ABSTRACT

Title of Dissertation: CONTROL THEORY-INSPIRED ACCELERATION OF THE GRADIENT-DESCENT METHOD: CENTRALIZED AND DISTRIBUTED

Kushal Chakrabarti
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Nikhil Chopra
Department of Electrical and Computer Engineering

Mathematical optimization problems are prevalent across various disciplines in science and engineering. Particularly in electrical engineering, convex and non-convex optimization problems are well-known in signal processing, estimation, control, and machine learning research. In many of these contemporary applications, the data points are dispersed over several sources. Restrictions such as industrial competition, administrative regulations, and user privacy have motivated significant research on distributed optimization algorithms for solving such data-driven modeling problems. The traditional gradient-descent method can solve optimization problems with differentiable cost functions. However, the speed of convergence of the gradient-descent method and its accelerated variants is highly influenced by the conditioning of the optimization problem being solved. Specifically, when the cost is ill-conditioned, these methods (i) require many iterations to converge and (ii) are highly unstable against process noise. In this dissertation, we propose novel optimization algorithms, inspired by control-theoretic tools, that can significantly

attenuate the influence of the problem's conditioning.

First, we consider solving the linear regression problem in a distributed server-agent network. We propose the Iteratively Pre-conditioned Gradient-Descent (IPG) algorithm to mitigate the deleterious impact of the data points' conditioning on the convergence rate. We show that the IPG algorithm has an improved rate of convergence in comparison to both the classical and the accelerated gradient-descent methods. We further study the robustness of IPG against system noise and extend the idea of iterative pre-conditioning to stochastic settings, where the server updates the estimate based on a randomly selected data point at every iteration. In the same distributed environment, we present theoretical results on the local convergence of IPG for solving convex optimization problems. Next, we consider solving a system of linear equations in peer-to-peer multi-agent networks and propose a decentralized pre-conditioning technique. The proposed algorithm converges linearly, with an improved convergence rate than the decentralized gradient-descent. Considering the practical scenario where the computations performed by the agents are corrupted, or a communication delay exists between them, we study the robustness guarantee of the proposed algorithm and a variant of it. We apply the proposed algorithm for solving decentralized state estimation problems. Further, we develop a generic framework for adaptive gradient methods that solve non-convex optimization problems. Here, we model the adaptive gradient methods in a state-space framework, which allows us to exploit control-theoretic methodology in analyzing Adam and its prominent variants. We then utilize the classical transfer function paradigm to propose new variants of a few existing adaptive gradient methods. Applications on benchmark machine learning tasks demonstrate our proposed algorithms' efficiency. Our findings suggest further exploration of the existing tools from control theory in complex machine learning problems.

The dissertation is concluded by showing that the potential in the previously mentioned idea of IPG goes beyond solving generic optimization problems through the development of a novel distributed beamforming algorithm and a novel observer for nonlinear dynamical systems, where IPG's robustness serves as a foundation in our designs. The proposed IPG for distributed beamforming (IPG-DB) facilitates a rapid establishment of communication links with far-field targets while jamming potential adversaries without assuming any feedback from the receivers, subject to unknown multipath fading in realistic environments. The proposed IPG observer utilizes a non-symmetric pre-conditioner, like IPG, as an approximation of the observability mapping's inverse Jacobian such that it asymptotically replicates the Newton observer with an additional advantage of enhanced robustness against measurement noise. Empirical results are presented, demonstrating both of these methods' efficiency compared to the existing methodologies.

CONTROL THEORY-INSPIRED ACCELERATION OF THE GRADIENT-DESCENT METHOD: CENTRALIZED AND DISTRIBUTED

by

Kushal Chakrabarti

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Nikhil Chopra, Chair/Advisor
Professor Nuno Martins
Professor Richard J. La
Professor André Tits
Professor Pratap Tokekar

to my beloved Ishita

Acknowledgments

This dissertation is one of my proud possession and it wont be in its final form without the help of others.

First and foremost, I'd express my sincere gratitude to my advisor, Professor Nikhil Chopra, for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past five years. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. It has been a pleasure to work with and learn from such an extraordinary individual.

Thanks are due to Professor Nuno Martins, Professor Richard La, Professor André Tits, and Professor Pratap Tokekar for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

Also, thanks to other faculty members and teaching assistants in the university for the course work which established my foundation for research.

My colleagues have enriched my professional life in many ways and deserve a special mention. Especially, a significant part of this research was in collaboration with Nirupam Gupta, Jeffrey N. Twigg, Fikadu T. Dagefu, and Amrit S. Bedi.

I would like to acknowledge financial support from the Petroleum Institute at UAE, the United States Department of Agriculture, and the Army Research Laboratory, for part of the projects discussed herein.

I want to thank my friends from here, for making my stay memorable. My warmest thanks go to my family for their support, love, encouragement, and patience. And at the last from Einstein's words:

Thanks to those who said NO !! because of them I did it myself.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Mathematical optimization is a common sub-problem in various disciplines, such as modeling, estimation, control systems, wireless communication, data analysis, finance, and operations research. Especially with the recent data-driven technological advancements, optimization is ubiquitous in several applications. A simple optimization technique, known as least-squares, dates back to 1795 A.D. when it was invented by a young scientist Karl Friedrich Gauss [1] for studying planetary motion using telescopic measurement data. Since then, several new optimization methodologies have been developed. The most attractive feature of optimization lies in the constructive formulation of a problem, followed by the availability of sophisticated and thoroughly studied tools to solve the formulated problem in a reliable and efficient manner. Therefore, the focus of this dissertation will be on developing novel tools for solving a class of optimization problems, aimed at balancing the decisive triad of performance, efficiency, and reliability.

Specifically, this dissertation will consider solving unconstrained optimization problems, where the objective is to minimize a smooth and differentiable cost function. The classical gradient-descent algorithm, attributed to Cauchy (1847 A.D.), is the basic first-order optimization algorithm to solve such problems. The gradient-descent algorithm is iterative, wherein the idea is to repeatedly refine an estimate of the minimum point by taking steps in the opposite direction of the gradient of the cost function at the current estimate [2]. In principle, the gradient-descent

method eventually converges to a minimum point of the cost. However, the speed of convergence of the gradient-descent method is highly influenced by the conditioning of the optimization problem being solved. Specifically, when the cost is ill-conditioned, this method (i) requires many iterations to converge and (ii) is highly unstable against process noise. This dissertation aims at developing novel optimization algorithms, inspired by classical tools from control theory, that can significantly attenuate the fundamental influence of the problem's conditioning.

In many contemporary applications, the data points are dispersed over several sources (or agents). Due to industrial competition, administrative regulations, and user privacy, it is nearly impossible to collect the individual data from the different agents at a single machine [3]. In recent years, these restrictions have motivated a significant amount of research on distributed algorithms for solving data-driven modeling problems. In most distributed algorithms, an individual agent is not required to transmit its raw data points [3, 4]. In the aforementioned cases, the data itself is distributed over multiple sources or agents. With the availability of a large amount of data, it is sometimes necessary to distribute the processing task over multiple machines or agents due to the inherent complexity of the problem regarding computational and/or memory requirements [5]. Therefore, apart from centralized optimization problems, this dissertation will also consider solving optimization problems with a differentiable cost function in a distributed manner. In Chapters 2-5, we propose our algorithms for solving specific optimization problems in centralized settings and in distributed network of multiple agents. In Chapters 6-7, we go beyond solving generic optimization problems by considering the distributed beamforming problem and the nonlinear observer problem, wherein we develop new techniques for these two problems by leveraging an idea from Chapter 4.

In Chapter 2, we consider the multi-agent linear least-squares problem in a server-agent

network architecture. The system comprises multiple agents, each with a set of local data points. The agents are connected to a server, and there is no inter-agent communication. The agents' goal is to compute a linear model that optimally fits the collective data. The agents, however, cannot share their data points. In principle, the agents can solve this problem by collaborating with the server using the server-agent network variant of the classical gradient-descent method. However, when the data points are ill-conditioned, the gradient-descent method requires a large number of iterations to converge. We propose a novel iterative pre-conditioning technique to mitigate the deleterious impact of the data points' conditioning on the convergence rate of the gradient-descent method. The idea of iterative pre-conditioning is inspired by adaptive control techniques. We rigorously show that our proposed algorithm converges with an improved rate of convergence in comparison to both the classical and the accelerated gradient-descent methods. In the presence of system uncertainties, we characterize the proposed algorithm's robustness against noise. We further extend our algorithm to the stochastic settings, where the server updates the estimate of a minimum point based on a single randomly selected data point at every iteration instead of the full batch of data points. Through numerical experiments on benchmark least-squares problems, we validate our theoretical findings. This work was carried out under the PSIM Project, supported by the Petroleum Institute, Khalifa University of Science and Technology, Abu Dhabi, UAE. The research works in this chapter have been published in [6–9].

In Chapter 3, we consider solving the linear least-squares problem in peer-to-peer multi-agent networks. Again, each agent has a set of local data points, and its goal is to compute a linear model that fits the collective data points. In principle, the agents can apply the decentralized gradient-descent method. However, when the data matrix is ill-conditioned, gradient-descent requires many iterations to converge and is unstable against system noise. We propose a decentralized

pre-conditioning technique to address these issues with the decentralized gradient-descent method. We show that the proposed algorithm has an improved convergence rate than the decentralized gradient-descent. Considering the practical scenario where the computations performed by the agents are corrupted, we also study the robustness guarantee of the proposed algorithm. Additionally, we apply the proposed algorithm for solving decentralized linear state estimation problems. The empirical results show our proposed state predictor's favorable convergence rate and robustness against system noise compared to prominent decentralized algorithms. When the communication links between agents are subject to potentially large but constant delays, we utilize a similar decentralized pre-conditioning technique to propose a delay-tolerant algorithm that solves the decentralized linear equations. Empirical results show that the accuracy of the solution obtained by the proposed algorithm is better or comparable to the existing methods. This work was partially supported by the USDA grant 2020-68012-31085 and partially carried out under the PSIM Project, supported by the Petroleum Institute, Khalifa University of Science and Technology, Abu Dhabi, UAE. The research works in this chapter have been published in [10, 11].

In Chapter 4, we study a distributed multi-agent convex optimization problem. The system architecture is the same as Chapter 2. However, the cost function here is convex and not limited to quadratic cost. The agents' goal is to learn a parameter vector that optimizes the aggregate of their local costs without revealing their local data points. We extend the idea of iterative pre-conditioning, proposed in Chapter 2, to convex cost functions and a class of non-convex functions. We present theoretical results on the local convergence of the proposed algorithm. We demonstrate our algorithm's superior performance to prominent distributed algorithms for solving real logistic regression problems and emulating neural network training via a noisy quadratic model, thereby signifying the proposed algorithm's efficiency for distributively solving

non-convex optimization. Further worth of the idea of IPG is investigated later in Chapter 6-7.

In Chapter 5, we consider solving smooth non-convex optimization problems, with a focus on deep neural network models. We aim to develop a generic framework for adaptive gradient-descent algorithms for solving non-convex problems in the continuous-time domain. Particularly, we model the adaptive gradient methods in a state-space framework, which allows us to exploit standard control-theoretic tools in analyzing existing prominent adaptive methods. We rigorously show that few other algorithms that are not encompassed by the proposed generic framework, can also be modeled and analyzed using a similar technique. The analyses of all all these fast gradient algorithms are unified by a common underlying proof sketch, relying upon Barbalat's lemma. Control-theoretic tools can also aid in developing novel adaptive gradient algorithms, as we show by proposing new variants of three existing adaptive gradient methods, using the concept of transfer functions. Applications on benchmark machine learning tasks demonstrate the proposed algorithms' efficiency. The findings in this chapter suggest further exploration of the existing tools from control theory in complex machine learning problems. Part of the research works in this chapter have been published in [12] and accepted for presentation at [13, 14].

Distributed multi-agent beampattern matching inherently enables covert communication by constructively combining the transmitted signals at target receivers and forming nulls towards the adversaries. Most existing beamforming methodologies achieve this by relying on partial or full-state real-time feedback from the receivers. Chapter 6 proposes a novel distributed beamforming technique that does not assume any feedback from the receivers or channel parameters such as multipath fading. The proposed algorithm works in a server-agent architecture of the beamforming agents, eliminating the need for receivers' feedback. Our algorithm is built on the classical gradient-descent (GD) method. However, when the problem is ill-conditioned, GD requires

many iterations to converge and is unstable against system noise. We propose an iterative pre-conditioning technique, founded upon the IPG algorithm proposed in Chapter 4, to mitigate the deleterious effects of the data points' conditioning on the convergence rate, facilitating a rapid establishment of communication links with far-field targets. The empirical results demonstrate the proposed beamforming algorithm's favorable convergence rate and robustness against unknown multipath fading in realistic environments. This work is part of an ArtIAMAS project[1] and is being partially funded by ARL Grant No. W911NF2120076. The results in this chapter has been accepted for presentation at [15].

Nonlinear observers are required for process monitoring, fault detection, signal reconstruction, and control in various applications. The previously proposed Newton observer has fast exponential convergence and applies to a wide class of problems. However, the Newton observer lacks robustness against measurement noise due to the computation of a matrix inverse at each step. In Chapter 7, we propose a novel observer for discrete-time plant dynamics with sampled measurements to alleviate the impact of measurement noise. The key to the proposed observer is an iterative pre-conditioning technique for the gradient-descent method, proposed in Chapter 4, for solving general optimization problems. The proposed observer utilizes a non-symmetric pre-conditioner to approximate the observability mapping's inverse Jacobian so that it asymptotically replicates the Newton observer with an additional advantage of enhanced robustness against measurement noise. Our observer applies to a wide class of nonlinear systems, as it is not contingent upon linearization or any specific structure in the plant nonlinearity and the measurement mapping. Its improved robustness against measurement noise than the prominent nonlinear observers is demonstrated through empirical results. Its relation with extended Kalman filter is also discussed.

[1] https://artiamas.umd.edu/

Finally, Chapter 8 summarizes the completed work and presents the future research directions resulting from this dissertation. Detailed proof of the theoretical results in this dissertation is presented in Appendix A.

Chapter 2: Distributed Linear Regression in Server-Agent Network

2.1 Introduction

In this chapter, we consider the multi-agent distributed linear least-squares problem. The nomenclature *distributed* here refers to the *data* being distributed across multiple agents. Specifically, we consider a system comprising multiple agents, each agent having a set of *local* data points. The agents can communicate bidirectionally with a central server, as shown in Fig. 2.1. However, there is no inter-agent communication, and the agents cannot share their local data points with the server. The agents' goal is to generate a linear mathematical model that optimally fits the data points held by all the agents. To solve this problem, as an individual agent cannot access the collective data points, the agents collaborate with the central server. Henceforth, we will refer to the above-described system architecture as *server-agent network*. Also, throughout this chapter, the system is assumed synchronous.

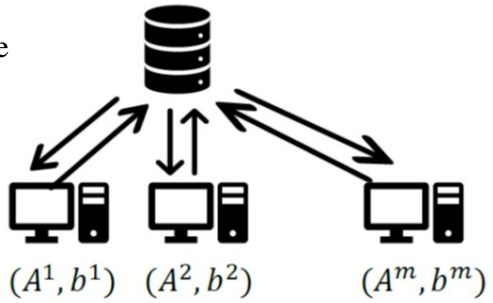

Figure 2.1: Distributed system architecture.

To be precise, we consider m agents in the system. Each agent i has a set of n_i data points represented by the rows of a $(n_i \times d)$-dimensional real-valued *local data matrix* A^i, and the elements of a n_i-dimensional real-valued *local observation vector* b^i. Typically, $n_i > d$ for each

i. The goal for the agents is to compute a *parameter vector* $x^* \in \mathbb{R}^d$ such that

$$x^* \in X^* = \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} \frac{1}{2} \left\| A^i x - b^i \right\|^2, \tag{2.1}$$

where $\|\cdot\|$ denotes the Euclidean norm. For each agent i, we define a *local cost function*

$$F^i(x) = \frac{1}{2} \left\| A^i x - b^i \right\|^2, \quad \forall x \in \mathbb{R}^d. \tag{2.2}$$

Note that solving for the optimization problem (2.1) is equivalent to computing a minimum point of the *aggregate cost function* $\sum_{i=1}^{m} F^i(x)$. An algorithm that enables the agents to jointly solve the above problem in the architecture of Fig. 2.1 without sharing their data points is defined as a *distributed algorithm*.

Common applications of the above linear least-squares problem include linear regression, state estimation, and hypothesis testing [16], [17]. Also, a wide range of supervised machine learning problems can be modeled as a linear least-squares problem, such as the supply chain demand forecasting [18], prediction of online user input actions [19], and the problem of selecting sparse linear solvers [20]. In many contemporary applications, the data points exist as dispersed over several sources (or agents). Due to industrial competition, administrative regulations, and user privacy, it is nearly impossible to collect the individual data from different agents at a single machine (or server) [3]. These restrictions have motivated a significant amount of research in recent years on distributed server-agent algorithms for solving data-driven modeling problems, such as (2.1) above [3], [4]. Herein lies our motivation to improve upon the state-of-the-art methods for solving (2.1) in a server-agent network.

To be able to present our key contributions, we review below the server-agent version of the traditional gradient-descent method [2].

2.1.1 Background on gradient-descent method

The *gradient-descent* (GD) method is an iterative algorithm wherein the server maintains an estimate of a minimum point, defined by (2.1), and updates it iteratively using the gradients of individual agents' local cost functions. To be precise, for each iteration $t = 0, 1, \ldots$, let $x(t) \in \mathbb{R}^d$ denote the estimate maintained by the server. The initial estimate $x(0)$ may be chosen arbitrarily from $\mathbb{R}^d$. For each iteration t, the server broadcasts $x(t)$ to all the agents. Each agent i computes the gradient, denoted by $g^i(t)$, of its local cost function $F^i(x)$ at $x(t)$. Specifically, for all $i \in \{1, \ldots, m\}$ and for all $t \in \{0, 1, \ldots\}$,

$$g^i(t) = \nabla F^i(x(t)) = \left(A^i\right)^T \left(A^i\, x(t) - b^i\right). \tag{2.3}$$

Here, $(\cdot)^T$ denotes the transpose. The agents send their computed gradients $\{g^i(t),\ i = 1, \ldots, m\}$ to the server. Upon receiving the gradients, the server updates $x(t)$ to $x(t+1)$ according to

$$x(t+1) = x(t) - \delta \sum_{i=1}^{m} g^i(t),\ t \in \{0, 1, \ldots\}, \tag{2.4}$$

where δ is a positive scalar real value commonly referred as the *step-size*. Let $g(t)$ denote the sum of all the agents' gradients, that is, $g(t) = \sum_{i=1}^{m} g^i(t)$. So, $g(t)$ is the gradient of the aggregate cost defined in (2.1). Substituting from above in (2.4), we obtain that convergence of the above server-agent version of the GD method is identical to its centralized counterpart with cost function

$\sum_{i=1}^{m} F^i(x)$.

It is known that for small enough step-size δ, the gradients $\{g(t),\ t = 0,\ 1, \ldots\}$ converge *linearly* to 0_d, the d-dimensional zero vector. Specifically, for δ small enough, there exists $\mu \in [0,\ 1)$ such that $\left\|g(t)\right\| \leq \mu^t \left\|g(0)\right\|, \quad \forall t \geq 0$ [21]. Commonly, the value μ is referred as the *convergence rate* [22]. The smaller the value of μ faster is the convergence, and vice-versa.

We propose an *iterative pre-conditioning* technique that improves upon the GD method's convergence rate in a server-agent network. Specifically, in each iteration, the server multiplies the aggregate of the agents' gradients $g(t)$ by a *pre-conditioner matrix* K before updating the local estimates. However, unlike the classical pre-conditioning techniques [21], the server iteratively updates the pre-conditioner matrix K. Hence, the name *iterative pre-conditioning*.

2.1.2 Related work

In his seminal work, Nesterov showed that the use of *momentum* could significantly *accelerate* the GD method [23]. Recently, there has been work on the applicability of the NAG method to the server-agent network, such as [5] and references therein. Azizan-Ruhi et al. [5] have proposed the APC method, a combination of the NAG method with a projection operation. Azizan-Ruhi et al. have shown through experiments that their APC method converges faster compared to the variants of the NAG and HBM [24]. However, they do not provide any theoretical guarantee for the improvement in the convergence speed. Also, Azizan-Ruhi et al. only consider a degenerate case of the optimization problem (2.1) where the set of linear equations $A^i x = b^i,\ i = 1, \ldots, m$, has a unique solution. We consider a more general setting wherein the minimum value of the aggregate cost function $\sum_{i=1}^{m} F^i(x)$ may not be zero, and the solution of the optimization problem (2.1) may

not be unique.

The HBM is another momentum-based accelerated variant of the GD method [24]. For the case when the optimization problem (2.1) has a unique solution, both these accelerated methods, namely HBM and GD, are known to converge linearly with a rate of convergence smaller than the above traditional GD method [25], [26].

Newton's method converges *quadratically*, hence *superlinearly* [22]. However, Newton's method does not apply to the server-agent networked system unless the agents share their local data points with the server. A quasi-Newton method, on the other hand, such as BFGS applies to the server-agent networks [22]. However, similar to Newton's method, BFGS also requires the solution of the optimization problem (2.1) to be unique. We consider a setting where the solution of the problem (2.1) may not be unique.

2.1.3 System noise

These distributed algorithms are iterative wherein the server maintains an estimate of a solution defined by (2.1), which is updated iteratively using the gradients of the individual agents' local cost functions defined in (2.2). In an ideal scenario with no noise, these algorithms converge to an optimal regression parameter defined in (2.1). Practical systems, however, inevitably suffers from uncertainties or *noise* [27,28]. Specifically, we consider two types of additive system noises, 1) *observation noise*, and 2) *process noise*. The *observation noise*, as the name suggests, models the uncertainties in the local data points observed by the agents [29]. The *process noise* models the uncertainties or jitters in the computation process due to hardware failures, quantization errors, or noisy communication links [28].

In this dissertation, we empirically show that the approximation error, or the *robustness*, of our method in the presence of the above system noises compares favorably to all the other aforementioned prominent distributed algorithms. Besides empirical results, we also present formal analyses on the proposed method's robustness when negatively impacted by additive system noises. Robustness analyses of some of the other aforementioned distributed algorithms are in [30, 31].

2.1.4 Stochastic settings

We also consider solving the distributed linear least-squares problem using stochastic algorithms. In particular, each agent i has n *local* data points, represented by a local data matrix A^i and a local observation vector b^i of dimensions $n \times d$ and $n \times 1$, respectively. Thus, for all $i \in \{1, \ldots, m\}$, $A^i \in \mathbb{R}^{n \times d}$ and $b^i \in \mathbb{R}^n$. For each agent i, we define a *local cost function* $F_i : \mathbb{R}^d \to \mathbb{R}$ such that for a given *parameter* vector $x \in \mathbb{R}^d$,

$$F^i(x) = \frac{1}{2n} \left\| A^i x - b^i \right\|^2. \tag{2.5}$$

The agents' objective is to compute an optimal parameter vector $x^* \in \mathbb{R}^d$ such that

$$x^* \in \arg\min_{x \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} F^i(x). \tag{2.6}$$

There are several theoretical and practical reasons for solving the distributed problem (2.6) using stochastic methods rather than batched optimization methods, particularly when the number of data-points is abundant [32]. The basic prototype of the stochastic optimization methods that

solve (2.1) is the traditional stochastic gradient (SGD) [32]. Several accelerated variants of the stochastic gradient descent algorithm have been proposed in the past decade [33–38]. A few of such well-known methods are the adaptive gradient descent (AdaGrad) [33], adaptive momentum estimation (Adam) [34], AMSGrad [37]. These algorithms are stochastic, wherein the server maintains an estimate of a solution defined by (2.6), which is refined iteratively by the server using the *stochastic gradients* computed by a randomly chosen agent.

In particular, Adam has been demonstrated to compare favorably with other stochastic optimization algorithms for a wide range of optimization problems. However, Adam updates the current estimate effectively based on only a window of the past gradients due to the exponentially decaying term present in its estimate updating equation, which leads to poor convergence in many problems [37]. A recently proposed variant of Adam is the AMSGrad algorithm, which proposes to fix Adam's convergence issue by incorporating "long-term memory" of the past gradients.

In this dissertation, we propose a *stochastic iterative pre-conditioning* technique for improving the rate of convergence of the distributed stochastic gradient descent method when solving the linear least-squares problem (2.6) in distributed networks. The idea of *iterative pre-conditioning* in the deterministic (batched data) case has been mentioned in Section 2.1.1 wherein the server updates the estimate using the sum of the agents' gradient multiplied with a suitable iterative pre-conditioning matrix. Updating the pre-conditioning matrix depends on the entire dataset at each iteration. We extend that idea to the *stochastic* settings, where the server updates both the estimate and the iterative pre-conditioning matrix based on a randomly chosen agents' *stochastic gradient* at every iteration. Each agent computes its stochastic gradient based on a single randomly chosen data point from its local set of data points. Using real-world datasets, we empirically show that the proposed algorithm converges in fewer iterations compared to the aforementioned state-of-

the-art distributed methods. Besides empirical results, we also present a formal analysis of the proposed algorithm's convergence in stochastic settings.

2.1.5 Summary of our contributions

In this chapter, we propose an *iterative pre-conditioning* technique for improving the rate of convergence of the traditional gradient-descent method when solving the linear least-squares problem in a server-agent network. We present below a summary of our key contributions and comparisons with other prominent algorithms applicable to the server-agent network architecture.

2.1.5.1 In deterministic settings

- We show that our algorithm, in general, converges *linearly*. For the special case when the solution of (2.1) is unique, the convergence of our algorithm is *superlinear*. Please refer Section 2.2.4 for details.

- We show that our algorithm has a favorable rate of convergence compared to the prominent distributed linear least-squares algorithms applicable to the server-agent network; the gradient-descent (GD) method, Nesterov's accelerated gradient-descent (NAG) method, the heavy-ball method (HBM), and the accelerated projection-consensus method (APC) [5]. These comparisons are summarized in Table 2.1. Please refer Section 2.3 for details.

  - In the particular case when the solution (2.1) is unique, our algorithm converges *superlinearly* which is only comparable to the BFGS method [22]. The convergence of the other aforementioned algorithms, on the other hand, is only *linear* [5], [25].

  - For the general case when the least-squares problem has multiple solutions (2.1),

both our algorithm and the GD method converge linearly. In this case, our algorithm provably improves upon the explicit rate of convergence of GD. Specifically, our algorithm converges *exponentially faster* compared to the GD method. The explicit *linear* rate of convergence of the NAG and HBM method relies on the uniqueness of the solution [25], [26]. For multiple solutions, these methods' convergence rate is available only in terms of the order of iteration numbers [26], [39]. However, not only the order of number of iterations but also the coefficient associated with the order contribute to characterizing the convergence rate of an algorithm. Hence, we provide the explicit convergence rate of our algorithm. Among the other aforementioned algorithms, convergence of the APC method and BFGS method requires uniqueness of the solution [22], [5].

- We validate our obtained theoretical results through numerical experiments, presented in Section 2.6, on benchmark real-world datasets.

Table 2.1: Comparisons between the theoretical convergence rate of different algorithms for distributed linear regression.

Algorithm		**Algorithm 1**	GD	NAG	HBM	APC	BFGS
Condition for convergence		any number of optimal solutions				unique optimal solution	
Explicit convergence rate	unique solution	superlinear	linear [5], [25]				superlinear [22]
	multiple solutions	linear, faster than GD	linear	not known [26], [39]		need not converge [22]	

2.1.5.2 In presence of system noise

Here, we make two key contributions, summarized as follows.

- In Section 2.4, we theoretically characterize the robustness guarantees of the IPG method against both the observation and the process noises.

- In Section 2.6, we empirically show the improved robustness of the IPG method, in comparison to the state-of-the-art algorithms.

2.1.5.3 In stochastic settings

- We present a formal convergence analysis of our proposed stochastic algorithm. Our convergence result can be informally summarized as follows. *Suppose the solution of problem* (2.6) *is unique, and the variances of the stochastic gradients computed by the agents are bounded. In that case, our proposed algorithm, i.e., Algorithm 2, converges linearly in expectation to a proximity of the solution of the problem* (2.6). The approximation error is proportional to the algorithm's stepsize and the variances of the stochastic gradients. Note that, as in the deterministic settings, our algorithm converges *superlinearly* to the exact solution when the gradient noise is zero. Formal details are presented in Theorem 2.5 in Section 2.5.3.

- Using real-world datasets, we empirically show that our proposed stochastic algorithm's convergence rate is superior to that of the state-of-the-art stochastic methods when distributively solving linear least-squares problems. These datasets comprise

  - four benchmark datasets from the SuiteSparse Matrix Collection;

- a subset of the *"cleveland"* dataset from the UCI Machine Learning Repository, which contains binary classification data of whether the patient has heart failure or not based on 13 features;
- a subset of the *"MNIST"* dataset for classification of handwritten digits one and five.

Please refer to Section 2.6 for further details.

2.2 Proposed algorithm: Iteratively Pre-Conditioned Gradient-Descent (IPG)

In this section, we present our algorithm, its computational complexity, and its convergence properties in the deterministic settings, i.e., with full-batch data and when there is no noise in the system.

2.2.1 Motivation for IPG

The proposed algorithm is similar to the gradient-descent method described in Section 2.1.1. However, a notable difference is that in our algorithm, the server multiplies the aggregate of the gradients received from the agents by a *pre-conditioner* matrix. The server uses the *pre-conditioned* aggregates of the agents' gradients to update its current estimates. In literature, this technique is commonly referred as *pre-conditioning* [40]. When the matrix A^TA is non-singular, the best pre-conditioner matrix for the gradient-descent method is the inverse Hessian matrix $\left(A^TA\right)^{-1}$, resulting in Newton's method which converges *superlinearly*. However, $\left(A^TA\right)^{-1}$ cannot be computed directly in a distributed setting as it requires the agents to send their local data points to the server. Thus, instead of a constant pre-conditioner matrix $\left(A^TA\right)^{-1}$, we propose a distributed scheme where the server iteratively updates the pre-conditioning matrix in such a

way that it converges to $\left(A^T A\right)^{-1}$ (ref. Lemma A.1). This specific iterative update rule of the pre-conditioning matrix is described later in Step 4 of the next subsection. Thus, our algorithm eventually converges to Newton's method and has *superlinear* convergence rate, as shown later in Section 2.2.4. Herein lies the motivation of our proposed algorithm.

The idea of iterative pre-conditioning is inspired by simple adaptive control techniques [41]. In adaptive control theory, the goal is to design a feedback control law, when the system parameters are unknown, so that the system state is driven to an equilibrium point. Analogous to such techniques, the proposed algorithm "adapts" to the unknown inverse Hessian matrix $\left(A^T A\right)^{-1}$ by tuning the iterative pre-conditioner matrix (the "feedback gain") which drives the estimate (system state) to a solution of (2.1). At the same time, the pre-conditioner matrix, which is an estimate of the unknown parameter $\left(A^T A\right)^{-1}$, also converges to the true parameter $\left(A^T A\right)^{-1}$. Thus, in the language of adaptive control theory, we have both the parameter convergence and the state convergence.

In each iteration $t \in \{0, 1, \ldots\}$, the server maintains an estimate $x(t)$ of a minimum point (2.1), and a pre-conditioner matrix $K(t) \in \mathbb{R}^{d\times d}$. The initial estimate $x(0)$ and the pre-conditioner matrix $K(0)$ are chosen arbitrarily from $\mathbb{R}^d$ and $\mathbb{R}^{d\times d}$, respectively. For each iteration $t \geq 0$, the algorithm steps are presented below.

2.2.2 Steps in each iteration t

In each iteration t, the algorithm comprises four steps, executed collaboratively by the server and the agents. Before initiating the iterative process, the server chooses three non-negative scalar real-valued parameters α, δ, and β whose specific values are presented later in

Section 2.2.4. The parameter β is broadcast to all the agents.

- *Step 1:* The server sends the estimate $x(t)$ and the matrix $K(t)$ to each agent i.

- *Step 2.1:* Each agent i computes the gradient $g^i(t)$, as defined in (2.3).

- *Step 2.2:* Each agent i computes a set of vectors $\left\{R_j^i(t) : \ j = 1, \ldots, \ d\right\}$ such that for each $j = 1, \ldots, d$,

$$R_j^i(t) \ = \left(\left(A^i\right)^T A^i + \left(\frac{\beta}{m}\right) \ I \right) k_j(t) - \left(\frac{1}{m}\right) e_j, \tag{2.7}$$

  where I denote the $(d \times d)$-dimensional identity matrix, e_j and $k_j(t)$ denote the j-th columns of matrices I and $K(t)$, respectively.

- *Step 3:* Each agent i sends gradient $g^i(t)$ and the set of vectors $\left\{R_j^i(t), \ j = 1, \ldots, \ d\right\}$ to the server.

- *Step 4:* The server updates the matrix $K(t)$ as

$$k_j(t+1) = k_j(t) - \alpha \sum_{i=1}^{m} R_j^i(t), \quad j = 1, ..., d. \tag{2.8}$$

  Finally, the server updates the estimate $x(t)$ to $x(t+1)$ such that

$$x(t+1) = x(t) - \delta K(t+1) \sum_{i=1}^{m} g^i(t). \tag{2.9}$$

  Parameter δ is a non-negative real value, commonly referred as the *step-size*.

  Our algorithm is summarized below in Algorithm 1.

Algorithm 1 *Iterative pre-conditioning* for the gradient-descent method in a server-agent network.

1: The server initializes $x(0) \in \mathbb{R}^d$, $K(0) \in \mathbb{R}^{d \times d}$ and selects the parameters $\alpha > 0$, $\delta > 0$ and $\beta \geq 0$.
2: **for** $t = 0,\ 1,\ 2, \ldots$ **do**
3: The server sends $x(t)$ and $K(t)$ to each agent $i \in \{1, \ldots,\ m\}$.
4: Each agent i computes the gradient $g^i(t)$ as defined by (2.3), and a set of vectors $\left\{R_j^i(t),\ j = 1, \ldots,\ d\right\}$ as defined by (2.7).
5: Each agent i sends $g^i(t)$ and the set $\left\{R_j^i(t),\ j = 1, \ldots,\ d\right\}$ to the server.
6: The server updates $K(t)$ to $K(t+1)$ as defined by (2.8).
7: The server updates the estimate $x(t)$ to $x(t+1)$ as defined by (2.9).
8: **end for**

2.2.3 Algorithm complexity

We now present the computational complexity of Algorithm 1, in terms of the total number of floating-point operations (*flops*) required per iteration.

For each iteration t, each agent i computes the gradient $g^i(t)$, defined in (2.3), and d vectors $\{\mathbb{R}_j^i(t) : \ j = 1, ..., d\}$, defined in (2.7). Computation of $g^i(t)$ requires two matrix-vector multiplications, namely $A^i\, x(t)$ and $(A^i)^T \left(A^i\, x(t) - b^i\right)$, in that order. As A^i is an $(n_i \times d)$-dimensional matrix and $x(t)$ is a d-dimensional vector, computation of gradient $g^i(t)$ requires $\mathcal{O}(n_i d)$ flops. From (2.7), computation of each vector $R_j^i(t)$ requires two matrix-vector multiplications, namely $A^i\, k_j(t)$ and $(A^i)^T(A^i k_j(t))$, in that order. As A^i is an $(n_i \times d)$-dimensional matrix, and both vectors $k_j(t)$ and $A^i\, k_j(t)$ are of dimensions d, computation of each $R_j^i(t)$ requires $\mathcal{O}(n_i d)$ flops. Thus, net computation of d vectors $\{\mathbb{R}_j^i(t) : \ j = 1, ..., d\}$ requires $\mathcal{O}(n_i d^2)$ flops. Therefore, the computational complexity of Algorithm 1 for each agent i is d numbers of $\mathcal{O}(n_i d)$ parallel flops, for each iteration. Note that, the computation of each member in the set $\{\mathbb{R}_j^i(t) : \ j = 1, ..., d\}$ is independent of each other. Hence, agent i can compute the d vectors $\{\mathbb{R}_j^i(t) : \ j = 1, ..., d\}$ in parallel.

For each iteration t, the server computes the matrix $K(t+1)$, defined in (2.8), and vector $x(t+1)$, defined in (2.9). Computing $K(t+1)$ only requires $\mathcal{O}(d)$ floating-point additions, and can be ignored. In (2.9), the computation of $K(t+1)\sum_{i=1}^{m} g^i(t)$ requires only one matrix-vector multiplication between the $d \times d$ dimensional matrix $K(t+1)$ and the d-dimensional vector $\sum_{i=1}^{m} g^i(t)$. Therefore, the computational complexity for the server is $\mathcal{O}(d^2)$ flops, for each iteration.

Communication complexity: For each iteration t, each agent sends a vector $g^i(t) \in \mathbb{R}^d$ and d vectors $R^i_j(t) \in \mathbb{R}^d$, which means (d^2+d) real scalar values are send to the server by each agent. Thus, the communication complexity of Algorithm 1 for each agent is $\mathcal{O}(d^2)$, compared to $\mathcal{O}(d)$ of the GD method, which is a drawback of Algorithm 1.

Next, we present convergence results for Algorithm 1.

2.2.4 Convergence guarantees

To be able to present the formal convergence guarantees of Algorithm 1, we note below some elementary facts and introduce some notation.

- We define the *collective data matrix* and the *collective observation vector* respectively to be $A = \left[(A^1)^T, \ldots, (A^m)^T\right]^T, b = \left[(b^1)^T, \ldots, (b^m)^T\right]^T$.

- Note that matrix product A^TA is positive semi-definite. Thus, if $\beta > 0$ then $\left(A^TA + \beta I\right)$ is positive definite, and therefore, invertible. Let $K_\beta = \left(A^TA + \beta I\right)^{-1}$, and let $\lambda_1, \ldots, \lambda_d$ denote the eigenvalues of matrix A^TA such that $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$.

- The rank of matrix A^TA is denoted by r. Note that $r = d$ if and only if the matrix A^TA is full rank. In general, when A^TA is not the trivial zero matrix, $1 \leq r \leq d$. Note that if

$r < d$ then $\lambda_1 \geq \ldots \geq \lambda_r > \lambda_{r+1} = \ldots = \lambda_d = 0$.

- For a matrix $M \in \mathbb{R}^{d \times d}$, $\|M\|_F$ denotes its Frobenius norm [42]. Specifically, if m_{ij} denote the $(i,\, j)$-th element of matrix M, then $\|M\|_F = \sqrt{\sum_{i=1}^{d} \sum_{j=1}^{d} m_{ij}^2}$.

For each agent i, recall from (2.2), the cost function $F^i(x)$ is convex and differentiable. Thus, the aggregate cost function $\sum_{i=1}^{m} F^i(x)$ is also convex. Therefore, $x^* \in X^*$ if and only if [43] $\nabla \sum_{i=1}^{m} F^i(x^*) = 0_d$, where 0_d denotes the d-dimensional zero vector. Recall, from Section 2.1.1,

$$g(t) = \nabla \sum_{i=1}^{m} F^i(x(t)) = \sum_{i=1}^{m} \nabla F^i(x(t)) = \sum_{i=1}^{m} g^i(t). \tag{2.10}$$

We now define below parameters that determine the *convergence rate* of Algorithm 1. Let,

$$\mu^* = \frac{\lambda_1 - \lambda_r}{\lambda_1 + \lambda_r + 2(\lambda_1 \lambda_r / \beta)}, \tag{2.11}$$

$$\varrho = \frac{\lambda_1 - \lambda_d}{\lambda_1 + \lambda_d + 2\beta}. \tag{2.12}$$

Since $\lambda_1, \lambda_r, \beta > 0$, (2.11)-(2.12) implies that $\mu^*, \varrho < 1$. We define the *optimal step-size* parameter

$$\delta^* = \frac{2}{\frac{\lambda_1}{\lambda_1+\beta} + \frac{\lambda_r}{\lambda_r+\beta}}. \tag{2.13}$$

The key result on the convergence of Algorithm 1 is presented next.

Theorem 2.1. *Consider Algorithm 1. Suppose,* $0 < \alpha < \frac{2}{\lambda_1+\beta}$ *and* $0 < \delta < 2\left(\frac{\lambda_1+\beta}{\lambda_1}\right)$. *Then,*

1. *there exists non-negative real values* μ *and* ρ *with* $\mu^* \leq \mu < 1$ *and* $\varrho \leq \rho < 1$ *such that*

for each iteration $t \geq 0$,

$$\left\|g(t+1)\right\| \leq \left(\mu + \delta\lambda_1 \left\|K(0) - K_\beta\right\|_F \rho^{t+1}\right) \left\|g(t)\right\|; \tag{2.14}$$

2. $\lim_{t\to\infty} \left\|g(t)\right\| = 0$;

3. *If* $\delta = \delta^*$ *then* (2.14) *holds with* $\mu = \mu^*$.

As $\rho < 1$, (2.14) of Theorem 2.1 implies that $\lim\limits_{t\to\infty} \frac{\left\|g(t+1)\right\|}{\left\|g(t)\right\|} \leq \mu < 1$, since $\rho^{t+1} \to 0$ as $t \to \infty$. Thus, Theorem 2.1 implies that the sequence of gradients $\{g(t)\}_{t\geq 0}$ converges linearly to 0_d with *convergence rate* no worse than μ. Since $g(t)$ is linearly related to $x(t)$ as presented in (2.10), linear convergence of $\{g(t)\}_{t\geq 0}$ to 0_d implies linear convergence of the sequence of estimators $\{x(t)\}_{t\geq 0}$ to a point in X^*.

Superlinear convergence: Consider the special case when x^* is the unique solution for the least-squares problem (2.1), i.e., the unique minimum point of the aggregate cost function $\sum_{i=1}^m F^i(x)$. In this particular case, the matrix $A^T A$ is full-rank, and therefore, $r = d$. Here, we show below that Algorithm 1 with parameter $\beta = 0$ converges *superlinearly* to x^*. Specifically, we obtain the following corollary of Theorem 2.1. Recall, from (2.12), that when $\beta = 0$ then $\varrho = \frac{\lambda_1 - \lambda_d}{\lambda_1 + \lambda_d} < 1$.

Corollary 2.1. *Consider Algorithm 1 with* $\beta = 0$. *If* x^* *defined by (2.1) is unique, and the parameter* α *satisfies the condition stated in Theorem 2.1, then for* $\delta = 1$ *there exists a non-negative real value* $\rho \in [\varrho, 1)$ *such that,*

1. *for each iteration* $t \geq 0$, $\left\|g(t+1)\right\| \leq \lambda_1 \left\|K(0) - K_\beta\right\|_F \rho^{t+1} \left\|g(t)\right\|$;

2. $\lim_{t\to\infty} \frac{\left\|x(t+1) - x^*\right\|}{\left\|x(t) - x^*\right\|} = 0$.

Remarks: Since the number of data points n_i held by each agent i is not necessarily the same across the agents, it may be of interest to minimize the weighted aggregate cost function $\sum_{i=1}^{m} \frac{n_i}{2\sum_{i=1}^{m} n_i} \left\| A^i x - b^i \right\|^2$, where each local cost function F^i is weighted by the fraction of the number of data points of the corresponding agent i relative to the total number of data points of all the agents combined. We note that the results in this section are still valid for this weighted cost function.

2.3 Comparisons with the existing methods

In this section, we present comparisons between the rates of convergence of Algorithm 1 and other prominent server-agent algorithms reviewed in Section 2.1. The algorithms are elaborated in [44].

Unique optimal solution: For the special case when the solution of the distributed least-squares problem (2.1) is unique, as shown in Section 2.2.4, Algorithm 1 converges *superlinearly*, which is comparable only to the BFGS method. The rest of the algorithms, namely GD, NAG, HBM, and APC, only converge *linearly* [5].

Multiple optimal solutions: In general when the solution for the distributed least-squares problem (2.1) is not unique, we show below that Algorithm 1 converges *exponentially faster* than the GD method. It should be noted that the convergence of BFGS and APC, and the explicit convergence rate of NAG and HBM can only be guaranteed for the special when the solution of (2.1) is unique; see [5], [25], [26] and references therein. Otherwise, the convergence rate of NAG and HBM is known only in terms of the order of iteration numbers [26], [39]. However, in practice the constant factor that is often ignored in order convergence analysis can be quite

critical. Thus, we provided an explicit convergence rate of Algorithm 1 in Theorem 2.1.

The above comparisons are succinctly summarized in Table 2.1 (ref. Page 16). Detailed comparison with the classical GD method is presented below.

2.3.1 The gradient-descent method

Consider the GD algorithm in server-agent networks, described in Section 2.1.1. To the best of our knowledge, in the open literature, the explicit rate of convergence for GD is mentioned only when the solution for (2.1) is unique [2], [25], [26]. We present below, formally in Lemma 2.1, the explicit convergence rate of GD for the general case. We define a parameter $\mu_{GD} = \dfrac{\lambda_1 - \lambda_r}{\lambda_1 + \lambda_r}$, where λ_1 and λ_r are respectively the largest and the smallest non-zero eigenvalue of $A^T A$.

Lemma 2.1. *Consider the gradient-descent algorithm in a server-agent network as presented in Section 2.1.1. In (2.4), if* $\delta \in \left(0, \frac{2}{\lambda_1}\right)$ *then there exists* μ *with* $\mu_{GD} \leq \mu < 1$ *such that, for each iteration* $t \geq 0$, $\left\|g(t+1)\right\| \leq \mu \left\|g(t)\right\|$.

We show formally below, in Theorem 2.2, that Algorithm 1 converges faster than GD method. Note that, in the special case when all the non-zero eigenvalues of the matrix $A^T A$ are equal, i.e. $A^T A = kI$ for some $k \geq 0$, both the GD algorithm and Algorithm 1 solve the optimization problem (2.1) in just one iteration. Now, Theorem 2.2 below presents the case when $\lambda_1 > \lambda_r$.

Theorem 2.2. *Consider Algorithm 1. Suppose that* $\lambda_1 > \lambda_r$. *If* $\beta > 0$ *then there exists a positive finite integer* τ, *and two positive finite real values* c *and* r *with* $r < 1$, *such that* $\left\|g(t+1)\right\| \leq c\,(r\,\mu_{GD})^{t+1} \left\|g(0)\right\|, \forall t > \tau$.

