(106)
\[
= (\rho_d D + I)^{-1} D (I - 2P) \eta_i^k - (\rho_d D + I)^{-1} c.
\]

And
\[
\eta_i^{k+1} = \rho_d \mu_i^{k+1} - \tilde{\beta}_i^k
\]
\[
= \rho_d (\rho_d D + I)^{-1} D (I - 2P) \eta_i^k + P \eta_i^k - \rho_d (\rho_d D + I)^{-1} c
\]
(107)
\[
= [(D + \rho_d I)^{-1} D (I - 2P) + P] \eta_i^k - \rho_d (\rho_d D + I)^{-1} c.
\]

We further have \(M_d\) is given by
\[
M_d = [(D + I / \rho_d)^{-1} D (I - 2P) + P].
\]
(108)

When \(\rho_d \rho_p = 1\),
\[
M_d = [(D + \rho_p I)^{-1} D (I - 2P) + P].
\]
(109)

It’s sufficient to show that \(M_p = (I - P - \rho_p (D + \rho_p I)^{-1})(I - 2P) = M_d\). Notice that \((I - P)(I - 2P) = I - P\), and \(M_d = (I - P - \rho_p (D + \rho_p I)^{-1})(I - 2P)\), it’s sufficient to show that
\[
I - P - \rho_p (D + \rho_p I)^{-1} (I - 2P) = (D + \rho_p I)^{-1} D (I - 2P) + P,
\]
(110)

which is obvious as
\[
I = (D + \rho_p I)(D + \rho_p I)^{-1}
\]
\[
I - \rho_p (D + \rho_p I)^{-1} = (D + \rho_p I)^{-1} D
\]
(111)
\[
I - 2P - \rho_p (D + \rho_p I)^{-1} (I - 2P) = (D + \rho_p I)^{-1} D (I - 2P).
\]
where the second equality holds because \((D + \rho_p I)^{-1}D = D(D + \rho_p I)^{-1}\), as \(D\) and \((D + \rho_p I)^{-1}\) are both symmetric. And by proving \(M_d = M_p\), we finish the proof on Proposition 4.1.

\[ \square \]

A.6. Proof on dual RP ADMM converges faster than dual distributed ADMM under worst-case data structure

Consider the following optimization problem

\[
\begin{align*}
\min_t & \quad \frac{1}{2} t^T t + y^T t \\
\text{s.t.} & \quad X^T t = 0
\end{align*}
\]

(112)

The augmented Lagrangian is thus given by

\[
L(t, \beta) = \frac{1}{2} t^T t + y^T t - \beta^T X^t + \frac{\rho_d}{2} t^T XX^t t
\]

(113)

Each agent \(i\) possesses \((X_i, y_i)\) with \(i = \{1, \ldots, b\}\) agents. Consider the following RP multi-block ADMM algorithm

**Algorithm 5** RP ADMM for solving (112)

**Initialization**: \(t = 0\), step size \(\rho_d \in \mathbb{R}^+\) \(t_i \in [t_1^i; \ldots; t_i^b] \in \mathbb{R}^n\), \(\beta_i \in \mathbb{R}^p\), and stopping rule \(\tau\)

while \(t \leq \tau\) do

random permute update order \(\sigma(b) = [\sigma_1, \ldots, \sigma_i, \ldots, \sigma_b]\)

while \(i \leq b\) do

Agent \(\sigma_i\) updates \(t_{i+1}^\sigma\) by

\[
t_{i+1}^\sigma = \arg\min_{t^\sigma_i} L(t_1^i; \ldots; t_{i+1}^{\sigma_i-1}; t_i^{\sigma_i}; t_{i+1}^{\sigma_i+1}; \ldots; t_b^i, \beta_i)
\]

end while

Decision maker updates \(\beta_{i+1} = \beta_i - \rho_d X^T t_{i+1}\)

end while

**Output**: \(\beta_\tau\) as global estimator

We prove that when \(\rho_d = 1\), under worst case data structure, dual RP ADMM in expectation converges faster than dual distributed ADMM for 2, 3 and 4 block ADMM. Our proving technique requires solving the polynomial function of degree equals to number of blocks, and we focus on the case where the polynomials have analytical solutions. The reason we take \(\rho_d = 1\) is because, when \(\rho_d = \rho_p = 1\), the dual distributed ADMM shares exactly same convergence rate as primal distributed ADMM, and we could utilize the previous theorem in order to fairly compare the convergence rate of primal algorithm and dual algorithm by separating the effect of step-size choice.
Theorem A.4. For \( \rho_p = \rho_d = 1 \), under the data structure of \( D_i = D_j \) for all \( i, j \in \{1, \ldots, b\} \), the expected convergence rate of dual RP-ADMM is smaller than the convergence rate of distributed ADMM for \( b \in \{2, 3\} \).