Consider the best possible rate of convergence for the GD algorithm. That is, substitute

$\mu = \mu_{GD}$ in Lemma 2.1. In that case, the gradients for the GD algorithm in a server-agent network satisfy $\|g(t+1)\| \leq (\mu_{GD})^{t+1} \|g(0)\|, \forall t \geq 0$. From Theorem 2.2, the gradients' norm for Algorithm 1 are given by $\|g(t+1)\| \leq c\,(r\,\mu_{GD})^{t+1} \|g(0)\|, \forall t > \tau$. Thus, assuming that both the algorithms are identically initialized with some $x(0)$, there exists a finite integer T such that the ratio between the upper bounds on the gradients of Algorithm 1 and GD in server-agent network is given by $c\ r^{t+1}$ for iteration $t > T$, where $r < 1$. This statement implies that, after a finite number of iterations, Algorithm 1 is guaranteed to have a smaller error bound compared to GD with identical initialization of $x(0)$ and arbitrary initialization of the iterative pre-conditioning matrix $K(0)$. More importantly, this error bound of Algorithm 1 decreases to zero at an *exponentially faster* rate compared to the latter one.

2.4 Robustness of the IPG method

In this section, we present our key results on the robustness of the IPG method [6], described in Algorithm 1 for the noise-free case, in the presence of additive system noises; the *observation noise* and the *process noise*.

We introduce below some notation, our main assumption, and review a pivotal prior result.

2.4.1 Notation, assumption, and prior results

Assumption 2.1. *Assume that the matrix $A^T A$ is full rank.*

Note that Assumption 2.1 holds true if and only if the matrix $A^T A$ is positive definite with $\lambda_d > 0$. Under Assumption 1, $A^T A$ is invertible, and we let $K^* = \left(A^T A\right)^{-1}$. Note, from (2.2), that the Hessian of the aggregate cost function $\sum_{i=1}^m F^i(x)$ is equal to $A^T A$ for all $x \in \mathbb{R}^d$.

Thus, under Assumption 1 when $A^T A$ is positive definite, the aggregate cost function has a unique minimum point. Equivalently, the solution of the least squares problem defined by (2.1) is unique.

We review below in Lemma 2.2 a prior result that is pivotal for our key results presented later in this section. We let

$$\varrho = (\lambda_1 - \lambda_d)/(\lambda_1 + \lambda_d). \tag{2.15}$$

For each iteration t, recall the *pre-conditioner* matrix $K(t)$ in Algorithm 1, and define

$$\widetilde{K}(t) = K(t) - K^*, \quad \forall t \geq 0. \tag{2.16}$$

Let $\widetilde{k}_j(t)$ denote the j-th column of $\widetilde{K}(t)$ and let k_j^* denote the j-th column of the matrix K^*. Lemma 2.2 below states sufficient conditions under which the sequence of the pre-conditioner matrices $\{K(t),\ t = 0,\ 1, \ldots\}$, in Algorithm 1, converges linearly to K^*.

Lemma 2.2. *Consider Algorithm 1 with $\alpha \in (0, \frac{2}{\lambda_1})$. If Assumption 1 holds true then there exists a real value $\rho \in [\varrho,\ 1)$ such that, for all $j \in \{1, \ldots,\ d\}$,*

$$\left\|\widetilde{k}_j(t+1)\right\| \leq \rho \left\|\widetilde{k}_j(t)\right\|, \quad \forall t \geq 0. \tag{2.17}$$

We first present in Section 2.4.2 below the robustness of the IPG method against observation noise, followed by the robustness against process noise in Section 2.4.3.

2.4.2 Robustness against observation noise

Based upon the literature [29], we model observation noise as follows. Each agent i observes a corrupted observation vector b^{io}, instead of the true observation vector b^i. Specifically,

$$b^{io} = b^i + w_b^i,\ i \in \{1, \ldots, m\}, \tag{2.18}$$

where $w_b^i \in \mathbb{R}^{n^i}$ is a random vector. Let $\mathbb{E}[\cdot]$ denote the expectation of a function of the random vectors $\{w_b^i,\ i = 1, \ldots, m\}$. Let $\|\cdot\|_1$ denote the l_1-norm [45].

Assumption 2.2. *Assume that there exists $\eta < \infty$ such that*

$$\mathbb{E}\left[\left\|w_b^i\right\|_1\right] \leq \eta,\ i \in \{1, \ldots, m\},. \tag{2.19}$$

In the presence of the above observation noise, in each iteration t of Algorithm 1, each agent i sends to the server a corrupted gradient $g^i(t)$ defined by (2.20) below, instead of (2.3). Specifically, for all i and t,

$$g^i(t) = \left(A^i\right)^T \left(A^i\, x(t) - b^{io}\right). \tag{2.20}$$

Due to the above corruption in gradient computation, Algorithm 1 no longer converges to an exact solution, defined by (2.1), but rather to an approximation.

Theorem 2.3 below presents a key result on the robustness of Algorithm 1 against the above additive observation noise. Recall from Section 2.4.1 that under Assumption 1 the solution, denoted by x^*, of the regression problem (2.1) is unique. For each iteration t, we define the

estimation error

$$z(t) = x(t) - x^*. \tag{2.21}$$

For a matrix M, with columns $M_1, \ldots, M_d$, its Frobenius norm [45] is defined to be $\|M\|_F = \sqrt{\sum_{j=1}^{d} \|M_j\|^2}$. Recall the definition of ϱ from (2.15).

Theorem 2.3. *Let $\{z(t)\}_{t=0}^{t=\infty}$ be constructed by Algorithm 1 with parameters $\alpha < \frac{2}{\lambda_1}$ and $\delta \leq 1$, in the presence of additive observation noise defined in (2.18) and the gradients for each iteration t, $\{g^i(t), i = 1, \ldots, m\}$, defined by (2.20). If Assumptions 2.1-2.2 hold then there exists $\rho \in [\varrho, 1)$ such that, for all $t \geq 0$,*

$$\mathbb{E}\left[\|z(t+1)\|\right] \leq \left(1 - \delta + \delta\lambda_1 \left\|\widetilde{K}(0)\right\|_F \rho^{t+1}\right) \|z(t)\| + \delta\eta m\sqrt{\lambda_1} \left\|\widetilde{K}(0)\right\|_F \rho^{t+1} + \delta\eta m\sqrt{1/\lambda_d}. \tag{2.22}$$

Additionally, $\lim_{t\to\infty} \mathbb{E}\left[\|z(t)\|\right] \leq \delta\eta m \sqrt{1/\lambda_d}$.

Since $\rho \in [0, 1)$ and $\delta \in (0, 1]$, Theorem 2.3 implies that the IPG method under the influence of additive observation noise (2.18) converges in expectation within a distance of $\delta\eta m\sqrt{1/\lambda_d}$ from the true solution x^* of the regression problem defined in (2.1). Since the gradient-descent (GD) method (with constant step-sizes) is a special case of the IPG method with $K(t) = I$ for all t, from the proof of Theorem 2.3 we obtain that the final error of GD is bounded by $\delta\eta m\sqrt{\lambda_1}$.

2.4.3 Robustness against process noise

In the presence of process noise [27,28], in each iteration t of Algorithm 1 (the IPG method in the noise-free case), the server computes corrupted values for both the pre-conditioner matrix $K(t)$ and the current estimate $x(t)$, described formally as follows. Specifically, in each iteration t, and for each $j \in \{1, \ldots, d\}$, instead of computing $k_j(t)$ (the j-th column of $K(t)$) accurately, the server computes

$$k_j^o(t) = k_j(t) + w_j^k(t), \tag{2.23}$$

where $w_j^k(t) \in \mathbb{R}^d$. Similarly, instead of computing $x(t)$ accurately, the server computes a corrupted estimate

$$x^o(t) = x(t) + w^x(t), \tag{2.24}$$

where $w^x(t) \in \mathbb{R}^d$. Together, the vectors $\zeta(t) = \{w^x(t), w_j^k(t), j = 1, \ldots, d\}$ are referred as additive process noise. For each iteration t, let $\mathbb{E}_{\zeta_t}[\cdot]$ denote the expectation of a function of the random vectors $\zeta(t)$. Let $\mathbb{E}_t[\cdot]$ denote the total expectation of a function of the random vectors $\{\zeta(0), \ldots, \zeta(t)\}$.

Assumption 2.3. *Assume that the random vectors* $\{w^x(t), w_j^k(t), j = 1, \ldots, d\}$ *are mutually independent for all* t*, and there exists* $\omega < \infty$ *such that for all* t, j*,*

$$\mathbb{E}_{\zeta_t}\left[\left\|w_j^k(t)\right\|_1\right] \leq \omega, \text{ and } \mathbb{E}_{\zeta_t}\left[\left\|w^x(t)\right\|_1\right] \leq \omega. \tag{2.25}$$

In the presence of above process noise, Algorithm 1 is modified as follows. In each iteration t, each agent i sends to the server a gradient $g^i(t)$ and d vectors $R_1^i(t), \ldots, R_d^i(t)$ defined by (2.26) and (2.27) below, respectively, instead of (2.3) and (2.7). Specifically, for all i and t,

$$g^i(t) = \left(A^i\right)^T \left(A^i\, x^o(t) - b^i\right), \tag{2.26}$$

and, for $j = 1, \ldots, d$,

$$R_j^i(t) = \left(A^i\right)^T A^i k_j^o(t) - \left(\frac{1}{m}\right) e_j. \tag{2.27}$$

Similarly, in each iteration t, the server now computes $K(t+1)$ whose j-th column is defined as follows, instead of (2.8), for all $j \in \{1, \ldots, d\}$,

$$k_j(t+1) = k_j^o(t) - \alpha \textstyle\sum_{i=1}^m R_j^i(t). \tag{2.28}$$

Recall that $k_j^o(t+1)$, defined by (2.23), is the corrupted value of $k_j(t+1)$. Instead of (2.9), the updated estimate $x(t+1)$ is defined by

$$x(t+1) = x^o(t) - \delta K^o(t+1) \textstyle\sum_{i=1}^m g^i(t), \ \forall t. \tag{2.29}$$

Recall that $x^o(t+1)$, defined by (2.23), is the corrupted value of $x(t+1)$.

To present our key result on the robustness of the IPG method, in Theorem 2.4 below, against the above process noise, we introduce some notation. From Lemma 2.2, recall that $\rho \in$

$[\varrho, 1)$. For each iteration t, we define

$$\widetilde{K}^o(t) = \left[\widetilde{k}_1^o(t), \ldots, \widetilde{k}_d^o(t)\right] = K^o(t) - K^*, \text{ and} \tag{2.30}$$

$$u(t) = 1 - \delta + \delta\lambda_1 \left(\rho^t \left\|\widetilde{K}(0)\right\| + \omega\sqrt{d}\, \textstyle\sum_{i=0}^{t} \rho^i\right). \tag{2.31}$$

Additionally, we let

$$\rho_{bd} = \frac{\left\|\widetilde{K}(0)\right\|}{\left\|\widetilde{K}(0)\right\| + \omega\sqrt{d}}, \text{ and } \omega_{bd} = \frac{(1-\rho)}{\lambda_1\sqrt{d}}. \tag{2.32}$$

Theorem 2.4 below characterizes the convergence of Algorithm 1 with modifications (2.26)-(2.29) in presence of process noise. Recall from definition (2.21) that $z(t) = x(t) - x^*$ for all t. Upon substituting $x(t)$ from (2.24), we obtain that $z(t) = x^o(t) - w^x(t) - x^*$. We let $z^0(t) = x^0(t) - x^*$. Thus, for each iteration t,

$$z^o(t) = z(t) + w^x(t). \tag{2.33}$$

Theorem 2.4. *Let $\left\{z^o(t)\right\}_{t=0}^{t=\infty}$ be constructed by Algorithm 1 with parameter $\alpha < \frac{2}{\lambda_1}$ and $\delta \leq 1$, in the presence of additive process noise defined in (2.23)-(2.24), and modifications defined in (2.26)-(2.29). If Assumptions 2.1 and 2.3 hold then there exists $\rho \in [\varrho, 1)$ such that*

$$\mathbb{E}_t\left[\left\|z^o(t)\right\|\right] \leq \Pi_{k=1}^{t} u(k) \left\|z(0)\right\| + \left(1 + u(t) + \ldots + \Pi_{k=1}^{t} u(k)\right)\omega, \ \forall t. \tag{2.34}$$

Moreover, if

$$\rho < \rho_{bd}, \text{ and } \omega < \omega_{bd}, \tag{2.35}$$

then $\lim_{t\to\infty} \mathbb{E}_t \left[\left\|z^o(t)\right\|\right] < \frac{\omega}{\delta\left(1-\left(\omega/\omega_{bd}\right)\right)}$.

Note that, from the proof of Theorem 2.4 when $K(t) = I$, $\forall t$, the asymptotic estimation error of the traditional GD method under the above process noise is bounded by $\frac{\omega}{1-\left\|I-\delta A^T A\right\|}$.

2.5 SGD with iterative pre-conditioning

In this section, we present our algorithm for solving (2.6) in stochastic settings. Our algorithm follows the basic prototype of the stochastic gradient descent method in distributed settings. However, unlike the traditional distributed stochastic gradient descent, the server in our algorithm multiplies the stochastic gradients received from the agents by a *stochastic pre-conditioner* matrix. These pre-conditioned stochastic gradients are then used by the server to update the current estimate.

In order to present the algorithm, we introduce some notation. The *individual data points* of the agents are represented by an *data row vector* a of dimensions $1 \times d$ and a *scalar observation* b. Thus, $a \in \mathbb{R}^{1\times d}$ and $b \in \mathbb{R}$. For each data point (a, b), we define *individual cost function* $f : \mathbb{R}^d \to \mathbb{R}$ such that for a given $x \in \mathbb{R}^d$,

$$f(x; a, b) = \frac{1}{2}\left(a\, x - b\right)^2, \tag{2.36}$$

and the gradient of the individual cost function f as

$$g(x; a, b) = \nabla_x f(x; a, b) = a^T \, (a\, x - b)\,. \tag{2.37}$$

In each iteration $t \in \{0, 1, \ldots\}$, the server maintains an estimate $x(t)$ of a minimum point (2.6), and a stochastic pre-conditioner matrix $K(t) \in \mathbb{R}^{d \times d}$. The initial estimate $x(0)$ and the pre-conditioner matrix $K(0)$ are chosen arbitrarily from $\mathbb{R}^d$ and $\mathbb{R}^{d \times d}$, respectively. For each iteration $t = 0, 1, \ldots$, the algorithm steps are presented below.

2.5.1 Steps in each iteration t

Before initiating the iterations, the server chooses a positive scalar real-valued parameter β and broadcast it to all the agents. We number the agents in order from 1 to m. In each iteration t, the proposed algorithm comprises of four steps described below. These steps are executed collaboratively by the server and the agents, without requiring any agent to share its local data points. For each iteration t, the server also chooses two positive scalar real-valued parameters α and δ.

- *Step 1:* The server sends the estimate $x(t)$ and the pre-conditioner matrix $K(t)$ to each agent $i \in \{1, \ldots, m\}$.

- *Step 2:* Each agent $i \in \{1, \ldots, m\}$ chooses a data point (a^{i_t}, b^{i_t}) *uniformly* at random from its local data points (A^i, b^i). Note that, a^{i_t} and b^{i_t} are respectively a row in the input matrix A^i and the output vector b^i of agent i. Each data point is independently and identically distributed (i.i.d.). Based on the selected data point (a^{i_t}, b^{i_t}), each agent i then computes a

stochastic gradient, denoted by $g^{i_t}(t)$, which is defined as

$$g^{i_t}(t) = g(x(t); a^{i_t}, b^{i_t}). \tag{2.38}$$

In the same step, each agent $i \in \{1, \ldots, m\}$ computes a set of vectors $\left\{h_j^{i_t}(t) : \ j = 1, \ldots, \ d\right\}$:

$$h_j^{i_t}(t) = h_j(k_j(t); a^{i_t}, b^{i_t}), \tag{2.39}$$

where the function $h_j : \mathbb{R}^d \to \mathbb{R}^d$ is defined below. Let I denote the $(d \times d)$-dimensional identity matrix. Let e_j and $k_j(t)$ denote the j-th columns of matrices I and $K(t)$, respectively. For each column $j \in \{1, \ldots, d\}$ of $K(t)$ and each individual data point (a, b), we define

$$h_j(k_j; a, b) = \left(a^T a + \beta I\right) k_j - e_j. \tag{2.40}$$

- *Step 3:* Each agent $i \in \{1, \ldots, m\}$ sends the stochastic gradient $g^{i_t}(t)$ and the set of stochastic vectors $\left\{h_j^{i_t}(t), \ j = 1, \ldots, \ d\right\}$ to the server.

- *Step 4:* The server draws an i.i.d. sample ζ_t *uniformly* at random from the set of agents $\{1, \ldots, m\}$ and updates the matrix $K(t)$ to $K(t+1)$ such that, for each $j \in \{1, \ldots, d\}$,

$$k_j(t+1) = k_j(t) - \alpha h_j^{\zeta_t}(t). \tag{2.41}$$

Finally, the server updates the estimate $x(t)$ to $x(t+1)$ such that

$$x(t+1) = x(t) - \delta K(t+1) g^{\zeta_t}(t). \tag{2.42}$$

Parameter δ is a non-negative real value, commonly referred as the *stepsize*.

These steps of our algorithm are summarized in Algorithm 2.

Algorithm 2 Iteratively Pre-Conditioned Stochastic Gradient-descent (IPSG).

1: The server initializes $x(0) \in \mathbb{R}^d$, $K(0) \in \mathbb{R}^{d\times d}$, $\beta > 0$ and chooses $\{\alpha > 0, \delta > 0 : \ t = 0, 1, \ldots\}$.
2: **Steps in each iteration** $t \in \{0,\ 1,\ 2, \ldots\}$:
3: The server sends $x(t)$ and $K(t)$ to all the agents.
4: Each agent $i \in \{1, \ldots,\ m\}$ *uniformly* selects an i.i.d. data point (a^{i_t}, b^{i_t}) from its local data points (A^i, b^i).
5: Each agent $i \in \{1, \ldots,\ m\}$ sends to the server a stochastic gradient $g^{i_t}(t)$, defined in (2.38), and d stochastic vectors $h_1^{i_t}(t), \ldots,\ \ h_d^{i_t}(t)$, defined in (2.39).
6: The server *uniformly* draws an i.i.d. sample ζ_t from the set of agents $\{1, \ldots, m\}$.
7: The server updates $K(t)$ to $K(t+1)$ as defined by (2.41).
8: The server updates the estimate $x(t)$ to $x(t+1)$ as defined by (2.42).

Next, we present the formal convergence guarantees of Algorithm 2. We begin by introducing some notation and our main assumptions.

2.5.2 Notation and assumptions

For each iteration $t \geq 0$ we define the following.

- Let $\mathbb{E}_{\zeta_t}[\cdot]$ and for each agent $i \in \{1, \ldots, m\}$ $\mathbb{E}_{i_t}[\cdot]$ denote the conditional expectation of a function the random variables ζ_t and i_t, respectively, given the current estimate $x(t)$ and the current pre-conditioner $K(t)$.

- Let $I_t = \{i_t,\ i = 1, \ldots, m\} \cup \{\zeta_t\}$ and $\mathbb{E}_{I_t}[\cdot] = \mathbb{E}_{1_t, \ldots m_t, \zeta_t}(\cdot)$.

Table 2.2: Additional notation for analysis of IPSG.

$A = \left[(a^1)^T, \ldots, (a^N)^T\right]^T$	$K_\beta = \left(\frac{1}{N}A^TA + \beta I\right)^{-1}$
$C_1 = \max_i \left\| \left(a^i\right)^T a^i - \frac{1}{N}A^TA \right\|$	$\rho = \frac{1}{N}\sum_{i=1}^N \left\| I - \alpha\left(\left(a^i\right)^T a^i + \beta I\right)\right\|$
$s_1 \geq \ldots \geq s_d \geq 0$: eigenvalues of the positive semi-definite matrix A^TA	
Λ_i and λ_i: the largest and the smallest eigenvalue of $\left(a^i\right)^T a^i$	$L = \beta + \max_{i=1,\ldots,N}\Lambda_i$
$\mu = \left(1 - \frac{2\alpha s_d}{N}(1-\alpha L)\right)$	$\sigma^2 = \max\limits_{j=1,\ldots,d}\frac{1}{N}\sum\limits_{i=1}^N \left\| \left(\left(a^i\right)^T a^i + \beta I\right) K_\beta\, e_j - e_j \right\|^2$
$C_3 = \frac{\alpha N\sigma^2}{s_d(1-\alpha L)}$	$C_2 = \frac{\alpha}{N}\sum_{i=1}^N \left\| \left(a^i\right)^T a^i - \frac{1}{N}A^TA\right\| \left\|K_\beta\right\|$
$\varrho = \left\| I - \alpha\left(\frac{1}{N}A^TA + \beta I\right)\right\|$	$C_5(t) = 2C_1E_2\frac{s_1}{N}\left(\left\|K_\beta\right\| + \left\|\widetilde{K}(0)\right\|\varrho^t\right)$
$C_6(t) = \frac{2s_d}{s_d+N\beta} - 2\frac{s_1}{N}\left\|\widetilde{K}(0)\right\|\varrho^{t+1}$	$C_7(t) = 2C_1E_1\left(\left\|K_\beta\right\| + \left\|\widetilde{K}(0)\right\|\varrho^t\right)$
$C_8(t) = (V_2+1)\frac{s_1^2}{N}(dC_3 + \left\|K_\beta\right\|^2 + 2C_2\left\|K_\beta\right\|\sum_{j=0}^t \rho^j + \left\|\widetilde{K}(0)\right\|_F^2\mu^{t+1}$ $+2\left\|K_\beta\right\|\left\|\widetilde{K}(0)\right\|\rho^{t+1}) + 0.5$	
$z(t) = x(t) - x^*$	$R_1(t) = 1 + \delta^2C_8(t) + \alpha\delta C_5(t) - \delta C_6(t)$
$R_2(t) = \delta^2V_1N(dC_3 + \left\|K_\beta\right\|^2 + 2C_2\left\|K_\beta\right\|\sum_{j=0}^t \rho^j + \left\|\widetilde{K}(0)\right\|_F^2\mu^{t+1}$ $+2\left\|K_\beta\right\|\left\|\widetilde{K}(0)\right\|\rho^{t+1}) + \frac{1}{2}\alpha^2C_7(t)^2$	

- Let $\mathbb{E}_t[\cdot]$ denote the total expectation of a function of the random variables $\{I_0, \ldots, I_t\}$ given the initial estimate $x(0)$ and initial pre-conditioner matrix $K(0)$. Specifically,

$$\mathbb{E}_t[\cdot] = \mathbb{E}_{I_0,\ldots,I_t}(\cdot),\ t \geq 0. \tag{2.43}$$

- Define the conditional variance of the stochastic gradient $g^{i_t}(t)$, which is a function of the random variable i_t, given the current estimate $x(t)$ and the current pre-conditioner $K(t)$ as

$$\mathbb{V}_{i_t}\left[g^{i_t}(t)\right] = \mathbb{E}_{i_t}\left[\left\|g^{i_t}(t) - \mathbb{E}_{i_t}\left[g^{i_t}(t)\right]\right\|^2\right] = \mathbb{E}_{i_t}\left[\left\|g^{i_t}(t)\right\|^2\right] - \left\|\mathbb{E}_{i_t}\left[g^{i_t}(t)\right]\right\|^2. \tag{2.44}$$

Additional notation are listed in Table 2.2. Among these notation, K_β is an approximation of the inverse Hessian matrix, to which the sequence of the pre-conditioner matrices converges in expectation. $R_1(t)$ and $R_2(t)$ characterize the estimation error after t iterations. The other notation in Table 2.2 are required to define $R_1(t)$ and $R_2(t)$.

We make the following assumption on the rank of the matrix $A^T A$.

Assumption 2.4. *Assume that the matrix $A^T A$ is full rank.*

Note that Assumption 2.4 holds true if and only if the matrix $A^T A$ is positive definite with $s_d > 0$. As the Hessian of the aggregate cost function $\sum_{i=1}^m F^i(x)$ is equal to $A^T A$ for all x (see (2.5), under Assumption 2.4, the aggregate cost function has a unique minimum point. Equivalently, the solution of the distributed least squares problem defined by (2.6) is unique.

We also assume, as formally stated in Assumption 2.5 below, that the variance of the stochastic gradient for each agent is bounded. This is a standard assumption for the analysis of stochastic algorithms [32].

Assumption 2.5. *Assume that there exist two non-negative real scalar values V_1 and V_2 such that, for each iteration $t = 0, 1, \ldots$ and each agent $i \in \{1, \ldots, m\}$,*

$$\mathbb{V}_{i_t}\left[g^{i_t}(t)\right] \leq V_1 + V_2 \left\| \textstyle\sum_{i=1}^m \nabla F^i(x(t))/m \right\|^2 .$$

Next, we present our key result on the convergence of Algorithm 2.

2.5.3 Convergence guarantees

Theorem 2.5. *Consider Algorithm 2 with parameters* $\beta > 0$, $\alpha < \min\left\{\frac{N}{s_d}, \frac{1}{L}, \frac{2}{(s_1/N)+\beta}\right\}$ *and* $\delta > 0$. *If Assumptions 2.4 and 2.5 are satisfied, then there exist two non-negative real scalar values* $E_1 \geq \sqrt{V_1 N}$ *and* $E_2 \geq \sqrt{V_2 N}$ *such that the following statements hold true.*

1. *If the stepsize* δ *is sufficiently small, then there exists a non-negative integer* $T < \infty$ *such that for any iteration* $t \geq T$, $R_1(t)$ *is positive and less than* 1.

2. *For an arbitrary time step* $t \geq 0$, *given the estimate* $x(t)$ *and the matrix* $K(t)$,

$$\mathbb{E}_t\left[\left\|z(t+1)\right\|^2\right] \leq R_1(t)\left\|z(t)\right\|^2 + R_2(t). \tag{2.45}$$

3. *Given arbitrary choices of the initial estimate* $x(0) \in \mathbb{R}^d$ *and the pre-conditioner matrix* $K(0) \in \mathbb{R}^{d\times d}$,

$$\lim_{t\to\infty} \mathbb{E}_t\left[\left\|z(t+1\right\|^2\right] \leq \delta^2 V_1 N\left(dC_3 + \left\|K_\beta\right\|^2 + \frac{2C_2\left\|K_\beta\right\|}{1-\rho}\right) + 2\alpha^2\left(C_1 E_1\left\|K_\beta\right\|\right)^2.$$

The implications of Theorem 2.5 are as follows.

- According to Part (1) and (2) of Theorem 2.5, for small enough values of the parameters α and stepsize δ, as $R_1(t) \in (0,1)$ after some finite iterations, Algorithm 2 converges *linearly* in expectation to a neighborhood of the minimum point x^* of the least-squares problem (2.6).

- According to Part (3) of Theorem 2.5, the neighborhood of x^*, to which the estimates of

Algorithm 2 converges in expectation, is $\mathcal{O}(V_1)$. In other words, the sequence of expected "distance" between the minima x^* of (2.6) and the final estimated value of Algorithm 2 is proportional to the variance of the stochastic gradients at the minimum point.

2.6 Experimental results

In this section, first, we present experimental results comparing Algorithm 1 with other state-of-the-art methods. We consider five benchmark collective data matrices A, namely *"ash608"*, *"bcsstm07"*, *"can_229"*, *"gr_30_30"*, and *"qc324"*, from the SuiteSparse Matrix Collection (http://sparse.tamu.edu/). The true value of the collective observation is $b = Ax^*$ where x^* is a d-dimensional vector with all entries equal to 1. The rows of (A, b) are distributed among $m = 10$ agents. As A^TA is positive definite except for *"can_229"*, the problem (2.1) has a unique solution x^* for the datasets except *"can_229"* which has multiple solutions.

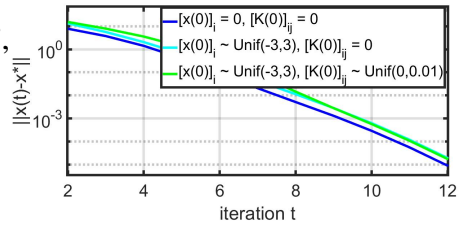

Figure 2.2: $\left\|x(t) - x^*\right\|$ under Algorithm 1 with different initialization on *"ash608"*. $\alpha = 0.1$, $\delta = 1$, $\beta = 0$.

We simulate Algorithm 1 with several initialization, for the datasets *"ash608"* and *"gr_30_30"*. For either of these datasets we consider three sets of initialization $(x(0), K(0))$: each entry of $x(0)$ and $K(0)$ is zero; each entry of $x(0)$ is selected *uniformly* at random within $(-3, 3)$ and each entry of $K(0)$ is zero; each entry of $x(0)$ and $K(0)$ is selected *uniformly* at random within $(-3, 3)$ and $(0, 0.01)$ respectively. We observe that, Algorithm 2.2.2 converges to x^* irrespective of the initial choice $x(0)$ and $K(0)$ (ref. Fig. 2.2-2.3).

The experimental results have been compared with the other distributed algorithms mentioned in Section 2.3. The parameters for all of these algorithms are chosen such that the respective algorithms achieve their optimal (smallest worst-case) convergence rate (ref. Table 2.3). For Algorithm 1 these optimal parameter values are given by $\alpha = \frac{2}{\lambda_1+\lambda_d+2\beta}$ and δ^* in (2.13). The optimal parameter expressions for the algorithms GD, NAG, HBM, and APC can be found in [5], [25]. The stepsize parameter for BFGS is selected using backtracking [22]. Note that evaluating the optimal tuning parameters for any of these algorithms requires knowledge about the smallest and largest eigenvalues of A^TA.

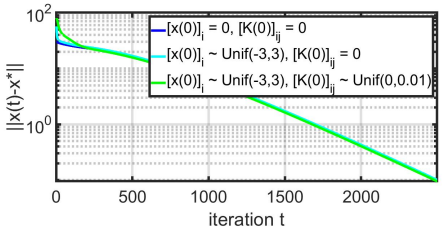

Figure 2.3: $\left\|x(t)-x^*\right\|$ under Algorithm 1 with different initialization on *"gr_30_30"*. $\alpha = 3\times10^{-3}$, $\delta = 0.4$, $\beta = 0$.

We compare the number of iterations needed by these algorithms to reach a *relative estimation error* defined as $\epsilon_{tol} = \left\|x(t)-x^*\right\|/\|x^*\|$ for unique solution or $\epsilon_{tol} = \left\|Ax(t)-b\right\|/\left\|Ax(0)-b\right\|$ for multiple solutions. Clearly, Algorithm 1 performs fastest among the algorithms except BFGS, significantly for the datasets *"bcsstm07"* and *"gr_30_30"* (Table 2.4). Thus, our theoretical claim on improvements over these methods is corroborated by the above experimental results.

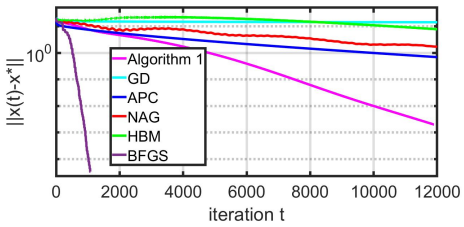

Figure 2.4: $\left\|x(t)-x^*\right\|$ in absence of noise for different algorithms on *"bcsstm07"*.

Observation Noise: We add *uniformly* distributed random noise vectors from $(-0.25, 0.25)$ and $(-0.15, 0.15)$, respectively, to the true output vectors of datasets *"ash608"* and *"gr_30_30"*. The algorithm parameters are chosen such that each algorithm achieves its minimum convergence

rate [5, 44] (ref. Table 2.3). Each iterative algorithm is run until the changes in its subsequent estimates is less than 10^{-4} over 20 consecutive iterations. We note the final estimation errors of the different algorithms in Table 2.5. We observe that the final estimation error of the IPG method is either comparable or favourable to all the other algorithms, for each dataset.

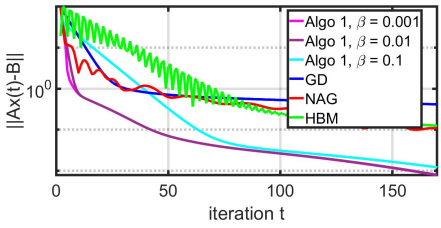

Figure 2.5: $\left\|x(t) - x^*\right\|$ in absence of noise for different algorithms on *"can_229"*.

Process noise: We simulate the algorithms by adding noise to the iterated variables. For the algorithms GD, NAG, HBM, and IPG, the process noise has been generated by rounding-off each entries of all the iterated variables in the respective algorithms to *four* decimal places. However, the rounding-off does not generate same values of noise w for the APC and BFGS algorithms. Therefore, for APC, we add *uniformly* distributed random numbers in the range $(0, 5 \times 10^{-5})$ for both the datasets, and similarly, for BFGS, we add *uniformly* distributed random numbers in the range $(0, 9 \times 10^{-5})$ and $(0, 2 \times 10^{-6})$ respectively for the datasets *"ash608"* and *"gr_30_30"*. The final estimation errors of different algorithms are noted in Table 2.5. We observe that the final error for IPG is less than all the other algorithms. Also, we observe that the estimation error for the BFGS algorithm on dataset *"ash608"* grows unbounded after 360 iterations. The cause for this instability is the violation of the non-singularity of the approximated Hessian matrix, which is a necessary condition for the convergence of BFGS [22].

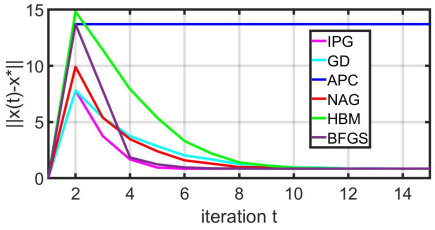

Figure 2.6: $\left\|x(t) - x^*\right\|$ in presence of observation noise for different algorithms on *"ash608"*.

2.6.1 Stochastic settings

We conduct experiments for different collective data points (A, b). Four of these datasets are from the the benchmark datasets available in SuiteSparse Matrix Collection. Particularly these four datasets are *"abtaha1"*, *"abtaha2"*, *"gre_343"*, and *"illc1850"*. The fifth dataset, *"cleveland"*, is from the UCI Machine Learning Repository [46]. The sixth and final dataset is the *"MNIST"* [47] dataset.

In the case of the first four aforementioned datasets, the problem is set up as follows. Consider a particular dataset *"abtaha1"*. Here, the matrix A has 14596 rows and $d = 209$ columns. The collective output vector b is such that $b = Ax^*$ where x^* is a 209 dimensional vector, all of whose entries are unity. The data points represented by the rows of the matrix A and the corresponding observations represented by the elements of the vector b are divided amongst $m = 4$ agents numbered from 1 to 4. Since the matrix A for this particular dataset has 14596 rows and 209 columns, each of the four agents $1, \ldots, 4$ has a data matrix A^i of dimension 3649×209 and a observation vector b^i of dimension 3649. The data points for the other three datasets,

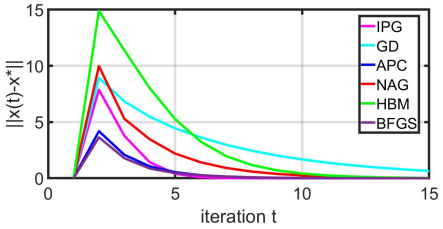

Figure 2.7: $\left\|x(t) - x^*\right\|$ in presence of process noise for different algorithms on *"ash608"*.

"abtaha2", *"gre_343"*, and *"illc1850"*, are similarly distributed among $m = 4$, $m = 7$ and $m = 10$ agents, respectively.

For the fifth dataset, 212 arbitrary instances from the *"cleveland"* dataset have been selected. This dataset has 13 numeric attributes, each corresponding to a column in the matrix A, and a target class (whether the patient has heart disease or not), which corresponds to the output vector b. Since the attributes in the matrix A has different units, its each column is shifted by the mean value of the corresponding column and then divided by the standard deviation of that column. Finally, a 212-dimensional column vector of unity is appended to this pre-processed matrix. This is our final input matrix A of dimension (212×14). The collective data points (A, B) are then distributed among $m = 4$ agents, in the manner described earlier.

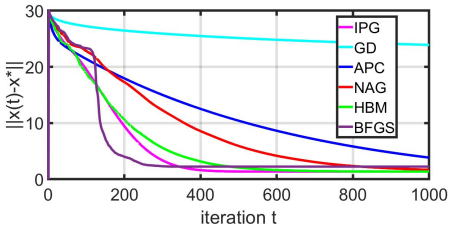

Figure 2.8: $\left\|x(t) - x^*\right\|$ in presence of observation noise for different algorithms on *"gr_30_30"*.

From the training examples of the *"MNIST"* dataset, we select 1500 arbitrary instances labeled as either the digit one or the digit five. For each instance, we calculate two quantities, namely the average intensity of an image and the average symmetry of an image [48]. Let the column vectors a_1 and a_2 respectively denote the average intensity and the average symmetry of those 1500. Then, our input matrix before pre-processing is $\begin{bmatrix} a_1 & a_2 & a_1.^2 & a_1.*a_2 & a_2.^2 \end{bmatrix}$. Here, $(.*)$ represents element-wise multiplication and $(.^2)$ represents element-wise squares. This raw input matrix is then pre-processed as described earlier

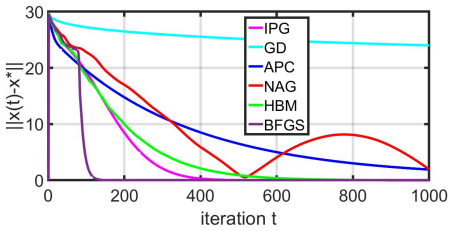

Figure 2.9: $\left\|x(t) - x^*\right\|$ in presence of process noise for different algorithms on *"gr_30_30"*.

for the *"cleveland"* dataset. Finally, a 1500-dimensional column vector of unity is appended to this pre-processed matrix. This is our final input matrix A of dimension (1500×6). The collective data points (A, B) are then distributed among $m = 10$ agents, in the manner already described for the other datasets. As the matrix $A^T A$ is positive definite in each of these cases, the optimization problem (2.6) has a unique solution x^* for all of these datasets.

We compare the performance of our proposed IPSG method (Algorithm 1) on the aforementioned datasets, with the other stochastic algorithms mentioned in Section 2.1.4. Specifically, these algorithms are stochastic gradient descent (SGD) [32], adaptive gradient descent (AdaGrad) [33], adaptive momentum estimation (Adam) [34], and AMSGrad [37] in the distributed network architecture of Fig. 2.1. These algorithms are implemented with different combinations of the respective algorithm parameters on the individual datasets. The parameter combinations are described below.

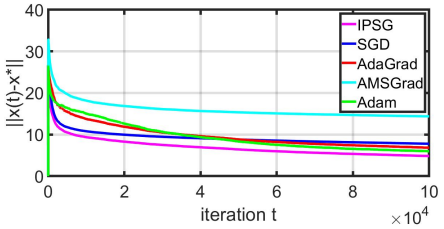

Figure 2.10: $\left\|x(t) - x^*\right\|$ for different stochastic algorithms on *"MNIST"*.

IPSG: The optimal (smallest) convergence rate of the deterministic version of the proposed IPSG method is obtained when $\alpha = \frac{2}{s_1+s_d}$ [44]. For each of the six datasets, we find that the IPSG method converges fastest when the parameter α is set similarly as $\alpha = \frac{2}{s_1+s_d}$. The stepsize parameter δ of the IPSG algorithm is chosen from the set $\{0.1, 0.5, 1, 2, 2.5\}$. The parameter β is chosen from the set $\{0.1, 0.5, 1, 5, 10, 30, 50\}$.

SGD: The SGD algorithm has only one parameter: the stepsize, denoted as α [32]. The deterministic version of the SGD method is the gradient-descent method, which has the optimal rate of convergence when $\alpha = \frac{2}{s_1+s_d}$. We find that the SGD method converges fastest when the stepsize parameter is similarly set as $\alpha = \frac{2}{s_1+s_d}$. s_1 and s_d depends on the collective data matrix

A, and their values may not be known to the server. When the actual values or estimates of s_1 and s_d are not known, the parameter α in both the IPSG and the SGD algorithm can be experimentally set by trying several values of different orders, as done for the parameters δ and β in the IPSG method.

AdaGrad: The stepsize parameter α of the AdaGrad algorithm [33] is selected from the set $\{1, 0.1, \frac{1}{t}\}$. The parameter ϵ is set at its usual value of 10^{-7}.

Adam and **AMSGrad**: The stepsize α [34, 37] is selected from the set $\{c, \frac{c}{\sqrt{t}}\}$ where c is from the set $\{1, 0.5, 0.1, 0.2, 0.05, 0.01\}$. The parameter β_1 is set at its usual value of 0.9. The parameter β_2 is selected from their usual values of $\{0.99, 0.999\}$. The parameter ϵ is set at 10^{-7}. The best parameter combinations from above, for which the respective algorithms converge in a fewer number of iterations, are reported for each dataset in Table 2.6.

The initial estimate $x(0)$ for all of these algorithms is chosen as the d-dimensional zero vector for each dataset except the *"cleveland"* dataset. For *"cleveland"* dataset, $x(0)$ is chosen as the d-dimensional vector whose each entry is 10. The initial pre-conditioner matrix $K(0)$ for the IPSG algorithm is the zero matrix of dimension $(d \times d)$.

We compare the number of iterations needed by these algorithms to reach a *relative estimation error* defined as $\epsilon_{tol} = \frac{\|x(t)-x^*\|}{\|x(0)-x^*\|}$. Each iterative algorithm is run (ref. Fig. 2.10-2.11) until its relative estimation error does not exceed ϵ_{tol} over a period of 10 consecutive iterations, and the smallest such iteration is reported in Table 2.7. The second column of Table 2.7 indicates the condition number $\kappa(A^TA)$ for each dataset. From Table 2.7, we see that the IPSG algorithm converges fastest among the algorithms

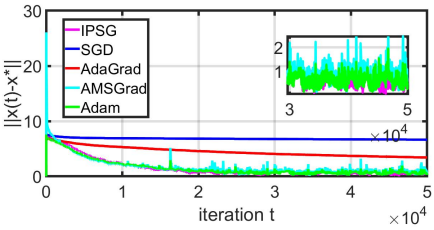

Figure 2.11: $\|x(t) - x^*\|$ for different stochastic algorithms on *"illc1850"*.

on four out of the six datasets, except for *"cleveland"* and *"abtaha2"*. Note that these two datasets have small condition number of order 10 and 10^2. Even for these two datasets, only the AMSGrad algorithm requires fewer iterations than IPSG. Moreover, from the datasets *"MNIST"*, *"gre_343"*, and *"illc1850"*, we observe that the differences between the proposed IPSG method and the other methods are significant when the condition number of the matrix A^TA is of order 10^3 or more. Thus, our claim on improvements over the prominent stochastic algorithms for solving the distributed least-squares problem (2.6) is corroborated by the above experimental results.

Table 2.3: The parameters used in different algorithms for their minimum convergence rate on distributed linear regression experiments.

Dataset	GD (δ)	NAG (α, β) [25]	HBM (α, β) [25]	APC (γ, η) [5]	Algorithm 1 (β, α, δ)
ash608	0.1163	(0.08, 0.5)	(0.15, 0.29)	(1.02, 5.27)	(0, 0.1163, 1)
bcsstm07	3×10^{-7}	(2×10^{-7}, 0.99)	(10^{-7}, 0.99)	(1.09, 12.8)	(0, 3×10^{-7}, 1)
can_229	0.024	(0.012, 0.98)	(0.012, 0.87)	N/A	(0.001, 0.022, 1.08)
gr_30_30	0.014	(0.009, 0.99)	(0.03, 0.98)	(1.09, 12.8)	(0, 0.014, 1)
qc324	0.85	(0.57, 0.99)	(0.03, 0.98)	(1.05, 18.9)	(0, 0.85, 1)

Table 2.4: The number of iterations required by different algorithms to attain relative estimation error ϵ_{tol} on distributed linear regression experiments.

Dataset	λ_1/λ_r	ϵ_{tol}	GD	NAG	HBM	APC	BFGS	Algo. 1
ash608	11.38	10^{-4}	37	23	21	15	15	**9**
bcsstm07	5.8×10^7	10^{-4}	$> 10^5$	5.64×10^4	4.87×10^4	4.85×10^4	**877**	1.19×10^4
can_229	1.4×10^4	10^{-4}	1.25×10^4	4.88×10^2	2.98×10^3	N/A	N/A	$\mathbf{1.6 \times 10^2}$
gr_30_30	3.79×10^4	10^{-4}	$> 10^5$	1.94×10^3	1.13×10^3	1.11×10^3	**85**	7.42×10^2
qc324	2.15×10^9	0.1	$> 10^5$	2.83×10^4	4.41×10^4	$> 10^5$	$\mathbf{1.74 \times 10^3}$	1.94×10^3

Table 2.5: Comparisons between the final estimation errors $\lim_{t\to\infty}\left\|x(t)-x^*\right\|$ for different algorithms on distributed linear regression experiments.

Noise type	Dataset	Noise level	IPG	GD	NAG	HBM	APC	BFGS
Obsrv. noise	ash608	$\eta = 8.23$	$\mathbf{0.86}$	$\mathbf{0.86}$	$\mathbf{0.86}$	$\mathbf{0.86}$	13.71	$\mathbf{0.86}$
	gr_30_30	$\eta = 7.21$	$\mathbf{1.35}$	2.05	$\mathbf{1.35}$	$\mathbf{1.35}$	1.82	2.25
Prcs. noise	ash608	$\omega = 9.3 \times 10^{-3}$	$\mathbf{0}$	3.46×10^{-4}	9.21×10^{-4}	10^{-4}	4.9×10^{-4}	∞
	gr_30_30	$\omega = 4.5 \times 10^{-2}$	$\mathbf{0}$	7.68	1.86	8.5×10^{-3}	3.72	1.49×10^{-2}

2.7 Summary

We have considered the multi-agent linear least-squares problem in a distributed server-agent network. Although several algorithms are available for solving this problem without requiring the agents to share their local data, their convergence speed is fundamentally limited by the condition number of the collective data. We have proposed an iterative pre-conditioning technique that is robust to the condition number. Thus, we can reach a satisfactory neighborhood of the desired solution in a provably fewer number of iterations than the existing state-of-the-art algorithms. The convergence analysis of our proposed Iteratively Pre-Conditioned Gradient-descent (IPG) algorithm and its comparison with related methods have been supported through experiments on real datasets.

We have considered practical settings with additive system noises: either observation noise or process noise. In this settings, our contribution has been analyzing the proposed IPG algorithm's robustness against such independent system noises whose magnitudes are bounded in expectation. The experimental results have reinforced our claim on the IPG method's superior accuracy compared to other state-of-the-art distributed algorithms when subjected to system noises.

Table 2.6: The parameters used in different stochastic algorithms on distributed linear regression experiments.

Dataset	IPSG	SGD [32]	AdaGrad [33]	AMSGrad [37]	Adam [34]
cleveland	$\alpha = 0.0031$, $\delta = 0.5$, $\beta = 30$	$\alpha = 0.0031$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha = 0.05$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	$\alpha = 0.05$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
abtaha1	$\alpha = 0.0052$, $\delta = 2$, $\beta = 5$	$\alpha = 0.0052$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{1}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.5}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
abtaha2	$\alpha = 0.0033$, $\delta = 2$, $\beta = 5$	$\alpha = 0.0033$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha = 1$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.5}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
MNIST	$\alpha = 0.0003$, $\delta = 0.1$, $\beta = 1$	$\alpha = 0.0003$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha = 1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	$\alpha = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
gre_343	$\alpha = 1.2$, $\delta = 2.5$, $\beta = 0.5$	$\alpha = 1.96$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.1}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.2}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
illc1850	$\alpha = 0.4436$, $\delta = 2$, $\beta = 1$	$\alpha = 0.4436$	$\alpha = 1$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.5}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-7}$	$\alpha_t = \frac{0.5}{\sqrt{t}}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$

We have further extended the idea of iterative pre-conditioning to the stochastic settings where, instead of full-batch data, only random data points are utilized in each iteration of the algorithms. We have presented convergence guarantee of the proposed Iteratively Pre-Conditioned Stochastic Gradient-descent (IPSG) and empirically show that the proposed IPSG method's convergence rate compares favorably to prominent stochastic algorithms for solving the distributed linear least-squares problem.

Table 2.7: Comparisons between the number of iterations required by different stochastic algorithms to attain the specified values for the relative estimation errors $\epsilon_{tol} = \left\|x(t) - x^*\right\| / \left\|x(0) - x^*\right\|$ on distributed linear regression experiments.

Dataset	$\kappa(A^T A)$	ϵ_{tol}	IPSG	SGD	AdaGrad	AMSGrad	Adam
cleveland	7.34	1.5×10^{-3}	4.11×10^3	4.71×10^3	6.04×10^3	$\mathbf{3.63 \times 10^3}$	4.11×10^3
abtaha1	1.5×10^2	10^{-3}	$\mathbf{7.35 \times 10^4}$	$> 10^5$	9.75×10^4	$> 10^5$	$> 10^5$
abtaha2	1.5×10^2	2×10^{-3}	9.86×10^4	$> 10^5$	$> 10^5$	$\mathbf{7.6 \times 10^4}$	$> 10^5$
MNIST	2.59×10^3	2.6×10^{-3}	$\mathbf{3.41 \times 10^4}$	$> 5 \times 10^4$	$> 5 \times 10^4$	$> 5 \times 10^4$	4.41×10^4
gre_343	1.25×10^4	4×10^{-3}	$\mathbf{3.88 \times 10^4}$	4.43×10^5	$> 10^5$	$> 10^5$	$> 10^5$
illc1850	1.93×10^6	0.2	$\mathbf{8.06 \times 10^4}$	3.31×10^5	2.81×10^5	$> 5 \times 10^5$	1.63×10^5

Chapter 3: Decentralized Linear Regression in Peer-to-Peer Network

3.1 Introduction

In this chapter, we consider solving a system of linear algebraic equations having at least one exact solution over a peer-to-peer network of agents. Specifically, we consider a network of m agents and the overall system is assumed to be synchronous. Each agent $i \in \{1, \ldots, m\}$ has n_i *local* data points, represented by a *data matrix* $A^i \in \mathbb{R}^{n_i \times d}$ and an *observation vector* $b^i \in \mathbb{R}^{n_i}$. In this network architecture, each agent $i \in \{1, \ldots, m\}$ can communicate with a set of certain other agents called its *neighbors*, represented by $\mathcal{N}^i$. We assume that the communication between the agents is bidirectional. This communication topology between the agents is represented by an undirected *graph* $\mathbb{G} = (\{1, \ldots, m\}, \mathcal{E})$, where an *edge* $(i, j) \in \mathcal{E}$ or $(j, i) \in \mathcal{E}$ if agent i and agent j are *neighbors*, for any $i, j \in \{1, \ldots, m\}$, $i \neq j$. Later, in Section 3.5, we consider the case of directed *graph*. The aim of the agents is to compute a common solution vector $x^* \in \mathbb{R}^d$ such that

$$A^i x^* = b^i, \quad \text{for all } i \in \{1, \ldots, m\}. \tag{3.1}$$

Since each agent only partially knows the *collective data points*, they collaborate with their own neighbors for solving the problem (3.1). However, the agents do not share their local data

points. An algorithm that prescribes instructions for the agents to jointly solve the above problem without sharing their data points is defined as a *decentralized algorithm*.

If $\mathbb{G}$ is *connected*, the solution vector x^* in (3.1) is a minima of the following least-squares problem [49]

$$x^* \in \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} \frac{1}{2} \left\| A^i x - b^i \right\|^2 . \tag{3.2}$$

The above optimization problem can be solved using the decentralized gradient-descent algorithm (DGD) [50]. To present our key contributions, we review below the DGD method.

3.1.1 Background on decentralized gradient-descent

The decentralized gradient-descent method is an iterative algorithm, wherein each agent maintains an estimate of a solution defined by (3.1) and updates it iteratively using its individual local cost function and its neighbors' estimates. To be precise, for each iteration $t = 0,\ 1, \ldots$, let $x^i(t) \in \mathbb{R}^d$ denote the estimate maintained by each agent $i \in \{1, \ldots, m\}$. The initial estimates $x^i(0)$ may be chosen arbitrarily from $\mathbb{R}^d$. For each iteration t, each agent broadcasts $x^i(t)$ to its neighbors $j \in \mathcal{N}^i$. Each agent i computes the gradient, denoted by $g^i(t)$, of its local cost function at $x(t)$. Specifically, for all $i \in \{1, \ldots,\ m\}$ and for all $t \in \{0,\ 1, \ldots\}$,

$$g^i(t) = \left(A^i\right)^T \left(A^i\, x^i(t) - b^i\right). \tag{3.3}$$

Having received the estimates $x^j(t)$ from its neighbors, each agent i updates $x^i(t)$ to $x^i(t+1)$ defined below. For all $t \geq 0$,

$$x^i(t+1) = x^i(t) - \alpha\delta\, g^i(t) + \delta \sum_{j \in \mathcal{N}^i} \Big(x^j(t) - x^i(t) \Big). \tag{3.4}$$

Here, δ is a positive scalar value commonly referred as the *step-size* and α is another positive scalar value attributing a relative weight to the local gradient $g^i(t)$ compared to the *consensus* terms $\sum_{j \in \mathcal{N}^i} \big(x^j(t) - x^i(t) \big)$.

For small enough step-size δ, the DGD algorithm has a *linear* convergence rate. However, its convergence rate is limited by the ratio between the largest eigenvalue and the smallest non-zero eigenvalue of the set of input matrices $\{ \big(A^i\big)^T A^i,\ i = 1, \ldots, m \}$ and that of the graph Laplacian, as we will show later in Section 3.3. We propose a *decentralized pre-conditioning* technique that works on top of the DGD algorithm. Specifically, each agent i computes a fixed *pre-conditioning matrix* K^i based only on its local input matrix. Before updating the local estimate, each agent multiplies its local gradient $g^i(t)$ and the consensus terms $\sum_{j \in \mathcal{N}^i} \big(x^j(t) - x^i(t) \big)$ by the matrix K^i. The classical pre-conditioning techniques require access to all the input matrices, and therefore, cannot be implemented in a decentralized network. Unlike these methods, in our case, each agent computes its pre-conditioning matrix based only on its local data. Hence, the name *decentralized pre-conditioning*.

3.1.2 Related Work

Several decentralized iterative algorithms have been proposed in the past decade for solving the system of linear equations (3.1). The notable ones among them can be found in [49, 51–59].

The projection-based algorithms in [51, 52] require each local input matrix A^i to have full row-rank. Otherwise, a set of linearly independent rows in each input matrix A^i needs to be obtained, which is computationally expensive, especially when the number of data points is large. Our proposed algorithm does not have such a requirement. Explicit *linear* convergence rates of these two algorithms have been provided [51, 52]. Explicit *linear* convergence rate has been provided also for the least-squares based solution in [53]. Linear convergence rates of the consensus-innovation and the consensus-residual algorithms proposed in [54] have been proved. The explicit expression of the convergence rate of the algorithm in [55] has been provided only under specific conditions, such as the *collective input matrix* $\left[(A^1)^T, \ldots, (A^m)^T\right]^T$ is orthogonal, or the communication graph is *complete*. The consensus flow-based algorithm in [56], the least-squares solver in [49], and the gradient-based algorithm in [57] require each agent to have a single data point, which is not the general case. A continuous-time finite-time solver has been proposed in [58]. However, no expression of its convergence time has been provided. The decentralized convex optimization algorithm DIGing in [60] solves (3.1). However, the convergence guarantee of the DIGing algorithm assumes the degenerate case of at least one input matrix A^i being full-rank. We consider a more general problem where none of the input matrices may be full-rank. Additionally, the convergence rate of DIGing increases with the number of agents in the network. The DADAM algorithm in [61] is a decentralized adaptive gradient algorithm for solving convex or non-convex optimization problems. The learning rate in DADAM is adaptively updated based on the past gradients. The theoretical regret of DADAM converges *sublinearly* for generic convex problems [61].

3.1.3 Communication delay

We consider a major practical challenge in solving (3.1): delay in the communication links between the agents. The communication delay from an agent i to agent j is modeled as τ^{ij}, where $\tau^{ij} = \tau^{ji} > 0$ is constant for any edge $(i, j) \in \mathcal{E}$. Other approaches to study the delay robustness problem can be found in [51], [62], [63].

The considered problem has also been addressed in prior works [64], [51], [62], [65], [66]. Initially [64] and later on [51] consider directed time-varying networks. [51] proves the global convergence of a projection-based asynchronous algorithm, with the assumption of extended neighbour graphs being repeatedly jointly strongly connected and bounded delays, and provides an upper bound on the convergence rate. Random communications have been considered in [62], [66] and algorithms have been provided with almost sure convergence to a solution of (3.1). [65] proves convergence of a communication-efficient extension of the algorithm in [64]. In the algorithm proposed in [65], each agent also needs to share the number of their neighbours. We guarantee convergence in case of a deterministic network topology, without assuming a bound on the constant delays and by the agents sharing only their estimates with neighbors.

The solution of (3.1) can also be obtained by solving a least-squares problem, such as in [51], [67], [59], [68]. When (3.1) is not solvable, the least-squares formulation has an advantage of finding the solution that best fits (3.1). The least-squares problem can also be solved by general distributed optimization algorithms that have been discussed in the literature. However, ill-conditioning of *collective input matrix* poses an additional challenge when there are communication delays between the agents. Existing distributed optimization algorithms [63], [69], [70], [71], [72] fare poorly if *collective input matrix* is ill-conditioned. The algorithms in [69], [70], [71] require

decreasing stepsize, which leads to slower convergence. [63] needs additional variables to be shared with neighbouring agents. [71] requires prior information on the delays. The algorithm proposed in [72] globally converges under strict convexity of each agent's cost function. Moreover, when *collective input matrix* is ill-conditioned, these gradient-based optimization approaches suffer from poor convergence rate or may even converge to an undesired point. We follow the same distributed optimization approach, however, our algorithm converges faster without requiring strict convexity of the local costs and shares only the estimates between neighbours. The key ingredient of our proposed approach is local pre-conditioning, which is obtained by solving an appropriate Lyapunov equation. Additionally, the proposed algorithm does not require any information about the heterogeneous communication delays, albeit that they are constant.

3.1.4 Summary of our contributions

In this dissertation, we propose a pre-conditioning technique that works on top of the aforementioned decentralized gradient-descent method when solving the system of linear algebraic equations in (3.1) over an undirected and connected peer-to-peer network. We rigorously analyze the convergence of our algorithm, and our key findings are summarized below.

3.1.4.1 In absence of communication delay

Rate of Convergence: We show that our algorithm converges *linearly*, when the system of linear equations (3.1) has at least one solution. Moreover, we provide an explicit convergence rate of our algorithm, detailed in Section 3.2.1 for unique solution and in Section 3.4 for multiple solutions. In Section 3.3, we show that our algorithm has a favorable convergence rate compared

to the decentralized gradient-descent algorithm, especially if the problem (3.1) is ill-conditioned.

Robustness: We analyse our algorithm's robustness against *computational process noise* when (3.1) has a unique solution. Please refer Section 3.2.2 for details. We demonstrate enhanced robustness when compared to the classical decentralized gradient-descent algorithm, and the difference between the robustness of the two algorithms is further accentuated when the linear system is ill-conditioned, as detailed in Section 3.3.

Empirical Study: In Section 3.8, we demonstrate the applicability of our algorithm to decentralized linear state estimation and propose a state predictor. In this context, we present empirical evidence of our algorithm's improved convergence rate and robustness when compared to most state-of-the-art decentralized algorithms mentioned in Section 3.1.2. The empirical results suggest the decentralized Kalman-consensus filter's (DKF) [73] faster convergence rate and smaller estimation error than our proposed state predictor. However, our algorithm has a smaller computational and communication cost than the DKF method. Please refer Section 3.8.1 for more details.

Finally, in Section 3.5, we extend our algorithm to strongly connected directed graphs, and show that our algorithm converges *linearly*, when (3.1) has a unique solution.

3.1.4.2 In presence of communication delay

Compared to the existing works that address communication delays in solving (3.1), our major contributions are follows:

- higher robustness to ill-conditioning of problem (3.1), unlike [69], [70], [71], [72].
- the communication delays are apriori unknown, unlike [71].

- addressing communication delays in solving least-squares formulation, unlike [64], [51].

3.2 Proposed algorithm

In this section, we present our algorithm, its computational complexity, its convergence, and its robustness properties, when there is no communication delay in the network.

We make the following assumptions about the problem.

Assumption 3.1. $x^* \in \mathbb{R}^d$ *is the unique solution of* (3.1).

Assumption 3.2. *The graph* $\mathbb{G}$ *is undirected and* connected.

The proposed algorithm is similar to the decentralized gradient-descent method described in Section 3.1.1. However, a notable difference is that in our algorithm, each agent multiplies its local gradient and consensus terms by a local *pre-conditioning* matrix. In literature, pre-multiplication of gradients by a matrix is commonly referred as *pre-conditioning* [40]. It should be noted that the conventional pre-conditioning techniques would require access to the agents' combined data points. However, each agent in our algorithm computes its pre-conditioning matrix based only on local data. The algorithm is iterative wherein each iteration $t \in \{0,\ 1, \ldots\}$, each agent $i \in \{1, \ldots, m\}$ maintains a local estimate $x^i(t)$ of the solution of linear equations (3.1). Each agent updates its local estimate using the steps presented below in Algorithm 3.

Initialization: Recall from Section 2.1 that, for each agent $i \in \{1, \ldots, m\}$, $\mathcal{N}^i$ denotes the set of its neighbors. Let $\left|\mathcal{N}^i\right|$ denote the cardinality of the set $\mathcal{N}^i$. Before starting the iterative process, each agent chooses two non-negative scalar real-valued parameters $\alpha,\ \delta$. The specific values of these parameters are presented later in Section 3.2.1. Further, each agent $i \in \{1, \ldots, m\}$

chooses an initial local estimate $x^i(0) \in \mathbb{R}^d$ and computes a local pre-conditioning matrix

$$K^i = \left(\alpha \left(A^i \right)^T A^i + \left| \mathcal{N}^i \right| I \right)^{-1}, \tag{3.5}$$

where I denote the $(d \times d)$-dimensional identity matrix. Since the matrix $\left(A^i\right)^T A^i$ is positive semi-definite, if $\alpha > 0$, $\left(\alpha \left(A^i \right)^T A^i + \left| \mathcal{N}^i \right| I \right)$ is positive definite and invertible.

Algorithm 3 Pre-conditioning for the decentralized gradient-descent method.

1: Each agent $i \in \{1, \ldots, m\}$ initializes $x^i(0) \in \mathbb{R}^d$, $\alpha > 0$ and $\delta > 0$.
2: Each agent $i \in \{1, \ldots, m\}$ computes its local pre-conditioning matrix K^i as defined by (3.5).
3: **for** each iteration $t = 0, 1, 2, \ldots$ **do**
4: Each agent $i \in \{1, \ldots, m\}$ receives the current estimates $x^j(t)$ from its neighbors $j \in \mathcal{N}^i$.
5: Each agent $i \in \{1, \ldots, m\}$ updates its current local estimate $x^i(t)$ to $x^i(t+1)$ such that

$$x^i(t+1) = x^i(t) - \alpha\delta K^i \left(A^i \right)^T \left(A^i\, x^i(t) - b^i \right) + \delta K^i \sum_{j \in \mathcal{N}^i} \left(x^j(t) - x^i(t) \right). \tag{3.6}$$

6: Each agent $i \in \{1, \ldots, m\}$ sends its updated local estimate $x^i(t+1)$ to all its neighbors $j \in \mathcal{N}^i$.
7: **end for**

Computational complexity: We now present the computational complexity of the proposed algorithm and compare it with related methods in terms of the total number of floating-point operations (*flops*). As floating-point multiplication is significantly costlier than additions [74], we ignore the additions while counting the total number of flops. Algorithm 3 requires $\mathcal{O}(n_i d)$ flops for each agent per iteration, which is the same as the DGD method. However, during initialization, each agent computes the pre-conditioning matrix K^i, which requires $\mathcal{O}(n_i d^2 + d^3)$ flops. Note that, initialization of the existing methods mentioned in Section 3.1.2 also has significant computational costs. The initialization of the projection-based algorithms in [51, 52] involves $\mathcal{O}(n_i d \min\{n_i, d\} + r_i^2 d + r_i^3)$ flops for each agent, where r_i is the rank of the matrix A^i.

The initialization of the least-squares solution in [53] requires $\mathcal{O}(n_i^2 d + n_i d^2 + n_i^3 + d^3)$ for each agent.

Next, we formally analyze the convergence of Algorithm 3.

3.2.1 Convergence guarantee

Notation: To formally state our convergence result, we note below some elementary facts and introduce some notation.

- For any pair of agents $i, j \in \{1, \ldots, m\}$, $i \neq j$, we define

$$c_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

- We define the matrix that represents all the agents' estimate update equation in Algorithm 3. Let,

$$M = \left[(M^1)^T, \ldots, (M^m)^T \right]^T, \tag{3.8}$$

whose i-th block-row M^i is defined as

$$M^i = K^i[-c_{i1}I, \ldots, -c_{i(i-1)}I, \left(\alpha \left(A^i \right)^T A^i + \left| \mathcal{N}^i \right| I \right), -c_{i(i+1)}I, \ldots, -c_{im}I]. \tag{3.9}$$

- For each $i \in \{1, \ldots, m\}$, the matrix $\left(A^i \right)^T A^i$ is positive semi-definite, and therefore, has non-negative eigenvalues. In general, when $\left(A^i \right)^T A^i$ is not the trivial zero matrix, at least

one of its eigenvalue is positive. We let $\overline{\lambda}^i$ and $\underline{\lambda}^i$ respectively denote the largest and the smallest non-zero eigenvalue of the matrix $\left(A^i\right)^T A^i$.

- We let L denote the Laplacian matrix of the graph $\mathbb{G}$ [75]. It is known that the Laplacian L is positive semi-definite, and therefore, has non-negative eigenvalues. We let $\overline{\lambda}_L$ and $\underline{\lambda}_L$ respectively denote the largest and the smallest non-zero eigenvalue of the matrix L.

- We let $\lambda_{max}(\cdot)$ and $\underline{\lambda}_{\min}(\cdot)$, respectively denote the largest and smallest non-zero eigenvalue of a matrix.

- For each iteration $t \geq 0$, we let $x(t)$ denote the md-dimensional vector obtained by vertical concatenation of all the agents' estimate at iteration t:

$$x(t) = \left[x^1(t)^T, \ldots, x^m(t)^T\right]^T. \tag{3.10}$$

Similarly, we let X^* denote the md-dimensional vector obtained by vertical concatenation of the unique solution x^* for all the agents:

$$X^* = \left[(x^*)^T, \ldots, (x^*)^T\right]^T. \tag{3.11}$$

Lemma 3.1 below states a preliminary result on the convergence of Algorithm 3 and is important for our key results presented afterward.

Lemma 3.1. *Consider Algorithm 3 with parameter $\alpha > 0$. Then, under Assumptions 3.1-3.2, all the eigenvalues of the matrix M are positive.*

Note that, under the conditions assumed in Lemma 3.1, $\lambda_{min}(M) > 0$. We now define

below a parameter that determines the *convergence rate* of Algorithm 3. Let,

$$\kappa_M = \frac{\max_{i=1,\ldots,m} \frac{\overline{\lambda}_L}{|\mathcal{N}^i|} + \max_{i=1,\ldots,m} \frac{\alpha\overline{\lambda}^i}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}}{\min\left\{\min_{i=1,\ldots,m} \frac{\underline{\lambda}_L}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}, \min_{i=1,\ldots,m} \frac{\alpha\underline{\lambda}^i}{\alpha\underline{\lambda}^i + |\mathcal{N}^i|}\right\}}. \tag{3.12}$$

The key result on the convergence of Algorithm 3 is presented below in the form of Theorem 3.1.

Theorem 3.1. *Consider Algorithm 3 with parameters $\alpha > 0$ and $\delta < \frac{2}{\lambda_{\max}(M)}$. Suppose each agent $i \in \{1, \ldots, m\}$ initializes $x^i(0) \in \mathbb{R}^d$. If Assumptions 3.1-3.2 hold, then there exists non-negative real values ρ with $\frac{\kappa_M - 1}{\kappa_M + 1} \leq \rho < 1$ such that, for each iteration $t \geq 0$ we have*

$$\left\|x(t+1) - X^*\right\| \leq \rho \left\|x(t) - X^*\right\|. \tag{3.13}$$

Since $0 \leq \rho < 1$, (3.13) in Theorem 3.1 implies that the sequence of estimates $\{x(t), t \geq 0\}$ converges to X^* with a *linear* convergence rate ρ. From the definitions (3.10)-(3.11), we conclude that the *local* estimates $\{x^i(t), t \geq 0\}$ of each agent $i \in \{1, \ldots, m\}$ converges to the solution x^* of (3.1).

Remark 3.1. *The explicit convergence rate of the projection-based algorithms in [51, 52] do not directly relate to the singular values of the local input matrices $\{A^i,\ i = 1, \ldots, m\}$ or parameters of the graph $\mathbb{G}$ that are well-known. On the other hand, the explicit convergence rate of Algorithm 3 directly relates to the eigenvalues of the set of matrices $\{\left(A^i\right)^T A^i,\ i = 1, \ldots, m\}$ and eigenvalues of the graph Laplacian (ref. (3.12)). Thus, the convergence rate of Algorithm 3 is easier to perceive than the algorithms in [51, 52].*

Next, we formally analyze the robustness of Algorithm 3 against *computational process*

noise.

3.2.2 Robustness against computational process noise

The convergence result in Theorem 3.1 considers an ideal setting where the system is free from *noise*. However, practical systems inevitably suffers from uncertainties or noise. The *computational process noise*, as the name suggests, models the uncertainties or jitters in the computation process due to hardware failures, quantization errors, or noisy communication links [28]. In the presence of computational process noise [27, 28], in each iteration t of Algorithm 3, each agent computes corrupted values for the current estimate $x(t)$, described formally as follows. Specifically, in each iteration t, and for each $i \in \{1, \ldots, m\}$, instead of computing $x^i(t)$ accurately, each agent i computes a corrupted estimate

$$x_o^i(t) = x^i(t) + \zeta^i(t), \tag{3.14}$$

where $\zeta^i(t) \in \mathbb{R}^d$. The vector $\zeta(t) = \left[\zeta^1(t)^T, \ldots, \zeta^m(t)^T\right]^T$ is referred as additive computational process noise. For each iteration t, we let $\mathbb{E}_{\zeta_t}[\cdot]$ denote the expectation of a function of the random vector $\zeta(t)$. We let $\mathbb{E}_t[\cdot]$ denote the total expectation of a function of the random vectors $\{\zeta(0), \ldots, \zeta(t)\}$.

In the presence of above computational process noise, Algorithm 3 is modified as follows. In each iteration t, each agent $i \in \{1, \ldots, m\}$ receives corrupted estimates $x_o^j(t)$ from its neighbors

$j \in \mathcal{N}^i$. Instead of (3.6), the updated estimate is

$$x^i(t+1) = x_o^i(t) - \alpha\delta K^i \left(A^i\right)^T \left(A^i\, x_o^i(t) - b^i\right) + \delta K^i \sum_{j\in\mathcal{N}^i} \left(x_o^j(t) - x_o^i(t)\right). \tag{3.15}$$

Recall that $x_o^i(t+1)$, defined by (3.14), is the corrupted value of $x^i(t+1)$.

To present our result, in Theorem 3.2 below, on the robustness of Algorithm 3 against the above computational process noise, we make the following assumption.

Assumption 3.3. *There exists $\omega < \infty$ such that for all $t \geq 0$ and $i \in \{1, \ldots, m\}$,*

$$\mathbb{E}_{\zeta_t}\left[\left\|\zeta^i(t)\right\|_1\right] \leq \omega. \tag{3.16}$$

For each iteration $t \geq 0$, we let $x_o(t)$ denote the corrupted md-dimensional vector obtained by vertical concatenation of all the agents' corrupted estimates at iteration t:

$$x_o(t) = \left[x_o^1(t)^T, \ldots, x_o^m(t)^T\right]^T. \tag{3.17}$$

Recall the definition of X^* in (3.11) and κ_M in (3.12).

Theorem 3.2. *Consider Algorithm 3 with parameters $\alpha > 0$ and $\delta < \frac{2}{\lambda_{\max}(M)}$, in the presence of computational process noise in (3.14), and modifications defined in (3.15). Suppose each agent $i \in \{1, \ldots, m\}$ initializes $x^i(0) \in \mathbb{R}^d$. If Assumptions 3.1-3.3 hold, then there exists non-negative real values ρ with $\frac{\kappa_M - 1}{\kappa_M + 1} \leq \rho < 1$ such that, for each iteration $t \geq 1$ we have*

$$\mathbb{E}_t\left[\left\|x_o(t) - X^*\right\|\right] \leq \rho^t \left\|x(0) - X^*\right\| + m\,\omega \sum_{i=0}^{t} \rho^i. \tag{3.18}$$

Additionally, $\lim_{t\to\infty} \mathbb{E}_t \left[\left\|x_o(t) - X^*\right\|\right] \leq \dfrac{m\,\omega}{1-\rho}$.

Remark 3.2. *Since* $0 \leq \rho < 1$*, Theorem 3.2 implies that Algorithm 3 under the influence of additive computational process noise* (3.14) *converges* linearly *in expectation within a distance of* $\frac{m\,\omega}{1-\rho}$ *from the true solution* x^* *of the linear equations defined in* (3.1)*.*

3.3 Comparison with decentralized gradient-descent

In this section, we present a discussion, theoretically comparing our proposed algorithm with the decentralized gradient-descent (DGD) method [50].

Convergence rate: First, we compare the smallest worst-case or *optimal* convergence rate of these two algorithms. From Theorem 3.1, the optimal convergence rate of Algorithm 3 is given by $\frac{\kappa_M - 1}{\kappa_M + 1}$. Thus, the optimal convergence rate is determined by the condition number bound κ_M of the matrix M, as defined by (3.12). The smaller is the value of κ_M, the faster is the convergence, and vice-versa. Note that, the DGD algorithm, as discussed in Section 3.1.1, is a special case of Algorithm 3 where the pre-conditioning matrix of each agent $i \in \{1, \ldots, m\}$ is the d-dimensional identity matrix, i.e., $K^i = I$. Thus, the optimal convergence rate of DGD is

$$\kappa_{DGD} = \frac{\overline{\lambda}_L + \max_{i=1,\ldots,m}\ \alpha\overline{\lambda}^i}{\min\left\{\underline{\lambda}_L,\ \min_{i=1,\ldots,m}\ \alpha\underline{\lambda}^i\right\}}. \tag{3.19}$$

For a comparison between κ_M, defined by (3.12), and κ_{DGD}, defined by (3.19), we consider the following case of problem (3.1). Suppose all the agents $i \in \{1, \ldots, m\}$ have the same number of neighbors, i.e., $\left|\mathcal{N}^i\right| = N$ for some positive integer N. We denote, $\overline{\mu} = \max_{i=1,\ldots,m}\ \overline{\lambda}^i$ and

$\underline{\mu} = \min_{i=1,\ldots,m} \underline{\lambda}^i$. From the definitions (3.12) and (3.19) then we have

$$\kappa_M = \frac{\frac{\overline{\lambda}_L}{N} + \frac{\alpha\overline{\mu}}{\alpha\overline{\mu}+N}}{\min\left\{\frac{\underline{\lambda}_L}{\alpha\overline{\mu}+N}, \frac{\alpha\underline{\mu}}{\alpha\underline{\mu}+N}\right\}}, \tag{3.20}$$

$$\kappa_{DGD} = \frac{\overline{\lambda}_L + \alpha\overline{\mu}}{\min\left\{\underline{\lambda}_L, \alpha\underline{\mu}\right\}}. \tag{3.21}$$

If we choose the parameter α in Algorithm 3 such that $\frac{\underline{\lambda}_L}{\alpha\overline{\mu}+N} \geq \frac{\alpha\underline{\mu}}{\alpha\underline{\mu}+N}$, such as small enough value of α, then from (3.20) we have $\kappa_M = \frac{\frac{\overline{\lambda}_L}{N} + \frac{\alpha\overline{\mu}}{\alpha\overline{\mu}+N}}{\frac{\alpha\underline{\mu}}{\alpha\underline{\mu}+N}} = \left(\frac{\overline{\mu}}{\underline{\mu}}\right)\left(\frac{\alpha\underline{\mu}+N}{\alpha\overline{\mu}+N}\right) + \frac{\overline{\lambda}_L}{\alpha\underline{\mu}} + \frac{\overline{\lambda}_L}{N}$. Since Gershgorin circle theorem [76] implies that $\overline{\lambda}_L \leq 2N$, from above we have that $\kappa_M \leq \left(\frac{\overline{\mu}}{\underline{\mu}}\right)\left(\frac{\alpha\underline{\mu}+N}{\alpha\overline{\mu}+N}\right) + \frac{\overline{\lambda}_L}{\alpha\underline{\mu}} + 2$. From (3.21), we have $\kappa_{DGD} = \begin{cases} \frac{\overline{\lambda}_L}{\underline{\lambda}_L} + \alpha\frac{\overline{\mu}}{\underline{\lambda}_L}, & \text{if } \alpha > \frac{\underline{\lambda}_L}{\underline{\mu}} \\ \frac{\overline{\mu}}{\underline{\mu}} + \frac{\overline{\lambda}_L}{\alpha\underline{\mu}}, & \text{if } \alpha \leq \frac{\underline{\lambda}_L}{\underline{\mu}} \end{cases} \geq \frac{\overline{\lambda}_L}{\underline{\lambda}_L} + \frac{\overline{\mu}}{\underline{\mu}}$. Since $\overline{\mu} > \underline{\mu}$ and $\alpha > 0$, we have $\left(\frac{\alpha\underline{\mu}+N}{\alpha\overline{\mu}+N}\right) < 1$. From the above two equations we conclude that, if the ratio $\frac{\overline{\mu}}{\underline{\mu}} >> 1$, then $\kappa_M < \kappa_{DGD}$. Hence, when (3.1) is ill-conditioned, Algorithm 3 provably improves upon the condition number κ_{DGD}, and therefore, the optimal convergence rate of DGD.

Robustness against computational process noise: Next, we compare the smallest worst-case or *optimal* final estimation error $\lim_{t\to\infty} \mathbb{E}_t\left[\left\|x_o(t) - X^*\right\|\right]$ of these two algorithms in the presence of computational process noise, as defined in Section 3.2.2. From Theorem 3.2, the optimal final estimation error is given by $m\,\omega^{\frac{\kappa_M+1}{2}}$ for Algorithm 3, and similarly, $m\,\omega^{\frac{\kappa_{DGD}+1}{2}}$ for the DGD method. From the earlier discussion in this section it follows that, when the problem (3.1) is ill-conditioned, Algorithm 3 provably improves upon the optimal final estimation error of the DGD method.

3.4 Multiple-solutions case

In this section, we consider the case when the linear equations in (3.1) have multiple exact solutions, defined as

$$X_m^* = \{x^* \in \mathbb{R}^d : \ A^i x^* = b^i, \forall i \in \{1, \dots, m\}\}. \tag{3.22}$$

Below, we present convergence analysis of the proposed algorithm in this case. Assumption 3.1 is not required.

To present our key result in this section, we define the matrix

$$M^{\mathcal{N}} = Diag(\frac{1}{|N^1|}I, \dots, \frac{1}{|N^m|}I)\,(L \otimes I)\,. \tag{3.23}$$

Then, the ratio between the largest and the smallest non-zero eigenvalue of $M^{\mathcal{N}}$ is

$$\kappa_{\mathcal{N}} = \frac{\overline{\lambda}_L \max_{i=1,\dots,m} |\mathcal{N}^i|}{\underline{\lambda}_L \min_{i=1,\dots,m} |\mathcal{N}^i|}.$$

The main result on the convergence of Algorithm 3, when the set X_m^* in (3.22) is not a singleton, is presented below in the form of Theorem 3.3.

Theorem 3.3. *Consider Algorithm 3 with parameters $\alpha > 0$ and $\delta < \frac{2}{\lambda_{max}(M)}$. Suppose each agent $i \in \{1, \dots, m\}$ initializes $x^i(0) \in \mathbb{R}^d$. If Assumption 3.2 holds, then there exists non-negative real values ρ with $\frac{\max\{\kappa_M, \kappa_{\mathcal{N}}\}-1}{\max\{\kappa_M, \kappa_{\mathcal{N}}\}+1} \leq \rho < 1$ such that, each $x^i(t)$ converges to the same solution in X_m^* at a linear convergence rate ρ.*

3.5 Directed-graph case

In this section, we consider the communication graph $\mathbb{G}$ to be directed. We make the following assumption.

Assumption 3.4. *The graph* $\mathbb{G}$ *is* strongly connected.

Since the graph is directed, we make two modifications to Algorithm 3 as follows. Instead of (3.5), each agent $i \in \{1, \ldots, m\}$ computes its local pre-conditioning matrix

$$K^i = \left(\alpha \left(A^i \right)^T A^i + \left| \mathcal{N}_{in}^i \right| I \right)^{-1}, \tag{3.24}$$

where the $\mathcal{N}_{in}^i$ represents the set of neighbors from which agent i receives information. Accordingly, each agent $i \in \{1, \ldots, m\}$ updates its current estimate to

$$x^i(t+1) = x^i(t) - \alpha \delta K^i \left(A^i \right)^T \left(A^i \, x^i(t) - b^i \right) + \delta K^i \sum_{j \in \mathcal{N}_{in}^i} \left(x^j(t) - x^i(t) \right). \tag{3.25}$$

Also, instead of (3.7), we define $c_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and zero otherwise. We have the following result on the convergence of Algorithm 3 in the case of directed graph.

Theorem 3.4. *Consider Algorithm 3, with the modifications defined in* (3.24)-(3.25) *and the parameter* $\alpha > 0$*. Suppose each agent* $i \in \{1, \ldots, m\}$ *initializes* $x^i(0) \in \mathbb{R}^d$*. If Assumption 3.1 and Assumption 3.4 hold and the step-size* $\delta > 0$ *is small enough, then each* $x^i(t)$ *converges to the unique solution* x^* *of* (3.1) *at a linear rate.*

3.6 Application: decentralized state estimation

In this section, we formulate the decentralized state estimation problem for linear time-invariant (LTI) systems [77] in the framework of problem (3.1). Consider an LTI system having the discrete-time dynamics

$$z(t+1) = \mathcal{A}z(t),\ t \in \{0,\ 1, \ldots\}, \tag{3.26}$$

where $z(t) \in \mathbb{R}^d$ is the system-state at time-instant t and the *state matrix* of the LTI system (3.26), denoted by $\mathcal{A}$, is a $(d \times d)$-dimensional real matrix. Each agent $i \in \{1, \ldots, m\}$ in the system observes d scalar local outputs, defined by

$$\overline{y}^i(t) = C^i z(t),\ t \in \{0,\ 1, \ldots,\ d-1\}. \tag{3.27}$$

Here, C^i is a d-dimensional real row vector. The agents' task is to estimate the state $z(t)$, without sharing their local outputs $\overline{y}^i(t)$. Since an agent cannot access the collective output vectors $\{\overline{y}^1(t), \ldots, \overline{y}^m(t),\ t = 0, \ldots, d-1\}$, they collaborate with their neighbors to estimate $z(t)$. The aforementioned problem is referred as *decentralized state estimation* [77].

We define the following notation.

- Define the $(d \times d)$-dimensional *local observability matrix* of each agent $i \in \{1, \ldots, m\}$ as

$$O^i = \begin{bmatrix} (C^i)^T & (C^i\mathcal{A})^T & \ldots & (C^i\mathcal{A}^{d-1})^T \end{bmatrix}^T. \tag{3.28}$$

- Define the $(m \times d)$-dimensional *global output matrix*

$$C = \begin{bmatrix} (C^1)^T & \dots & (C^m)^T \end{bmatrix}^T. \tag{3.29}$$

- Define $(md \times d)$-dimensional *global observability matrix*

$$\overline{O} = \begin{bmatrix} C^T & (CA)^T & \dots (CA^{d-1})^T \end{bmatrix}^T. \tag{3.30}$$

- For each agent $i \in \{1, \dots, m\}$, we let y^i denote the d-dimensional column vector obtained upon stacking the local outputs $\{\overline{y}^i(t) \in \mathbb{R},\ t = 0, \dots, d-1\}$ as

$$y^i = \begin{bmatrix} \overline{y}^i(0) & \dots & \overline{y}^i(d-1) \end{bmatrix}^T. \tag{3.31}$$

- We let $\overline{Y}$ denote the md-dimensional column vector obtained upon stacking the local outputs of all the agents:

$$\overline{Y} = \begin{bmatrix} \overline{y}^1(0) \dots \overline{y}^m(0) \dots \overline{y}^1(d-1) \dots \overline{y}^m(d-1) \end{bmatrix}^T. \tag{3.32}$$

Assumption 3.5. *The* global system $(\mathcal{A}, C)$ *is jointly observable, i.e., the global observability matrix* $\overline{O}$ *is full-rank.*

First, we estimate the initial state $z(0)$ of the system (3.26)-(3.27) as follows. Upon

substituting from (3.29) in (3.30) we have

$$\overline{O} = \begin{bmatrix} (C^1)^T \dots (C^m)^T \dots (C^1\mathcal{A}^{d-1})^T \dots (C^m\mathcal{A}^{d-1})^T \end{bmatrix}^T. \tag{3.33}$$

Upon substituting from (3.27) and (3.26) in (3.32) we obtain that

$$\overline{Y} = \begin{bmatrix} (C^1)^T \dots (C^m)^T \dots (C^1\mathcal{A}^{d-1})^T \dots (C^m\mathcal{A}^{d-1})^T \end{bmatrix}^T z(0) \overset{(3.33)}{=} \overline{O} z(0). \tag{3.34}$$

We define the following vector Y, which is a rearrangement of the rows in $\overline{Y}$ (ref. (3.32)) so that the measurements of each agent $i \in \{1, \dots, m\}$ are stacked together as follows:

$$Y = \begin{bmatrix} \overline{y}^1(0) \dots \overline{y}^1(d-1) \dots \overline{y}^m(0) \dots \overline{y}^m(d-1) \end{bmatrix}^T. \tag{3.35}$$

Similarly, we define the following matrix O as a rearrangement of the rows in the global observability matrix $\overline{O}$ (ref (3.33)):

$$O = \begin{bmatrix} (C^1)^T \dots (C^1\mathcal{A}^{d-1})^T \dots (C^m)^T \dots (C^m\mathcal{A}^{d-1})^T \end{bmatrix}^T. \tag{3.36}$$

Upon substituting from (3.28) in (3.36) we have

$$O = \begin{bmatrix} (O^1)^T & \dots & (O^m)^T \end{bmatrix}^T. \tag{3.37}$$

Upon substituting from (3.27) and (3.26) in (3.35) we obtain that

$$Y = \begin{bmatrix} (C^1)^T \dots (C^1\mathcal{A}^{d-1})^T \dots (C^m)^T \dots (C^m\mathcal{A}^{d-1})^T \end{bmatrix}^T z(0) \overset{(3.36)}{=} Oz(0). \tag{3.38}$$

From (3.35) and (3.36), it follows that $\overline{Y} = \overline{O}z(0)$ in (3.34) is a rearrangement of the equations $Y = Oz(0)$ in (3.38). So, (3.34) and (3.38) are equivalent. Upon substituting from (3.31) in (3.35), $Y = \begin{bmatrix} (y^1)^T & \dots & (y^m)^T \end{bmatrix}^T$. From (3.37) and above, (3.38) is equivalent to

$$y^i = O^i z(0),\ i = 1, \dots, m. \tag{3.39}$$

Under Assumption 3.5, the matrix $\overline{O}$ is full-rank. Since O is a rearrangement of the rows in $\overline{O}$, O is full-rank. From (3.37), then it follows that the problem (3.39) has a unique solution $z(0)$ which is the true initial state of (3.26)-(3.27).

Upon iterating (3.26) from t to 0,

$$z(t) = \mathcal{A}^t z(0). \tag{3.40}$$

Thus, each agent estimates $z(t)$ at any instant $t > 0$ from the decentralized solution of the equations (3.39) and using (3.40).

Comparison with related work: The hybrid observer proposed in [78] partially uses a decentralized solution to linear algebraic equations. However, a major advantage of our aforementioned approach is its scalability. Since the per-iteration computational complexity is comparable to our algorithm and the required minimum number of iterations in each estimation loop is $\mathcal{O}(m^2)$, the

total minimum computational cost of the hybrid observer in [78] increases quadratically with the number of agents. The per-iteration computational cost of the continuous-time decentralized observer in [79] is $\mathcal{O}(n_i d_i^2 + (\sum_{j=1}^{i} d_j)^3)$ for agent i, where $\{d_i,\ i = 1, \ldots, m\}$ is any partition of the dimension of the state vector such that $\sum_{j=1}^{m} d_j = d$. Thus, the computational cost of our algorithm is less than the observer in [79]. Additionally, the explicit linear convergence of the observer in [79] is known only after a finite time. Moreover, unlike the aforementioned observers in [78]- [79], we have rigorously analyzed the robustness of our method in the presence of computational process noise and characterized the worst-case estimation error.

3.7 Proposed algorithm in presence of communication delay

In this section, we propose a distributed optimization algorithm for solving (3.1) that is robust to communication delays. The proposed algorithm is a modification of proportional-integral (PI) consensus-based gradient descent method [72], wherein we (a) remove the integral terms from the algorithm and (b) introduce pre-conditioning at every node. It is evident from the analysis of the proposed method, that integral terms are not required for linear problems. The pre-conditioning matrices have been introduced in the algorithm to take care of ill-conditioning of (3.1). Even in the synchronous setting without any communication delay, ill-conditioning of (3.1) slows down convergence or, even worse, can make the agents converge to an undesired solution. Proper choice of conditioning matrices can tolerate ill-conditioning.

For each agent $i \in [m]$, we denote the local gradients as

$$\phi^i(x^i) = (A^i)^T(A^i x^i - b^i). \tag{3.41}$$

Table 3.1: Required notations for solving decentralized linear equations subject to communication delay.

Symbol	Meaning
$\mathcal{N}^i$	neighbor set of node i
$\dot{x}$	first derivative of x w.r.t time t
$\|S\|$	cardinality of a set S
$A \succ 0$	matrix A is positive definite
I	identity matrix of order n
$\mathcal{L}_\infty$	set of bounded functions
S_+^d	set of symmetric positive semi-definite matrices of order d
S_{++}^d	set of symmetric positive definite matrices of order d
$[m]$	$\{1, \ldots, m\}$

Define local preconditioner matrix K^i which is the solution of the Lyapunov equation [80]

$$-N^i K^i - K^i N^i = -2kI, \tag{3.42}$$

with $k > 0$ and $N^i = (A^i)^T A^i + |\mathcal{N}^i| I$. Since $N^i \succ 0$ and $k > 0$, (3.42) has a unique symmetrical solution that is specified later in the convergence analysis of the algorithm. Using the above definitions, we describe Algorithm 4 below.

Algorithm 4

1: **for** $t \geq 0$ **do**
2: **for** each agent $i \in [m]$ **do**
3: receive $x^j(t - \tau^{ji}), \forall j \in \mathcal{N}^i$
4: update estimate

$$\dot{x}^i(t) = K^i[\eta(\eta I + K^i)^{-1} \sum_{j \in \mathcal{N}^i} (x^j(t - \tau^{ji}) - x^i(t)) - \phi^i(x^i(t))]. \tag{3.43}$$

5: transmit updated estimate to all $j \in \mathcal{N}^i$
6: **end for**
7: **end for**

In step 3 of Algorithm 4, each $x^j(t - \tau^{ji})$ can be set to any value for $t < \tau^{ji}$.

3.7.1 Convergence guarantee

Lemma 3.2. *For every* $i \in [m]$, $\overline{K}^i := (K^i + \eta I)^{-1}(K^i - \eta I)$ *is Schur for any* $\eta > 0$.

In order to establish convergence of Algorithm 4, we develop a framework based on [72] and pre-conditioning. The analysis of our algorithm then easily follows from this framework. Note that, the following framework is solely for analysis purposes.