The sketch of proof is as follows. To prove Theorem A.4, we show that for any random permuted update order across blocks, the spectrum of linear mapping matrix under cyclic ADMM is upper bounded and is smaller than the distributed ADMM. We then use Weyl’s theorem to show that the spectrum of the expected mapping matrix of RP-ADMM is upper bounded by the average of the spectrum of cyclic ADMM. In order to prove Theorem A.4, we first introduce the following theorem to provide the tight upper bound of linear convergence rate of cyclic ADMM.

Theorem A.5. For \( \rho_d = 1 \), under the data structure of \( D_i = D_j \) for all \( i, j \in \{1, \ldots, b\} \), the convergence rate of dual cyclic ADMM \( \rho(M_c) \) is unique solution to the function \( f(x) = \frac{x}{1-x} \left( 1 - \left( \frac{2x-1}{x^2} \right)^{1/b} \right) \).

with \( b \in \{2, 3\} \). Moreover, dual cyclic ADMM converges faster than distributed ADMM under such data structure with \( \rho_p = \rho_d = 1 \).

Proof. Without loss of generosity, in this proof we consider the ascending update order from block 1 to block \( b \). Similarly, introducing \( \mu_i = X_i^T t_i \), we have at period \( k \), updating \( t_i^{k+1} \) and \( \beta^{k+1} \) is equivalent as

\[
\rho_d X_i^T X_i \left( \sum_{j=1}^{i} \mu_j^{k+1} + \sum_{j=i+1}^{b} \mu_j^k \right) + \mu_i^{k+1} = X_i^T X_i \beta^k - X_i^T y_i;
\]

(114)

\[
\beta^{k+1} = \beta^k - \rho_d \sum_{i=1}^{b} \mu_i^{k+1}.
\]

We first show that Let \( L \) be the lower block triangular matrix with \( L_{i,j} = X_i^T X_j \) for \( j \leq i \), and \( L_{i,j} = 0 \) for \( j > i \). For example, when \( b = 3 \), we have

\[
L = \begin{bmatrix}
X_1^T X_1, & 0, & 0 \\
X_1^T X_2, & X_2^T X_2, & 0 \\
X_1^T X_3, & X_2^T X_3, & X_3^T X_3
\end{bmatrix}.
\]

(115)

Following the same definition on \( P \) and \( D \), and let \( E = [I_p; \ldots; I_p] \in \mathbb{R}^{p \times b} \), \( \mu^{k+1} = [\mu_1^{k+1}; \ldots; \mu_b^{k+1}] \in \mathbb{R}^{b_p \times 1} \), the previous updating system could be written as

\[
\begin{bmatrix}
I + \rho_d L, & 0 \\
0, & I + \rho_d E E^T
\end{bmatrix} \begin{bmatrix}
\mu_k^{k+1} \\
\beta_k^{k+1}
\end{bmatrix} = \begin{bmatrix}
(I + \rho_d L) - (I + \rho_d b \Phi P) \Phi E & I \\
0 & I
\end{bmatrix} \begin{bmatrix}
\mu_k^k \\
\beta_k^k
\end{bmatrix} - \begin{bmatrix}
X_i^T y_i \\
0
\end{bmatrix}.
\]

(116)

And the linear mapping matrix of cyclic updating is given by

\[
M_c = \begin{bmatrix}
(I + \rho_d L)^{-1}, & 0 \\
-\rho_d E E^T (I + \rho_d L)^{-1}, & I
\end{bmatrix} \begin{bmatrix}
(I + \rho_d L) - (I + \rho_d b \Phi P) \Phi E & I \\
0 & I
\end{bmatrix}.
\]