Define consensus terms for each agent:

$$v^i = \sum_{j \in \mathcal{N}^i} v^{ij},\ v^{ij} = K^i(r_{ij} - x^i). \tag{3.44}$$

In this framework, r_{ij} is an external input to agent i from its neighbor $j \in \mathcal{N}^i$. In order to evaluate these r_{ij}'s, we define the following transformations [72]:

$$\vec{s}_{ij} = \frac{1}{\sqrt{2\eta}}(-v^{ij} + \eta r_{ij}),\ \overleftarrow{s}_{ij} = \frac{1}{\sqrt{2\eta}}(v^{ij} + \eta r_{ij}), \tag{3.45}$$

$$\overleftarrow{s}_{ji} = \frac{1}{\sqrt{2\eta}}(v^{ji} + \eta r_{ji}),\ \vec{s}_{ji} = \frac{1}{\sqrt{2\eta}}(-v^{ji} + \eta r_{ji}), \tag{3.46}$$

where $\eta > 0$. Information about agent j's estimate is contained in v^{ji} which is sent to neighbor agent $i \in \mathcal{N}^j$ in the form of $\vec{s}_{ji}$. Considering the delay model defined in Section 3.1.3, these communication variables are related as

$$\overleftarrow{s}_{ji}(t) = \vec{s}_{ij}(t - \tau^{ij}),\ \overleftarrow{s}_{ij}(t) = \vec{s}_{ji}(t - \tau^{ji}). \tag{3.47}$$

Upon receiving $\overleftarrow{s}_{ij}$, which is the variable $\vec{s}_{ji}$ sent by agent j but delayed in time by τ^{ji}, agent i

then recovers r_{ij} from it using (3.45). Hence, r_{ij} contains information about delayed estimates of its neighbor j. These transformations are commonly known as scattering transformation [72]. We assume, $\vec{s}_{ij}(t) = \vec{s}_{ji}(t) = 0 \; \forall t < 0$.

Using the above definitions, we propose the following lemma which will be useful in proving convergence of Algorithm 4.

Lemma 3.3. *Consider the following dynamics for each agent $i \in [m]$:*

$$\dot{x}^i = v^i - K^i \phi^i(x^i), \tag{3.48}$$

the transformation (3.45)-(3.46) *and delays* (3.47) *for $j \in \mathcal{N}^i$, $\forall i$. If Assumptions 3.1-3.2 hold, then $x^i \to x^* \; \forall i \in [m]$ as $t \to \infty$.*

The following theorem shows global convergence of Algorithm 4.

Theorem 3.5. *Consider Algorithm 4. If Assumptions 3.1-3.2 hold, then $x^i \to x^* \; \forall i \in [m]$ as $t \to \infty$.*

3.8 Experimental results

First, we present experimental results for solving the decentralized state estimation problem described in Section 3.6. We consider an LTI system with $m = 5$ agents, as described in (3.26)-

(3.27), given by

$$\mathcal{A} = \begin{bmatrix} -3 & -0.1 & 0 & 0 & 0 \\ 0.1 & 1 & -0.1 & 0 & 0 \\ 0 & 0.1 & -3 & 0.1 & 0 \\ 0 & 0 & 0.1 & 1 & -0.1 \\ 0 & 0 & 0 & 0.1 & -3 \end{bmatrix} C^1 = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 \end{bmatrix}, C^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$C^3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}, C^4 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}, C^5 = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 \end{bmatrix}.$$

The graph G is considered to be a ring. The true initial state is $z(0) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. We denote the combined true initial state of the LTI system by $Z_0 = \left[z(0)^T, \ldots, z(0)^T\right]^T$. For this system, Assumption 3.5 is satisfied. The agents estimate $z(0)$ by solving (3.39) using a decentralized algorithm. To solve (3.39), we apply Algorithm 3 and compare the experimental results with the prominent decentralized algorithms in [50–53, 55, 58, 60, 61].

We simulate each algorithm in the presence of computational process noise defined by (3.14). This noise is generated by rounding off each entry of all the iterated variables in the respective algorithms to *three* decimal places. For an unbiased comparison, the parameters of the respective algorithms are selected such that each algorithm converges in a fewer number of iterations. Specifically, these parameters are selected as follows. **Algorithm 3**: The optimal convergence rate of Algorithm 3 is obtained when the step-size $\delta = \frac{2}{\lambda_{\max}(M)+\lambda_{\min}(M)}$ [21]. The parameter α is chosen from the set $\{10^p : p = -4, -3, \ldots, 4\}$. **DGD**: The optimal convergence rate of the DGD algorithm is similarly given by $\delta = \frac{2}{\lambda_{\max}(M)+\lambda_{\min}(M)}$, where $K^i = I$ for each agent

$i \in \{1, \ldots, m\}$. The other parameter α is chosen from the set $\{10^p : p = -4, -3, \ldots, 4\}$. **Algorithm in [53]**: The parameter c is chosen from the set $\{0.1, 1, 3, 5, 10, 100\}$. **Algorithm in [52]**: The step-size γ_i is chosen from the set $\{0.25, 0.5, 0.75, 0.9\}$ for each agent i. **Algorithm in [55]**: The parameter γ_i is chosen from the set $\{0.25, 0.5, 0.75, 0.99\}$ for each agent i. **Algorithm in [58]**: The parameter $k(t) = \frac{1}{t+1} + \delta$, where δ is chosen from the set $\{0, 0.001, 0.01, 0.1\}$. **DIGing**: The step-size α is chosen from the set $\{0.01, 0.02..., 0.09\}$ for each agent. **DADAM**: The parameters $\beta_1, \beta_2, \beta_3$ are set to their default values $0.9, 0.999, 0.9$, respectively. The step-size α_t is chosen from the set $\{c \times 10^p : c = 1, \frac{1}{\sqrt{t}}, \frac{1}{t}, p = 0, -1, -2\}$. The best parameter choice for each of the aforementioned algorithms is tabulated in Table 3.2. The algorithm in [58] is initialized as per instructions. All the other algorithms are initialized with $x^i(0) = 0_d$ for each agent $i \in \{1, \ldots, m\}$.

First, we compare the number of iterations needed by these algorithms to reach a *relative estimation error* defined as $\epsilon_{tol} = \frac{\|x(t) - Z_0\|}{\|x(0) - Z_0\|}$. Each iterative algorithm is simulated (ref. Fig. 3.1) until its relative estimation error does not exceed ϵ_{tol} over a period of 10 consecutive iterations. We observe that the proposed Algorithm 3 requires the least number of iterations among all algorithms. Next, we compare the final estimation error $\lim_{t\to\infty} \|x(t) - Z_0\|$ of these algorithms in Table 3.2. We observe that the final estimation error of Algorithm 3 is either favourable or comparable to the other algorithms.

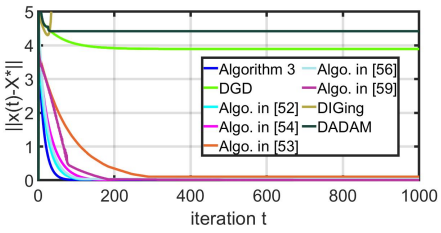

Figure 3.1: Error in estimating the true initial state $z(0)$ in presence of computational process noise, for different algorithms.

Next, we compare the aforementioned state predictor with the decentralized Kalman-consensus filter proposed in [73].

3.8.1 Comparison with decentralized Kalman filter

We consider the LTI system described above in this section but in the presence of *process noise* and *measurement noise*. Specifically, at each sampling instant of the state dynamics (3.26), we add Gaussian zero-mean process noise of standard deviation 10^{-3}. We add Gaussian zero-mean measurement noise of standard deviation 10^{-3} to each local output in (3.27).

We simulate the decentralized Kalman-consensus filter (DKF) [73] with parameters $\epsilon = 0.1$, process noise covariance $Q = 10^{-3}I$, measurement noise variance $R = 10^{-3}$, and initial estimation error covariance $P_0 = 5I$. We simulate Algorithm 3 with the parameters listed in Table 3.2. For both algorithms, each entry of each agent's initial estimate is randomly selected from Gaussian distribution of zero mean and 0.1 standard deviation. We let $Z(t) = \left[z(t)^T, \ldots, z(t)^T\right]^T$ denote the combined true states.

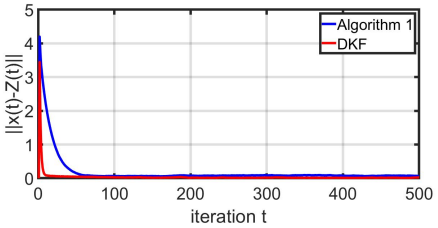

Figure 3.2: Error in estimating the system states $z(t)$ in presence of process noise and measurement noise, for Algorithm 3 and DKF.

From Figure 3.2, DKF converges in fewer iterations than Algorithm 3. However, in DKF, each agent computes inverses of two $(d \times d)$-dimensional matrices at each iteration. So, the per-iteration computational complexity of each agent in the DKF algorithm is $\mathcal{O}(d^3)$, compared to $\mathcal{O}(n_i d)$ of Algorithm 3. Besides, the communication complexity of DKF per-agent is $\mathcal{O}(d^2)$, compared to $\mathcal{O}(d)$ of Algorithm 3. Next, we compare the final estimation error $\lim_{t\to\infty} \left\|x(t) - Z_t\right\|$ of both algorithms. Each algorithm is simulated until the changes in its subsequent estimates are less than 0.005 over 10 consecutive iterations. We note the final estimation error of Algorithm 3 and DKF respectively are 0.03 and 0.009. The better estimation error of the DKF can be attributed

to it being a filtering technique, i.e., the estimate is based on the current outputs. In contrast, our proposed estimation technique involving Algorithm 3 is a predictor, i.e., its estimates are based only on the outputs from sampling instant 0 to $d-1$.

Table 3.2: Comparisons between the performance of different algorithms on decentralized state prediction experiments.

Algorithm	Algo. 3	DGD	Algo. in [51]	Algo. in [53]	Algo. in [52]	Algo. in [55]	Algo. in [58]	DIGing [60]	DADAM [61]
Para-meters	$(\alpha, \delta) = (10^3, 1.07)$	$(\alpha, \delta) = (0.001, 0.12)$	N/A	$c = 10^2$	$\gamma_i = 0.5$	$\gamma_i = 0.99$	$\delta = 0$	$\alpha = 0.00005$	$(\beta_1, \beta_2, \beta_3 \alpha_t) = (0.9, 0.999, 0.9, \frac{0.1}{\sqrt{t}})$
Iterations ($\epsilon_{tol} = 0.01$)	67	$> 10^3$	105	133	$> 10^3$	113	181	$> 10^3$	$> 10^3$
Final error	0.01	3.89	0.03	0.02	0.11	0.04	0.005	∞	4.42

3.8.2 In presence of delay

Consider the following numerical example. The matrix A is a real symmetric positive definite matrix from the benchmark dataset[1] *"bcsstm19"* which is part of a suspension bridge problem. The dimension of A is (817×817) and its condition number is 2×10^5. The rows of A and the corresponding rows of b are distributed among $m = 5$ agents, so that four of them know 163 such rows and the remaining 165 rows are known by the fifth agent. The network is of ring topology and $\tau^{ij} = 5$. We set $b = Ax^*$ where $x^* = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$ is the unique solution.

We implement Algorithm 4 on this numerical example. The differential equations governing x^i's are solved numerically with small stepsizes $h = 10^{-3}$. Each agent converge to the desired

[1] https://sparse.tamu.edu

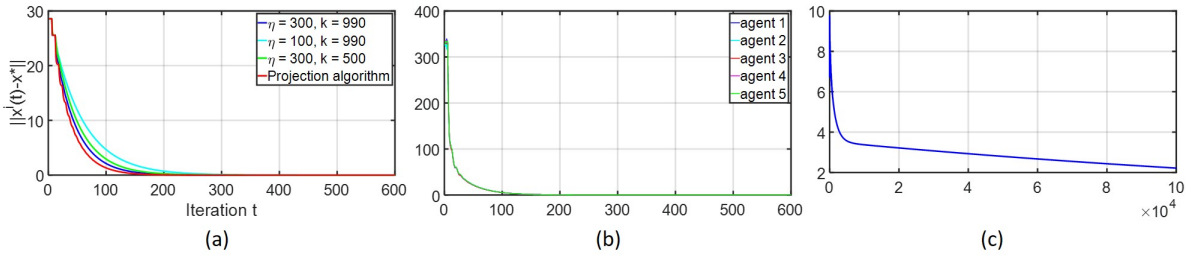

Figure 3.3: $\left\|x^i(t) - x^*\right\|$ under (a) Algorithm 4 and projection-based algorithm [51], (b) Algorithm 4, (c) PI consensus algorithm [72]. (a) $x^i(0) = [0, \ldots, 0]^T \ \forall i$ in each of the cases. (b) $\eta = 300, k = 990$, $x^i(0)$ are randomly initialized within $[-20, 20]$. (c) $x^i(0) = [3, \ldots, 3]^T \ \forall i$.

solution x^* irrespective of their initial choice of estimate (ref. Fig. 3.3) and convergence rate can be changed by tuning the algorithm parameters (ref. Fig. 3.3a). We compare this result with that of two other algorithms in [72] and [51]. The algorithm in [72] has slower convergence rate (ref. Fig. 3.3c) due of ill-conditioning, and the projection-based algorithm in [51] is comparable with the proposed Algorithm 4 (ref. Fig. 3.3a).

3.9 Summary

We have proposed a locally pre-conditioned decentralized gradient-descent algorithm for solving systems of linear algebraic equations over peer-to-peer networks. First, we have assumed the communication graph is synchronous. We have presented explicit linear convergence rate of the proposed algorithm and characterized its robustness against process noise. The theoretical analyses have been supported through experiments by applying the proposed algorithm to develop a linear state predictor. Our results show that the proposed algorithm requires fewer iterations to reach a satisfactory neighborhood of the true solution than the existing decentralized methods except for the decentralized Kalman filter. However, the decentralized Kalman filter has a larger computational and communication cost. Future works include extending the proposed pre-conditioning

scheme to time-varying networks and when the equations have no exact solution.

Additionally, we have proposed a continuous time algorithm for solving the systems of linear algebraic equations when there are delays in the communication links between the agents. In this case also, our algorithm utilizes local pre-conditioners in conjunction with the classical decentralized gradient method. Global convergence and consensus of the proposed algorithm have been proved, assuming undirected network and constant delay. To illustrate the effectiveness of the proposed algorithm, we have applied it to solving a numerical example. The results show that the rate of convergence can be controlled by tuning the algorithm parameters, although convergence rate has not been analyzed. From simulations, we have seen that the proposed algorithm's convergence speed is either favorable or comparable to existing prominent methods.

Chapter 4: Distributed Convex Optimization in Server-Agent Network

4.1 Introduction

In this chapter, we consider solving multi-agent distributed convex optimization problems. Precisely, we consider m agents in the system, as shown in Fig. 2.1. The agents can only interact with a common server and the overall system is assumed to be synchronous. Each agent $i \in \{1, \ldots, m\}$ has a set of n^i number of local data points and a *local cost function* $f^i : \mathbb{R}^d \to \mathbb{R}$ associated with its local data points. The aim of the agents is to compute a minimum point of the *aggregate cost function* $f : \mathbb{R}^d \to \mathbb{R}$, taking values $f(x) = \sum_{i=1}^{m} f^i(x)$ for each $x \in \mathbb{R}^d$, across all the agents in the system. Formally, the goal of the agents is to *distributively* compute a common parameter vector $x^* \in \mathbb{R}^d$ such that

$$x^* \in X^* = \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} f^i(x). \tag{4.1}$$

Since each agent only partially knows the collective data points, they collaborate with the server for solving the distributed problem (4.1). However, the agents cannot share their local data points to the server. An algorithm that enables the agents to jointly solve the above problem in the aforementioned settings is defined as a *distributed algorithm*.

In principle, the optimization problem (4.1) can be solved using the distributed gradient-

descent (GD) algorithm [2]. Built upon the prototypical gradient-descent method, several *adaptive gradient algorithms* have been proposed, which distributively solve (4.1) [22–24, 33–35, 38]. Amongst them, some notable distributed algorithms are Nesterov's accelerated gradient-descent (NAG) [23], heavy-ball method (HBM) [24], and Adabelief [81]. All these methods are iterative in nature, wherein the server maintains an estimate of a solution defined in (4.1) and iteratively refines it using the sum of *local gradients* computed by the agents. The per-iteration computational cost of these algorithms at each agent is $\mathcal{O}(n^i d)$. The above momentum-based methods improve upon the convergence rate of GD. In particular, the recent Adabelief method has been demonstrated to compare favorably for machine learning problems [81]. If the aggregate cost function f is convex, these algorithms converge at a *sublinear* rate [82, 83]. For the special case of strongly convex cost f, the aforementioned methods converge *linearly* [21, 82, 83].

Newton's method [22] explores second-order information of f. Specifically, when f is strongly convex, Newton's method pre-multiplies the aggregate gradient with the inverse Hessian matrix of f at every iteration, resulting in local *quadratic* convergence rate [22]. Despite of faster convergence rate, there are several issues in Newton's method. (i) For empirical risk minimization, the per-iteration computational cost of Newton's is $\mathcal{O}(n^i d^2)$ at each agent i and $\mathcal{O}(d^3)$ at the server. (ii) Secondly, convergence of Newton's method is guaranteed only if f is strongly convex and the convergence is local. (iii) Additionally, it involves computing a matrix inverse at every iteration, which is highly unstable against *process noise*, such as hardware failures and quantization errors. Here, we propose a novel algorithm, addressing the lack of robustness against noise of Newton's method.

There is a vast literature of Newton-like methods which address the aforementioned first and second issues [22, 84, 85]. For least-squares problems, the Gauss-Newton (GN) and the

damped Levenberg–Marquardt (LM) methods have been proposed, which discard the second order terms in Hessian. The GN method converges locally to a minima at *quadratic* rate if the residual at the minimum point is zero, i.e., $f(x^*) = 0$, and the *Jacobian* matrix is invertible [22]. The damped LM methods converges globally to a minima at a local quadratic rate if $f(x^*) = 0$ and the Jacobian is bounded [22]. However, $f(x^*) = 0$ is not the general case. Moreover, the per-iteration cost of damped LM method is the same as Newton during the initial guess of damping parameter, and $\mathcal{O}(n^i d)$ at each agent and $\mathcal{O}(d^3)$ at the server during the updated damping parameter phase [84]. Cost-efficient variants of the classical LM method have been proposed in [84,85], but without convergence guarantee. The Armijo-Newton method is globally convergent with a local quadratic rate, if the cost is strongly convex [86]. Although, our proposed algorithm is guaranteed to converge locally, $f(x^*)$ need not be zero.

Unlike the aforementioned variations of Newton's method, the quasi-Newton methods approximate the full Hessian matrix [22]. The most popular quasi-Newton method is the BFGS algorithm. The per-iteration computational cost of the BFGS method is $\mathcal{O}(n^i d)$ at each agent i and $\mathcal{O}(d^2)$ at the server. If the Hessian at a minimum point x_* is positive definite, then the BFGS method locally converges to x^* at a *superlinear* rate [22]. The Armijo-BFGS method is globally convergent at a *superlinear* rate if the cost is strongly convex [22]. Another class of algorithms are the nonlinear conjugate gradient (CG) methods [87, 88]. Their per-iteration computational cost is same as GD or BFGS, depending on the particular update parameter. The Polyak-Ribière CG method with periodic resetting converges globally with d-step quadratic rate if f is strongly convex [86]. If f is convex and the Hessian at a minimum point is positive definite, then the above convergence result holds locally. However, these algorithms also suffer from poor robustness against noise.

The aforementioned works on the formal convergence of Newton-like methods only consider a Utopian setting wherein the system is free from noise. The algorithms in [89–95] consider a mini-batch of data points at every iteration. However, *process noise* is generic and not limited to any particular source. Specifically, *process noise* models the uncertainties in the computation due to hardware failures, quantization errors, adversarial perturbations, or finite difference approximations [28, 96]. While the literature of quasi-Newton methods is vast, only a few of them have addressed robustness against process noise [97–101]. Sufficient conditions for local quadratic convergence of Newton's method executed on finite precision machines and local linear convergence of a class of quasi-Newton method satisfying a certain deterioration condition have been presented in [97] and [98]. The analysis of BFGS in [99] assumes progressively diminishing noise. [100] and the most recent work [101] have proposed noise-tolerant variations of the BFGS method, assuming that an estimate of the noise is known. In comparison, our proposed algorithm is the same in deterministic and in noisy settings. Our algorithm does not require any noise estimate.

We propose an *adaptive gradient algorithm* for improving the convergence rate of the distributed gradient-descent method when solving the convex optimization problem (4.1). The key concept in our proposed method is *iterative pre-conditioning*. The idea of *iterative pre-conditioning* has been proposed in Chapter 2, wherein the server updates the estimate using the sum of the agents' gradient multiplied with a suitable iterative pre-conditioning matrix. However, Chapter 2 considers only quadratic cost functions. Note that the iterative pre-conditioning in Chapter 2 does not trivially extend to general cost functions due to non-linearity in the gradients. The proposed algorithm in this chapter rigorously extends that idea of iterative pre-conditioning to general convex optimization problems (4.1). Using real-world datasets, we empirically show that the proposed algorithm converges in fewer iterations compared to the aforementioned methods

Table 4.1: Comparison between convergence rate and per-iteration computational complexity of different algorithms, for solving distributed convex optimization problems. t is the number of iterations.

Algorithm	Strongly Convex		Convex		Per-iteration complexity	
	global	local	global	local	agent i	server
IPG (Proposed)	X	superlinear	X	linear	$\mathcal{O}(n^i d^2 + d^3)$	$\mathcal{O}(d^2)$
NAG, HBM	linear	linear	$\mathcal{O}(\frac{1}{t^2})$	$\mathcal{O}(\frac{1}{t^2})$	$\mathcal{O}(n^i d)$	$\mathcal{O}(d)$
Armijo-Newton	unknown	quadratic	X	X	$\mathcal{O}(n^i d^2)$	$\mathcal{O}(d^3)$
Damped LM	unknown	quadratic[b]	unknown	quadratic[b]	$\mathcal{O}(n^i d^2)$	$\mathcal{O}(d^3)$
Armijo-BFGS	unknown	superlinear	X	X	$\mathcal{O}(n^i d)$	$\mathcal{O}(d^2)$
CG	unknown	d-step quadratic	X	d-step quadratic	same as GD or BFGS	

[a] X indicates "need not converge"
[b] if squared cost, $f(x^*) = 0$, Jacobian uniformly bounded, and Jacobian non-singular at x^*

for solving the distributed convex optimization problem (4.1). Besides empirical results, we also present a formal analysis of the proposed algorithm's convergence.

Several research works have used quadratic models for approximating the loss functions of neural networks [102–106]. Quadratic model-based analyses of neural networks have produced vital insights, including learning rate scheduling [103], and the Bayesian viewpoint of SGD with fixed stepsize [107]. The noisy quadratic model (NQM) with carefully chosen model parameters is a proxy for neural network training. The NQM parameters proposed in [108] make predictions that are aligned with deep neural networks in realistic experimental settings. The results in [108] are supported by further evidence from [109], which rigorously show that quadratic loss function model governs infinite width neural network of arbitrary depth. The theoretical NQM in [105] correctly captures the short-horizon bias of learning rates in neural network training. Thus, although the general optimization model of neural networks is non-

convex, noisy convex quadratic models such as [108] trim away inessentials features while capturing the key aspects of real neural network training, including generalization performance [106] and the effects of pre-conditioning on gradients. This motivates us to implement our proposed iterative pre-conditioning scheme on the noisy quadratic model [108] of neural networks and empirically validating it for solving general non-convex optimization problems, including neural network training.

4.1.1 Summary of our contributions

Our key contributions can be summarized as follows.

- We formally show that our algorithm converges locally at *linear* rate for a general class of convex cost functions when the Hessian is non-singular at a solution of (4.1). In the special case when the solution of (4.1) is unique, the convergence of our algorithm is locally *superlinear*. Formal details are presented in Theorem 4.1 and Theorem 4.2 in Section 4.2.2.

- We rigorously show that the local convergence rate of our algorithm compares favorably to prominent distributed algorithms, namely the GD, NAG, HBM, and AdaBelief methods. So, compared to the first order methods for general convex problems, the total number of computations required to reach an *optimality gap* of ϵ is eventually smaller for the proposed algorithm, i.e., for small-enough $\epsilon > 0$. This improvement is further accentuated when the dimension d is small.

  - When the solution (4.1) is unique, our algorithm converges locally at *superlinear* rate. The convergence of GD, NAG, HBM, and AdaBelief, on the other hand, is only *linear* [21, 82, 83].

- Moreover, in the general case, our algorithm converges locally at *linear* rate. On the other hand, the convergence of GD, NAG, HBM, and AdaBelief is *sublinear* [21, 82, 83].

- Though numerical experiments on real-world data analysis problems, we demonstrate the improved computation time and robustness of our algorithm.

  - The noisy quadratic model in [108] has been claimed to emulate neural network training. Our empirical study shows that the proposed algorithm converges faster than the aforementioned distributed methods on this model, thereby demonstrating the proposed algorithm's efficiency for distributed solution of non-convex optimization problems. Please refer to Section 4.3.1 for further details.

  - Our empirical results for distributed binary logistic regression problem on the "MNIST" datasets validate the proposed algorithm's superior convergence rate and comparable test accuracy, under the influence of process noise. Please refer to Section 4.3.2 for further details.

4.2 Proposed algorithm: Iteratively Pre-conditioned Gradient-descent (IPG)

In this section, we present our algorithm. Our algorithm is an extension of the IPG algorithm proposed in Chapter 2 for linear regression problems, as described below. The algorithm is iterative when in each iteration $t \in \{0, 1, \ldots\}$, the server maintains an estimate $x(t)$ of a minimum point (4.1), and a pre-conditioner matrix $K(t) \in \mathbb{R}^{d \times d}$. Both the estimate and the pre-conditioner matrix are updated using steps presented below.

Initialization: Before starting the iterative process, the server chooses an initial estimate $x(0)$ and a pre-conditioner matrix $K(0)$ from $\mathbb{R}^d$ and $\mathbb{R}^{d\times d}$, respectively. Further, the server chooses a sequence of non-negative scalar parameters $\{\alpha(t), t \geq 0\}$ and two non-negative scalar real-valued parameters $\delta,\ \beta$. The server broadcasts parameter β to all the agents. The specific values of these parameters are presented later in Section 4.2.2.

4.2.1 Steps in each iteration t

In each iteration t, the algorithm comprises four steps that are executed collaboratively by the server and the agents.

- *Step 1*: The server sends the estimate $x(t)$ and the matrix $K(t)$ to each agent $i \in \{1, \ldots, m\}$.
- *Step 2*: Each agent $i \in \{1, \ldots, m\}$ computes the gradient $g^i(t)$ of its local cost function, defined as

$$g^i(t) = \nabla f^i(x(t)). \tag{4.2}$$

Let I denote the $(d \times d)$-dimensional identity matrix. Let e_j and $k_j(t)$ denote the j-th columns of matrices I and $K(t)$, respectively, so that $K(t) = \Big[k_1(t), \ldots, k_d(t)\Big]$.
In the same step, each agent i computes a set of vectors $\Big\{R_j^i(t) :\ j = 1, \ldots,\ d\Big\}$ such that for each j,

$$R_j^i(t) = \left(\nabla^2 f^i(x(t)) + \left(\frac{\beta}{m}\right) I\right) k_j(t) - \left(\frac{1}{m}\right) e_j. \tag{4.3}$$

- *Step 3*: Each agent i sends gradient $g^i(t)$ and the set of vectors $\left\{R_j^i(t),\ j=1,\ldots,\ d\right\}$ to the server.

- *Step 4*: The server updates the estimate $x(t)$ to $x(t+1)$ such that

$$x(t+1) = x(t) - \delta K(t) \sum_{i=1}^{m} g^i(t). \tag{4.4}$$

The server updates the pre-conditioner matrix $K(t)$ to $K(t+1)$ such that

$$k_j(t+1) = k_j(t) - \alpha(t) \sum_{i=1}^{m} R_j^i(t), \quad j = 1, ..., d. \tag{4.5}$$

Next, we formally analyze convergence of the proposed IPG algorithm.

4.2.2 Convergence guarantees

We make the following assumptions for our theoretical results. Recall that f denotes the aggregate cost function, i.e., $f = \sum_{i=1}^{m} f^i$, and x^* denotes a minimum point of f, defined in (4.1).

Assumption 4.1. *Assume that the minimum of function f exists and is finite, i.e., $\left|\min_{x\in\mathbb{R}^d} f(x)\right| < \infty$.*

Assumption 4.2. *Assume that each local cost function f^i is convex and twice continuously differentiable over a convex domain $\mathcal{D} \subseteq \mathbb{R}^d$ containing the set of minimum points X^*, defined in (4.1).*

Assumption 4.3. *Assume that the gradient ∇f is Lipschitz continuous over the domain $\mathcal{D}$ with*

respect to the 2*-norm with Lipschitz constant* l*. Specifically, for any* $x, y \in \mathcal{D}$*,*

$$\left\|\nabla f(x) - \nabla f(y)\right\| \leq l\|x - y\|\,. \tag{4.6}$$

Assumption 4.4. *Assume that the Hessian* $\nabla^2 f$ *is Lipschitz continuous over the domain* $\mathcal{D}$ *with respect to the* 2*-norm with Lipschitz constant* γ*. Specifically, for any* $x, y \in \mathcal{D}$*,*

$$\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \leq \gamma\|x - y\|\,.$$

Assumption 4.5. *Assume that the Hessian* $\nabla^2 f$ *is non-singular at any minimum point* $x^* \in X^*$*.*

Notation: To formally state our convergence result we introduce some notation below.

- For $\beta > 0$, we define

$$K^* = \left(\nabla^2 f(x^*) + \beta I\right)^{-1}.$$

  Under Assumption 4.2; the function f is convex, thus $\left(\nabla^2 f(x^*) + \beta I\right)$ is positive definite when $\beta > 0$, and hence K^* is well-defined.

- We let η denote the induced 2-norm of K^*, i.e., $\eta = \|K^*\|$.

- For each iteration $t \geq 0$, we define

$$\rho(t) = \left\|I - \alpha(t)\left(\nabla^2 f(x(t)) + \beta I\right)\right\|.$$

- We let $\lambda_{max}[\cdot]$ and $\lambda_{min}[\cdot]$, respectively denote the largest and smallest eigenvalue of a matrix.

Lemma 4.1 below states a preliminary result about the convergence of the sequence of pre-conditioner matrices $\{K(t), t \geq 0\}$ to K^*. This lemma is important for our key results presented afterward.

Lemma 4.1. *Consider the IPG algorithm with parameters $\beta > 0$ and $\alpha(t)$ subject to*

$$0 < \alpha(t) < \frac{1}{\lambda_{max}\left[\nabla^2 f(x(t))\right] + \beta}, \quad \forall t \geq 0. \tag{4.7}$$

Then, under Assumptions 4.1 and 4.2, $\rho(t) \in [0,\ 1)$ for all $t \geq 0$.

Note that under the conditions assumed in Lemma 4.1,

$$\rho := \sup_{t \geq 0} \rho(t) \text{ exists, and is less than } 1.$$

We now present below in Theorem 4.1 a key convergence result of our proposed IPG method. Recall that x^* denotes a minimum point of f, defined in (4.1).

Theorem 4.1. *Suppose that Assumptions 4.1-4.5 hold true. Consider the IPG algorithm with parameters $\beta > 0$, $\delta = 1$ and $\alpha(t)$ satisfying (4.7) for all $t \geq 0$. Let the parameter β, the initial estimate $x(0) \in \mathcal{D}$ and pre-conditioner matrix $K(0)$ be chosen such that*

$$\frac{\eta\gamma}{2}\left\|x(0) - x^*\right\| + l\left\|K(0) - K^*\right\| + \eta\beta \leq \frac{1}{2\mu} \tag{4.8}$$

where $\mu \in \left(1,\ \frac{1}{\rho}\right)$ and $\eta = \|K^*\|$. If for $t \geq 0$,

$$\alpha(t) < \min\left\{\frac{1}{\lambda_{max}\left[\nabla^2 f(x(t))\right] + \beta},\ \frac{\mu^t(1-\mu\rho)}{2l(1-(\mu\rho)^{t+1})}\right\}, \tag{4.9}$$

then we obtain that, for all $t \geq 0$*,*

$$\left\|x(t+1) - x^*\right\| \leq \frac{1}{\mu}\left\|x(t) - x^*\right\|. \tag{4.10}$$

Since $\mu > 1$, Theorem 4.1 implies that the sequence of estimates $\{x(t), t \geq 0\}$ locally converges to a solution $x^* \in X^*$ with a *linear* convergence rate $\frac{1}{\mu}$. To obtain a simpler condition on $\alpha(t)$, compared to (4.9), we can use a more conservative upper bound. Specifically, let Λ be an upper bound of $\lambda_{max}\left[\nabla^2 f(x(t))\right]$ that holds true for each iteration t. From condition (4.8) in Theorem 4.1, we have $0 < \mu\rho < 1$ and $\mu > 1$. Thus,

$$\frac{\mu^t(1-\mu\rho)}{2l(1-(\mu\rho)^{t+1})} > \frac{\mu^t(1-\mu\rho)}{2l}.$$

Ultimately, from above we infer that if

$$\alpha(t) < \min\left\{\frac{1}{\Lambda+\beta}, \frac{\mu^t(1-\mu\rho)}{2l}\right\}, \quad \forall t \geq 0,$$

then condition (4.9) is implied, and Theorem 4.1 holds true. Since $\mu > 1$, there exists $T < \infty$ such that for all $t \geq T$ the above condition is equivalent to the simpler condition

$$\alpha(t) < \frac{1}{\Gamma+\beta}, \quad \forall t \geq T.$$

The case of strong convexity: We further show that our algorithm attains *superlinear* convergence when the aggregate cost function f is *strongly convex* (however, the local costs may only be convex), as stated in the assumption below.

Assumption 4.6. *Assume that the aggregate cost function f is strongly convex over the domain $\mathcal{D}$.*

In this case, the Hessian $\nabla^2 f$ is positive definite over the entire domain $\mathcal{D}$. Thus, solution to problem (4.1) is unique, which we denote by x^*. Under Assumption 4.6, we show that IPG method with parameter $\beta = 0$ converges *superlinearly*.

Theorem 4.2. *Consider the IPG algorithm presented in Section 4.2.1, with parameters $\beta = 0$, $\delta = 1$, and $\alpha(t)$ for each iteration $t \geq 0$. If Assumptions 4.1-4.4, Assumption 4.6, and the conditions (4.8)-(4.9) are satisfied, then the following statement holds true:*

$$\lim_{t\to\infty} \frac{\left\|x(t+1) - x^*\right\|}{\left\|x(t) - x^*\right\|} = 0. \tag{4.11}$$

The case of non-convexity: Finally, we consider a general class of non-convex cost functions when the Hessian is non-singular at a solution of (4.1) and make the following assumption.

Assumption 4.7. *Assume that the smallest eigenvalue of the Hessian $\nabla^2 f$ is uniformly bounded below over domain $\mathcal{D}$, i.e., $\lambda_{min}\left[\nabla^2 f(x)\right] > \underline{\gamma}$ for any $x \in \mathcal{D}$. Moreover, $\lambda_{min}\left[\nabla^2 f(x^*)\right] > 0$.*

Assumption 4.7 implies that, for $\beta > \left|\underline{\gamma}\right|$, $\left(\nabla^2 f(x) + \beta I\right)$ is positive definite for $x \in \mathcal{D}$. Also, $\lambda_{min}\left[\nabla^2 f(x^*)\right] > 0$ in Assumption 4.7 implies that $\eta\beta < 1$. So, under Assumption 4.7, with the algorithm parameter $\beta > \left|\underline{\gamma}\right|$, the result in Theorem 4.1 still holds for a class of non-convex aggregate cost.

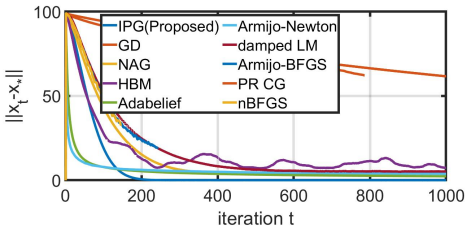

Figure 4.1: $\left\|x(t) - x^*\right\|$ for different algorithms on the noisy quadratic model.

4.3 Experimental results

This section presents our results on two experiments, validating the convergence of our proposed algorithm on real-world problems and its comparison with related methods, namely the distributed versions of GD [2], NAG [23], HBM [24], Adabelief [34], Armijo-Newton [86], damped LM [22], Armijo-BFGS [22], Polyak-Ribière CG (PR CG) [87], and the recent noise-tolerant BFGS (nBFGS) [101], on two different experiments.

4.3.1 Distributed noisy quadratic model

In the first experiment, we implement our proposed algorithm for distributively solving (4.1) in the case of the noisy quadratic model (NQM) [108], which has been shown to agree with the results of training real neural networks.

The noisy quadratic model of neural networks consists of a quadratic aggregate cost function $f(x) = \frac{1}{2}x^T Hx,\ x \in \mathbb{R}^d$ where H is a $(d \times d)$-dimensional diagonal matrix whose i-th element in the diagonal is $\frac{1}{i}$, and each gradient query is corrupted with independent and identically distributed noise of zero mean and the diagonal covariance matrix H. In the experiments, we choose $d = 10^4$, which is also the condition number of the Hessian H. The distributed implementation of the aforementioned noisy quadratic model is set up as follows. The data

points represented by the rows of the matrix H are divided amongst $m = 10$ agents. Thus, each agent $1, \ldots, 10$ has a data matrix of dimension $10^3 \times 10^4$. Since the Hessian matrix H in the noisy quadratic model is positive definite, the optimization problem (4.1) has a unique solution $x^* = 0_d$.

The parameters of the respective distributed algorithms are selected such that each of these methods converges in a fewer number of iterations. Specifically, these parameters are selected as described below. For the GD, NAG, and HBM methods, the definition of optimal parameters can be found in [25]. Since the Hessian of the noisy quadratic model is positive definite, Assumption 4.6 holds. So, we set the parameter $\beta = 0$ for the IPG method. The optimal convergence rate of the linear version of the proposed IPG method is obtained when $\alpha = \frac{2}{\lambda_1+\lambda_d}$ [44]. Here, $\lambda_1 = 1$ and $\lambda_d = \frac{1}{d}$ respectively denote the largest and the smallest eigenvalue of H. We find that the IPG method implemented for the NQM converges fastest when the parameter α is set similarly as $\alpha(t) = \frac{2}{\lambda_1+\lambda_d}$. The step-size parameter $\alpha(t)$ of AdaBelief is selected from the set $\{c, \frac{c}{\sqrt{t}}, \frac{c}{t}\}$ where c is from the set $\{0.01, 0.05, 0.1, 0.5, 1, 2\}$. The other parameters of Adabelief are set at their usual values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The step-size for Armijo-Newton and Armijo-BFGS is either obtained following the backtrack routine or chosen from the set $\{ce{-}p : c = 1, 2, 5, p = 2, 3, 4, 5\}$. The best parameter combinations from above are reported in Table 4.2.

The initial estimate $x(0)$ for these algorithms is randomly drawn from the standard normal distribution. The initial pre-conditioner matrix $K(0)$ for the IPG algorithm is the zero matrix of dimension $(d \times d)$. The initial approximation of the Hessian matrix $B(0)$ for the Armijo-BFGS and nBFGS algorithm is the identity matrix of dimension $(d \times d)$.

We compare the number of iterations and total floating point operations ($flops$) needed by

these algorithms to reach an *estimation error* $\left\|x(t) - x^*\right\| = 0.05$. Each iterative algorithm is run until it does not exceed this specified tolerance over 15 consecutive iterations. The results are recorded in Table 4.3 and Figure 4.1. We observe that the proposed IPG algorithm converges faster than the other algorithms.

4.3.2 Distributed logistic regression

In the second experiment, we implement the proposed algorithm for distributively solving the logistic regression problem on the "MNIST" [47] dataset.

From the training examples of the "MNIST" dataset, we select 10^4 arbitrary instances labeled as either the digit one or the digit five. For each instance, we calculate two quantities, namely the average intensity and the average symmetry of the image [48]. Let the column vectors a_1 and a_2 respectively denote the average intensity and the average symmetry of those 10^4 instances. These two features are then mapped to a second-order polynomial space. Then, our input data matrix before pre-processing is $\begin{bmatrix} a_1 & a_2 & a_1.^2 & a_1.*a_2 & a_2.^2 \end{bmatrix}$. Here, $(.*)$ represents element-wise multiplication and $(.^2)$ represents element-wise squares. This raw data matrix is then pre-processed as follows. Each column is shifted by the mean value of the corresponding column and then divided by the standard deviation of that column. Finally, a 10^4-dimensional column vector of unity is appended to this pre-processed matrix. This is our final input matrix A of dimension $(10^4 \times 6)$. The collective data points (A, B) are then distributed among $m = 10$ agents, in the

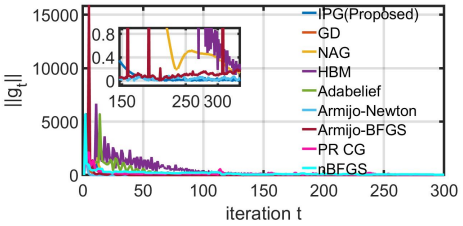

Figure 4.2: $\left\|g(t)\right\|$ for different algorithms on MNIST.

manner already described in Section 4.3.1.

Table 4.2: The parameters used in different algorithms on distributed convex optimization experiments.

Algorithm	Parameters		Algorithm	Parameters	
	NQM	MNIST		NQM	MNIST
IPG $(\alpha(t), \delta, \beta)$	$(1.99, 1, 0)$	$(5e{-}4, 1, 0)$	GD (α)	(1.99)	$(5e-4)$
NAG (α, β)	$(1.33, 0.97)$	$(5e-4, 0.97)$	HBM (α, β)	$(3.92, 0.96)$	$(1e-3, 0.94)$
Adabelief $(\alpha(t), \beta_1, \beta_2, \epsilon)$	$(1/t, 0.9.0.999, 1e-8)$		Armijo-BFGS (α_t)	backtrack [22]	$(5e-2)$

Table 4.3: Comparisons between the number of iterations and total floating point operations ($flops$) required by different algorithms to attain estimation error $\|x(t) - x^*\| = 0.05$ for the NQM.

Algorithm	iterations	total $flops/(nd)$	Algorithm	iterations	total $flops/(nd)$
IPG	$3e2$	$3e6$	GD	$> 5e6$	$> 5e6$
NAG	$> 5e6$	$> 5e6$	HBM	$> 5e6$	$> 5e6$
Adabelief	$> 5e6$	$> 5e6$	Armijo-Newton	$> 3e6$	$> 3e10$
damped LM	$> 1e4$	$> 1e8$	Armijo-BFGS	unstable after 245	N/A
PR CG	unstable after 787	N/A	nBFGS	unstable after 197	N/A

For the MNIST dataset, the parameter α_t in IPG, GD, NAG, and HBM is selected from $\{ce{-}3, ce{-}4 : c = 1, 2, 5\}$. In IPG, the other parameters δ is chosen from $\{1, 0.1, 0.05\}$ and β from $\{0, 0.1, 1\}$. For NAG and HBM, β is from $\{0.91, 0.92, \ldots, 0.99\}$. The parameters of

Table 4.4: Comparisons between the number of iterations and total floating point operations ($flops$) required by different algorithms to attain $\left\|g(t)\right\| = 0.02$ for binary classification on MNIST data, subject to process noise.

Algorithm	iterations	total $flops/(nd)$	Algorithm	iterations	total $flops/(nd)$
IPG	$3e2$	$2e3$	GD	$> 5e3$	$> 5e3$
NAG	$> 5e3$	$> 5e3$	HBM	$> 5e3$	$> 5e3$
Adabelief	$> 5e3$	$> 5e3$	Armijo-Newton	$> 1e3$	$> 6e3$
damped LM	N/A	N/A	Armijo-BFGS	$> 5e3$	$> 5e3$
PR CG	$> 5e3$	$> 5e3$	nBFGS	$> 5e3$	$> 5e3$

Adabelief. Armijo-Newton, and Armijo-BFGS are set following the steps mentioned in Section 4.3.1. at their usual values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial estimate x_0 for these algorithms is randomly drawn from the standard *normal* distribution. The initial pre-conditioner matrix K_0 for IPG is O_d. The initial approximation of the Hessian matrix for Armijo-BFGS and nBFGS is I_d. We note that the damped LM algorithm is not applicable to logistic regression problems. The algorithms are initialized as described in Section 4.3.1. The best parameter combinations, for which the respective algorithms converge in a fewer number of iterations, are then reported for each dataset in Table 4.2.

We compare the performance of the proposed IPG method with other distributed algorithms for solving the logistic regression problem on the "MNIST" and "CIFAR-10" datasets. The algorithm parameters are selected such that each of these methods converges in a fewer number of iterations. Specifically, the parameter selection is described below. The best parameter combinations, for which the respective algorithms converge in a fewer number of iterations, are then reported

for each dataset in Table 4.2. The algorithms are initialized as described in Section 4.3.1.

We add random process noise to each iterated variable of the respective algorithms. Specifically, the additive noise is generated from *normal* distribution with zero mean and $5e-3$ standard deviation. We compare the number of iterations and total floating point operations ($flops$) needed by these algorithms to reach the norm of the gradient $\|g_t\| = 0.02$. Each iterative algorithm is run until it does not exceed this specified tolerance over 15 consecutive iterations. The results are recorded in Table 4.4 and Figure 4.2. We observe that the proposed IPG algorithm converges faster than the other algorithms.

4.4 Summary and future work

We have extended the idea of iterative pre-conditioning for solving convex optimization in distributed server-agent network architecture. We have presented some preliminary theoretical results on the local convergence of the proposed IPG algorithm. Our theoretical results have been derived for the full-batch settings, convex cost functions, and a class of non-convex cost. Through experiments we have also shown the efficacy of our algorithm in a binary classification task and in training of non-convex neural network via a noisy quadratic model.

There is a significant amount of future research scope on the IPG algorithm. First, the IPG algorithm's global convergence rate will be rigorously analyzed. There are a few more classes of other distributed methods which solves the problem (4.1). One such notable class is stochastic quasi-Newton methods, which include recent optimization methods [22, 84, 85, 89–94, 94, 95, 110]. Detailed theoretical comparison with the aforementioned algorithms will be investigated, in terms of the per-iteration computational complexity and global convergence

rate. The experimental results in Section 4.3 suggests improved robustness of the IPG algorithm than the existing optimizers. Therefore, IPG's robustness against process noise will be formally characterized and compared with the Newton and quasi-Newton methods which have addressed robustness against noise [97–101].

As indicated, the IPG algorithm has the potential to be applicable to non-convex optimization. Specifically, ongoing research includes applying the iterative pre-conditioning technique in distributed beamforming problems and nonlinear observer problem. This direction of research is explored later in this dissertation.