(117)
By the fact $eig(AB) = eig(BA)$ for matrix $A, B \in \mathbb{R}^{n \times n}$, it's sufficient to consider the eigenvalue of the following matrix

$$M'_c = \begin{bmatrix} (I + \rho_d L) - (I + \rho_d \Phi P) & \Phi E \\ 0 & I \end{bmatrix} \begin{bmatrix} (I + \rho_d L)^{-1}, & 0 \\ -\rho_d E^T(I + \rho_d L)^{-1}, & I \end{bmatrix} = \begin{bmatrix} I - (I + 2\rho_d \Phi P)(I + \rho_d L)^{-1} \Phi E & \rho_d E^T(I + \rho_d L)^{-1} I \end{bmatrix}.$$ (118)

We have, when $\rho_d = 1$,

$$M'_c = \begin{bmatrix} (L - 2\rho \Phi P)(I + L)^{-1} \Phi E \\ -E^T(I + L)^{-1} & I \end{bmatrix}.$$ (119)

Let $v = [v_1; v_2]$ be the associated eigenvector pair of $\lambda$, the following equations holds

$$(L - 2\rho \Phi P)(I + L)^{-1}v_1 + \Phi E v_2 = \lambda v_1,$$

$$-E^T(I + L)^{-1}v_1 + v_2 = \lambda v_2.$$ (120)

Under the data structure of $D_i = D_j$ for all $i, j \in \{1, \ldots, b\}$, we first prove that $\lambda \neq 1$. Suppose $\lambda = 1$, we have

$$-E^T(I + L)^{-1}v_1 = 0,$$ (121)

which implies

$$L(I + L)^{-1}v_1 + \Phi E v_2 = v_1.$$ (122)

Firstly, $v_1 \neq 0$, if $v_1 = 0$, we have $\Phi E v_2 = 0$, which implies $v_2 = 0$, and that contradicts to $v = [v_1; v_2]$ being an eigenvector. Let $m = (I + L)^{-1}v_1$, we have $\Phi E v_2 = m$, and by equation (120), $Pm = 0$, and $P\Phi E v_2 = \Phi E v_2$, hence $P\Phi E v_2 = \Phi E v_2 = Pm = 0$, which implies $v_2 = 0$, and we have $m = 0$. This contradicts to $v = [v_1; v_2]$ being an eigenvector, as $m = 0$ implies $v_1 = 0$ given $(I + L)^{-1} \succ 0$.

Since $\lambda \neq 1$, we have

$$v_2 = \frac{1}{1 - \lambda} E^T(I + L)^{-1}v_1,$$ (123)

substituting $v_2$ into previous equation,

$$(L - 2b \Phi P)(I + L)^{-1}v_1 + \frac{1}{1 - \lambda} \Phi EE^T(I + L)^{-1}v_1 = \lambda v_1.$$ (124)

We claim that $v_1 \neq 0$, if $v_1 = 0$, $v_2 = \lambda v_2$, since $\lambda \neq 1$, $v_2 = 0$ which contradicts to the fact that $[v_1, v_2]$ is an eigenvector. We then introduce $m = [m_1; \ldots; m_b] = (I + L)^{-1}v_1 \neq 0$ where $m_i \in \mathbb{R}^{p \times 1}$.

With some algebra,

$$(1 - \lambda)^2 L m + (2\lambda - 1)b \Phi P m = \lambda(1 - \lambda)m.$$ (125)

Let $D = \frac{X^T X}{b} = D_i$ for all $i$, this implies for all $i \in \{1, \ldots, b\}$, we have the following equations holds

$$(1 - \lambda)^2 D \sum_{j=1}^i m_j + (2\lambda - 1)D \sum_{j=1}^b m_j = \lambda(1 - \lambda)m_i \quad \forall i,$$ (126)
and

\[ m_{i-1} = \left( I - \frac{1-\lambda}{\lambda} \bar{D} \right) m_i. \]  

\( (126) \)

We first show that \( m_b \neq 0 \). Suppose \( m_b = 0 \), we have from the \( b^{th} \) equation in (125),

\[ \lambda^2 \bar{D} \sum_{j=1}^{b} m_j = 0. \]

\( (127) \)

Since we consider non-zero eigenvalues, and by the fact \( \bar{D} \succ 0 \), equation (127) implies that \( \sum_{j=1}^{b} m_j = 0 \) which further implies for the \((b-1)^{th}\) equation, we have

\[ (1-\lambda)^2 \bar{D} \sum_{j=1}^{b-1} m_j = \lambda(1-\lambda)m_{b-1}. \]