Chapter 5: Non-Convex Optimization

5.1 Introduction

In this chapter, we consider the problem of minimizing a non-convex function. The *objective function* $f : \mathbb{R}^d \to \mathbb{R}_{>0}$ is smooth and, without loss of generality, positive-valued. Formally, the problem is defined as

$$\min_{x \in \mathbb{R}^d} f(x). \tag{5.1}$$

Prominent machine learning models aim to learn a hypothesis parameterized by a vector $x \in \mathbb{R}^d$, when provided with n numbers of input-output training data pairs $\{a_i, b_i\}_{i=1}^n$. The learning task is typically formulated as minimization of the *empirical risk* function $f(x) = \frac{1}{n} \sum_{i=1}^n l(x; a_n, b_n)$, where $l(x; a, b)$ is the *loss function* measuring the discrepancy between the prediction by the learnt model characterized by x on the input data a and the true output b. The most successful machine learning models used by practitioners, such as neural networks, are non-convex. Instead of searching a global optimal solution, a more meaningful and attainable goal for such models is to train the model for finding a *critical point* of the objective function f.

First-order optimization algorithms are widely used for training complex machine learning models. The classical gradient-descent (GD) is the simplest first-order optimization method [2].

Its stochastic counterpart, known as the stochastic gradient-descent (SGD) has achieved tremendous success in training deep neural networks, despite its simplicity [32]. The momentum-based accelerated variants of SGD, such as SGD with momentum [111], Nesterov's accelerated gradient-descent (NAG) [23], are popular algorithms for image classification and language modeling problems [112, 113]. The aforementioned algorithms iteratively update all the model parameters using a global learning rate, and are classified as *accelerated gradient algorithms*.

To achieve faster convergence, several *adaptive gradient algorithms* have been proposed recently. These methods update each of the model parameters with an individual learning rate, resulting in faster training than accelerated gradient methods. AdaGrad [33] is the first prominent adaptive gradient method that significantly improved upon SGD, especially for sparse-gradients problems [114]. We propose a more general AdaGrad algorithm, referred as G-AdaGrad. G-AdaGrad can improve upon the convergence rate of AdaGrad by tuning an additional scalar parameter. We review the AdaGrad algorithm below.

AdaGrad is a prominent optimization method that achieves significant performance gains compared to SGD. As the name suggests, AdaGrad adaptively updates the *learning rate* based on the information of all the previous gradients. Specifically [33], for each iteration $k \in \{0, 1, \ldots\}$, let $x_k = [x_{k,1}, \ldots, x_{k,d}]^T$ denote the estimate of a local minima in (5.1) maintained by AdaGrad. In addition, AdaGrad maintains a set of real valued scalar parameters denoted by $\{b_{k,i} : i = 1, \ldots, d\}$. The algorithm is initialized with arbitrarily chosen initial estimate $x_0 \in \mathbb{R}^d$ and $\{b_{k,i} > 0 : i = 1, \ldots, d\}$. Let the gradient of the *objective function* evaluated at $x \in \mathbb{R}^d$ be denoted as $\nabla f(x) \in \mathbb{R}^d$, and its i-th element in be denoted as $\nabla_i f(x)$ for each dimension $i \in \{1, \ldots, d\}$. At each iteration $k \in \{0, 1, \ldots\}$, each of the parameters $\{b_{k,i} : i = 1, \ldots, d\}$ are updated according to $b^2_{k+1,i} = b^2_{k,i} + \left\|\nabla_i f(x_k)\right\|^2$. At the same iteration k, the estimate is updated to $x_{k+1,i} =$

$x_{k,i} - \eta \frac{\nabla_i f(x_k)}{b_{k+1,i}}$ for each $i \in \{1, \ldots, d\}$. The real valued scalar parameter $\eta > 0$ is called the *step-size*. Thus, the *learning rate* in AdaGrad is adaptively weighted along each dimension by the sum of squares of the past gradients. AdaGrad has been shown to particularly effective for sparse gradients [115], but has under-performed for some applications [116].

The Adam algorithm has been observed to compare favorably with other optimization methods for a wide range of optimization problems, including deep learning [117–119]. Like AdaGrad, Adam also updates the *learning rate* based on the information of past gradients. However, unlike AdaGrad, Adam effectively updates the *learning rate* based on only a moving window of the past gradients. Specifically [34], Adam maintains two sets of d-dimensional vectors, respectively denoted by $\mu_k = [\mu_{k,1}, \ldots, \mu_{k,d}]^T$ and $v_k = [v_{k,1}, \ldots, v_{k,d}]^T$. μ_k and v_k are respectively known as the biased first moment estimate and biased second raw moment estimate. These vectors are initialized with $\mu_0 = 0_d$ and $\{v_{k,i} > 0 : i = 1, \ldots, d\}$. Three parameters $\eta > 0$, $\beta_1 \in [0, 1)$, and $\beta_2 \in [0, 1)$ are chosen before the iterations begin. At each iteration $k \in \{0, 1, \ldots\}$, the vectors μ_k and v_k are updated according to $\mu_{k+1,i} = \beta_1 \mu_{k,i} + (1 - \beta_1) \nabla_i f(x_k)$ and $v_{k+1,i} = \beta_2 v_{k,i} + (1 - \beta_2) \left\| \nabla_i f(x_k) \right\|^2$ along each dimension $i \in \{1, \ldots, d\}$. Next, the estimate x_k is updated to $x_{k+1,i} = x_{k,i} - \eta \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{\mu_{k+1,i}}{\sqrt{v_{k+1,i}}}$ for each $i \in \{1, \ldots, d\}$. The factor $\frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}$ is responsible for the initial bias correction, as proposed in the original Adam algorithm [34]. Thus, the learning rate in Adam is weighted by the exponentially moving averages of the past gradients.

Despite its widespread use, the poor generalization ability of Adam has led to several variants of Adam. The notable ones among them include AdaBelief [81], AdaBound [120], AdamW [121], AMSGrad [114], Fromage [122], MSVAG [123], RAdam [124], Nadam [38] and Yogi [125]. Among these modifications of Adam, AdaBelief has been proposed most recently. Through extensive empirical studies, AdaBelief has been shown to reduce the generalization gap

between SGD and adaptive gradient methods, by improving upon the generalization performance of the aforementioned algorithms including SGD. However, in these experiments, few other existing methods such as Yogi, MSAVG, AdamW, RAdam, and Fromage have achieved faster convergence of the training cost than AdaBelief [81]. We review the AdaBelief algorithm below.

The AdaBelief algorithm is similar to Adam, except that $\left\|\nabla_i f(x(k))\right\|^2$ is replaced by $\left\|\nabla_i f(x(k)) - \mu_i(k)\right\|^2$ so that $\frac{1}{\sqrt{\nu_i(k)}}$ represents the belief in the observed gradient [81]. In other words, the following set of equations represents the AdaBelief algorithm. For each $i \in \{1, \ldots, d\}$ and each iteration $k \in \{0, 1, \ldots\}$,

$$\begin{aligned}
\mu_{k+1,i} &= \beta_1 \mu_{k,i} + (1-\beta_1)\nabla_i f(x_k), \\
v_{k+1,i} &= \beta_2 v_{k,i} + (1-\beta_2)\left\|\nabla_i f(x_k) - \mu_i(k)\right\|^2, \\
x_{k+1,i} &= x_{k,i} - \eta \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{\mu_{k+1,i}}{\sqrt{v_{k+1,i}}}.
\end{aligned}$$

Another adaptive gradient method is AdaBound [120]. AdaBound imposes a dynamic bound on the learning rates, enabling a smooth transition throughout iterations from adaptive gradient to the classical stochastic gradient-descent (SGD) method, which is known to have better generalization ability. We briefly review the AdaBound algorithm below. AdaBound is an iterative algorithm. For each iteration $k \in \{0, 1, \ldots\}$, let $x_k = [x_{k,1}, \ldots, x_{k,d}]^T$ denote the estimate of a local minima in (2.1) maintained by AdaBound. In addition, AdaBound maintains three d-dimensional vectors, respectively denoted by $\mu_k = [\mu_{k,1}, \ldots, \mu_{k,d}]^T$, $\nu_k = [\nu_{k,1}, \ldots, \nu_{k,d}]^T$, and $\overline{\eta}_k = [\overline{\eta}_{k,1}, \ldots, \overline{\eta}_{k,d}]^T$. μ_k and ν_k are respectively known as the biased first moment estimate and biased second raw moment estimate. $\overline{\eta}_k$ is the learning rate, which is bounded below by a predefined non-decreasing function $\eta_l : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ and upper bounded by another non-

increasing function $\eta_u : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{>0}$. Moreover, η_l and η_u asymptotically converge to some positive-valued scalar α^*. We define an operator $Clip\left(x, y, z\right) = \max\left(\min\left(x, z\right), y\right)$, for $x, y, z \in \mathbb{R}$. The vectors are initialized with $\mu_0 = 0_d$ and $\{\nu_{k,i} > 0 : i = 1, \ldots, d\}$. Three parameters $\alpha > 0$, $\beta_1 \in [0, 1)$, and $\beta_2 \in [0, 1)$ are chosen before the iterations begin. At each iteration k, the vectors μ_k, ν_k, and $\overline{\eta}_k$ are updated as $\mu_{k+1,i} = \beta_1 \mu_{k,i} + (1 - \beta_1)\nabla_i f(x_k)$, $\nu_{k+1,i} = \beta_2 \nu_{k,i} + (1 - \beta_2)\left\|\nabla_i f(x_k)\right\|^2$, and $\overline{\eta}_{k+1,i} = Clip\left(\frac{\alpha}{\sqrt{\nu_{k+1,i}}}, \eta_{l,k}, \eta_{u,k}\right)$ along each dimension $i \in \{1, \ldots, d\}$. Next, the estimate x_k is updated to $x_{k+1,i} = x_{k,i} - \frac{1}{\sqrt{k+1}}\overline{\eta}_{k+1,i}\mu_{k+1,i}, \forall i$. In this way, AdaBound restricts the dimension-wise learning rates $\overline{\eta}_{k,i}$ to be within $[\eta_{l,k}, \eta_{u,k}]$. As a result, AdaBound replicates Adam at the initial stage of iterations as the bounds have very little impact on learning rates, and it gradually transforms to SGD as the bounds become progressively restricted.

Recently, the MAdam algorithm [126] has been proposed to improve the stability of adaptive learning. Unlike AdaBound, MAdam updates ν_k based on dynamic values of β_2 so that the estimated variance of gradients is maximized. MAdam has been shown to improve over existing adaptive gradient algorithms on several machine learning experiments. We briefly review the MAdam algorithm below. For each iteration $k \in \{0, 1, \ldots\}$, MAdam maintains four d-dimensional vectors, denoted by $\tilde{\mu}_k$, $\tilde{u}_k$ $\tilde{\nu}_k$, and w_k. The zeroth moment estimate w_k of the gradients is used for the initial bias correction. These vectors are initialized with $\tilde{\mu}_0 = \tilde{u}_0 = w_0 = 0_d$ and $\tilde{\nu}_{k,i} > 0 \; \forall i$. Three parameters $\alpha \in (0, 1)$, $\underline{\beta}$, and $\overline{\beta}$ are chosen such that $0 < \underline{\beta} < \overline{\beta} < 1$. At each iteration k, first $\tilde{\mu}_k$ is updated as $\tilde{\mu}_{k+1,i} = \alpha\tilde{\mu}_{k,i} + (1 - \alpha)\nabla_i f(x_k)$. Next, the dynamic coefficient for updating the moment estimates are computed as $\beta_{k,i} = Clip\left(\tilde{\beta}_{k,i}, \underline{\beta}, \overline{\beta}\right)$, where $\tilde{\beta}_{k,i} = \arg\max_\beta \nu_{k,i}(\beta) - u_{k,i}(\beta)^2$. Here, $u_k = \frac{\tilde{u}_k}{w_k}$ and $\nu_k = \frac{\tilde{\nu}_k}{w_k}$ are the bias corrected estimate of the first and the second moment, respectively. Using the computed $\beta_{k,i}$, the vectors

$\tilde{u}_k$ $\tilde{\nu}_k$, and w_k are updated respectively as $\tilde{u}_{k+1,i} = \beta_{k,i}\tilde{u}_{k,i} + (1-\beta_{k,i})\nabla_i f(x_k)$, $\tilde{\nu}_{k+1,i} = \beta_{k,i}\tilde{\nu}_{k,i} + (1-\beta_{k,i})\left\|\nabla_i f(x_k)\right\|^2$, and $w_{k+1,i} = \beta_{k,i}w_{k,i} + (1-\beta_{k,i})$. Finally, x_k is updated to $x_{k+1,i} = x_{k,i} - \eta_k \frac{\sqrt{w_{k+1,i}}}{1-\alpha^{k+1}} \frac{\tilde{\mu}_{k+1,i}}{\sqrt{\tilde{\nu}_{k+1,i}}}$, where η_k is the step-size. The closed-form expression of $\beta_{k,i}$ can be found in the original paper [126].

Two more variants of Adam are the Nadam [38] and the RAdam [124]. By incorporating Nesterov's momentum into Adam, Nadam enables faster convergence and smaller generalization errors on several problems, including natural language processing (NLP) [127], image analysis [128, 129], and automated path planning [130]. RAdam rectifies the variance of dynamic learning rate in Adam by introducing a constructive *warmup* (learning rate scheduling). RAdam is being applied for training transformers [131], general adversarial networks [132], for NLP [133] and medical imaging tasks [134]. We briefly review the Nadam and RAdam algorithms below.

Nadam and RAdam are iterative methods. For each iteration $k \in \{0, 1, \ldots\}$, let $x_k = [x_{k,1}, \ldots, x_{k,d}]^T$ denote the estimate of a local minima in (2.1) maintained by the algorithms. In addition, they maintain two d-dimensional vectors, respectively denoted by $\mu_k = [\mu_{k,1}, \ldots, \mu_{k,d}]^T$ and $\nu_k = [\nu_{k,1}, \ldots, \nu_{k,d}]^T$. μ_k and ν_k are respectively known as the first moment estimate and biased second raw moment estimate, initialized with $\mu_0 = 0_d$ and $\{\nu_{k,i} > 0 : i = 1, \ldots, d\}$. Three parameters $\eta > 0$ (step-size), $\beta_1 \in (0, 1)$, and $\beta_2 \in (0, 1)$ are chosen before the iterations begin. At each iteration $k \in \{0, 1, \ldots\}$, Nadam updates the iterables as $\mu_{k+1,i} = \beta_1\mu_{k,i} + (1-\beta_1)\nabla_i f(x_k)$, $\nu_{k+1,i} = \beta_2\nu_{k,i} + (1-\beta_2)\left\|\nabla_i f(x_k)\right\|^2$, and $x_{k+1,i} = x_{k,i} - \eta\frac{\beta_1}{\sqrt{\beta_2}}\frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+2}}\frac{\mu_{k+1,i}}{\sqrt{\nu_{k+1,i}}} - \eta\frac{1-\beta_1}{\sqrt{\beta_2}}\frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+1}}\frac{\nabla_i f(x_k)}{\sqrt{\nu_{k+1,i}}}$, for each dimension $i \in \{1, \ldots, d\}$. RAdam updates μ_k and ν_k in exactly the same way as Nadam. The difference lies in updating the estimate x_k. Specifically, let $\rho_\infty = \frac{2}{1-\beta_2} - 1$, $\rho_{k+1} = \rho_\infty - 2(k+1)\frac{\beta_2^{k+1}}{1-\beta_2^{k+1}}$, and $r_{k+1} = \sqrt{\frac{(\rho_{k+1}-4)(\rho_{k+1}-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_{k+1}}}$. When $\rho_{k+1} \leq 4$, x_k

is updated as $x_{k+1,i} = x_{k,i} - \eta \frac{\mu_{k+1,i}}{1-\beta_1^{k+1}}$. Otherwise, when $\rho_{k+1} > 4$, x_k is updated as $x_{k+1,i} = x_{k,i} - \eta r_{k+1} \frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+1}} \frac{\mu_{k+1,i}}{\sqrt{\nu_{k+1,i}}}$.

5.1.1 Related work

We aim to present simplified proofs of convergence of the aforementioned prominent adaptive gradient algorithms to a *critical point* for non-convex *objective functions* in the deterministic settings. The first convergence guarantee of a generalized AdaGrad method for non-convex functions has been presented recently in [135], where the additional parameter $\epsilon \geq 0$ generalizes the AdaGrad method. However, the parameter ϵ in [135] has been assumed to be strictly positive for the convergence guarantee, which excludes the case of the original AdaGrad method [33] where $\epsilon = 0$. We propose a more general AdaGrad model, coined G-AdaGrad, that subsumes the work in [135]. Our model and corresponding convergence proof allow the parameter ϵ to be negative, as well as the case of the original AdaGrad. Besides, our proof provides intuition behind how this generalization of AdaGrad impacts its convergence. The analysis for AdaGrad in [136] assumes the gradients to be uniformly bounded. We do not make such an assumption. Other works also analyze the convergence of AdaGrad-like algorithms for non-convex objective functions, notable among them being WNGrad [137] and AdaGrad-Norm [138]. Note that all of the aforementioned analyses of AdaGrad and AdaGrad-like algorithms are in discrete-time. We analyze AdaGrad in the continuous-time domain.

Previous works that prove convergence of Adam for non-convex problems include [82, 125, 139–142]. In [125], the proof for Adam considers the algorithm parameter $\beta_1 = 0$, which is essentially the RMSProp algorithm. We consider the general parameter settings where $\beta_1 \geq 0$.

An Adam-like algorithm has been proposed and analyzed in [136]. The proofs in [82, 125, 139–141] do not consider the initial bias correction steps in the original Adam [34]. Our analysis of Adam considers the bias correction steps. The analyses in [82, 125, 136, 139, 141] assume uniformly bounded gradients. We do not make such an assumption. The aforementioned analyses of Adam are in discrete-time. A continuous-time model of Adam has been proposed in [142], which includes the bias correction steps. However, compared to the convergence proof in [142], our proof for Adam is simpler. In addition, [142] assumes that the parameters β_1 and β_2 in the Adam algorithm are functions of the *step-size* η such that β_1 and β_2 tends to one as the *step-size* $\eta \to 0$. We do not make such an assumption in our analysis.

The convergence guarantee of the AdaBelief algorithm has been provided by the authors in discrete-time [81]. Here, we present a simpler proof for AdaBelief in continuous-time.

There is substantial literature on analysis of iterative optimization algorithms in discrete-time [143]. However, there has been a recent surge in utilizing tools from control theory for that purpose. Convergence of the classical gradient-descent algorithm and *accelerated gradient algorithms* were characterized in [25] using integral quadratic constraints. Standard control theoretic tools have been used to gain new insights into the aforementioned accelerated gradient methods [144–146]. Differential equation-based modeling of these algorithms in the continuous-time limit has been used for their convergence analysis [83, 147–149]. Appropriate discretization of these continuous-time models has led to the design of new optimization algorithms [83, 147, 148, 150]. Recently, a unified model, refered as *accelerated gradient flow*, for several accelerated proximal gradient methods and momentum-based methods such as heavy-ball and Nesterov's method have been proposed [148]. Driven by this theme, we view adaptive gradient algorithms as dynamical systems, which we use for the purpose of convergence analysis and also for proposing

a novel adaptive gradient algorithm. However, most of the existing research works on continuous-time modeling of optimization methods have only considered *accelerated gradient methods*. Here, we focus specifically on *adaptive gradient methods*. It must be noted that the convergence results of continuous-time optimization algorithms or *gradient flow* not necessarily extend to their discrete-time counterparts. In fact, different discretization schemes of the same continuous-time algorithm can result in different algorithms in discrete-time [148].

5.1.2 Our contributions

First, we propose a more general AdaGrad algorithm, which we refer to as Generalized AdaGrad (G-AdaGrad). The proposed optimizer improves upon the convergence rate of the original AdaGrad algorithm. The original AdaGrad, discussed in Section 5.1, is a special case of the proposed G-AdaGrad algorithm. We propose a state-space models for the G-AdaGrad algorithm in continuous time-domain. The proposed state-space model of G-AdaGrad is an autonomous system of ordinary differential equations. Using a simple analysis of the proposed state-space model, we prove the convergence of the G-AdaGrad algorithm to a *critical point* of the possibly non-convex optimization problem (5.1) in the deterministic settings. Our analysis requires minimal assumptions about the optimization problem (5.1).

Next, we propose a general class of adaptive gradient algorithms for solving non-convex optimization problems. We represent the proposed algorithm in the state-space form. This state-model of adaptive gradient algorithms is a non-autonomous system of ODEs, and includes the aforementioned AdaGrad, G-AdaGrad, Adam, and AdaBelief algorithms as special cases. We formally analyze the proposed adaptive gradient algorithm framework from state-space perspective,

which enables us in presenting simplified convergence proofs of AdaGrad, G-AdaGrad, Adam, and AdaBelief in continuous-time for non-convex optimization problems.

However, this generic framework does not include the AdaBound, Nadam, and RAdam algorithms. We analyze these four algorithms in a state-space model, which shows that the aforementioned intuitive and unifying proof sketch can be applied to other adaptive gradient algorithms.

Convergence of AdaBound for convex objective function has been analyzed in the AdaBound paper [120] and later in [151]. Its convergence in the non-convex case has been recently presented in [152]. Although the monotonicity requirement on the bound functions η_l and η_u has been removed in [152], we assume η_u to be monotonous and a dynamic bound on the derivative of η_l. Unlike the aforementioned discrete-time analyses of AdaBound, our continuous-time analysis has the following advantages. The aforementioned works assume a diminishing step-size sequence of AdaBound, which slows down the convergence rate when closer to a minimum point. Instead, we employ a constant step-size in our AdaBound model. The existing analyses do not include the initial bias correction steps in AdaBound. Consistent with the practical implementation of AdaBound [120] and other adaptive gradient algorithms [34, 81], our analysis considers the bias correction steps. Moreover, instead of specifying the exact expression for bias correction, we use a generic expression satisfying a certain condition which includes the bias correction term used in practice.

To the best of our knowledge, the theoretical convergence guarantee of the aforementioned Nadam and RAdam algorithms is not available. We prove the convergence of these two algorithms for non-convex optimization problems in continuous-time. We note that the Nadam and RAdam algorithms are not included in our generic framework of adaptive gradient methods. Specifically,

the two different bias correction terms $\frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+2}}$ and $\frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+1}}$ prohibits Nadam to be included in this generic framework. Similarly, the additional time-varying factor r_{k+1} and a separate estimate update law in case of $\rho_{k+1} \leq 4$ prohibits RAdam to be included in the generic framework. Recently, an alternate perspective for the warmup of Adam and a few "rule-of-thumb" warmup schedules have been presented in [153]. However, no convergence analysis of RAdam or the proposed warmup schedules has been presented in [153].

Furthermore, we show that our technique for proving convergence of optimizers for non-convex functions is not limited to adaptive gradient methods. Specifically, we present a convergence proof of the rescaled gradient flow [154] for *smooth* non-convex problems by invoking Barbalat's lemma. The fixed-time convergence guarantee of the gradient flow algorithm [154] assumes a unique minimum point of (5.1) and quadratic growth of the objective function f, i.e., $f(x) - f(x^*) \geq \frac{\mu}{2}\|x - x^*\|^2 \, \forall x \in \mathbb{R}^d$ and for some $\mu > 0$. When f is non-convex, it can be easily shown that the rescaled gradient flow [154] is a descent method, i.e., $\frac{df}{dt}(x(t)) < 0$, as in [147]. However, it is a known fact that f being smooth, lower bounded, and decreasing not necessarily imply that $\frac{df}{dt}$ converges to zero in the limit $t \to \infty$. Convergence of a discrete-time variation of the rescaled gradient flow has been analyzed in [155] under the assumption of *strongly smooth* f, a stronger assumption than mere smoothness. In continuous-time, we show that the convergence of gradient $\nabla f(x(t))$ to zero in the rescaled gradient flow can be proved by Barbalat's lemma for *smooth* non-convex problem (5.1).

Finally, motivated by the issues in training neural network models as mentioned above, we propose novel adaptive gradient algorithms for solving (5.1), aimed at balancing improved generalization and faster convergence of machine learning models. Facilitated by our aforementioned

state-space framework for adaptive gradient methods, the proposed algorithms are built on top of Adam, MAdam, and Nadam, respectively. Our proposed algorithms add a specific pole-zero pair to the dynamics of the second raw moment estimate $\tilde{\nu}$ in MAdam, thereby improving its convergence speed and steady-state behavior.

Our key findings are summarized below.

- We propose the Generalized AdaGrad (G-AdaGrad) algorithm, which improves upon the convergence rate of the original AdaGrad. From a state-space perspective, we present a simple yet intuitive convergence analysis of the proposed G-AdaGrad algorithm. Please refer Section 5.2 for the details.

- We develop a state-space framework for a general adaptive gradient algorithms that solves the optimization problem defined in (5.1). Using a simple analysis of the proposed state-space model, we prove the convergence of the proposed adaptive gradient algorithm to a *critical point* of the possibly non-convex optimization problem (5.1) in the deterministic settings. Since our framework is inclusive of the AdaGrad, G-AdaGrad, Adam, and AdaBelief optimizers, an intuitive and simple convergence proof for each of these methods follows as a special case of our analysis. Please refer Section 5.4 for the details.

- Compared to existing works, our continuous-time model of AdaBound in Section 5.5 better resembles the practical implementation of AdaBound and enables a simpler analysis. We model the aforementioned AdaBound algorithm in the state-space framework. Our model of AdaBound is in continuous-time and represented by a set of non-autonomous ordinary differential equations (ODE). We rigorously analyze the proposed state model using tools from adaptive control theory and present a simple convergence proof of AdaBound to a

critical point of the non-convex objective function (5.1) in the deterministic settings.

- We propose two state-space models, each for the Nadam and the RAdam algorithm, in continuous-time. By utilizing Barbalat's lemma [156], we present a simple analysis of the proposed state-space models, proving convergence of Nadam and RAdam to a *critical point* of the non-convex optimization problem (5.1) in the deterministic settings for the first time. Our analysis requires minimal assumptions about the optimization problem (5.1). Further details are in Section 5.6-5.7.

- In Section 5.8, we further extend the applicability of our proof technique, relying on Barbalat's lemma, by proving convergence of the rescaled gradient flow [154] for non-convex problem.

- We propose new variants of Adam, MAdam, and Nadam, which we refer as AdamSSM, MAdamSSM, and NadamSSM, respectively, for solving the general non-convex optimization problem defined in (5.1). The proposed algorithm adds a specific pole-zero pair to the dynamics of the second raw moment estimate ν in Adam, thereby improving its convergence. Utilizing the aforementioned framework, we guarantee convergence of the proposed AdamSSM algorithm to a *critical point* of (5.1). Please refer Section 5.9 for the details.

- Finally, in Section 5.12, we demonstrate the applicability of the proposed AdamSSM, MAdamSSM, and NadamSSM algorithms to benchmark machine learning problems. In this context, we conduct image classification experiments with convolutional neural network (CNN) models on CIFAR10 and CIFAR100 datasets and language modeling experiments with long short-term memory (LSTM) models. These results present empirical evidence of

the proposed algorithms' capability in balancing better generalization and faster convergence than the existing state-of-the-art optimizers.

5.2 Proposed algorithm: Generalized AdaGrad (G-AdaGrad)

In this section, we propose a set of autonomous ordinary differential equations. Using first-order Euler discretization, we show that the proposed set of differential equations coincides with a general version of the AdaGrad algorithm, which we refer to as the Generalized AdaGrad (G-AdaGrad). The proposed differential equations include the original AdaGrad as a special case.

We make the following assumptions in order to present our algorithms and their convergence results.

Assumption 5.1. *Assume that the minimum of function f exists and is finite. In other words,* $\left|\min_{x\in\mathbb{R}^d} f(x)\right| < \infty$.

Assumption 5.2. *Assume that f is twice differentiable over its domain $\mathbb{R}^d$ and the entries in the Hessian matrix $\nabla^2 f(x)$ are bounded above for all $x \in \mathbb{R}^d$.*

5.2.1 Description of G-AdaGrad

We propose the Generalized AdaGrad (G-AdaGrad) method which is parameterized by a positive real scalar c. For each dimension $i \in \{1, \ldots, d\}$ and $t \geq 0$, consider the following pair

of differential equations

$$\dot{x}_{ci}(t) = \left\|\nabla_i f(x(t))\right\|^2, \tag{5.2}$$

$$\dot{x}_i(t) = -\frac{\nabla_i f(x(t))}{\left(x_{ci}(t)\right)^c}, \tag{5.3}$$

with initial conditions $x_c(0) \in \mathbb{R}^d$ and $x(0) \in \mathbb{R}^d$. We assume that the initial condition $\{x_{ci}(0) > 0 : i = 1, \ldots, d\}$. The variable $x_{ci}, \forall i$ can be abstracted as dynamic controller state.

The above pair of differential equations (5.2)-(5.3) can be seen as a continuous-time variation of the following algorithm, when (5.2)-(5.3) are discretized with a fixed sampling time $\delta > 0$. For each $i \in \{1, \ldots, d\}$ and $k \in \{0, 1, \ldots\}$,

$$x_{ci}((k+1)\delta) = x_{ci}(k\delta) + \delta\left\|\nabla_i f(x(k\delta))\right\|^2, \tag{5.4}$$

$$x_i((k+1)\delta) = x_i(k\delta) - \delta\frac{\nabla_i f(x(k\delta))}{\left(x_{ci}(k\delta)\right)^c}. \tag{5.5}$$

This can be seen from the following argument. Since $\delta > 0$, (5.4)-(5.5) can be rewritten as

$$\frac{x_{ci}((k+1)\delta) - x_{ci}(k\delta)}{\delta} = \left\|\nabla_i f(x(k\delta))\right\|^2,$$
$$\frac{x_i((k+1)\delta) - x_i(k\delta)}{\delta} = -\frac{\nabla_i f(x(k\delta))}{\left(x_{ci}(k\delta)\right)^c}.$$

Defining $t = k\delta$, in the limit $\delta \to 0$, the above equations coincide with (5.2)-(5.3). Note that, (5.4)-(5.5) represents the AdaGrad algorithm in discrete-time [33] with *step-size* $\eta = \delta$.

Note that, (5.4)-(5.5) represents a generalization of the AdaGrad algorithm discussed in Section 5.1 with *step-size* $\eta = \delta$ and an additional parameter c. The controller states $x_c(t)$

in continuous-time corresponds to the variable b_k in discrete-time of the AdaGrad algorithm. When we set $c = 0.5$, (5.4)-(5.5) correspond to the original AdaGrad algorithm. Introducing the parameter c can further improve its convergence. This is discussed in the following subsection.

5.2.2 Convergence of G-AdaGrad

Define the set of *critical points* of the *objective function* f as

$$X^* = \{x \in \mathbb{R}^d : \nabla f(x) = 0_d\}. \tag{5.6}$$

Theorem 5.1 below presents a key result on the convergence of the G-AdaGrad algorithm (5.2)-(5.3) in continuous-time to a *critical point* in X^*.

Theorem 5.1. *Consider the pair of differential equations* (5.2)-(5.3) *with initial conditions* $x_c(0) \in \mathbb{R}^d$ *and* $x(0) \in \mathbb{R}^d$ *such that* $\{x_{ci}(0) > 0 : i = 1, \ldots, d\}$. *Let the parameter* $c \in (0, 1)$. *If Assumptions 5.1-5.2 hold, then*

$$\lim_{t\to\infty} \nabla f(x(t)) = 0_d. \tag{5.7}$$

Moreover, for all $t \geq 0$, *we have*

$$f(x(t)) = f(x(0)) + \sum_{i=1}^{d} \frac{\left(x_{ci}(0)\right)^{1-c} - \left(x_{ci}(0) + \int_0^t \left\|\nabla_i f(x(s))\right\|^2 ds\right)^{1-c}}{1-c}. \tag{5.8}$$

Proof. The time-derivative of f along the trajectories $x(t)$ of (5.3) is given by

$$\dot{f}(x(t)) = (\nabla f(x(t))^T \dot{x}(t) = \sum_{i=1}^{d} \nabla_i f(x(t)) \dot{x}_i(t).$$

Substituting (5.3) yields,

$$\dot{f}(x(t)) = -\sum_{i=1}^{d} \frac{\left\|\nabla_i f(x(t))\right\|^2}{\left(x_{ci}(t)\right)^c}.$$

Further utilizing (5.2) we get,

$$\dot{f}(x(t)) = -\sum_{i=1}^{d} \frac{\dot{x}_{ci}(t)}{\left(x_{ci}(t)\right)^c}. \tag{5.9}$$

Integrating both sides above with respect to (w.r.t) t from 0 to t, we get

$$f(x(t)) - f(x(0)) = -\sum_{i=1}^{d} \int_0^t \frac{\dot{x}_{ci}(s)}{\left(x_{ci}(s)\right)^c} ds. \tag{5.10}$$

Since $c < 1$, upon evaluating the integral we have

$$f(x(t)) = f(x(0)) + \sum_{i=1}^{d} \frac{\left(x_{ci}(0)\right)^{1-c} - \left(x_{ci}(t)\right)^{1-c}}{1-c}. \tag{5.11}$$

Integrating both sides of (5.2) w.r.t t from 0 to t, we have

$$x_{ci}(t) = x_{ci}(0) + \int_0^t \left\|\nabla_i f(x(s))\right\|^2 ds,\ i \in \{1, \ldots, d\}.$$

Using the above equation in (5.11) proves (5.8).

Since $x_{ci}(0) > 0$, we have $x_{ci}(t) > 0$. The above equation implies that $x_{ci}(t)$ is non-decreasing w.r.t t, which combined with (5.8) and $c \in (0,1)$ implies that $f(x(t))$ in non-increasing w.r.t. t. From Assumption 5.1, f is bounded below. Thus, $\lim_{t\to\infty} f(x(t))$ is finite. From (5.11) then it follows that, $\lim_{t\to\infty} x_c(t)$ is finite. Thus, the above equation implies that $\left\|\nabla f(x(t))\right\|$ is square-integrable w.r.t t. Hence, $\nabla f(x(t))$ is bounded above.

Since $\nabla f(x(t))$ is bounded and $x_{ci}(t) > 0$, from (5.3) we have that $\dot{x}(t)$ is bounded above. Now, the time-derivative of $\|\nabla f\|^2$ along the trajectories $x(t)$ is given by

$$\frac{d}{dt}\left\|\nabla f(x(t))\right\|^2 = 2\nabla f(x(t))^T \nabla^2 f(x(t))\dot{x}(t).$$

We have shown that $\nabla f(x(t))$ and $\dot{x}(t)$ are bounded above. From Assumption 5.2, we have all the entries in $\nabla^2 f(x(t))$ bounded above. Then, from the above equation we have $\frac{d}{dt}\left\|\nabla f(x(t))\right\|^2$ bounded above. Thus, $\left\|\nabla f(x(t))\right\|^2$ is uniformly continuous.

We have shown that, $\left\|\nabla f(x(t))\right\|$ is square-integrable and $\left\|\nabla f(x(t))\right\|^2$ is uniformly continuous. From Barbalat's lemma [157] it follows that $\lim_{t\to\infty}\left\|\nabla f(x(t))\right\|^2 = 0$. This proves (5.7). □

Theorem 5.1 implies that the G-AdaGrad algorithm, proposed in (5.2)-(5.3), converges to a *critical point* in X^* of the non-convex optimization problem (5.1). Furthermore, (5.8) implies that the convergence of G-AdaGrad is affected by the algorithm parameter c. As we will show through simulations in Section 2.6, $c = 0.5$, which corresponds to the original AdaGrad method [33], is not the optimal value of c.

Another significance of the above proof is that, it explains why the exponent c of $x_c(t)$ (equivalently, b_k in discrete-time) in the update equation of the estimate $x(t)$ is limited to $c < 1$. If $c > 1$, (5.8) implies that $f(x(t))$ will be increasing in t. If $c = 1$, evaluating the integral

in (5.10) we have

$$f(x(t)) = f(x(0)) + \sum_{i=1}^{d} \log\left(\frac{x_{ci}(0)}{x_{ci}(t)}\right).$$

Thus, f decreases at a slower rate for $c = 1$ compared to $c < 1$, because of logarithmic decrements in case of $c = 1$ compared to exponential decrements.

Note that, the parameter ϵ in [135] plays the same role as c. However, the convergence results in [135] is only for $\epsilon \in (0, 0.5]$ which corresponds to $c \in (0.5, 1]$. Thus, our analysis is more general compared to [135]. In addition, our analysis in Theorem 5.1 explains the significance of the parameter c, as discussed in the previous paragraph.

5.3 State-model of Adam and AdaBelief

In continuous-time domain, the Adam algorithm can be modeled using a set of non-autonomous ODEs as follows. We define two real-valued scalar parameters $b_1, b_2 \in (0, 1)$. We further define a positive-valued function $\alpha : [0, \infty) \to \mathbb{R}$ by

$$\alpha(t) = \frac{1 - (1 - b_1)^{t+1}}{\sqrt{1 - (1 - b_2)^{t+1}}},\ t \geq 0. \tag{5.12}$$

For each $i \in \{1, \ldots, d\}$ and $t \geq 0$, we consider the following set of ODEs

$$\dot{\mu}_i(t) = -b_1 \mu_i(t) + b_1 \nabla_i f(x(t)), \tag{5.13}$$

$$\dot{\nu}_i(t) = -b_2 \nu_i(t) + b_2 \left\| \nabla_i f(x(t)) \right\|^2, \tag{5.14}$$

$$\dot{x}_i(t) = -\frac{1}{\alpha(t)} \frac{\mu_i(t)}{\sqrt{\nu_i(t)}}, \tag{5.15}$$

with initial conditions $\mu(0) \in \mathbb{R}^d$, $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$, and $x(0) \in \mathbb{R}^d$. The variables μ and v are estimates of, respectively, the biased first moment and the biased second raw moment of the gradients. These variables μ and ν can be abstracted as dynamic controller states. The term $\alpha(t)$ in (5.15) is responsible for the initial bias corrections in Adam [34]. Note that, the algorithm parameters b_1 and b_2 in the continuous-time representation above are related to the parameters β_1 and β_2 in discrete-time implementation of Adam [34], according to $\beta_1 = (1 - \delta b_1)$ and $\beta_2 = (1 - \delta b_2)$ where δ is the sampling-time for first-order Euler discretization of (5.13)-(5.15).

The AdaBelief algorithm is similar to Adam, except that $\left\| \nabla_i f(x(t)) \right\|^2$ in (5.14) is replaced by $\left\| \nabla_i f(x(t)) - \mu_i(t) \right\|^2$. In other words, the following set of ODEs represents the AdaBelief algorithm. For each $i \in \{1, \ldots, d\}$ and $t \geq 0$,

$$\dot{\mu}_i(t) = -b_1 \mu_i(t) + b_1 \nabla_i f(x(t)), \tag{5.16}$$

$$\dot{\nu}_i(t) = -b_2 \nu_i(t) + b_2 \left\| \nabla_i f(x(t)) - \mu_i(t) \right\|^2, \tag{5.17}$$

$$\dot{x}_i(t) = -\frac{1}{\alpha(t)} \frac{\mu_i(t)}{\sqrt{\nu_i(t)}}, \tag{5.18}$$

with initial conditions $\mu(0) \in \mathbb{R}^d$, $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$, and $x(0) \in \mathbb{R}^d$.

5.4 A general adaptive gradient algorithm

In this section, we propose a class of adaptive gradient algorithms, creating a state-space framework that encompasses Adam-like adaptive gradient algorithms. We follow by presenting convergence proofs of some existing adaptive gradient optimizers from a state-space perspective. These optimizers are G-AdaGrad, Adam, and AdaBelief.

Our framework of adaptive gradient algorithms hinges on their state-space representation in terms of ODEs. The proposed class of adaptive gradient algorithms is similar to the Adam method described in Section 5.3. However, there are two notable differences.

- Firstly, we replace the squared-gradient term $\left\|\nabla_i f(x(t))\right\|^2$ in the evolution of second raw moment estimate in (5.14) with $\psi(\nabla_i f(x(t)), \mu_i(t))$, where we define ψ as an well-behaved and non-negative valued function $\psi : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. The sufficient conditions on the general function ψ which guarantees convergence of the proposed algorithm are specified later in this section. Needless to say, the proposed setup includes the original Adam algorithm, i.e., $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$, as a special case.

- The other difference is that, in our algorithm, the ODE governing the second raw moment estimate of the gradients has a higher order. From simulations, we observed an issue with the Adam algorithm that the ODEs (5.13)-(5.14) for the first and second moments can converge slowly, especially if the optimization problem (5.1) is ill-conditioned, leading to a slower convergence of the estimate. A similar transient property of momentum-based algorithms have been theoretically proved in [158]. Now, each of the ODEs (5.13)-(5.14) can be seen as a controlled dynamical system. The controlled input for (5.13) and (5.14)

is, respectively, $\nabla_i f(x(t))$ and $\left\|\nabla_i f(x(t))\right\|^2$. The controlled output for (5.13) and (5.14) is, respectively, $\mu_i(t)$ and $\nu_i(t)$. It is known that addition of a zero improves the transient response of a linear time-invariant (LTI) system. Note that, the Adam algorithm does not have any zero in the dynamics (5.14). So, in our algorithm, we add a left-half plane zero to the dynamics in (5.14) for improving the convergence rate. However, in the Adam algorithm, the dynamics in (5.14) is *strictly proper*. Thus, we also add an additional left half-plane pole in (5.14). Hence, for each dimension $i \in \{1, \ldots, d\}$, the transfer function from $\psi(\nabla_i f(x(t)), \mu_i(t))$ to the second raw moment estimate $\nu_i(t)$ in our general setup has an additional pole-zero pair in the left-half plane, as compared to only one pole and no zero for its counterpart in the Adam method. Herein lies the intuition behind our algorithm.

Our general adaptive gradient algorithm is iterative in nature where, at each time $t \geq 0$, it maintains four d-dimensional vectors: an estimate $x(t)$ of a minimum point of (5.1), an estimate $\mu(t)$ of the first moment of gradients, an estimate $\nu(t)$ of the second raw moment of gradients, and an additional $\zeta(t)$ due to higher order of the transfer functions from $\psi(\nabla_i f(x(t)), \mu_i(t))$ to $\nu_i(t)$. The initial estimate $x(0)$ is chosen arbitrarily from $\mathbb{R}^d$. The real-valued vector variables μ and ζ are initialized at $t = 0$ as the d-dimensional zero vector, denoted by 0_d. The remaining variable ν is initialized according to $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. Before initiating the iterative process, the algorithm chooses nine non-negative real-valued scalar parameters $\lambda_1, \ldots, \lambda_8$ and c. Their specific conditions are presented later in this section. Given the parameters $\lambda_2 \in (0, 1)$ and

$\lambda_6 \in (0,1)$, we define a positive-valued function $\alpha_g : [0,\infty) \to \mathbb{R}$ by

$$\alpha_g(t) = \begin{cases} \dfrac{1-(1-\lambda_2)^{t+1}}{\left(1-(1-\lambda_6)^{t+1}\right)^c}, & \text{if } \lambda_7 > 0 \\ 1, & \text{if } \lambda_7 = 0. \end{cases} \tag{5.19}$$

For time $t \geq 0$, the proposed adaptive gradient optimizer in continuous-time domain are governed by the set of ODEs

$$\dot{\mu}_i(t) = -\lambda_1 \mu_i(t) + \lambda_2 \nabla_i f(x(t)), \tag{5.20}$$

$$\dot{\zeta}_i(t) = -\lambda_3 \zeta_i(t) + \lambda_3 \nu_i(t), \tag{5.21}$$

$$\dot{\nu}_i(t) = \lambda_4 \zeta_i(t) - \lambda_5 \nu_i(t) + \lambda_6 \psi(\nabla_i f(x(t)), \mu_i(t)), \tag{5.22}$$

$$\dot{x}_i(t) = -\frac{1}{\alpha_g(t)} \frac{\lambda_7 \mu_i(t) + \lambda_8 \nabla_i f(x(t))}{\nu_i(t)^c}. \tag{5.23}$$

The d-dimensional vectors μ, ζ, and ν can be abstracted as dynamic controller states. Here, $\alpha_g(t)$ represents the initial bias correction. When the moment estimate is not used in updating $x(t)$, there is no bias correction needed in the algorithm, e.g. AdaGrad [33]. Such cases are represented by $\lambda_7 = 0$ and $\alpha_g(t) = 1$ because of no bias correction.

Assumption 5.3. *We assume that the function ψ satisfies the following properties.*

A: *ψ is differentiable over $\mathbb{R}^2$. Moreover, $\psi(x,y)$ and its gradient $\nabla\psi(x,y)$ are bounded if both x and y are bounded, for any $x, y \in \mathbb{R}$.*

B: *$\psi(x,.)$ is bounded only if x is bounded, for $x \in \mathbb{R}$.*

C: $\lim_{t\to\infty} \psi(\nabla_i f(x(t)), \mu_i(t)) = 0$ *only if* $\lim_{t\to\infty} \nabla_i f(x(t)) = 0$.

The following theorem guarantees convergence of the proposed adaptive gradient optimizer, presented above in (5.20)-(5.23), to a critical point of problem (5.1).

Theorem 5.2. *Consider the set of differential equations* (5.20)-(5.23) *with initial conditions* $\mu(0) = \zeta(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{v_i(0) > 0 : i = 1, \dots, d\}$. *Let the parameters satisfy*

$$0 < c < 1,\ \lambda_2 > 0,\ \lambda_3 > 0,\ \lambda_4 \geq 0,\ \lambda_4 \leq \lambda_5 < \frac{2\lambda_1}{c},\ \lambda_6 > 0,\ \lambda_7 \geq 0,\ \lambda_8 \geq 0, \lambda_7 + \lambda_8 > 0. \tag{5.24}$$

Additionally, let $0 < \lambda_6 < \lambda_2 < 1$ *if* $\lambda_7 > 0$. *If Assumptions 5.1-5.3 hold,* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

Now we are ready to show that the AdaGrad, G-AdaGrad, Adam, and AdaBelief algorithms are included in our proposed adaptive gradient algorithm (5.20)-(5.23). Therefore, simple convergence proofs of the aforementioned algorithms for non-convex problems follow from Theorem 5.2. The following results present these convergence guarantees in continuous-time. Specifically, we obtain the following corollaries of Theorem 5.2. Recall the G-AdaGrad, the Adam, and the AdaBelief algorithms from Section 5.2 and Section 5.3.

Corollary 5.1 (G-AdaGrad). *Consider the set of differential equations* (5.2)-(5.3) *with initial conditions* $x(0) \in \mathbb{R}^d$, *and* $\{x_{ci}(0) > 0 : i = 1, \dots, d\}$. *Let the parameter* c *satisfy* $0 < c < 1$. *If Assumptions 5.1-5.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

Proof. The set of equations (5.2)-(5.3) is a special case of (5.20)-(5.23) with $\nu = x_c$, $\lambda_4 = \lambda_5 = 0$, $\lambda_6 = 1$, $\lambda_7 = 0$, $\lambda_8 = 1$, and $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$. Clearly, $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$ satisfies Assumption 5.3. Thus, Theorem 5.2 is applicable, and the proof follows.

□

Since the AdaGrad algorithm, represented by (5.2)-(5.3), is a special case of AdaGrad with $c = 0.5$, the convergence of AdaGrad is included in Corollary 5.1 above.

Corollary 5.2 (Adam). *Consider the set of differential equations* (5.13)-(5.15) *with initial conditions* $\mu(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{v_i(0) > 0 : i = 1, \ldots, d\}$. *Let the parameters* b_1 *and* b_2 *satisfy* $0 < b_2 < b_1 < 1$. *If Assumptions 5.1-5.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

Proof. The set of equations (5.13)-(5.15) is a special case of (5.20)-(5.23) with $c = 0.5$, $\lambda_1 = \lambda_2 = b_1$, $\lambda_4 = 0$, $\lambda_6 = b_2$, $\lambda_7 = 1$, $\lambda_8 = 0$, and $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$. Clearly, $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$ satisfies Assumption 5.3. Thus, Theorem 5.2 is applicable, and the proof follows. □

Corollary 5.3 (AdaBelief). *Consider the set of differential equations* (5.16)-(5.18) *with initial conditions* $\mu(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{v_i(0) > 0 : i = 1, \ldots, d\}$. *Let the parameters* b_1 *and* b_2 *satisfy* $0 < b_2 < b_1 < 1$. *If Assumptions 5.1-5.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

Proof. The set of equations (5.17)-(5.18) is a special case of (5.20)-(5.23) with $c = 0.5$, $\lambda_1 = \lambda_2 = b_1$, $\lambda_4 = 0$, $\lambda_6 = b_2$, $\lambda_7 = 1$, $\lambda_8 = 0$, and $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2$. Clearly, $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2$ satisfies Assumption 5.3.A-5.3.B. We will show that it also satisfies Assumption 5.3.C.

Suppose that $\lim_{t\to\infty} \left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2 = 0$. Then, $\lim_{t\to\infty} \nabla_i f(x(t)) = \lim_{t\to\infty} \mu_i(t)$. Then, from (5.17), $\lim_{t\to\infty} \dot{\mu}_i(t) = 0$. We define the notation $\mu_i^* = \lim_{t\to\infty} \mu_i(t) = \lim_{t\to\infty} \nabla_i f(x(t))$. We claim that $\mu_i^* = 0$. Otherwise, suppose that $\mu_i^* \neq 0$. Then, there exists $T < \infty$ such that $\mu_i(t)$ and $\nabla_i f(x(t))$ have the same sign for all $t \geq T$. Then, from (A.151), $\dot{f}(x(t)) < 0$ for all $t \geq T$. Since f is bounded below, due to Assumption (5.1), and decreasing for all $t \geq T$, $f(x(t))$

converges to its minimum value as $t \to \infty$. Upon differentiating both sides of (A.151) w.r.t. t,

$$\ddot{f}(x(t)) = \sum_{i=1}^{d} \nabla_i f(x(t)) \ddot{x}_i(t) + \sum_{i=1}^{d} \left[\nabla^2 f(x(t))\right]_i \dot{x}(t) \dot{x}_i(t). \tag{5.25}$$

Upon differentiating both sides of (5.18) w.r.r. t and substituting from (5.17),

$$\ddot{x}_i(t) = -\frac{\dot{\mu}_i(t)}{\alpha(t)\nu_i(t)^{0.5}} + \frac{\dot{\alpha}(t)\mu_i(t)}{\alpha(t)^2\nu_i(t)^{0.5}} + \frac{b_2\mu_i(t)\left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2}{2\alpha(t)\nu_i(t)^{1.5}} - \frac{b_2\mu_i(t)}{2\alpha(t)\nu_i(t)^{0.5}}.$$

From the definition of α in Section 2.1, $\alpha(t) > 0$ and $\dot{\alpha}(t)$ is bounded. Also, $\nu_i(t) > 0$. Since $\lim_{t\to\infty}\left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2 = \lim_{t\to\infty} \dot{\mu}_i(t) = 0$ and $\lim_{t\to\infty} \mu_i(t) = \mu_i^*$, we have $\left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2$, $\dot{\mu}_i(t)$, and $\mu_i(t)$ bounded. Then, $\ddot{x}_i(t)$ is bounded. Since $\lim_{t\to\infty} \nabla_i f(x(t)) = \mu_i^*$, $\nabla_i f(x(t))$ is bounded. Hence, under Assumption 5.2, (5.25) implies that $\ddot{f}(x(t))$ is bounded. Since we have $\lim_{t\to\infty} f(x(t))$ finite and $\ddot{f}(x(t))$ bounded, Barbalat's lemma [157] implies that $\lim_{t\to\infty} \dot{f}(x(t)) = 0$. But we also shown that $\dot{f}(x(t)) < 0$ for all $t \geq T$. This is a contradiction. So, Assumption 5.3.C holds and the proof is complete following Theorem 5.2. □

5.5 Continuous-time AdaBound

5.5.1 State-space model of AdaBound

In order to present our state-model of AdaBound, we define four real valued positive scalar parameters $\alpha > 0$, $\gamma > 0$, $\lambda_1 \in (0, 1)$ and $\lambda_2 \in (0, 1)$. Recall from Section 2.1 that η_l and η_u are, respectively, the time-varying lower bound and the upper bound on the learning rate. In the original paper [120], these bounds η_l and η_u are defined as positive valued functions $\eta_l : [0, \infty) \to$

$\mathbb{R}_{>0}$ and $\eta_u : [0, \infty) \to \mathbb{R}_{>0}$ such that η_l is non-decreasing and η_u is non-increasing. Additionally, both the functions η_l and η_u are assumed to asymptotically converge to some positive scalar α^*, i.e., $\lim_{t\to\infty} \eta_l(t) = \lim_{t\to\infty} \eta_u(t) = \alpha^*$. Note that, η_l and η_u are defined by the user before the algorithm proceeds. We define a positive valued function $h : [0, \infty) \to \mathbb{R}_{>0}$, which signifies the initial bias correction term, as is used in the practice while implementing AdaBound or any other adaptive gradient algorithms [81, 120]. However, our model does not require an explicit expression of h. The constraints on h and the bound functions η_l, η_u are presented below.

Assumption 5.4. *Assume that there exists $T_1 \in [0, \infty)$ such that the function h is non-increasing for all $t \geq T_1$.*

Assumption 5.5. *Assume that η_l and η_u are differentiable. Moreover, there exists $T_2 \in [0, \infty)$ such that η_u is non-increasing and η_l satisfies $\dot{\eta}_l(t) \leq 2\alpha\lambda_1\eta_l(t)$ for $t \geq T_2$.*

Assumption 5.6. *Assume that there exists two positive constants L and R such that $[\eta_l(t), \eta_u(t)] \subseteq [L, R]$ for $t \geq 0$.*

The conventional bias correction $h(t) = \frac{1-(1-\lambda_1)^{t+1}}{\sqrt{1-(1-\lambda_2)^{t+1}}}$ is included in Assumption 5.4 as a special case, when the parameters satisfy $0 < \lambda_2 < \lambda_1 < 1$. The bound functions $\eta_l(t) = \alpha^* \left(1 - \frac{1}{(1-\beta)t+1}\right)$ and $\eta_u(t) = \alpha^* \left(1 + \frac{1}{(1-\beta)t}\right)$ with $\beta \in (0, 1)$ proposed in the original AdaBound paper [120] satisfy Assumption 5.5. Assumption 5.6 is the standard boundedness assumption of η_l and η_u [120, 151].

For each $i \in \{1, \ldots, d\}$ and $t \geq 0$, we consider the following set of equations

$$\dot{\mu}_i(t) = -\lambda_1 \mu_i(t) + \lambda_1 \nabla_i f(x(t)), \tag{5.26}$$

$$\dot{\nu}_i(t) = -\lambda_2 \nu_i(t) + \lambda_2 \left\| \nabla_i f(x(t)) \right\|^2, \tag{5.27}$$

$$\eta_i(t) = Clip\left(\frac{1}{\sqrt{\nu_i(t)}}, \frac{\eta_l(t)}{\alpha}, \frac{\eta_u(t)}{\alpha} \right), \tag{5.28}$$

$$\dot{x}_i(t) = -\frac{\gamma}{h(t)} \mu_i(t) \eta_i(t), \tag{5.29}$$

with initial conditions $\mu(0) \in \mathbb{R}^d$, $\nu_i(0) > 0 \ \forall i$, and $x(0) \in \mathbb{R}^d$. Equations (5.26)-(5.29) represents a dynamical system: x is the state-vector and the variables μ, ν, and η can be abstracted as dynamic controller states. We assume that the functions η_l and η_u are smooth so that η is differentiable.

When (5.26)-(5.29) are *explicitly* discretized as $t = k\delta$ with a fixed sampling time $\delta > 0$, (5.26)-(5.29) is a continuous-time equivalent in the limit $\delta \to 0_+$ of the following algorithm. For each $i \in \{1, \ldots, d\}$ and sampling instant $k \in \{0, 1, \ldots\}$, $\mu_i((k+1)\delta) = (1 - \delta\lambda_1)\mu_i(k\delta) + \delta\lambda_1 \nabla_i f(x(k\delta))$, $\nu_i((k+1)\delta) = (1 - \delta\lambda_2)\nu_i(k\delta) + \delta\lambda_2 \left\| \nabla_i f(x(k\delta)) \right\|^2$, $\eta_i(k\delta) = Clip(\frac{1}{\sqrt{\nu_i(k\delta)}}, \frac{\eta_l(k\delta)}{\alpha}, \frac{\eta_u(k\delta)}{\alpha})$, $x_i((k+1)\delta) = x_i(k\delta) - \frac{\delta\gamma}{h(k\delta)} \mu_i(k\delta)\eta_i(k\delta)$. The above set of equations represents the AdaBound algorithm in discrete-time, as reviewed in Section 5.1, with the parameters $\beta_1 = 1 - \delta\lambda_1$, $\beta_2 = 1 - \delta\lambda_2$, $\alpha = \delta\gamma$, and the controller state $\overline{\eta}(t) = \alpha\eta(t)$. Note that, the step-size γ in our state-update equation (5.29) is constant, unlike the original AdaBound algorithm which employs a decreasing step-size $\frac{1}{\sqrt{k}}$ (ref. Section 5.1).

5.5.2 Convergence of AdaBound

The following theorem guarantees convergence of the state-model of AdaBound, presented above in (5.26)-(5.29), to a critical point in X^*. The initial states of μ, x are the same as AdaBound paper [120], and ν is assumed positive to avoid division by zero.

Theorem 5.3. *Consider the set of equations* (5.26)-(5.29) *with initial conditions* $\mu(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. *Let the model parameters satisfy* $\alpha > 0$, $\gamma > 0$, *and* $0 < \lambda_2 < 4\lambda_1 < 1$. *If Assumptions 5.1-5.2 and Assumptions 5.4-5.6 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

5.6 Convergence analysis of Nadam

To present our state-model of Nadam, we define four real valued positive scalar parameters $\gamma_1, \gamma_2 > 0$, $\lambda_1, \lambda_2 \in (0, 1)$, and two positive-valued functions $\alpha_1, \alpha_2 : [0, \infty) \to \mathbb{R}_{>0}$ as

$$\alpha_1(t) = \frac{1 - (1 - \lambda_1)^{t+1}}{\sqrt{1 - (1 - \lambda_2)^{t+1}}}, \; t \geq 0, \tag{5.30}$$

$$\alpha_2(t) = \frac{1 - (1 - \lambda_1)^{t+2}}{\sqrt{1 - (1 - \lambda_2)^{t+1}}}, \; t \geq 0. \tag{5.31}$$

For each $i \in \{1, \ldots, d\}$ and $t \geq 0$, we consider the following set of ordinary differential equations

$$\dot{\mu}_i(t) = -\lambda_1 \mu_i(t) + \lambda_1 \nabla_i f(x(t)), \tag{5.32}$$

$$\dot{\nu}_i(t) = -\lambda_2 \nu_i(t) + \lambda_2 \left\| \nabla_i f(x(t)) \right\|^2, \tag{5.33}$$

$$\dot{x}_i(t) = -\frac{\gamma_1}{\alpha_2(t)} \frac{\mu_i(t)}{\sqrt{\nu_i(t)}} - \frac{\gamma_2}{\alpha_1(t)} \frac{\nabla_i f(x(t))}{\sqrt{\nu_i(t)}}, \tag{5.34}$$

with initial conditions $\mu(0), x(0) \in \mathbb{R}^d$ and $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. The above ODEs (5.32)-(5.34) are continuous-time equivalent of the discrete-time Nadam algorithm, as reviewed in Section 2.1, with the parameters $\beta_1 = 1-\delta\lambda_1$, $\beta_2 = 1-\delta\lambda_2$, $\gamma_1 = \frac{\eta}{\delta}\frac{1-\delta\lambda_1}{\sqrt{1-\delta\lambda_2}}$, $\gamma_2 = \frac{\eta}{\delta}\frac{\delta\lambda_1}{\sqrt{1-\delta\lambda_2}}$. It can be easily seen by *explicitly* discretizing (5.32)-(5.34) with $t = k\delta$ and taking the limit at $\delta \to 0_+$, where $\delta > 0$ is the sampling time. The ODEs (5.32)-(5.34) represent a closed-loop system: x is the state-vector and μ, ν are dynamic controller states. The terms $\alpha_1(t), \alpha_2(t)$ in (5.34) capture the initial bias corrections in Nadam.

The following theorem proves convergence of the state-model (5.32)-(5.34) of Nadam to a critical point in X^*.

Theorem 5.4. *Consider the ODEs* (5.32)-(5.34) *with initialization* $\mu(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. *Let the model parameters satisfy* $\gamma_1, \gamma_2 > 0$ *and* $0 < \lambda_2 < \lambda_1 < 1$. *If Assumptions 5.1-5.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

5.7 Convergence analysis of RAdam

To present our state-model of RAdam, we define a scalar parameter $\gamma > 0$. The update law for the dynamic controllers μ and ν in RAdam is same as Nadam in (5.32)-(5.33). The state

dynamics of RAdam is described by

$$\rho(t) = \rho_\infty - 2(t+1)\frac{(1-\lambda_2)^{t+1}}{1-(1-\lambda_2)^{t+1}}, \tag{5.35}$$

$$r(t) = \sqrt{\frac{(\rho(t)-4)(\rho(t)-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho(t)}}, \tag{5.36}$$

$$\dot{x}_i(t) = \begin{cases} -\dfrac{\gamma}{\alpha_1(t)}\dfrac{r(t)\mu_i(t)}{\sqrt{\nu_i(t)}}, & \text{if } \rho(t) > 4 \\ -\dfrac{\gamma}{1-(1-\lambda_1)^{t+1}}\mu_i(t), & \text{if } \rho(t) \leq 4, \end{cases} \tag{5.37}$$

with initial conditions $\mu(0), x(0) \in \mathbb{R}^d$ and $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. The ODEs (5.32), (5.33), and (5.37) are continuous-time equivalent of the discrete-time RAdam algorithm, as reviewed in Section 2.1, with the parameters $\beta_1 = 1 - \delta\lambda_1$, $\beta_2 = 1 - \delta\lambda_2$, $\gamma = \frac{\eta}{\delta}$.

The following theorem proves convergence of the state-model of RAdam to a critical point in X^*.

Theorem 5.5. *Consider the ODEs* (5.32), (5.33), *and* (5.37) *with initialization* $\mu(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{\nu_i(0) > 0 : i = 1, \ldots, d\}$. *Let the model parameters satisfy* $\gamma > 0$ *and* $0 < \lambda_2 < \lambda_1 < 1$. *If Assumptions 5.1-5.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

5.8 Convergence of rescaled gradient flow

The rescaled gradient flow proposed in [154] is governed by the state-dynamics

$$\dot{x}(t) = -\frac{c_1 \nabla f(x(t))}{\left\|\nabla f(x(t))\right\|^{\frac{p_1-2}{p_1-1}}} - \frac{c_2 \nabla f(x(t))}{\left\|\nabla f(x(t))\right\|^{\frac{p_2-2}{p_2-1}}}, \tag{5.38}$$

paremeterized by $c_1, c_2 > 0$, $p_1 > 2$, and $p_2 \in (1, 2)$. The following theorem proves convergence of (5.38) for non-convex objective function f to a critical point in X^*.

Theorem 5.6. *Consider the ODE* (5.38) *with initial condition* $x(0) \in \mathbb{R}^d$. *Let the model parameters satisfy* $c_1, c_2 > 0$, $p_1 > 2$, *and* $p_2 \in (1, 2)$. *If Assumptions 5.1-5.2 hold,* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

5.9 Proposed algorithm: AdamSSM

In this section, we present the novel AdamSSM algorithm and its convergence guarantee. The proposed AdamSSM algorithm is in the form of the generic adaptive gradient setup, presented in Section 5.4, where the function ψ in (5.22) is defined as $\psi(\nabla_i f(x(t)), \mu_i(t)) = \left\|\nabla_i f(x(t))\right\|^2$. Specifically, the AdamSSM algorithm in continuous-time is described by the following set of ODEs, parameterized by b_1, b_2, and b_3. For each dimension $i \in \{1, \ldots, d\}$ and $t \geq 0$,

$$\dot{\mu}_i(t) = -b_1\mu_i(t) + b_1\nabla_i f(x(t)),\ \dot{\zeta}_i(t) = -b_2\zeta_i(t) + b_2\nu_i(t), \tag{5.39}$$

$$\dot{\nu}_i(t) = b_3\zeta_i(t) - (b_2 + b_3)\nu_i(t) + b_2\left\|\nabla_i f(x(t))\right\|^2,\ \dot{x}_i(t) = -\frac{1}{\alpha(t)}\frac{\mu_i(t)}{\sqrt{\nu_i(t)}}. \tag{5.40}$$

The function ψ and the parameters $\beta_1 = (1 - \delta b_1)$ and $\beta_2 = (1 - \delta b_2)$ in the proposed AdamSSM algorithm is same as Adam. However, unlike Adam, b_3 in AdamSSM is strictly positive. As a result, for each dimension $i \in \{1, \ldots, d\}$, the transfer function [80] from $\left\|\nabla_i f(x(t))\right\|^2$ to $\nu_i(t)$ in AdamSSM is $\frac{b_2(s+b_2)}{s^2+(2b_2+b_3)s+b_2^2}$ where s denotes the Laplace variable. On the other hand, for Adam we have $b_3 = 0$, and the above transfer function $\frac{b_2}{(s+b_2)}$, due to cancellation of a pole-zero pair at $s = -b_2$. The addition of a pole and a zero in AdamSSM is due to introduction of the positive-valued parameter b_3 which improves the convergence of the estimated objective function

Algorithm 5 AdamSSM: Adaptive momentum estimation with Second-order dynamics of Second Moment

Initialize $x(0) \in \mathbb{R}^d$, $\mu(0) = \zeta(0) = \nu(0) = 0_d$, $b_1, b_2, b_3 > 0$, $\epsilon > 0$, and the learning rate schedule $\eta(t)$.
for each iteration $t = 0,\ 1,\ 2, \ldots$ **do**
    Compute gradient $\nabla f(x(t))$ at the current estimate.
    Update biased first moment estimate $\mu(t+1) = (1 - \delta b_1)\mu(t) + \delta b_1 \nabla f(x(t))$.
    Update the vector $\zeta(t)$ to $\zeta(t+1) = (1 - \delta b_2)\zeta(t) + \delta b_2 \nu(t)$.
    Update biased second raw moment estimate $\nu_i(t+1) = \delta b_3 \zeta_i(t) + (1 - \delta b_2 - \delta b_3)\nu_i(t) + \delta b_2 \left\| \nabla_i f(x(t)) \right\|^2$, for each $i = 1, \ldots, d$.
    Compute bias corrected first moment estimate $\hat{\mu}(t+1) = \frac{\mu(t+1)}{1-(1-b_1)^{t+1}}$.
    Compute bias corrected second raw moment estimate $\hat{\nu}(t+1) = \frac{\nu(t+1)}{1-(1-b_2)^{t+1}}$.
    Updates the current estimate $x(t)$ to $x_i(t+1) = x_i(t) - \eta(t) \frac{\hat{\mu}_i(t+1)}{\sqrt{\hat{\nu}_i(t+1)} + \epsilon}$, $i = 1, \ldots, d$.
end for

values $f(x(t))$. This can be seen from (A.155) in the proof of Theorem 5.2, presented later in Appendix A.19. The additional parameter $b_3 > 0$ in the AdamSSM algorithm corresponds to $\lambda_4 > 0$ in the generic adaptive gradient algorithm (5.20)-(5.23). But in case of Adam we have $\lambda_4 = b_3 = 0$. Thus, for AdamSSM, there is an additional negative term on the R.H.S. of (A.155), contributing to faster decrements of $f(x(t))$. Note that the order of computational cost of the AdamSSM algorithm is same as Adam.

The formal convergence guarantee of the proposed AdamSSM algorithm in continuous-time is presented below in the form of Corollary 5.4 of Theorem 5.2.

Corollary 5.4. *Consider the set of differential equations* (5.39)-(5.40) *with initial conditions* $\mu(0) = \zeta(0) = 0_d$, $x(0) \in \mathbb{R}^d$, *and* $\{v_i(0) > 0 : i = 1, \ldots, d\}$. *Let the parameters* b_1, b_2 *and* b_3 *satisfy* $0 < b_2 < b_1 < 1$, $b_3 > 0$, $b_2 + b_3 < 4b_1$. *If Assumptions 3.1-3.2 hold, then* $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$.

The AdamSSM algorithm in discrete-time is summarized above in Algorithm 5. Specifically, we use first-order *explicit* Euler discretization with a fixed sampling rate to discretize the set of

ODEs (5.39)-(5.40). To be consistent with the format of the existing algorithms, we replace the condition $\nu_i(0) > 0 \, \forall i$ with an additional parameter $\epsilon > 0$, as it is in the existing algorithms. The purpose of either of these conditions is the same: to avoid division by zero in (5.40). Additionally, we denote the learning rate parameter for updating the estimate $x(t)$ for each iteration $t = 0, 1, \ldots$ in discrete-time by $\eta(t)$.

5.10 Proposed Algorithm: NadamSSM

Recall the state-space model of Nadam in (5.32)-(5.34), particularly the dynamic update of the controller state ν in (5.33) which is the second raw moment estimate of the gradients. It has been proved that the transient dynamics of a momentum-based algorithm, such as Nadam, is negatively impacted by the condition number of f, leading to slower convergence [158]. Therefore, the motivation behind our proposed algorithm is to improve upon the convergence rate of Nadam. We note that the transfer function from input $\left\|\nabla_i f(x(t))\right\|^2$ to output $\nu_i(t)$ in (5.33) of Nadam has no *zero* and one *pole*. In the above transfer function of our algorithm, therefore, we add (i) an LHP zero for faster transient response of controller state ν, and (ii) an LHP pole for improving stability of ν.

The proposed *Nadam with Second-order Dynamics of Second Moment* (NadamSSM) algorithm

is governed by the following state-space model. For $i \in \{1, \ldots, d\}$ and $t \geq 0$,

$$\dot{\mu}_i(t) = -\lambda_1 \mu_i(t) + \lambda_1 \nabla_i f(x(t)), \tag{5.41}$$

$$\dot{\zeta}_i(t) = -\lambda_2 \zeta_i(t) + \lambda_2 \nu_i(t), \tag{5.42}$$

$$\dot{\nu}_i(t) = \lambda_3 \zeta_i(t) - (\lambda_2 + \lambda_3)\nu_i(t) + \lambda_2 \left\| \nabla_i f(x(t)) \right\|^2, \tag{5.43}$$

$$\dot{x}_i(t) = -\frac{\gamma_1}{\alpha_2(t)} \frac{\mu_i(t)}{\sqrt{\nu_i(t)} + \epsilon} - \frac{\gamma_2}{\alpha_1(t)} \frac{\nabla_i f(x(t))}{\sqrt{\nu_i(t)} + \epsilon}, \tag{5.44}$$

with $x(0) \in \mathbb{R}^d$, $\mu(0) = \zeta(0) = \nu(0) = 0_d$. The algorithm parameters $\lambda_1, \lambda_2, \gamma_1, \gamma_2$ are the same as Nadam. The additional parameter $\lambda_3 > 0$. To be consistent with the format of the existing algorithms, we replace $\nu_i(0) > 0$ with an additional parameter $\epsilon > 0$. The purpose of either of these conditions is to avoid division by zero in (5.44). From (5.42)-(5.43), the transfer function from $\left\| \nabla_i f(x(t)) \right\|^2$ to $\nu_i(t)$ in NadamSSM is $H(s) = \frac{\lambda_2(s+\lambda_2)}{s^2+(2\lambda_2+\lambda_3)s+\lambda_2^2}$, where s denotes the Laplace variable. As described above, $H(s)$ in NadamSSM has a LHP zero at $s = -\lambda_2$ and two real valued LHP poles, as compared to no zero and one LHP pole $s = -\lambda_2$ (ref. (5.33)) in Nadam. In our experiments, we implement the proposed NadamSSM algorithm by discretizing (5.41)-(5.44) by *explicit* Euler discretization with a sampling rate $\delta = 0.15$.

5.11 Proposed Algorithm: MAdamSSM

We utilize the transfer function-based modification of Adam and Nadam algorithms on the MAdam algorithm as well. To provide the intuition of our proposed algorithm, we begin with a state-space model of the MAdam algorithm, reviewed in Section 5.1, as follows. The MAdam algorithm is a discrete-time feedback control system, with x being the state-vector and the

Algorithm 6 MAdamSSM: Maximum variation averaging Adaptive momentum estimation with Second-order dynamics of Second Moment

Initialize $x(0) \in \mathbb{R}^d$, $\tilde{\mu}(0) = \tilde{u}(0) = \zeta(0) = \tilde{\nu}(0) = w(0) = 0_d$, $\alpha \in (0,1)$, $0 < \underline{\beta} < \overline{\beta} < 1$, $\beta_3 \in (0,1)$, $\beta_3 < \underline{\beta}^2$, $\epsilon > 0$, and the step-size schedule $\eta(t)$.
for each iteration $k = 0,\ 1,\ 2, \ldots$ **do**
  Compute gradient $\nabla f(x(t))$ at the current estimate.
  for each dimension $i = 1, \ldots, d$ **do**
    Update $\tilde{\mu}_{k+1,i} = \alpha \tilde{\mu}_{k,i} + (1-\alpha)\nabla_i f(x_k)$.
    Compute $\tilde{\beta}_{k,i} = \arg\max_{\beta} \nu_{k,i}(\beta) - \mu_{k,i}(\beta)^2$.
    Compute $\beta_{k,i} = Clip\left(\tilde{\beta}_{k,i}, \underline{\beta}, \overline{\beta}\right)$.
    Update $\tilde{u}_{k+1,i} = \beta_k \tilde{u}_{k,i} + (1-\beta_k)\nabla_i f(x_k)$.
    Update $\zeta_{k+1,i} = \beta_{k,i}\zeta_{k,i} + (1-\beta_{k,i})\nu_{k,i}$.
    Update $\tilde{\nu}_{k+1,i} = \beta_3 \zeta_{k,i} + (\beta_{k,i} - \beta_3)\nu_{k,i} + (1-\beta_{k,i})\left\|\nabla_i f(x_k)\right\|^2$.
    Update $w_{k+1,i} = \beta_k w_{k,i} + (1-\beta_k)$.
    Compute $u_{k+1,i} = \frac{\tilde{u}_{k+1,i}}{w_{k+1,i}}$.
    Compute $\nu_{k+1,i} = \frac{\tilde{\nu}_{k+1,i}}{w_{k+1,i}}$.
    Update $x_{k+1,i} = x_{k,i} - \eta_k \frac{\sqrt{w_{k+1,i}}}{1-\alpha^{k+1}} \frac{\tilde{\mu}_{k+1,i}}{\sqrt{\tilde{\nu}_{k+1,i}+\epsilon}}$.
  end for
end for

variables $\tilde{\mu}$, $\tilde{u}$, $\tilde{\nu}$, w being dynamic controller states. From simulations, we observed an issue with adaptive gradient algorithms, including Adam and MAdam, that the first and second moments of gradients can converge slowly, especially if the optimization problem is ill-conditioned. This property of momentum-based algorithms have been theoretically proved in [158]. Specifically, the peak overshoot and the rise time of these methods are directly influenced by the condition number [158]. Now, the difference equation $\tilde{\nu}_{k+1,i} = \beta_k \tilde{\nu}_{k,i} + (1-\beta_k)\left\|\nabla_i f(x_k)\right\|^2 \ \forall i$ can be viewed as a controlled dynamical system. Specifically, the controlled input for updating the controller state $\tilde{\nu}_{k,i}$ is $\left\|\nabla_i f(x_k)\right\|^2$ and the controlled output is $\tilde{\nu}_{k+1,i}$. From classical control theory, we know that the addition of a zero improves the transient response of a linear system. Now, the MAdam algorithm does not have any zero in its dynamics of $\tilde{\nu}_{k,i}$. So, in our algorithm, we add a zero inside the unit circle $|z| < 1$ to the dynamics of $\tilde{\nu}_{k,i}$ for improving the convergence

rate. However, in the MAdam algorithm, the dynamics of $\tilde{\nu}_{k,i}$ is *strictly proper*. Thus, we also add a pole inside the unit circle to the dynamics of $\tilde{\nu}_{k,i}$. Hence, for each dimension i, the transfer function from $\left\|\nabla_i f(x_k)\right\|^2$ to $\tilde{\nu}_{k,i}$ in our proposed algorithm has an additional pole-zero pair inside the unit circle, as compared to only one pole and no zero for its counterpart in the MAdam method. Herein lies the intuition behind our algorithm.

The proposed MAdamSSM algorithm is summarized above in Algorithm 1. To avoid division by zero at the first iteration of Algorithm 1, we define $\tilde{u}_{1,i} = (1-\beta_1)\nabla_i f(x_1)$ and $\tilde{\nu}_{1,i} = (1-\beta_1)\left\|\nabla_i f(x_1)\right\|^2$, and $w_{1,i} = (1-\beta_1)$, as done in the MAdam algorithm [126]. The major difference between MAdam and MAdamSSM is the addition of an appropriate pole-zero pair to the dynamics from $\left\|\nabla_i f(x_k)\right\|^2$ to $\tilde{\nu}_{k+1,i}$. Specifically, unlike MAdam, the parameter β_3 in our algorithm is a strictly positive scalar. If the coefficient β_k is a constant β_2, as in Adam or AdaBound algorithm, then for each each dimension i, the transfer function from $\left\|\nabla_i f(x_k)\right\|^2$ to $\tilde{\nu}_{k,i}$ in the MAdamSSM algorithm is $T(z) = \frac{(z-\beta_2)(1-\beta_2)}{z^2-(2\beta_2-\beta_3)z+(\beta_2^2-\beta_3)}$, where z denotes the operator in Z-transform. The hyperparameter β_3 controls the placement of both the zero and the additional pole, balancing stability and acceleration. For MAdam, $\beta_3 = 0$ and the above transfer function is $T'(z) = \frac{(1-\beta_2)}{(z-\beta_2)}$, due to cancellation of a pole-zero pair at $z = \beta_2$. The parametric conditions $\beta_2 \in (0,1)$, $\beta_3 \in (0,1)$ and $\beta_3 < \beta_2^2$ ensure that the poles and zeros of $T(z)$ are within unit circle.

5.12 Experimental results

In this section, we present our experimental results validating the efficiency of the proposed G-AdaGrad and AdamSSM algorithms.

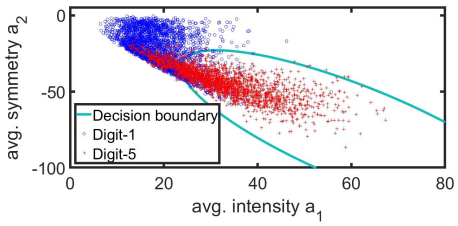

Figure 5.1: Decision boundary in the $a_1 - a_2$ plane, obtained from training a linear regression model for classification of digit-1 and digit-5 using G-AdaGrad. The data points from MNIST training set are plotted in $a_1 - a_2$ plane.

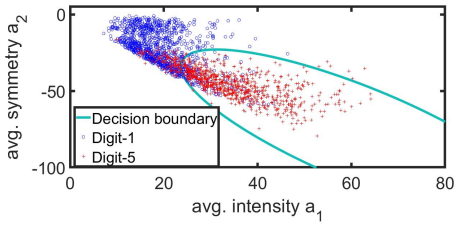

Figure 5.2: Decision boundary in the $a_1 - a_2$ plane, obtained from training a linear regression model for classification of digit-1 and digit-5 using G-AdaGrad. The data points from MNIST test set are plotted in $a_1 - a_2$ plane.

5.12.1 G-AdaGrad

We consider the problem of recognizing handwritten digit one and digit five. Although it is a binary classification problem between the digits one and five, we solve it as a regression problem first. The obtained linear regression model can be a good initial decision boundary (ref. Fig. 5.1-5.2) for classification algorithms. We conduct experiments for minimizing the *objective function* $f(x) = \frac{1}{2}\|Ax - b\|^2$. The training data points (A, b) are obtained from the *"MNIST"* [47] dataset as follows. We select 5000 arbitrary training instances labeled as either the digit one or the digit five. For each instance, we calculate two quantities, namely the average intensity of an

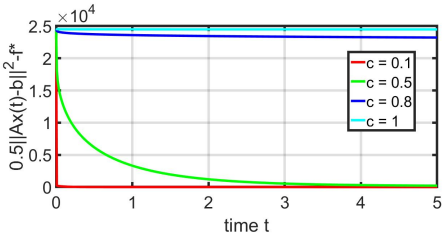

Figure 5.3: $\frac{1}{2}\left\|Ax(t) - b\right\|^2 - f^*$ of linear regression for classifying digit-1 and digit-5 from the MNIST dataset with G-AdaGrad.

image and the average symmetry of an image [48]. Let the column vectors a_1 and a_2 respectively denote the average intensity and the average symmetry of those 5000 instances. We perform a quadratic *feature transform* of the data (a_1, a_2). Then, our input matrix before pre-processing is $\tilde{A} = \begin{bmatrix} a_1 & a_2 & a_1.^2 & a_1.*a_2 & a_2.^2 \end{bmatrix}$. Here, $(.*)$ represents element-wise multiplication and $(.^2)$ represents element-wise squares. This raw input matrix $\tilde{A}$ is then pre-processed as follows. Each column of $\tilde{A}$ is shifted by the mean value of the corresponding column and then divided by the standard deviation of that column. Finally, a 5000-dimensional column vector of unity is appended to this pre-processed matrix. This is our final input matrix A of dimension (5000×6). Next we consider the logistic regression model and conduct experiments for minimizing the cross-entropy error on the raw training data.

We train both of these models with the G-AdaGrad algorithm (5.2)-(5.3). We initialize the algorithm according to the conditions in Theorem 5.1. Specifically, we set $x_c(0) = x(0) = [0.01, \ldots, 0.01]^T$. G-AdaGrad converges for different values of c (ref. Fig. 5.3 and Fig. 5.4). We observe that the convergence is faster when c is smaller. Thus, the coefficient $c = 0.5$, which corresponds to the original AdaGrad method, is not the optimal choice. In addition, $c = 1$ leads to poor convergence, as we have theoretically explained in Section 5.2.2.

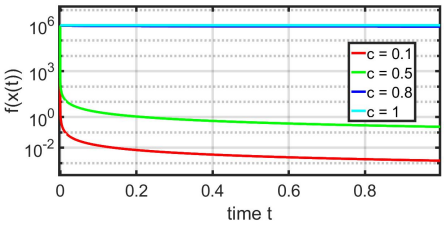

Figure 5.4: Training loss of logistic regression model for classifying digit-1 and digit-5 from the MNIST dataset with G-AdaGrad.

5.12.2 AdamSSM

We present experimental results on benchmark machine learning problems, comparing the convergence rate and test-set accuracy [48] of the proposed AdamSSM algorithm with several other adaptive gradient methods. These methods are AdaBelief [81], AdaBound [120], Adam [34], AdamW [121], Fromage [122], MSVAG [123], RAdam [124], SGD [32], and Yogi [125].

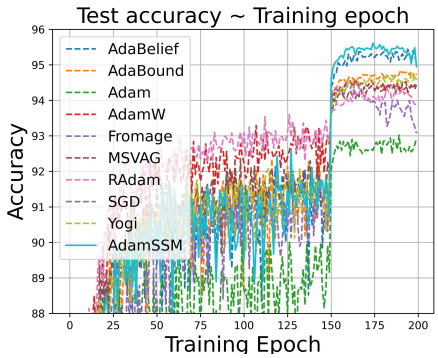

Figure 5.5: Test set accuracy for image classification task on CIFAR-10 dataset with ResNet34 architecture trained with different algorithms.

In the experiments, we consider two machine learning tasks: image classification on CIFAR-10 dataset [159] and language modeling on Penn TreeBank (PTB) dataset [160]. The CIFAR-10 dataset consists of $60k$ tiny colour images with 32×32 pixels in 10 mutually exclusive classes, with $6k$ images per class. There are $50k$ training images and $10k$ test images. The PTB dataset consists of $929k$ training words, $73k$ validation words, and $82k$ test words.

For image classification task, we use two CNN architectures: ResNet34 [161] and VGG11 [162]. The numeral after the keyword signifies the number of weighted layers in that architecture. ResNet34 and VGG11 has approximately $d = 21$ million and $d = 133$ million parameters, respectively. These are the state-of-the-art architectures for image classification. ResNet, in particular, solves the famous vanishing gradient problem, where the computed gradients get truncated to

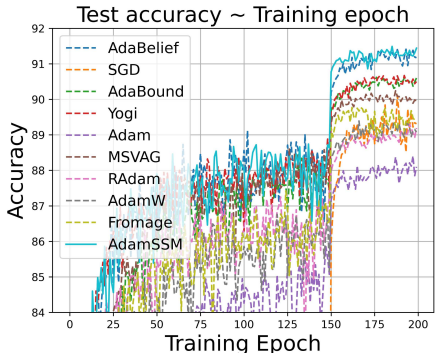

Figure 5.6: Test set accuracy for image classification task on CIFAR-10 dataset with VGG11 architecture trained with different algorithms.

zero due to repeated application of chain rule across deep layers during back-propagation and due to finite precision.

For language modeling task, we use the long short-term memory (LSTM) [163] architecture with respectively 1-layer, 2-layers, and 3-layers. LSTM is a widely used language model in different applications, including text generation and speech recognition. It is a recurrent neural network with 'gates' which are neural network that learns the important information from the training data corpus. Perplexity [164] is a metric for measuring performance of a language model. Technically, a language model computes the joint probability of a word sequence from product of conditional probabilities of each word. Perplexity is defined as the inverse probability of the test set, as predicted by a trained model, normalised by the number of words. Perplexity can also be interpreted as the number of words that can be encoded with the cross-entropy. Thus, lower the perplexity, more confident the model is in predicting the next word in a sequence. So, a lower perplexity is preferred.

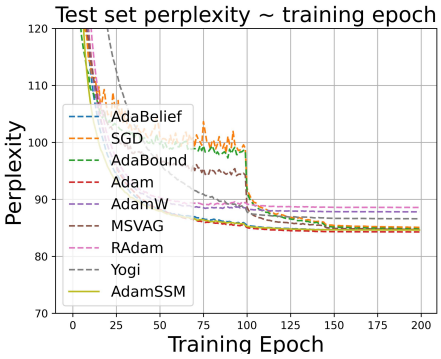

Figure 5.7: Test set perplexity for language modeling task on PTB dataset with 1-layer LSTM architecture trained with different algorithms.

To conduct these experiments, we adapt the experimental setup used in the recent AdaBelief paper [81] and the AdaBound paper [120]. The estimate $x(t)$ is initialized randomly from $\mathbb{R}^d$ for all the algorithms. The hyperparameters of the respective algorithms are tuned such that the individual algorithms achieves a better generalization on the test dataset. Following [81], these hyperparameters are selected as described below.

AdaBelief: The standard parameter values $\beta_1 = 0.9$, $\beta_2 = 0.999$ are used. The parameter

ϵ is set to 10^{-8} for image classification tasks, 10^{-16} for 1-layer LSTM and 10^{-12} for 2-layer and 3-layer LSTM. The learning rate η is set to 10^{-2} for 2-layer and 3-layer LSTM, and 10^{-3} for all other models. These parameter values are set according to the implementation of AdaBelief in GitHub [1].

AdaBound, Adam, MSVAG, RAdam, Yogi: The parameter β_1 is selected from the set $\{0.5, 0.6, 0.7, 0.8, 0.9\}$. The learning rate η is selected from the set $\{10^p : p = 1, 0, -1, -2, -3\}$. Standard values are used for the other parameters.

AdamW: The weight decay parameter is chosen from the set $\{10^{-2}, 10^{-3}, 5\times10^{-4}, 10^{-4}\}$. The other parameters are selected in the same way as Adam.

Fromage, SGD: The learning rate is selected as described above for Adam. The momentum is chosen as the default value of 0.9.

AdamSSM: The parameters $\beta_1 = (1 - \delta b_1)$ and $\beta_2 = (1 - \delta b_2)$ are similar to Adam. The standard choices for β_1 and β_2 in Adam are respectively 0.9 and 0.999 [34]. With a sampling time $\delta = 0.15$, therefore we set $b_1 = 0.67$ and $b_2 = 0.0067$. The parameter b_3 is chosen from the set $\{\frac{c\times10^{-3}}{\delta} : c = 1, 2, 3, 4, 5\}$. The parameter ϵ and the learning rate η are selected in the same way as AdaBelief.

In our experiments, we have considered the *cross-entropy* loss function [48]. To avoid overfitting, we have used l_2-regularization [48] while training the architectures. Following [81], the regularization hyperparameter is set to 5×10^{-4} for the image classification tasks and 1.2×10^{-6} for the language modeling task, for each of the aforementioned algorithms.

For the image classification tasks, the model is trained for 200 epochs; the learning rate is multiplied by 0.1 at epoch 150; and a mini-batch size of 128 is used [81, 120]. We compare

[1] https://github.com/juntang-zhuang/Adabelief-Optimizer

the training-set and test-set accuracy of different training algorithms in Table 5.1 and Table 5.2. We observe that the proposed AdamSSM algorithm has the best test-set accuracy among all the algorithms, on both the architectures ResNet34 and VGG11. Some other algorithms achieve a better training-set accuracy than the proposed method. However, the test-set accuracy of those algorithms is less than AdamSSM.

For the language modeling tasks, the model is trained for 200 epochs; the learning rate is multiplied by 0.1 at epoch 100 and 145; and a mini-batch size of 20 is used [81]. We compare the training-set and test-set perplexity of different training algorithms in Table 5.3-5.5. Note that a lower perplexity means better accuracy. For 1-layer LSTM, only the Adam method generates lower test set perplexity than the proposed method. For 2-layer LSTM, only the AdaBelief method generates lower test set perplexity than the proposed method. For the more complex 3-layer LSTM, the proposed method achieves both the least test set and the least training set perplexity.

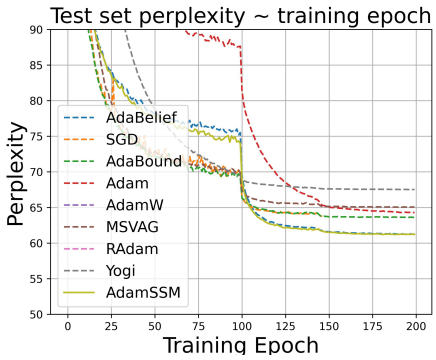

Figure 5.8: Test set perplexity for language modeling task on PTB dataset with 3-layer LSTM architecture trained with different algorithms.

5.12.3 NadamSSM

We implement our NadamSSM algorithm in discrete-time and compare its performance with AdaBelief [81], AdamSSM, and Nadam [38] algorithms for solving the following machine learning tasks: image classification on CIFAR-10 dataset [159], with ResNet34 [161] and VGG11

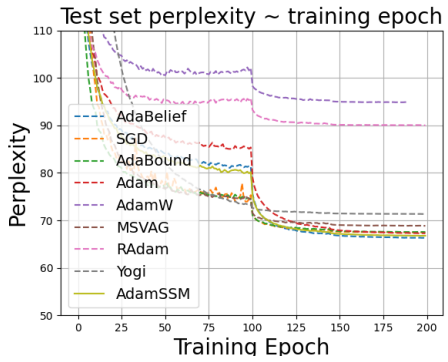

Figure 5.9: Test set perplexity for language modeling task on PTB dataset with 2-layer LSTM architecture trained with different algorithms.

[162] models, and language modeling on Penn TreeBank (PTB) dataset [160], with 3-layer long short-term memory (LSTM) [163] model. AdaBelief has been shown to be more efficient than the popular optimizers [81] on benchmark machine learning tasks. However, the experimental results in [81] have not compared AdaBelief with Nadam.

We use the experimental setup as in the recent AdaBelief paper [81]. Following [81], the l_2-regularization hyperparameter is set to 5×10^{-4} for image classification and 1.2×10^{-6} for language modeling. The hyperparameters are tuned such that the individual algorithms achieves a better generalization on the test set. Specifically, all the parameter values of AdaBelief and AdamSSM are set as per the implementation in the AdaBelief paper [81] and Section 5.12.2. For Nadam and NadamSSM, $\lambda_1 = 0.67, \lambda_2 = 0.0067$ are such that $\beta_1 = (1-\delta\lambda_1)$ and $\beta_2 = (1-\delta\lambda_2)$ are same as AdaBelief. λ_3 is chosen from $\{c \times 10^{-3}/\delta : c = 1, 2, 3, 4, 5\}$. The parameter ϵ and the learning rate η are same as AdaBelief and AdamSSM.