\( (128) \)

Since \( m_b = 0 \), \( \sum_{j=1}^{b-1} m_j = \sum_{j=1}^{b} m_j = 0 \), and \( m_{b-1} = 0 \), by the induction we would have \( m = [m_1; \ldots; m_b] = 0 \) which contradicts to the fact that \( m \neq 0 \). With some algebra, equations (125) implies that

\[ \frac{\lambda}{1-\lambda} \bar{D} \sum_{j=1}^{b} \left( I - \frac{1-\lambda}{\lambda} \bar{D} \right)^{b-j} m_b = m_b. \]

\( (129) \)

Let \( M_{\lambda} = \frac{\lambda}{1-\lambda} \bar{D} \sum_{j=1}^{b} \left( I - \frac{1-\lambda}{\lambda} \bar{D} \right)^{b-j} \), if \( \lambda \in \text{eig}(M_c) \), \( 1 \in \text{eig}(M_{\lambda}) \). Moreover, since \( M_{\lambda} \) is a polynomial function of \( \bar{D} \), let \( p \in \text{eig}(\bar{D}) \), the eigenvalue of \( M_{\lambda} \) is given by

\[ \frac{\lambda^2}{(1-\lambda)^2} \left( 1 - \left( 1 - \frac{1-\lambda}{\lambda} p \right)^b \right). \]

\( (130) \)

And if \( \lambda \neq 1 \in \text{eig}(M_c) \), \( \lambda \) is the solution of the following equation with \( p \in (0, \frac{1}{b}) \)

\[ f(\lambda) = \left( 1 - \frac{1-\lambda}{\lambda} p \right)^b - \frac{2\lambda - 1}{\lambda^2}. \]

\( (131) \)

Rearranging previous polynomial, the largest solution (in absolute value) to equation (131) which is not equal to 1 is given by the unique solution to the function \( f(x) = \frac{q}{2} \), where

\[ g(x) = \frac{x}{1-x} \left( 1 - \left( \frac{2x - 1}{x^2} \right)^{1/b} \right). \]

\( (132) \)

Also one could verify that for \( b = 2, 3 \), the largest solution (in absolute value) is given by

\[ x_2 = \frac{1+q^2}{(1+q)^2}, \]

\[ x_3 = \frac{1+3q^2+2q^3+\sqrt{1+6q^2-3q^4}}{2(1+3q+3q^2+q^3)}. \]

\( (133) \)
One could further check for $q \in (0, \frac{1}{b})$, $1 + 6q^2 - 3q^4 > 0$. Further, for distributed ADMM, by Theorem 3.2, for $\rho_p = 1$ the spectrum of distributed ADMM mapping matrix is given by $\frac{b}{b+q}$. One can check for $b = 2$, $\frac{b}{b+q} - \frac{1+q^2}{(1+q)^2} = \frac{q(3-q^2)}{(1+q)^2(2+q)} > 0$, and for $b = 3$, $\frac{b}{b+q} - \frac{1+q^2+2q^3+\sqrt{1+6q^2-3q^4}}{2(1+3q^2+3q^4+q^6)} = \frac{3+q(17+q(3+q)(3-2q))-(3+q)(\sqrt{1+6q^2-3q^4})}{(3+q)(2(1+q)^3)} > 0$, where $(3+q(17+q(3+q)(3-2q))^2-(3+q)(\sqrt{1+6q^2-3q^4})^2 = 4q(1+q)^3(24-9q^2+q^4) > 0$. Hence, dual cyclic ADMM converges faster than distributed ADMM under such data structure with $\rho_p = \rho_d = 1$.

While we conjecture similar result holds for general $b$, there is no explicit expression for the solution of higher order polynomials. One could further show that the expected mapping matrix of RP-ADMM is the average of cyclic ADMM with different update orders across blocks, and for each specific update order, by Theorem A.5, the spectrum of cyclic ADMM mapping matrix is upper bounded by the spectrum of distributed ADMM mapping matrix. Hence, by Weyl’s theorem, the expected spectrum of RP-ADMM mapping matrix is smaller than the spectrum of distributed ADMM mapping matrix. With the theory guidance, when designing DRAP-ADMM, we apply the random updated order across block as random permuted.