All the models are trained for 200 epochs. At each training epoch, we evaluate the performance of the models, trained with different optimizers, on the test data and compare the smallest error of individual optimizers. For image classification, the learning rate is multiplied by 0.1 at epoch 150; the mini-batch size is 128 [81]. For language modeling, the learning rate is multiplied by 0.1

at epochs 100 and 145; the mini-batch size is 20 [81]. From Table 5.1 and Table 5.2, the proposed NadamSSM algorithm has the highest test set and training set accuracy on the VGG11 model, but AdaBelief and AdamSSM perform better on ResNet34. From Table 5.5, NadamSSM achieves the least test set perplexity (smaller perplexity is better) on 3-layer LSTM. Since each optimizer is run for 200 epochs and NadamSSM obtains a smaller test error on VGG11 and LSTM over the same number (200) of epochs, NadamSSM is faster among the algorithms on VGG11 and LSTM. We note that when Nadam is better than AdaBelief/AdamSSM, NadamSSM significantly improves on Nadam. Similarly, on ResNet34, when Nadam is poorer than AdaBelief/AdamSSM, so is NadamSSM but with better performance than Nadam. This behavior can be attributed to our transfer function-based modification of Nadam.

5.12.4 MAdamSSM

We present experimental results on a benchmark machine learning problem, comparing the proposed MAdamSSM algorithm and the MAdam algorithm. MAdam has been demonstrated to be more efficient than the AdaBelief optimizer [126], which in turn is superior to the other existing optimizers [81], on several machine learning tasks.

In the experiments, we consider the machine learning task of image classification on the CIFAR-10 and CIFAR-100 datasets [159] using two CNN architectures: ResNet18 and ResNet34 [161]. We adapt the experimental setup used in the MAdam paper [126]. The objective function f is *cross-entropy*. The estimate $x(t)$ is initialized randomly from $\mathbb{R}^d$ for both the algorithms. Following [126], the hyperparameters of the respective algorithms are tuned such that the individual algorithms achieve better accuracy on the test set, as described below. The step-size schedule is

cosine [165], which starts with a large value and rapidly decreases to a minimum value before being increased rapidly again. This minimum value is set to $2e{-}6$. **MAdam**: The L_2 regularization parameter is selected from $\{0.025, 0.05, 0.1, 0.2, 0.4, 0.8, 1\}$, the best value being 0.05 for CIFAR-10 and 0.2 for CIFAR-100. The learning rate is selected from $\{ce-3 : c = 0.5, 1, 2, 3, 4, 6, 8\}$, the best value being $8e-3$ for CIFAR-10 and $4e-3$ for CIFAR-100. The other parameters are $\alpha = 0.9, \underline{\beta} = 0.5, \overline{\beta} = 0.999, \epsilon = 1e-8$ [126]. **MAdamSSM**: The regularization parameter, learning rate, $\alpha, \underline{\beta}, \overline{\beta}$ and ϵ are same as MAdam. β_3 is selected from $\{ce-3 : \ c = 1, 2, 3, 4, 5\}$ and the best value is observed to be $2e-3$ for CIFAR-10 and $1e-3$ for CIFAR-100. The ResNet models are trained over 200 epochs with a mini-batch size of 128 [126]. We compare the training-set and test-set accuracy of MAdamSSM with MAdam over five runs. From Table 5.6-5.7, we observe that the proposed MAdamSSM algorithm has better test-set accuracy on both architectures. Additionally, MAdamSSM achieves this accuracy in fewer training epochs.

Table 5.1: Comparisons between best training accuracy, best test accuracy, and number of training epochs required to achieve these accuracies for different algorithms on image classification task with ResNet34.

Training algorithm	Test accuracy	Epoch	Train accuracy	Epoch
AdaBelief	95.44	194	99.988	193
AdaBound	94.85	190	99.998	191
Adam	93.02	189	99.308	190
AdamW	94.59	164	**100.0**	169
Fromage	94.51	165	99.992	165
MSVAG	94.44	199	99.996	185
RAdam	94.33	182	**100.0**	179
SGD	94.64	155	99.272	169
Yogi	94.71	182	99.972	192
AdamSSM (Proposed)	**95.61**	174	99.99	188
Nadam	95.31	189	99.99	182
NadamSSM (Proposed)	95.38	178	99.98	193

Table 5.2: Comparisons between best training accuracy, best test accuracy, and number of training epochs required to achieve these accuracies for different algorithms on image classification task with VGG11.

Training algorithm	Test accuracy	Epoch	Train accuracy	Epoch
AdaBelief	91.41	193	99.784	197
AdaBound	90.62	176	99.914	193
Adam	88.40	197	94.028	199
AdamW	89.39	166	99.312	198
Fromage	89.77	162	99.730	170
MSVAG	90.24	187	**99.948**	192
RAdam	89.30	195	98.984	196
SGD	90.11	188	96.436	195
Yogi	90.67	192	99.868	196
AdamSSM (Proposed)	91.49	185	99.792	187
Nadam	91.52	191	99.82	185
NadamSSM (Proposed)	**91.81**	173	99.84	197

Table 5.3: Comparisons between best training set perplexity, best test set perplexity, and number of training epochs required to achieve these perplexities for different algorithms on language modeling task with 1-layer LSTM.

Training algorithm	Test accuracy	Epoch	Train accuracy	Epoch
AdaBelief	84.63	199	58.25	192
AdaBound	84.78	199	62.36	155
Adam	**84.28**	196	58.26	155
AdamW	87.80	194	**55.33**	155
MSVAG	84.68	199	63.59	167
RAdam	88.57	196	55.81	155
SGD	85.07	199	63.64	155
Yogi	86.59	199	69.22	155
AdamSSM (Proposed)	84.61	199	58.93	192

5.13 Summary and discussions

We have proposed a fast optimizer, named Generalized AdaGrad (G-AdaGrad), a generalization of the prototypical AdaGrad algorithm. We have acquired state-space framework of the G-

Table 5.4: Comparisons between best training set perplexity, best test set perplexity, and number of training epochs required to achieve these perplexities for different algorithms on language modeling task 2-layer LSTM.

Training algorithm	Test accuracy	Epoch	Train accuracy	Epoch
AdaBelief	**66.29**	199	45.48	184
AdaBound	67.53	199	**43.65**	165
Adam	67.27	199	46.86	184
AdamW	94.86	186	67.51	184
MSVAG	68.84	199	45.90	184
RAdam	90.00	199	61.48	184
SGD	67.42	197	44.79	165
Yogi	71.33	199	54.53	143
AdamSSM (Proposed)	66.75	198	44.92	190

Table 5.5: Comparisons between best training set perplexity, best test set perplexity, and number of training epochs required to achieve these perplexities for different algorithms on language modeling task 3-layer LSTM.

Training algorithm	Test accuracy	Epoch	Train accuracy	Epoch
AdaBelief	61.24	194	37.06	197
AdaBound	63.58	195	37.85	193
Adam	64.28	199	43.11	197
AdamW	104.49	159	104.94	155
MSVAG	65.04	192	39.64	185
RAdam	93.11	199	90.75	185
SGD	63.77	146	38.11	146
Yogi	67.51	196	51.46	164
AdamSSM (Proposed)	61.18	188	36.82	197
Nadam	61.10	181	**35.95**	197
NadamSSM (Proposed)	**60.85**	187	36.44	197

AdaGrad algorithm, governed by a set of ordinary differential equations. From the proposed state-space viewpoint, we have presented simple convergence proof of G-AdaGrad for non-convex optimization problems. Our analysis of G-AdaGrad has provided further insights into the AdaGrad method. The theoretical results have been validated empirically on *MNIST* dataset.

Most importantly, we have proposed a generic adaptive gradient-descent optimizer to accelerate

Table 5.6: Comparisons between mean (and std.) of the best training and test accuracies, and the number of training epochs required to achieve them for the MAdamSSM (proposed) and MAdam algorithms over five runs.

Model, Dataset	MAdamSSM (Proposed)				MAdam			
	Test accuracy	Epoch	Train accuracy	Epoch	Test accuracy	Epoch	Train accuracy	Epoch
ResNet18, CIFAR-10	**95.36** (**0.09**)	189.8 (2.3)	100.0 (0)	191.2 (4.3)	95.29 (0.1)	194.6 (4.7)	100.0 (0)	187.2 (4.7)
ResNet34, CIFAR-10	**95.46** (**0.14**)	187.0 (6.3)	100.0 (0)	195.4 (2.9)	95.37 (0.14)	188.0 (3.2)	100.0 (0)	190.8 (3.6)
ResNet18, CIFAR-100	**78.95** (**0.25**)	187.8 (2.5)	99.98 (0.0)	194.8 (2.9)	78.83 (0.26)	192.6 (4.2)	99.98 (0.0)	193.2 (2.9)

Table 5.7: Comparisons between the best training and test accuracies, and the number of training epochs required to achieve them for the MAdamSSM (proposed) and MAdam algorithms applied to train ResNet34 with CIFAR-10 data over five runs.

	MAdamSSM (Proposed)				MAdam			
	Test accuracy	Epoch	Train accuracy	Epoch	Test accuracy	Epoch	Train accuracy	Epoch
Run 1 (seed= 0)*	95.30	179	100.0	192	**95.49**	185	100.0	191
Run 2 (seed= 50)	**95.53**	189	100.0	199	95.18	189	100.0	196
Run 3 (seed= 100)	95.32	187	100.0	196	**95.47**	193	100.0	192
Run 4 (seed= 500)	**95.51**	196	100.0	197	95.43	186	100.0	188
Run 5 (seed= 1000)	**95.63**	184	100.0	193	95.27	187	100.0	187

* *seed* initializes the random number generator of Python's NumPy package

gradient-based optimization of non-convex optimization problems. Adaptive gradient optimizers are an integral component of modern-day machine learning pipelines. We have presented a

state-space framework for such optimizers, facilitating a dynamical system perspective of some prominent adaptive gradient optimizers and their potential variations. Specifically, the estimate of the true minima is the system state, and updating the learning rate of the state can be described as a control input. When the learning rates are updated adaptively, the control input is dynamically evolving. Hence, the dynamic controller states are governed by feedback inputs that are a function of the current estimate (the current system state). The existing optimizers specify the feedback input to update the dynamic controller states. We have proposed a general class of feedback inputs to the controller states and specified sufficient conditions for the feedback to guarantee convergence of our general algorithm for adaptive optimization. Another salient feature of our generic algorithm is the addition of a suitable pole and zero in the transfer function to the controller state from its corresponding input. By rigorously analyzing this state-space model of the proposed algorithm, we have developed a framework from which simplified proofs of existing algorithms, such as G-AdaGrad, Adam, and AdaBelief, follow. This framework can yield constructive insights to design new optimizers, as we have shown by developing the AdamSSM algorithm. AdamSSM is a variant of Adam, where adding a pole-zero pair in the second-moment estimate of gradients improves the convergence through a re-balance of transient and steady state. We proposed such transfer function-based variants of Nadam and MAdam algorithms also. Through extensive experiments on complex machine learning problems, we have demonstrated that these proposed algorithms reduces the gap between better accuracy on unseen data and faster convergence than the state-of-the-art algorithms.

Notable fast gradient optimizers AdaBound, Nadam, RAdam, and the recent rescaled gradient flow are not included in our proposed framework of generic adaptive gradient methods. However, we presented convergence proofs of these methods from a state-space analysis in continuous time

using Barbalat's lemma for non-convex optimization problems, similar to our unifying approach for other adaptive gradient methods. These theoretical results suggest that investigating the convergence analysis of other fast gradient-based optimizers in the mold of the simple proof sketch presented in this chapter may be fruitful.

A new optimizer can also be developed by modifying the AdaBelief algorithm similarly. Formally speaking, this new variant can be described by our generic state-space framework (5.20)-(5.23) as follows:

$$\dot{\mu}_i(t) = -b_1\mu_i(t) + b_1\nabla_i f(x(t)),\ \dot{\zeta}_i(t) = -b_2\zeta_i(t) + b_2\nu_i(t),$$
$$\dot{\nu}_i(t) = b_3\zeta_i(t) - (b_2+b_3)\nu_i(t) + b_2\left\|\nabla_i f(x(t)) - \mu_i(t)\right\|^2,\ \dot{x}_i(t) = -\frac{1}{\alpha(t)}\frac{\mu_i(t)}{\sqrt{\nu_i(t)}}.$$

Convergence of the above algorithm directly follows from Theorem 5.2. However, further studies are needed to validate the efficacy of this optimizer compared to the existing optimizers. In this work, we modified the transfer function only for the second moment estimate of the gradients. The idea can be extended to modifying the transfer function for the first moment estimate as well. Additionally, other input functions ψ that satisfy Assumption 3.3 can be explored for developing potentially better optimizers. In this work, we have analyzed asymptotic convergence. Another direction of work is characterizing the (finite-time) rate of convergence.

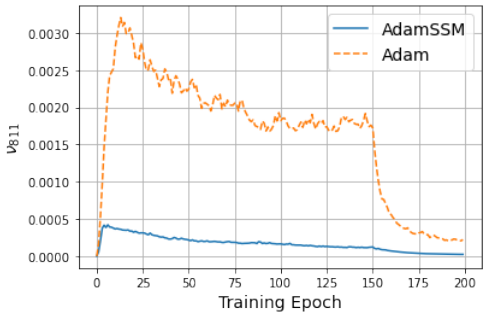

Figure 5.10: Second raw moment estimate of gradient along dimension $i = 811$, for image classification task on CIFAR-10 dataset with VGG11 architecture trained with Adam and the proposed AdamSSM algorithms.

Recall that the transfer function from the squared gradient to $\nu_i(t)$ is $\frac{b_2(s+b_2)}{s^2+(2b_2+b_3)s+b_2^2}$ for the proposed AdamSSM algorithm and $\frac{b_2}{s+b_2}$ for the Adam algorithm. So, the two poles and the single zero of this transfer function depend on the parameter values b_2, b_3, where b_3 is an additional parameter compared to Adam. By tuning the hyperparameters b_2, b_3 we create a signal $\nu_i(t)$ that is less impacted by the noise than the input signal of the squared gradient. The transfer functions for both the Adam and AdamSSM methods act as a low-pass filter on the squared gradient input to the output $\nu_i(t)$. So, in other words, by properly tuning the hyperparameters b_2, b_3, the AdamSSM algorithm attenuates frequency-specific noise in the squared gradient during the training better than Adam, which leads to improved generalization. We numerically validate our above hypothesis on the image classification task with VGG11. In Figure 5.10-5.11, we have plotted the curves of $\nu_i(t)$ when the VGG11 model is trained with Adam and AdamSSM algorithms, along two different dimensions $i = 811, 1728$. The output signal $\nu_i(t)$ increases initially, albeit with larger damping for AdamSSM due to the addition of an LHP pole compared to Adam. After the initial rise, $\nu_i(t)$ in the Adam method fluctuates much more throughout the stable part of the signal. It implies that the output $\nu_i(t)$ of Adam is significantly more impacted by the noise, which supports our hypothesis that the AdamSSM algorithm supports better generalization by being more robust to noisy signals.

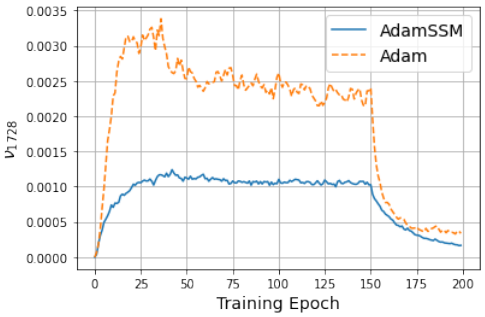

Figure 5.11: Second raw moment estimate of gradient along dimension $i = 1728$, for image classification task on CIFAR-10 dataset with VGG11 architecture trained with Adam and the proposed AdamSSM algorithms.

Chapter 6: Distributed Beamforming

6.1 Introduction

Distributed beamforming involves coordination among a group of autonomous transmitting antennas, referred to as *beamforming agents*, in such a manner that constructive interference of the transmitted signals occurs at the desired receiver locations while ensuring destructive interference at the unintended receivers. The major research trends in distributed beamforming problem over the past two decades has been classified in the survey [166]. In this chapter, we consider the problem of beampattern matching, where the beamforming agents aim to optimize the phase and amplitude of the transmitted signals to achieve the desired beampattern at the target receivers. The desired beampattern comprises target *beams* and *nulls* at specific locations, enabling directed communication links.

6.1.1 Related works

The majority of research works in optimizing beampattern aims at forming nulls at unintended directions [167–169] or maximizing the beampattern directivity [170–172]. An extensive literature of related works can be found in [166]. Another class of works consider simultaneous beamforming and nullforming [173–177] which utilize the periodic feedback from the beam and null receivers.

However, most of the aforementioned works aim to maximize the gain or steering nulls at specific locations rather than match the radiation pattern. The problem of matching the radiation pattern is more general than the above objectives, since it offers precise control over the desired power level at multiple directions and inherently includes simultaneous beamforming and nullforming at specified locations in a constructive manner. Moreover, formulation of the beampattern matching problem as an optimization problem offers ability to include additional constraints, such as limited transmit power of the agents, or derivative constraints [178] when the receiver locations are not precisely known.

Recent methodologies in [179–183] have considered distributively matching the desired beampattern. The algorithms in [179–181] use feedback from the receivers. The beamforming techniques in [182, 183] focus on selecting a subset of active transmitters, also known as sparse beamforming. In this chapter, we consider the number of transmitting agents to be fixed, and that the agents are stationary. However, our proposed algorithm can potentially be combined with the optimal antenna placement and the sparse beamforming approach in [182] and will be explored in our future work. The existing algorithms in [181, 184] do not assume a channel model for beamforming. A probabilistic channel prediction and its integration technique with path planning for mobile beamforming agents have been proposed in [184]. However, the objective in [184] is energy efficiency in transmitting power and movement of the agents, rather than matching the desired beampattern. Unlike [181], our proposed algorithm does not require feedback from any of the receivers. Most recently, a receiver-feedback free approach has been proposed in [185]. Unlike [185], we do not assume line-of-sight channels between receivers and the beamforming agents.

6.1.2 Summary of our contributions

To distributively solve the problem of beampattern matching at a set of desired receiver locations, we propose a novel algorithm, coined Iteratively Pre-Conditioned Gradient-descent for Distributed Beamforming (IPG-DB). The proposed IPG-DB algorithm works in a distributed *server-agent* architecture and is built on top of the classical gradient-descent algorithm. The server can be an auxiliary node, acting as a coordinator between the agents. Our key findings are summarized below.

- Our proposed algorithm achieves significant acceleration over the recently proposed gradient-based distributed beamforming algorithm in [180]. This feature enables the proposed algorithm to be potentially incorporated in the alternate optimization framework [180, 182] for the joint optimization of position, sparsity, and transmitted signal.

- Unlike most of the existing works on distributed beampattern matching with a fixed set of transmitting agents [179–181], our algorithm does not require feedback from any of the receivers.

- Unlike the existing works [179–181], our algorithm does not require information on channel fading parameters, thereby demonstrating robustness.

- Through simulations on a synthetic problem and a simulated urban environment of Los Angeles, we validate the aforementioned claims of our proposed beamforming algorithm.

6.2 Problem formulation

Given a fixed number n of beamforming agents, the objective is to match the desired beampattern at certain locations in space. Each agent is equipped with a transmitting antenna. Let there be s number of samples, where each sample $i \in \{1, \ldots, s\}$ represents the location of the receiver in two-dimensional polar coordinates (ρ_i, θ_i) and the amplitude of the desired array factor f_i at that receiver. Note that f_i can be a null, and therefore, the considered problem includes simultaneous beamforming and nullforming. We assume that the agents' locations are fixed, and each agent $m \in \{1, \ldots, n\}$ is located at $(x_m, y_m) \in \mathbb{R}^2$ in Cartesian co-ordinates. We let the Euclidean distance between an agent m and the receiver i be denoted by

$$d_{i,m} = \left\| \begin{bmatrix} x_m \\ y_m \end{bmatrix} - \rho_i \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \right\|. \tag{6.1}$$

Throughout this chapter, the system is assumed synchronous. Carrier frequency and phase synchronization of the agents can be done prior to deploying the optimization algorithm for solving the beamforming problem [166], and is not in the scope of this work. We let k denote the synchronized carrier wave number. Then, the array factor constructed by the agents at receiver $i \in \{1, \ldots, s\}$ is given by [180]

$$AF_i = \sum_{m=1}^{n} h_{i,m} I_m \exp jk \left(x_m \cos\theta_i + y_m \sin\theta_i\right), \tag{6.2}$$

where $I_m \in \mathbb{C}$ is a complex number that represents the excitation signal of the agent m and $h_{i,m} \in \mathbb{C}$ is a complex number that represents the channel gain from agent m to receiver i. The

excitation signal of agent m with amplitude a_m and phase α_m is given by

$$I_m = a_m \exp j\alpha_m. \tag{6.3}$$

We model the channel gain as

$$h_{i,m} = \frac{\gamma_{i,m}}{d_{i,m}} \exp jkd_{i,m}, \tag{6.4}$$

where $\gamma_{i,m} \in \mathbb{C}$ represents the unknown multipath fading between agent m and receiver i. Upon substituting from (6.3)-(6.4) into (6.2) we obtain that

$$AF_i = \sum_{m=1}^{n} \frac{a_m \gamma_{i,m}}{d_{i,m}} \exp j\left(\alpha_m + kx_m \cos\theta_i + ky_m \sin\theta_i + kd_{i,m}\right). \tag{6.5}$$

We have assumed there is no mutual coupling between the agents. The agents collaborate to construct the desired beampattern at all the receivers $i \in \{1, \ldots, s\}$. Specifically, each agent $m \in \{1, \ldots, n\}$ controls their excitation amplitude a_m and phase α_m with the aim that the combined array factor $|AF_i|$ at the receiver i is as close as possible to f_i. The aforementioned beamforming task of the agents can be formulated as the following optimization problem [180]

$$(a_m^*, \alpha_m^*)_{m=1}^n = \arg \min_{(a_m,\alpha_m)_{m=1}^n} \sum_{i=1}^{s} \frac{w_i}{2} \left\| f_i - |AF_i| \right\|^2, \tag{6.6}$$

so that the sum-of-squares error between the desired and constructed array factor amplitude at all the receivers is minimized. Here, w_i is a penalty weight assigned to the cost at receiver $i \in \{1, \ldots, s\}$, which characterize the relative importance of receiver i. Note that the cost function in (6.6) is non-convex with respect to the optimization variables $(a_m^*, \alpha_m^*)_{m=1}^n$. We assume that

each agent knows the synchronized carrier frequency, its own location (x_m, y_m), the location of the receivers $\{(\rho_i, \theta_i),\ i \in \{1, \ldots, s\}\}$, and the desired amplitudes $\{f_i,\ i \in \{1, \ldots, s\}\}$.

Recently proposed approaches in [179–181] utilize the gradient-descent (GD) method for solving the beamforming problem (6.6). However, It is known that convergence rate of classical gradient-descent method is fundamentally limited by the conditioning of the problem (6.6) [22]. The larger the *condition number* of the Hessian of the cost function in (6.6), slower is the convergence, and vice-versa. Additionally, these algorithms in [179–181] depend on feedback from the receivers. Even if feedback from the intended receivers are available, it is not likely to be available from the unwanted receivers where the nulls are formed.

In the next section, we propose our IPG-DB algorithm for addressing the above challenges in distributively solving the beamforming problem defined by (6.6). Specifically, the proposed algorithm is significantly robust to the *condition number* of (6.6) and does not require feedback from the receivers.

6.3 Proposed algorithm: Iteratively Pre-conditioned Gradient-descent for Distributed Beamforming

We consider a network architecture among the beamforming agents where the agents can communicate bidirectionally with a central *server*, as shown in Fig. 2.1. However, there is no inter-agent communication. The server can be an auxiliary node and it acts as a coordinator between the agents. Specifically, the agents collaborate with the server and exchange certain information with it to solve the optimization problem (6.6) using our proposed algorithm. However, the agents never share its excitation amplitude and phase, thereby maintaining autonomy. Solving

the problem (6.6) in the above architecture eliminates the necessity of feedback from the receivers, which will be evident from the algorithm described below. Henceforth, we will refer to the above-described system architecture as *server-agent network*.

We propose the following algorithm, coined Iteratively Pre-conditioned Gradient-descent for Distributed Beamforming (IPG-DB), for solving (6.6). Note that the idea of iterative pre-conditioning was proposed in Section 4.2 for solving convex empirical risk optimization problem in server-agent framework. However, the algorithm in Section 4.2 does not trivially apply to (6.6) due to the following reasons. First, as opposed to the data points being distributed in Section 4.2, the estimates are distributed in our problem (6.6). The proposed algorithm is built on top of the classical gradient-descent (GD) method. In GD, each agent updates its amplitude and phase estimates using the corresponding row of the gradient vector [180]. The iterative pre-conditioning technique in Section 4.2 pre-multiplies the gradient vector with an iterative matrix. The beamforming agents, in this case, update its estimates using the corresponding row of the product of pre-conditioning matrix and the gradient vector. However, computing any row of this matrix-vector product requires access to the full gradient vector, unlike the GD method. Thus, the algorithm in Section 4.2 does not trivially apply to (6.6) in distributed settings. Second, the problem (6.6) is non-convex, unlike Section 4.2.

To provide the motivation behind using the idea of iterative pre-conditioning in solving the beamforming problem, first, we consider the special case of strongly convex cost functions. Then, we generalize it to our problem (6.6), where the cost is non-convex. The well-known Newton's method [22] can be restated as a pre-conditioned gradient-descent method $x(t+1) = x(t) - \delta(t)K(t)g(t)$ wherein the pre-conditioner matrix $K(t)$ is the inverse Hessian $H(t)^{-1}$, and hence, iteration-dependent. Here, $x(t)$ denotes the estimate of a minimum point at iteration t

and $g(t)$ denotes the cost's gradient evaluated at $x(t)$. Although Newton's method converges fast at a *quadratic* rate, matrix inversion is vulnerable to process noise. We note that, practically, the beamforming problem is always impacted by channel noise. Now, to mitigate the effect of process noise, instead of computing $H(t)^{-1}$ at each iteration t, we proposed a scheme in Section 4.2 wherein the pre-conditioner matrix $K(t)$ is updated in such a way that it eventually converges to $H(t)^{-1}$. Specifically, it used a fixed-point iteration on the pre-conditioning matrix $K(t+1) = K(t) - \epsilon(t)\left(H(t)K(t) - I\right)$, where I denotes the identity matrix. For small enough step-size $\alpha(t)$, the pre-conditioner matrix $K(t)$ in the above equation converges to the fixed point K_* which satisfies $H_*K_* = I$, i.e., $K_* = H_*^{-1}$. As a result, the algorithm eventually converges to Newton's method and has *superlinear* convergence rate. However, the cost in (6.6) is non-convex. Therefore, the Hessian matrix in the beamforming problem is not necessarily invertible. Thus, in our proposed IPG-DB algorithm to solve the beamforming problem, we introduce a stabilization parameter $\beta(t) > -\lambda_{min}(H(t))$, where $\lambda_{min}(\cdot)$ denotes the smallest eigenvalue of a matrix. The idea is to drive the sequence of the iterative pre-conditioning matrices towards an "approximate inverse Hessian" $\left(H(t) + \beta(t)I\right)^{-1}$. In other words, for the non-convex beamforming problem, the fixed point K_* in the iteration of the pre-conditioning matrices satisfies $H_*K_* + \beta_*K_* = I$. Herein lies the intuition behind the idea of iterative pre-conditioning and our extension of it to solve the non-convex beamforming problem (6.6).

Below, we elaborate on how our aforementioned extension of the original iterative pre-conditioning scheme is applied to solve the beamforming problem in (6.6). The proposed IPG-DB algorithm is iterative, wherein each agent maintains an estimate of its own optimum amplitude and phase defined by (6.6) and updates it iteratively using the gradient of the cost function. To be precise, for each iteration $t = 0, 1, \ldots$, let $a_m(t) \in \mathbb{R}$ and $\alpha_m(t) \in \mathbb{R}$ denote the estimate of a_m^*

and α_m^*, respectively, maintained by agent $m \in \{1, \ldots, n\}$. Each of the initial estimates $a_m(0)$ and $\alpha_m(0)$ may be chosen arbitrarily from the search space $\mathbb{R}$. In each iteration $t \in \{0, 1, \ldots\}$, the server maintains a pre-conditioner matrix $K(t) \in \mathbb{R}^{2n \times 2n}$. The initial pre-conditioner matrix $K(0)$ is chosen arbitrarily from $\mathbb{R}^{2n \times 2n}$. Let, $x(t) = \big[a_1(t), \ldots, a_n(t), \alpha_1(t), \ldots, \alpha_n(t)\big]^T$ denote the combined $2n$-dimensional estimate of a minimum point in (6.6). Let, $g(t)$ and $H((t)$ respectively denote the gradient and the Hessian of the cost function defined in (6.6), evaluated at the current estimate $x(t)$. Then, in the centralized settings, the extension of the iterative pre-conditioning scheme in solving (6.6) results in the following two updates at each iteration t. First, the current estimate $x(t)$ is updated to

$$x(t+1) = x(t) - \delta K(t) g(t), \tag{6.7}$$

where $\delta > 0$ is the step-size. Next, the current pre-conditioning matrix $K(t)$ is updated to

$$K(t+1) = K(t) - \epsilon \left(H(t)K(t) + \beta K(t) - I \right), \tag{6.8}$$

where $\epsilon > 0$ is an algorithm parameter. In order to implement the steps (6.7)-(6.8) in a distributed manner, each agent $m \in \{1, \ldots, n\}$ requires the product of its corresponding m-th row of the matrix $K(t)$ and the full gradient vector $g(t)$. However, in the distributed settings, an agent does not have access to the full gradient vector, as the full gradient vector depends on the estimated amplitude and phase of all n agents. So, implementation of the centralized steps (6.7)-(6.8) in the distributed settings is not trivial. In the next subsection, we present the distributed counterpart of our proposed solution in (6.7)-(6.8), which is the IPG-DB algorithm. Specifically, for each

iteration t, the IPG-DB algorithm steps are as follows.

6.3.1 Steps in each iteration $t \geq 0$

In each iteration t, the IPG-DB algorithm comprises of five steps, executed collaboratively by the server and the agents. Before initiating the iterative process, the server chooses three non-negative scalar real-valued parameters ϵ, δ, β and broadcasts them along with the matrix $K(0)$ to all the agents.

- *Step 1 (local to each agent):* For each agent $m \in \{1, \ldots, n\}$ and each receiver $i \in \{1, \ldots, s\}$, we define

$$\zeta_{i,m} = kx_m \cos\theta_i + ky_m \sin\theta_i + kd_{i,m}. \tag{6.9}$$

For each receiver $i \in \{1, \ldots, s\}$, each agent $m \in \{1, \ldots, n\}$ computes two real-valued scalars $u_{i,m}(t)$ and $v_{i,m}(t)$ such that

$$u_{i,m}(t) = \frac{1}{d_{i,m}} \cos\big(\alpha_m(t) + \zeta_{i,m}\big), \tag{6.10}$$

$$v_{i,m}(t) = \frac{1}{d_{i,m}} \sin\big(\alpha_m(t) + \zeta_{i,m}\big), \tag{6.11}$$

and a complex-valued scalar $y_{i,m}(t)$ such that

$$y_{i,m}(t) = a_m(t)\left(u_{i,m}(t) + jv_{i,m}(t)\right). \tag{6.12}$$

- *Step 2 (agents → server):* Each agent $m \in \{1, \ldots, n\}$ sends the set of scalars

$$\{u_{i,m}(t), v_{i,m}(t), y_{i,m}(t),\ i \in \{1, \ldots, s\}\}$$

to the server. We let $k_j(t)$ denote the j-th row of the matrix $K(t)$, for $j \in \{1, \ldots, 2n\}$. If $t > 0$, then each agent $m \in \{1, \ldots, n\}$ also sends the updated m-th and the $(m+n)$-th row of the matrix $K(t)$, i.e., $k_m(t)$ and $k_{m+n}(t)$ to the server.

- *Step 3 (at the server):* The server computes the complex-valued scalar $y_s(t) = \sum_{m=1}^{n} y_{i,m}(t)$, for each $i \in \{1, \ldots, s\}$. Note that, $y_s(t)$ is the constructed array factor at receiver i in absence of multipath fading.

- *Step 4 (server → agents):* For each $i \in \{1, \ldots, s\}$, let

$$u_i(t) = \left[u_{m,1}(t), \ldots, u_{m,n}(t)\right]^T, v_i(t) = \left[v_{m,1}(t), \ldots, v_{m,n}(t)\right]^T, y_i(t) = \left[y_{m,1}(t), \ldots, y_{m,n}(t)\right]^T.$$

The server sends the set of scalars $\{y_s(t),\ i \in \{1, \ldots, s\}\}$, the set of n-dimensional vectors $\{u_i(t), v_i(t), y_i(t),\ i \in \{1, \ldots, s\}\}$, and the matrix $K(t)$ to each agent $m \in \{1, \ldots, n\}$.

- *Step 5 (local to each agent):* We let $\Re(\cdot)$ and $\Im(\cdot)$ respectively denote the real and imaginary component of a complex number. Each agent $m \in \{1, \ldots, n\}$ updates its

amplitude and phase estimates, $a_m(t)$ and $\alpha_m(t)$, according to

$$a_m(t+1) = a_m(t) - \delta k_m(t) \sum_{i=1}^{s} w_i \frac{\left|y_s(t)\right| - f_i}{\left|y_s(t)\right|} \times \left(\Re\left(y_s(t)\right) u_i(t) + \Im\left(y_s(t)\right) v_i(t)\right), \tag{6.13}$$

$$\alpha_m(t+1) = \alpha_m(t) - \delta k_{m+n}(t) \sum_{i=1}^{s} w_i \frac{\left|y_s(t)\right| - f_i}{\left|y_s(t)\right|} \times \left(-\Re\left(y_s(t)\right) \Im\left(y_i(t)\right) + \Im\left(y_s(t)\right) \Re\left(y_i(t)\right)\right). \tag{6.14}$$

The above two equations, combined for all agents n, is the distributed implementation of the estimate update equation (6.7) at each agent. Specifically, the summation terms on the R.H.S. above is equal to the gradient of the aggregate cost (6.6) with respect to the current estimates $a_m(t)$ and $\alpha_m(t)$ of each agent m.

In the same step, each agent $m \in \{1, \ldots, n\}$ updates the m-th and the $(m+n)$-th row of the matrix $K(t)$ as follows. We let I_{2n} denote the $(2n \times 2n)$-dimensional identity matrix. We let $H_j(t)$ denote the j-th row of the Hessian matrix of the cost defined in (6.6). Then,

$$k_j(t+1) = k_j(t) - \epsilon \left(H_j(t) K(t) + \beta k_j(t) - I_{2n,j}\right), \tag{6.15}$$

for each column $j \in \{m, m+n\}$. Here, $I_{2n,j}$ denote the j-th row of the $(2n \times 2n)$-dimensional identity matrix. Note that, $H_m(t)$ and $H_{m+n}(t)$ can be locally computed by each agent m. Thus, (6.15) above, combined for all agents n, is the distributed implementation of the pre-conditioner update (6.8).

6.4 Experimental results

In this section, we present simulation results comparing the proposed IPG-DB algorithm with the existing gradient-descent (GD) method, such as in [180].

Synthetic environment: First, we consider a synthetic environment. There are $n = 19$ beamforming agents, with fixed positions in two-dimensional space as shown in Figure 6.1. $s = 49$ receivers are located at a radial distance $\rho = 10\lambda$ from the origin, uniformly spaced along the angular direction θ from 0 to 2π radians. The desired array factor amplitudes at these 49 receivers are shown by the dashed line in Figure 6.2 in decibels (dB). The carrier frequency is 40 MHz. We apply the classical GD and the proposed IPG-DB algorithm for solving this beamforming problem. The agents' amplitude and phase estimates in both of these algorithms are identically initialized as $a_m(0) = 10$ and $\alpha_m(0) = 10$ for each agent m. For an unbiased comparison, the parameters in both algorithms are set at their optimum values so that the individual algorithms attains its best convergence rate. Specifically, the step-size parameter $\delta(t) = \frac{1}{\lambda_{max}(H(t))}$ for GD method, where $\lambda_{max}(\cdot)$ denotes the largest eigenvalue of a square matrix. For IPG-DB, we set the parameters $\epsilon(t) = \frac{1}{\lambda_{max}(H(t))+\beta}$, $\beta = 0.1$, and $\delta = 1$. We artificially add narrow-band fading in each of the channels. The fading parameter $\gamma_{m,i}$ is generated independently at each iteration of the algorithm from the Rayleigh distribution. Note that the algorithms do not use the value of $\gamma_{m,i}$.

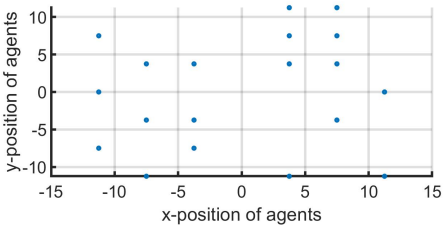

Figure 6.1: Cartesian coordinates of $n = 19$ beamforming agents in the synthetic problem.

The beampatterns generated by the GD and the IPG-DB algorithms are compared in Figure 6.2-6.3. We observe that the IPG-Db algorithm converges to a neighborhood of the desired pattern within a significantly smaller number of iterations t than the GD method.

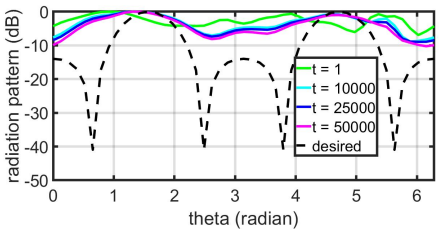

Figure 6.2: Array factors constructed by the GD algorithm for solving the synthetic beamforming problem, after different number of iterations t.

Los Angeles urban environment: Next, we consider a realistic urban environment in a section of Los Angeles (ref. Fig. 6.4). There are $n = 5$ agents and $s = 3$ receivers, each with their fixed positions in two-dimensional space as shown in Figure 6.6. The desired array factor amplitudes at these 3 receivers are shown by the blue colored bars in Figure 6.7. Two of them are client receivers, indicated in brown, and the third is a null receiver, indicated in red. The carrier frequency is 40 MHz. We apply the proposed IPG-DB algorithm for solving this problem.

The agents' amplitude and phase estimates are initialized randomly from the *uniform* distribution in $(0, 10)$. The parameters $\epsilon(t)$ and δ are chosen as before. The third parameter β is set initially at 0 and reduced to 0.1 after 3000 iterations. To test the performance of IPG in this problem, we use a dataset generated from physics-based simulations using EM.CUBE, provided by the U.S. Army Research Laboratory (ref. Fig. 6.5).

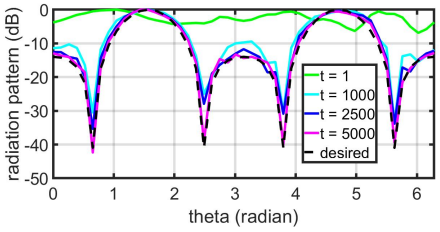

Figure 6.3: Array factors $\{|AF_i|\,,\ i = 1, \ldots, 49\}$ constructed by the IPG-DB algorithm for solving the synthetic beamforming problem, after different number of iterations t.

After 25000 iterations of the proposed IPG-DB algorithm, the array factors at each of the three receivers are shown in Fig. 6.7. We observe that the obtained array factors using IPG-DB are close to the desired array factor values.

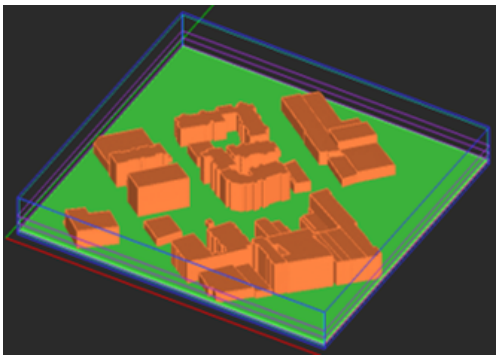

Figure 6.5: CAD model of the section of Los Angeles from Figure 6.4.

6.5 Summary

We have considered the multi-agent beamforming problem with the agents' objective being able to match the desired beampattern, which includes target beams at intended receivers and nulls at unintended receivers. Majority of the existing beamforming approaches rely on feedback from the receivers. We have proposed a distributed iterative pre-conditioning technique in the server-agent architecture of the beamforming agents. The proposed algorithm is robust to the condition number of the problem and noisy channels, and hence, can reach a satisfactory neighborhood of the desired pattern in a fewer number of iterations. Future work involves incorporating channel estimates or feedback in the proposed algorithm to further improve its robustness against fading and utilize the proposed algorithm in the joint optimization framework of position and excitation of the agents.

Figure 6.4: A section of Los Angeles used as the environment for beamforming problem.

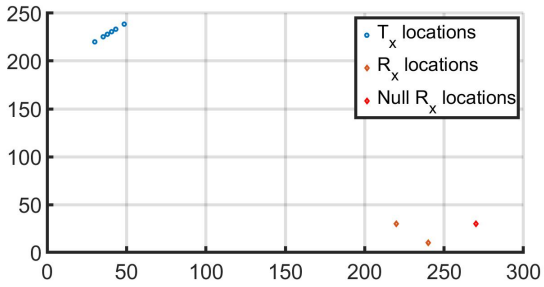

Figure 6.6: Cartesian coordinates of $n = 5$ beamforming agents in the LA urban environment problem.

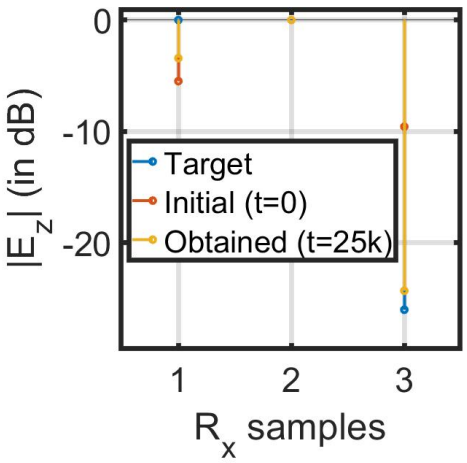

Figure 6.7: Array factors for the electric field along the z-direction constructed by the IPG-DB algorithm for solving the beamforming problem in LA urban environment, initially ($t = 0$) and after $t = 25000$ iterations.

Chapter 7: Nonlinear Observer

7.1 Introduction

We consider the classical observer design problem for a nonlinear discrete-time system with sampled noisy measurements. Specifically, let $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}^p$, and $\eta_k \in \mathbb{R}^p$ respectively denote the state of the dynamical system, the input to the system, the observed measurement or output of the system, and the measurement noise vector at the k^{th} sampling instant. Then, for each $k \in \mathbb{N}$, the dynamical system is described in the state-space form

$$x_{k+1} = F(x_k, u_k), \tag{7.1}$$

$$y_k = h(x_k) + \eta_k, \tag{7.2}$$

where the state-dynamics function $F : (\mathbb{R}^n, \mathbb{R}^m) \to R^n$ and the output function $h : \mathbb{R}^n \to \mathbb{R}^p$ are nonlinear. For each $k \geq N$, we let $Y_k = [y_{k-N+1}, \ldots, y_k]^T \in \mathbb{R}^{Np}$ and $U_k = [u_{k-N+1}, \ldots, u_{k-1}]^T \in \mathbb{R}^{(N-1)m}$ denote the column vectors of the past $N \in \mathbb{N}$ consecutive measurements and inputs, respectively, at the k^{th} sampling instant. In general, the number p of measured variables or sensors is less than the state dimension (n), due to limited budget or physical constraints, leading to the natural question of estimating the full state vector from the limited dimension of measurements, which is essentially the observer design problem. In other words, our aim is to design a discrete-time dynamical system to estimate the sampled system trajectory using a moving window of the past N measurements and inputs so that the output $\hat{x}_k$ of the designed system is an estimate of the

system states x_k in the sense that $\lim_{k\to\infty}\|x_k - \hat{x}_k\| = 0$. We assume that the functions F and h are known. Such an observer is required in several applications including control, monitoring, reconstruction, and modeling. While observers are essential in various applications, constructing a nonlinear observer is in general challenging.

7.1.1 Prior works

Generally speaking, the existing nonlinear observers belongs to the class of extended Luenberger observers with state-dependent gains [186]. Depending on the mathematical technique used in observer formulation, the existing observers can be categorized into several classes. We briefly review the classes and the notable works in each class below.

The exact linearization-based observers [187–190] use an invertible functional to transform the observer into linear coordinates. The advantage of this approach is that the coefficient of exponential rate in the error can be tuned constructively. However, a key step in this approach is solving linear functional equations in discrete-time or linear partial differential equations in continuous-time. The proposed observers in this category approximately solve such equations using a power-series solution, truncating upto certain order of terms. Thus, the computational complexity of these observers increases with the dimension n and the truncation order. Our proposed observer does not rely on the Taylor series approximation.

A majority of the existing observers [191–201] are built for the system (7.1)-(7.2) having some special structure in F and h. For example, the observers proposed in [194, 199–201] require (7.1)-(7.2) in the observer canonical form. While this form does not lose generality in the case of observable linear systems, for nonlinear systems only input-affine systems have been shown to be only the locally equivalent to observer canonical form by a change of coordinates [202].

The observers in [197] require the output mapping h to be linear and assume the nonlinearity in F to be either quadratically bounded or one-sided Lipschitz. The observers in [191,192,198] require the output mapping h to be linear. The observers in [189,193,195,196] require the linearity in F to be upper-triangular and h to be linear. Compared to these vast literature, we do not restrict our observer to any particular structures in F, h.

Another class is the high-gain observers [200,203–206] which involves computing a robust inverse of the nonlinear observability matrix. The observer in [200] requires (7.1)-(7.2) to be in the observer canonical form. The observers in [206] combine the high-gain formulation in [203] with a robust inversion strategy, without explicitly computing the inverse observability matrix. However, [206] requires continuous-time measurements of the system. A unified framework of high-gain observers for uncertain systems with continuous-time measurements is presented in [205]. We use only sampled measurements (7.2), which is more practical. The qDES observer framework developed in [207] in presence of measurement noise includes the aforementioned exact linearization-based and high-gain observers among others. However, constructing a qDES observer relies on choosing suitable storage and class $\mathcal{K}$ functions so that certain conditions are satisfied [207].

Moving horizon estimation (MHE) techniques have also been used in constructing nonlinear observers [208–211]. This class of observers minimizes a cost function at each step, where the cost comprises an weighted error in estimating the measurements, an weighted error in state estimation, and an approximate arrival cost summarizing the past history. However, for general nonlinear dynamics, no constructive method for choosing the arrival cost is known [208]. A quadratic arrival cost is proposed in [210], in which case the stability of the observer depends on the choice of the corresponding weight matrix. The sufficient condition in [210] on choosing

the weight matrix to guarantee stability is not constructive. The one-step ahead MHE algorithm in [210] requires computing the inverse of the Hessian at each sampling instant, which is prone to instability in presence of measurement noise. We do not compute any matrix inverse in our algorithm. Additionally, the MHE observers in [209, 210] solve a nonlinear programming (NLP) at each step, and hence, the efficiency and computational effort of the observer depends on the chosen NLP solver. Thus, instead of solving the full NLP at each step, three solvers based on gradient-descent (GMHE), conjugate gradient (CGMHE), and Newton's method (NMHE) are proposed in [211], and shown to have much less computational cost than a full NLP optimizer, at the price of slight decay in performance.

Our proposed observer belongs to the class of Newton and approximate Newton observers [212–215]. Based on the Newton observer in [212], the finite-difference approximation-based observer in [213] and the GMRES-based observer in [214] are proposed for the case when, instead of (7.1), the continuous-time dynamics is known. A Broyden's method-based quasi-Newton observer has been proposed in [212]. However, these Newton-type observers result in large steady-state error, even instability, when subjected to measurement noise. We propose an approximate Newton observer which is more robust to measurement noise than these methods.

When, instead of (7.1), the continuous-time dynamics is known, an observer based on Carleman linearization-based discretization of the continuous time-dynamics is proposed in [216]. As shown in [216], the performance of the observer improves with the order of truncation in the discretization, but at the expense of computing higher order coefficients which adds to the computational cost. In our method, when the continuous-time dynamics is known, we use the simple first-order Euler discretization.

Among other related works, the observer in [217] is a partial state observer. The differential-

algebraic equation-based observer in [218], the observability function-based observer in [219], and the observer in [220] require continuous-time measurements of the system. We use only sampled measurements, which is more practical. An observer for nonlinear discrete-time dynamics based on linear parameter varying optimization is proposed in [221]. The extended Kalman filter (EKF) [222] has also been implemented in practice for nonlinear state estimation in several applications. However, being reliant on linearization, the EKF has serious difficulties when the dynamics F is strongly nonlinear [186].

7.1.2 Summary of our contributions

While there exists a vast literature on nonlinear observers for (7.1)-(7.2), the majority of them consider an Utopian setting where the system is noise-free. The performance of most of the aforementioned observers in presence of measurement noise has not been investigated. Process and measurement noise in the system (7.1)-(7.2) have been considered in the observers in [197, 209]. However, the observer in [197] is only applicable to a certain structure in (7.1)-(7.2), as mentioned earlier, and the limitation of MHE observer in [209] is discussed in Section 7.1.1.

We propose an observer for estimating the full states of the discrete-time nonlinear system defined by (7.1)-(7.2), from a moving window of the past N measurements and inputs. Our observer is built upon the centralized counterpart of the iteratively pre-conditioned gradient-descent (IPG) algorithm that was proposed for solving distributed optimization problems in Section 4.2. The proposed observer leverages the robustness of the general-purpose IPG optimizer towards system noise, leading to a Newton-type nonlinear observer that is robust to measurement noise w_k in (7.2). This observer does not require any specific structure in the nonlinear functions F, h. Our observer is developed in Section 7.3. The relationship between the proposed observer

and EKF is discussed in Section 7.3.3. The empirical results, in Section 7.4, support our hypothesis that the proposed observer has an improved robustness against measurement noise than the well-known nonlinear observers that are applicable to general nonlinear functions F, h.

7.2 Problem formulation

To mathematically formulate the observer problem in [212], we introduce the following notation. For simplicity, we define $F^{u_k}(x_k) = F(x_k, u_k)$ for each $k \in \mathbb{N}$ and let $\circ$ denote the composition of functions. Recall the definition of the past N measurement vector $Y_k = [y_{k-N+1}, \ldots, y_k]^T \in \mathbb{R}^{Np}$ from Section 2.1. Then, from (7.1)-(7.2), for each $k \geq N$,

$$Y_k = \begin{bmatrix} h(x_{k-N+1}) \\ \vdots \\ h \circ F^{u_{k-1}} \circ \ldots \circ F^{u_{k-N+1}}(x_{k-N+1}) \end{bmatrix} + \begin{bmatrix} \eta_{k-N+1} \\ \vdots \\ \eta_k \end{bmatrix}.$$

We let $H^{U_k}(x_{k-N+1})$ and W_k respectively denote the first and the second vector on the R.H.S. above, and obtain that

$$Y_k = H^{U_k}(x_{k-N+1}) + W_k,\ k \geq N. \tag{7.3}$$

Here, $H^{U_k} : \mathbb{R}^n \to \mathbb{R}^{Np}$ is known as the *observability function* of (7.1)-(7.2). The observer problem involves solving for x_{k-N+1} from the above set of nonlinear equations online, followed by propagating the obtained solution of x_{k-N+1} forward by N sampling instants using the state-dynamics F, defined in (7.1), to obtain an estimate of x_k for each $k \geq N$.

Newton observer [212] solves the above problem by solving the nonlinear equations (7.3) at each sampling instant $k \geq N$ using the well-known Newton optimizer [22]. For each $k \geq N$, it computes $d+1$ estimates $\{w_k^i \in \mathbb{R}^n : i = 0, \ldots, d\}$ of x_{k-N+1}. Specifically, the iterative

Newton optimizer is applied d times, $i = 0, \ldots, d-1$, for solving (7.3) at each k:

$$w_k^{i+1} = w_k^i - \left(\frac{\partial H^{U_k}}{\partial w}(w_k^i)\right)^{\dagger} \left(H^{U_k}(w_k^i) - Y_k\right), \tag{7.4}$$

where $(\cdot)^{\dagger}$ denotes the pseudo-inverse. From w_k^d, after the end of the above d Newton iterations, the estimate $\hat{x}_k$ of the true state x_k is obtained by forward propagating w_k^d as

$$\hat{x}_k = F^{u_{k-1}} \circ \ldots \circ F^{u_{k-N+1}}(w_k^d). \tag{7.5}$$

The initial condition w_{k+1}^0 for the Newton iterations in the next sampling period $k+1$ is set by propagating w_k^d forward by one sampling instant as

$$w_{k+1}^0 = F^{u_{k-N+1}}(w_k^d). \tag{7.6}$$

Under certain assumptions, if the parameter d is large enough and the initial estimate w_N^0 is chosen close enough to the true initial state x_1, then the Newton observer (7.4)-(7.6) exponentially converges to x_k when the measurement noise $\eta_k = 0_p$ [212]. However, in presence of measurement noise, the Newton observer can display instability due to inverse computation in (7.4). In the next section, we propose an observer based on the IPG method (ref. Section 4.2) to alleviate the impact of measurement noise.

7.3 Proposed observer

First, we provide the intuition behind using the idea of iterative pre-conditioning (ref. Section 4.2) in solving the observer problem. The well-known Newton's method [22] solves the minimization problem $x^* = \arg\min_{x \in \mathbb{R}^n} f(x)$ of a strongly convex cost function $f : \mathbb{R}^n \to \mathbb{R}$. We let w^i denote the estimate of a minimum point of f, maintained by Newton's method at

iteration $i \in \mathbb{N}$. We let g^i and H^i respectively denote the cost's gradient and Hessian evaluated at w^i. The, Newton's method can be restated as a pre-conditioned gradient-descent method $w^{i+1} = w^i - \delta^i K^i g^i$ wherein the pre-conditioner matrix K^i is the inverse Hessian $(H^i)^{-1}$, and hence, iteration-dependent. Although Newton's method converges fast at a *quadratic* rate, matrix inversion is vulnerable to system noise. Now, to mitigate the effect of noise, instead of computing $(H^i)^{-1}$ at each iteration i, the IPG algorithm updates the pre-conditioner matrix K^i in such a way that it eventually converges to $(H^i)^{-1}$. Specifically, IPG uses a fixed-point iteration on the pre-conditioning matrix $K^{i+1} = K^i - \alpha^i \left(H^i K^i - I\right)$, where I denotes the identity matrix. For small enough step-size α^i, the pre-conditioner matrix K^i in the above equation converges to the fixed point K^* which satisfies $H^* K^* = I$, i.e., $K^* = (H^*)^{-1}$. As a result, the IPG algorithm eventually converges to the Newton's method and has *superlinear* convergence rate. IPG does not involve any inverse computation, and also the pre-conditioning matrix is not required to preserve symmetric positive definiteness, unlike the quasi-Newton methods. Thus, IPG is robust against system noise when compared with other general-purpose optimizers, as shown by simulation in Section 4.3. Consequently, our proposed observer relies on the IPG algorithm to utilize its robustness.

We propose to solve the noisy nonlinear equations (7.3) at each sampling instant $k \geq N$ in the observer problem using an algorithm similar to the aforementioned IPG algorithm. In other words, we replace the step (7.4) in Newton observer with an IPG-like iteration as presented below.

The proposed observer is iterative, wherein at each sampling instant $k \geq N$, it applies d iterations indexed by $i = 0, \ldots, d-1$. At each iteration i and sampling instant k, it maintains an n-dimensional estimate w_k^i of the state x_{k-N+1}, and an n-dimensional estimate $\hat{x}_k$ of the actual

state x_k, and an $n \times n$-dimensional matrix K_k^{i+1}. Before initiating the observer, it chooses an initial estimate $w_N^0 \in \mathbb{R}^n$ and an initial pre-conditioning matrix $K_N^0 \in \mathbb{R}^{n\times n}$. The algorithm parameter $d \in \mathbb{N}$ is also chosen. At each $k \geq N$, the observer comprises the following three steps.

7.3.1 Steps in each sampling instant $k \geq N$

Step-I: The observer executes d iterations to update the current estimate w_k^0 and the pre-conditioner K_k^0. Before initiating these iterations, two step-size sequences $\{\alpha^i, \delta^i : i = 0, \ldots, d-1\}$ are chosen. Their specific values are described in Section 7.3.2. At each iteration $i = 0, \ldots, d-1$, the pre-conditioning matrix K_k^i and the estimate w_k^i are updated according to whether the set of equations (7.3) is square. If $n = Np$, then in the same iteration i,

$$K_k^{i+1} = K_k^i - \alpha^i \left(\frac{\partial H^{U_k}}{\partial w}(w_k^i) K_k^i - I \right), \tag{7.7}$$

$$w_k^{i+1} = w_k^i - \delta^i K_k^i \left(H^{U_k}(w_k^i) - Y_k \right), \tag{7.8}$$

and, if $n \neq Np$, then in the same iteration i,

$$K_k^{i+1} = K_k^i - \alpha^i \left(\left(\frac{\partial H^{U_k}}{\partial w}(w_k^i) \right)^T \frac{\partial H^{U_k}}{\partial w}(w_k^i) K_k^i - I \right), \tag{7.9}$$

$$w_k^{i+1} = w_k^i - \delta^i K_k^i \left(\frac{\partial H^{U_k}}{\partial w}(w_k^i) \right)^T \left(H^{U_k}(w_k^i) - Y_k \right). \tag{7.10}$$

Here I denotes the $n \times n$-dimensional identity matrix.

Step-II: Using w_k^d, after the end of the above d IPG iterations in Step-I, the estimate $\hat{x}_k$ of the true state x_k is obtained by forward propagating $(N-1)$ times the vector w_k^d, as described in (7.5).

Step-III: The final step involves setting the initial estimate and the initial pre-conditioning matrix for the next sampling instant, based on w_k^d and K_k^d obtained after the d-th IPG iteration in Step-I. Specifically, w_{k+1}^0 is obtained according to (7.6) and K_{k+1}^0 is set as

$$K_{k+1}^0 = K_k^d. \tag{7.11}$$

Note that the computational complexity of each iteration within the proposed observer is same as the Newton observer.

7.3.2 Step-size selection

Now, we present our argument for choosing the step-sizes α^i, δ^i in (7.9)-(7.10). Since the proposed observer applies IPG iterations at each sampling instant $k \geq N$, the observer's convergence depends on the convergence result of the IPG algorithm which we review below. We again consider the minimization problem $x^* = \arg\min_{x\in\mathbb{R}^n} f(x)$ of an l-strongly convex cost function $f : \mathbb{R}^n \to \mathbb{R}$. Note that the Hessian of f is then full-rank and invertible. We let η denote the induced 2-norm $\eta = \left\|(H^*)^{-1}\right\|$. Assume that the gradient and Hessian of f are Lipschitz continuous with Lipschitz coefficient L and γ respectively. We define $\rho = \max_{i\geq 0} 1 - \alpha^i l$. If the initial estimate w^0 and initial pre-conditioning matrix K^0 are chosen such that $\frac{\eta\gamma}{2}\left\|w^0 - x^*\right\| + L\left\|K^0 - (H^*)^{-1}\right\| \leq \frac{1}{2r}$, where $r \in \left(1,\ \frac{1}{\rho}\right)$, $\alpha^i < \min\left\{\frac{1}{L},\ \frac{r^{(i)}(1-r\rho)}{2L}\right\}$, and $\delta^i = 1$, then the IPG algorithm has a guaranteed local superlinear convergence 4.1. Here, $(\cdot)^{(i)}$ denotes the i-th power. We further note that $\frac{1}{L} < \frac{r^{(i)}(1-r\rho)}{2L} \iff r^{(i)} > \frac{2}{1-r\rho}$. Since $r > 1$, there exists $T < \infty$ such that above condition is necessarily satisfied for all $i \geq T$. So, for all $i \geq T$, the aforementioned condition on α^i is equivalent to $\alpha^i < \frac{1}{L}$.

Hence, to ensure that the IPG iterations (7.7)-(7.10) nested within each sampling instant

$k \geq N$ is a contraction, we select the step-size parameters as follows. Assume that the standard *uniform N-observability* and the *N-observability rank* conditions hold true, as in the Newton observer paper [212]. These conditions imply that the Jacobian $\frac{\partial H^{U_k}}{\partial w}$ is full-rank, which is assumed in [213, 214]. Note that, this is similar to the Hessian of f being full-rank in the minimization problem of strongly convex cost f. As in [212–214], we assume that the Jacobian is uniformly bounded, i.e., $\left\| \frac{\partial H^{U_k}}{\partial w} \right\| \leq L$ when the system of equations (7.3) is square, and $\left\| \left(\frac{\partial H^{U_k}}{\partial w} \right)^T \frac{\partial H^{U_k}}{\partial w} \right\| \leq L$ otherwise. Owing to the mean value theorem, the Jacobian being bounded by L implies that H^{U_k} is L-Lipschitz continuous. Again, this is similar to the the gradient of f being Lipschitz in the minimization problem of f. Thus, following the convergence result of IPG iterations as mentioned earlier, we set

$$\alpha^i < \frac{1}{L}, \text{ and } \delta^i = 1,\ i = 0, \ldots, d-1, \tag{7.12}$$

in (7.7)-(7.10) of our proposed observer's Step-I.

The convergence of the proposed observer can be proved following the proof sketch of Newton observer [212, Theorem 3.2]. Following 4.1, at each k, the IPG iterations (7.7)-(7.10) with the above step-size choices and local initialization as described above, solves the underlying nonlinear equations (7.3) with *superlinear* convergence as $d \to \infty$. So, the idea is to find a minimum number of nested IPG iterations d_{min} within each k, so that F applied to the final estimate w_k^d after d iterations (ref. (7.6)) and $\left\| K_k^d - \frac{\partial H^{U_k}}{\partial w}(\hat{x}_{k-N+1})^{-1} \right\| = \left\| K_{k+1}^0 - \frac{\partial H^{U_k}}{\partial w}(\hat{x}_{k-N+1})^{-1} \right\|$ (ref. (7.11)) are such that $\left\| w_{k+1}^0 - \hat{x}_{k-N+2} \right\|$ and $\left\| K_{k+1}^0 - \frac{\partial H^{U_{k+1}}}{\partial w}(\hat{x}_{k-N+2})^{-1} \right\|$ are small enough, satisfying the local initial condition required to invoke 4.1 at the next instant $(k+1)$. The detailed proof based on this outline is left for a future work.

7.3.3 Relation with extended Kalman filter

As mentioned in Section 7.3.2, the IPG iteration for minimizing a strongly convex cost function locally converges to the Newton's method [22], when the initial estimate and the initial pre-conditioning matrix are chosen close enough to the true mimimum point and the inverse Hessian at the true minimum point 4.1. Similarly, the proposed observer (7.5)-(7.11) with stepsize (7.12) is expected to converge to Newton observer as the sampling instant $k \to \infty$ and the iterations $i \to \infty$ within each k, if w_N^0 and K_N^0 are suitably initialized, in the sense that $\lim_{k,i\to\infty} K_k^i = \left(\frac{\partial H^{U_k}}{\partial w}(x_{k-N+1})\right)^{-1}$, if $n = Np$, else $\lim_{k,i\to\infty} K_k^i \left(\frac{\partial H^{U_k}}{\partial w}(w_k^i)\right)^T = \left(\frac{\partial H^{U_k}}{\partial w}(x_{k-N+1})\right)^{\dagger}$.

Now, the Newton observer with $d = 1$ iteration within each sampling instant has been shown to be exactly the extended Kalman filter for the extended dynamical system

$$x_{k+1} = F(x_k) + q\zeta_k,\ \overline{y}_k = H^{U_k}(x_k) + \epsilon\eta_k,$$

in the limit of measurement noise covariance $\epsilon \to 0$ [212]. The extended output mapping H^{U_k} is defined in (7.3). So, we hypothesize that the proposed observer (7.5)-(7.11) with the stepsize (7.12) and suitable range of w_N^0 and K_N^0 for the dynamical system (7.1)-(7.2) would converge to the EKF with measurement noise covariance $\epsilon \to 0$ for the above extended dynamical system in the limit $k, i \to \infty$. However, this relation with EKF will be confirmed by a rigorous convergence analysis of the proposed observer, which is left for a future work.

7.4 Experimental results

In this section, we present empirical results, comparing robustness of the proposed IPG observer against measurement noise with prominent observers that are applicable to general nonlinear plant (7.1) with sampled measurements (7.2). Specifically, these observers are: Newton

observer [212], Broyden observer [212], Levenberg-Marquardt (LM) [22] optimizer-based observer, GMRES observer [214], extended Kalman filter (EKF) [222], gradient-descent moving horizon estimation (GMHE) [211], conjugate gradient-descent moving horizon estimation (CGMHE) [211], Newton moving horizon estimation (NMHE) [211], pseudo-Newton observer [215], and linear parameter varying observer (LPV) [221]. We consider two example systems where we compare with all the aforementioned observers except LPV. Then, in the third example, we compare our IPG observer with LPV observer only.

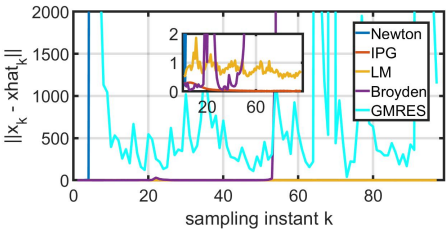

Figure 7.1: State estimation error of different observers for bio-reactor system (7.13)-(7.16) with measurement noise's standard deviation 0.01.

First, we consider the mixed-culture bio-reactor system from [212]. The system dynamics in discrete-time is

$$x_{(1),k+1} = x_{(1),k} + h\left(\frac{0.4S(x_{(1),k}, x_{(2),k})}{0.05 + S(x_{(1),k}, x_{(2),k})}x_{(1),k} - 0.3x_{(1),k}\right), \tag{7.13}$$

$$x_{(2),k+1} = x_{(2),k} + h\left(\frac{0.01S(x_{(1),k}, x_{(2),k})x_{(2),k}}{0.05 + S(x_{(1),k}, x_{(2),k})(0.02 + x_{(3),k})} - 0.3x_{(2),k}\right), \tag{7.14}$$

$$x_{(3),k+1} = x_{(3),k} + h\left(-0.5x_{(1),k}x_{(3),k} - 0.3x_{(3),k} + 0.3 \times 0.0067\right), \tag{7.15}$$

$$y_k = x_{(1),k} + x_{(2),k} + \eta_k, \tag{7.16}$$

where $S(x_1, x_2) = 2 - 5x_1 - 6.667x_2$, sampling period $h = 1$, and the true initial state $[0.2, 0.02, 0.005]^T$. The measurement noise η_k is generated from zero mean Gaussian distribution with two cases of standard deviation: 0.01 and 0.001. The Newton, IPG, LM, Broyden, GMRES, GMHE, CGMHE, NMHE, and pseudo-Newton observers are implemented with the window

length $N = 3$ and initial state estimate $[0.02, 0.2, 0.015]^T$, following the implementation in [212]. The number of nested iterations for Newton, IPG, LM, GMRES, NMHE, and pseudo-Newton is $d = 2$ within each sampling instant. Since Broyden, GMHE, and CGMHE have per-iteration complexity n times smaller than the other algorithms above, for Broyden, GMHE and CGMHE we set $d = 6$ for a fair amount of computational effort at each sampling instant across all algorithms. The initial pre-conditioner for IPG observer is set at $10^{-4}I_3$, tuned from $\{10^pI_3 : p = -4, -3, \ldots, 4\}$ and the parameter α^i is set at a constant value 0.01, tuned from $\{0.1, 0.01, 0.001\}$. The initial approximate of the Hessian in Broyden observer is set at $0.1I_3$. In GMRES, the parameters are set at $\rho = 10^{-7}$ and $\eta = 0.01$. The parameter α in GMHE and CGMHE is set at 0.001, tuned from $\{0.1, 0.01, 0.001\}$, for their fastest convergence. The initial state estimate in EKF is same as the other algorithms above. The initial error covariance and the process noise covariance matrix are both set at $0.1I_3$, tuned from $\{10^pI_3 : p = -3, \ldots, 3\}$. The measurement noise variance in EKF is set at 100, tuned from $\{100, 10, 0.1, 0.01, 0.001\}$.

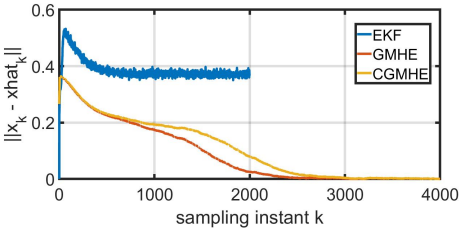

Figure 7.2: State estimation error of different observers for bio-reactor system (7.13)-(7.16) with measurement noise's standard deviation 0.01.

Next, we consider the inverted pendulum-cart system from [223]. The system dynamics in

discrete-time is

$$x_{(1),k+1} = x_{(1),k} + hx_{(2),k}, \tag{7.17}$$

$$x_{(2),k+1} = x_{(2),k} + h\left(\frac{100x_{(4),k}^2 \sin(x_{(3),k}) - 25g\sin(2x_{(3),k}) + 5}{2 + 50(1 - \cos(x_{(3),k})^2}\right), \tag{7.18}$$

$$x_{(3),k+1} = x_{(3),k} + hx_{(4),k}, \tag{7.19}$$

$$x_{(4),k+1} = x_{(4),k} - h\left(\frac{50x_{(4),k}^2 \sin(2x_{(3),k}) - 52g\sin(x_{(3),k}) + 5\cos(x_{(3),k})}{4 + 100(1 - \cos(x_{(3),k})^2}\right), \tag{7.20}$$

$$y_k = [x_{(1),k}, x_{(3),k}]^T + \eta_k, \tag{7.21}$$

where $g = 9.8$, $h = 10^{-3}$, and the true initial state $[0, 1, \pi, 1]^T$. The measurement noise η_k is generated from zero mean Gaussian distribution with two cases of diagonal covariance with standard deviation: 0.1 and 0.01. Since the system of equations (7.3) is not square in this case, the Broyden, GMRES, and pseudo-Newton observers are not applicable here. We implement Newton, IPG, LM, GMHE, CGMHE, and NMHE with the window length $N = 4$ and initial state estimate $[5, 5, 5, 5]^T$. For Newton, IPG, LM, and NMHE we set $d = 2$ within each sampling instant. Since GMHE and CGMHE have per-iteration complexity n times smaller than the other algorithms above, for GMHE and CGMHE we set $d = 8$ for a fair amount of computational effort across all algorithms. The initial pre-conditioner for IPG observer is set at $10^4 I_4$, tuned from $\{10^p I_4 : p = -4, -3, \ldots, 4\}$ and the parameter α^i is set at a constant value 0.1, tuned from $\{0.1, 0.01, 0.001\}$. The parameter α in

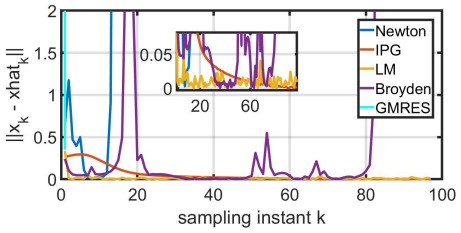

Figure 7.3: State estimation error of different observers for bio-reactor system (7.13)-(7.16) with measurement noise's standard deviation 0.001.

GMHE and CGMHE is set at 0.1, tuned from $\{0.1, 0.01, 0.001\}$, for their fastest convergence. The initial state estimate in EKF is same as the other algorithms above. The initial error covariance and the process noise covariance matrix are both set at $0.1I_4$, tuned from $\{10^p I_4 : p = -3, \ldots, 3\}$. The measurement noise covariance matrix in EKF is set at $0.1I_2$, tuned from $\{10^p I_2 : p = -3, \ldots, 3\}$.

We define the *steady-state estimation error* as the mean of $\|x_k - \hat{x}_k\|$ over a period of 20 consecutive sampling instants k after the observer has reached steady-state. For the bio-reactor system, the steady-state estimation errors of IPG observer are approximately 0.03 and 0.002, respectively for standard deviation 0.01 and 0.001 in the measurement noise, which are smaller than the other observers except for GMHE and CGMHE (ref. Fig. 7.1-7.4). The steady-state estimation errors of GMHE and CGMHE are approximately 0.001 and 0.0004, respectively for standard deviation 0.01 and 0.001 in the measurement noise. However, with the identical computational load per sampling instant, GMHE and CGMHE attain these steady-state errors much slower than the IPG observer (ref. Fig. 7.1-7.4). From Fig. 7.1 and Fig. 7.3, the Newton, Broyden, and GMRES observers are highly unstable. We further report that the NMHE and the pseudo-Newton observer diverge from the first sampling instant, and hence, are not shown in the plots.

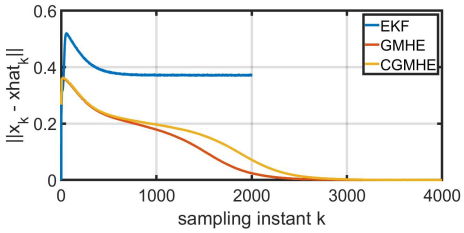

Figure 7.4: State estimation error of different observers for bio-reactor system (7.13)-(7.16) with measurement noise's standard deviation 0.001.

For the inverted pendulum-cart system, the steady-state estimation errors of IPG observer are approximately 4 and 0.75, respectively for standard deviation 0.1 and 0.01 in the measurement noise, which are smaller than the other observers except for GMHE and CGMHE (ref. Fig. 7.5-

7.8). The steady-state estimation errors of GMHE and CGMHE are approximately 2, for standard deviation 0.1 and 0.01 in the measurement noise. Thus, IPG has a smaller steady-state error than all other observers for standard deviation 0.01. For standard deviation 0.1, only GMHE and CGMHE have a smaller steady-state error than IPG observer. However, with the identical computational load per sampling instant, GMHE and CGMHE attain these steady-state errors much slower than IPG (ref. Fig. 7.5-7.8). Also, NMHE diverges from the first sampling instant, and hence, not shown in the plots.

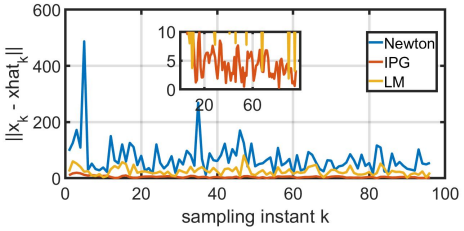

Figure 7.5: State estimation error of different observers for inverted pendulum-cart system (7.17)-(7.21) with measurement noise's standard deviation 0.1.

Finally, to compare the IPG observer with LPV, we consider the system from the LPV observer paper [221]:

$$\dot{x}_{(1)} = \exp(-x_{(1)}^2)\log(1+x_{(1)}^2) + 0.5\frac{\sin(x_{(2)})}{x_{(2)}}, \tag{7.22}$$

$$\dot{x}_{(2)} = -\frac{x_{(2)}}{1+x_{(2)}^2} - \frac{7}{4}\frac{x_{(2)}^3}{1+x_{(2)}^2} - 0.1y^3, \tag{7.23}$$

$$y = x_{(1)} + 0.5\sin x_{(1)} + \eta, \tag{7.24}$$

with initial state $[0.5, 0.5]^T$. The measurement noise η is generated from zero mean Gaussian distribution with standard deviation 0.1. The LPV observer for this system has been designed in (48) of [221] in continuous-time, which we solve using *ode45*. To implement the IPG observer, we discretize (7.22)-(7.24) with sampling period $h = 1$ and the IPG observer is built upon this discretized model. Both the observers are initialized with the state estimate $[3, 2]^T$. For IPG,

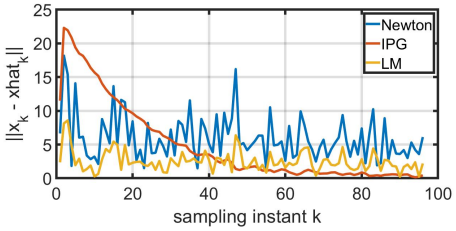

Figure 7.6: State estimation error of different observers for inverted pendulum-cart system (7.17)-(7.21) with measurement noise's standard deviation 0.01.

we choose $N = 2$ and $d = 2$. The initial pre-conditioner for IPG observer is set at $10^{-4}I_2$, tuned from $\{10^p I_2 : p = -4, -3, \ldots, 4\}$ and $\alpha^i = 0.01$, tuned from $\{0.1, 0.01, 0.001\}$. From Fig. 7.10, the steady-state estimation error of LPV observer is approximately 0.13. From Fig. 7.9, the steady-state estimation error of IPG observer is approximately 0.02, smaller than LPV.

7.5 Summary

We proposed a Newton-type observer to address the lack of robustness of Newton observer and its existing variants for discrete-time systems with sampled outputs. The proposed observer is based upon a non-symmetric iteratively pre-conditioned gradient-descent (IPG) technique, used for solving a set of nonlinear equations at each sampling instant, leading to improved robustness against measurement noise. The proposed IPG observer applies to a wide class of nonlinear systems. We presented extensive empirical results comparing the accuracy of our IPG observer with the prominent observers. Future work involves rigorously obtaining an upper bound on the estimation error of IPG observer in presence

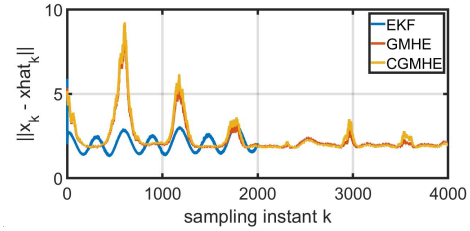

Figure 7.7: State estimation error of different observers for inverted pendulum-cart system (7.17)-(7.21) with measurement noise's standard deviation 0.1.

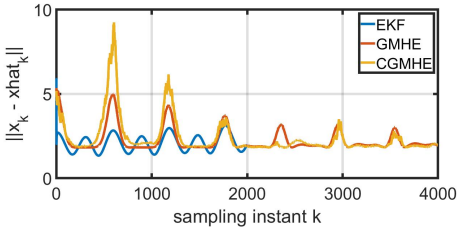

Figure 7.8: State estimation error of different observers for inverted pendulum-cart system (7.17)-(7.21) with measurement noise's standard deviation 0.01.

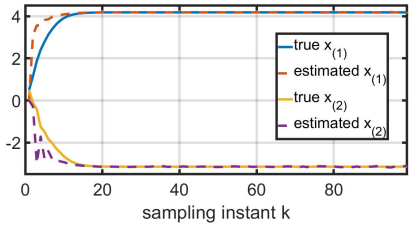

Figure 7.9: True system trajectory and its estimate for the system (7.22)-(7.24) obtained by IPG observer.

of measurement noise and demonstrating the error's convergence to zero in absence of noise.

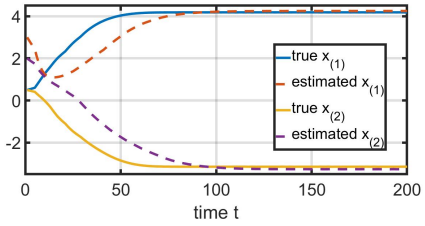

Figure 7.10: True system trajectory and its estimate for the system (7.22)-(7.24) obtained by LPV observer.

Chapter 8: Summary and Future Work

This dissertation aims to propose robust and efficient optimization algorithms for unconstrained problems in centralized and distributed settings. The specific optimization problems that have been considered can be classified into distributed optimization in server-agent architecture, decentralized linear regression in peer-to-peer network architecture, and centralized non-convex optimization with an emphasis on deep neural networks. Novel optimization schemes have been developed to solve each of these three classes of problems. Additionally, a distributed beamforming algorithm and a nonlinear observer have been proposed, built upon one of these proposed schemes.

8.1 Completed work

First, the Iteratively Pre-Conditioned Gradient-descent (IPG) algorithm has been proposed for solving distributed convex optimization. The key to the IPG method is a particular iterative pre-conditioning scheme, inspired by classical adaptive control laws, facilitating the IPG method to be executed in distributed settings. For the class of distributed linear regression problems, thorough theoretical and empirical studies have been presented, which entail comparing IPG with existing distributed methods. Rigorous characterization of IPG's robustness against system noise of arbitrary sources and IPG's extension (IPSG) to the stochastic settings have been presented, along with the relevant empirical comparison with existing methods. Theoretical results have

been presented for a class of distributed convex cost functions, involving IPG's local convergence analysis and comparison with some of the existing distributed optimization methods. Empirical results indicate IPG's improved convergence rate and robustness against system noise compared to some of the existing methods for distributed logistic regression.

Next, two locally pre-conditioned decentralized gradient-descent algorithms have been proposed for solving systems of linear algebraic equations over peer-to-peer networks, respectively, in the absence and presence of communication delay. The key to the proposed methods is a particular decentralized pre-conditioning scheme. The presented theoretical results include the proposed algorithms convergence guarantee and its robustness against system noise. The theoretical analyses have been supported through experiments by applying the proposed algorithm to develop a linear state predictor and comparing it with existing decentralized methods, including the decentralized Kalman filter.

Finally, solving the centralized non-convex optimization problems has been addressed from a state-space perspective. The contributions towards this class of problems are motivated by the recent flourish of considering optimization algorithms as dynamical systems. The eventual aim is to develop a generic framework encompassing the adaptive gradient-descent algorithms used in machine learning. To reach this goal, a class of adaptive gradient-descent methods has been proposed in a state-space framework. A well-known tool from adaptive control, Barbalat's lemma, is then utilized to analyze the proposed class of methods, leading to simple and intuitive proofs of some prominent adaptive gradient-descent optimizers in the continuous-time domain. It has been shown that the proof sketch build upon Barbalat's lemma can also be applied for analyzing other adaptive and accelerated gradient methods that are not included in the proposed generic framework. The advantage of the state-space perspective for optimization algorithms is

not limited to analysis, which we show by proposing variants of some of the prominent adaptive gradient methods in machine learning, by utilizing the concept of transfer functions. Benchmark deep neural network experiments suggest that the proposed transfer function-based modification contributes towards balancing smaller generalization errors and faster training, a dilemma in applying adaptive methods to train machine learning models.

In addition to the aforementioned generic optimization problems, the distributed beamforming problem and the nonlinear observer design problem have been considered, which can be formulated as specific optimization problems. Built upon the aforementioned IPG optimizer, the IPG for distributed beamforming (IPG-DB) and the IPG observer have been proposed, respectively to solve these two problems, which leverages the iterative pre-conditioning technique's robustness against system noise. Experimental results have been presented, demonstrating the efficiency of IPG-DB in contested uncertain environments and the robustness of IPG observer against measurement noise.

8.2 Future work

For general convex cost functions, although the IPG method has been proved to have a locally superior convergence rate than the existing first-order methods, it lacks a global "picture". Hence, the global convergence of the proposed IPG method will be addressed in future. This will help in a complete comparison with the existing methods, including the classes of nonlinear conjugate gradient, Newton and quasi-Newton methods with line-search, Levenberg–Marquardt methods, and the recent stochastic quasi-Newton methods. While the literature on quasi-Newton methods is vast, only a few have formally addressed robustness against process noise. Most

importantly, Newton's method and the quasi-Newton methods are not reliable in the presence of uncertainties, as shown in Section 4.3. Sufficient conditions for local quadratic convergence of Newton's method executed on finite precision machines and local linear convergence of a class of quasi-Newton method satisfying a certain deterioration condition have been presented in [97] and [98]. The analysis of BFGS in [99] assumes progressively diminishing noise. [100], and the most recent work [101] has proposed noise-tolerant variations of the BFGS method, assuming that an estimate of the noise is known. In comparison, the IPG algorithm will be the same in deterministic and noisy settings, and would not require any noise estimate. Instead of presenting sufficient condition for convergence in the presence of noise, IPG's robustness against bounded noise will be characterized.

An immediate research direction from the distributed optimization in server-agent settings is the federated optimization [224]. The challenge in federated settings is the heterogeneity of the agents' data, system-level heterogeneity, the asynchronous nature of the agents in updating the local gradients, limited and delayed communication between the agents and the server. It is expected that the robustness of IPG will be useful in addressing these challenges in federated optimization, possibly resulting in an algorithm superior to the existing federated optimization techniques [225, 226].

Regarding non-convex optimization, this dissertation hypothesized that by adding poles and zeros in the transfer function, and by tuning these hyper parameters, we create a signal, representing the second raw moment estimate of gradients, that is less impacted by the noise as compared to the squared gradient input. Attenuation of the frequency specific noise may help by not leading the learning astray and consequently lead to better generalization. Although, numerical evidence have been presented for this hypothesis, it needs to be rigorously evaluated in

a subsequent study. This study will help in understanding whether the idea of transfer functions can be exploited to improve any existing adaptive gradient optimizer in long-term.

Finally, the ongoing work in distributed beamforming includes integrating feedback from an auxiliary node into the proposed IPG-DB algorithm to address the additional objective of suppressing the minor lobes in generated radiation pattern, and covert communication with the client when the precise locations of the adversaries are unknown. Performance of the IPG-DB algorithm with higher carrier frequency (2.4 GHz) is being tested using channel data generated from EM.CUBE simulations.

Appendix A: Proofs of the Theoretical Results

A.1 Proof of Theorem 2.1

In this section, we present a proof of Theorem 2.1. Throughout this section we assume that A^TA is not the trivial zero matrix, $\beta > 0$, and $0 < \alpha < \frac{2}{\lambda_1+\beta}$. The proof relies on the following lemma, Lemma A.1, which shows the *linear* convergence of the sequence of matrices $\{K(t),\ t = 0,\ 1, \ldots\}$ to K_β. Let $k_{j\beta}$ denote the j-th column of matrix K_β where $j = 1, \ldots,\ d$, and recall the definition of ϱ from (2.12).

Lemma A.1. *Consider Algorithm 1. If* $\alpha \in \left(0, \frac{2}{\lambda_1+\beta}\right)$ *then there exists* ρ *with* $\varrho \leq \rho < 1$ *such that for each* $j \in \{1, \ldots,\ d\}$, $\left\|k_j(t+1) - k_{j\beta}\right\| \leq \rho \left\|k_j(t) - k_{j\beta}\right\|, \forall t \geq 0$.

Proof. From (2.7) we have $\sum_{i=1}^m R_j^i(t) = \left[\left(A^TA + \beta I\right) k_j(t) - e_j\right]$. Upon substituting from above in (2.8), for $j = 1, \ldots, d$,

$$k_j(t+1) \ = k_j(t) - \alpha \left(A^TA + \beta I\right) k_j(t) + \alpha e_j. \tag{A.1}$$

Recall the definition $K_\beta = \left(A^TA + \beta I\right)^{-1}$. Then for each column $j = 1, \ldots, d$ of K_β we obtain that $\left(A^TA + \beta I\right) k_{j\beta} = e_j$. For each iteration t, let $\widetilde{K}(t)$ denote the matrix obtained by stacking

the column vectors $\widetilde{k}_1(t), \ldots, \widetilde{k}_d(t)$:

$$\widetilde{K}(t) = \left[\widetilde{k}_1(t), \ldots, \widetilde{k}_d(t)\right] = K(t) - K_\beta, \quad \forall t \geq 0. \tag{A.2}$$

Upon substituting from above in (A.1), and using the definition of $\widetilde{k}_j(t)$ in (A.2), we have $\widetilde{k}_j(t+1) = \left[I - \alpha\left(A^TA + \beta I\right)\right]\widetilde{k}_j(t)$. Since $(A^TA + \beta I)$ is positive definite for $\beta > 0$, there exists $\alpha \in \left(0, \frac{2}{\lambda_1+\beta}\right)$ for which there is a positive $\rho < 1$ such that $\left\|\widetilde{k}_j(t+1)\right\| \leq \rho\left\|\widetilde{k}_j(t)\right\|$ for $j = 1, \ldots, d$ and $\forall t \geq 0$ [21]. Note that [21] $\rho \geq \frac{(\lambda_1+\beta)-(\lambda_d+\beta)}{(\lambda_1+\beta)+(\lambda_d+\beta)} = \frac{\lambda_1-\lambda_d}{\lambda_1+\lambda_d+2\beta} \overset{(2.12)}{=} \varrho$. Hence, the proof. □

Now, for each i and t, upon substituting $g^i(t)$ from (2.3) in (2.10) we obtain that $g(t) = \left(\sum_{i=1}^m (A^i)^TA^i\right)x(t) - \left(\sum_{i=1}^m (A^i)^Tb^i\right)$. As $A^TA = \sum_{i=1}^m (A^i)^TA^i$ and $A^Tb = \sum_{i=1}^m (A^i)^Tb^i$, the above implies that

$$g(t) = \textstyle\sum_{i=1}^m g^i(t) = A^T(Ax(t) - b). \tag{A.3}$$

From (A.3) and the definition of X^* in (2.1), $X^* = \left\{x \in \mathbb{R}^d \ : \ A^T(Ax - b) = 0_d\right\}$. Consider an arbitrary point $x^* \in X^*$. Define $z(t) = x(t) - x^*$. As $A^T(Ax^* - b) = 0_d$, upon substituting from above in (A.3) we obtain that $g(t) = A^TA\,z(t)$. Let $\mathcal{N}(A^TA)$ denote the nullspace of matrix A^TA, and $\mathcal{N}(A^TA)^\perp$ denote the orthogonal vector space of $\mathcal{N}(A^TA)$. Due to the fundamental theorem of linear algebra [45], $\mathbb{R}^d = \mathcal{N}(A^TA) \oplus \mathcal{N}(A^TA)^\perp$. Therefore, for each $t \geq 0$, we can decompose vector $z(t)$ into two orthogonal vectors $z(t)^\perp$ and $z(t)^\mathcal{N}$, such that $z(t)^\perp \in \mathcal{N}(A^TA)^\perp$ and $z(t)^\mathcal{N} \in \mathcal{N}(A^TA)$. Specifically, for each $t \geq 0$, $z(t) = z(t)^\mathcal{N} + z(t)^\perp$. As $A^TA\,z(t)^\mathcal{N} = 0_d$,

we have

$$g(t) = A^T A\, z(t)^{\perp}, \quad \forall t \geq 0. \tag{A.4}$$

The remainder of the proof is divided into three steps.

Step I: Upon substituting from (A.3) in (2.9), using the definitions of $z(t)$ and X^*, we obtain that

$$z(t+1) = \Big(I - \delta K(t+1) A^T A\Big)\ z(t). \tag{A.5}$$

Recall that $\lambda_1 \geq \ \ldots \ \geq \lambda_d \geq 0$ denote the eigenvalues of the positive semi-definite matrix $A^T A$. We denote by $Diag(.)$ a diagonal matrix of appropriate dimensions, with the arguments denoting the diagonal entries in the same order. Let $S = Diag(\lambda_1, \ldots, \lambda_d)$, and let the matrix V consists of the corresponding orthonormal eigenvectors $[V_1, \ldots, V_d]$. Note that $V^T V = I$, and $A^T A = V S V^T$. Recall that r denotes the rank of matrix $A^T A$. In general, $1 \leq r \leq d$. If $r < d$ then $\lambda_1 \geq \ \ldots \ \geq \lambda_r > \lambda_{r+1} = \ \ldots \ = \lambda_d = 0$. Thus, $S = Diag(\lambda_1, \ldots, \lambda_r, \underbrace{0, \ldots, 0}_{d-r})$. Let $span\,\{V_1, \ldots, V_r\}$ denote the vector space spanned by the orthonormal eigenvectors $V_1, \ldots, V_r$, defined as $span\,\{V_1, \ldots, V_r\} = \big\{\sum_{i=1}^{r} u_i V_i \ : \ u_i \in \mathbb{R},\ \forall i\big\}$. As the eigenvectors $V_1, \ldots, V_d$ are orthogonal [45], $\mathcal{N}(A^T A)^{\perp} = span\,\{V_1, \ldots, V_r\}$ and $\mathcal{N}(A^T A) = span\,\{V_{r+1}, \ldots, V_d\}$. Let, $S^{\perp} = Diag(\underbrace{1, \ldots, 1}_{r}, \underbrace{0, \ldots, 0}_{d-r})$. Define a *projection matrix*

$$Q = V S^{\perp} V^T. \tag{A.6}$$

Note that for a vector $v \in \mathbb{R}^d$, due to the fundamental theorem of linear algebra, the vectors Qv and $(v - Qv)$ belong to the orthogonal vector spaces $\mathcal{N}(A^TA)^\perp$ and $\mathcal{N}(A^TA)$, respectively. Thus, from the definition of $z(t)^\perp$, $z(t)^\perp = Qz(t)$. This implies that, for all $t \geq 0$,

$$\begin{aligned} z(t+1)^\perp &= Q\, z(t+1) \overset{(A.5)}{=} Q\, \left(I - \delta K(t+1) A^T A\right)\, z(t) \\ &= z(t)^\perp - \delta Q K(t+1) A^T A\, z(t) \\ &= z(t)^\perp - \delta Q K(t+1) A^T A\, \left(z(t)^{\mathcal{N}} + z(t)^\perp\right). \end{aligned} \tag{A.7}$$

As $z(t)^{\mathcal{N}} \in \mathcal{N}(A^TA)$, $A^TA\, z(t)^{\mathcal{N}} = 0_d$. Upon substituting this in (A.7) we obtain that $z(t+1)^\perp = z(t)^\perp - \delta Q K(t+1) A^T A\, z(t)^\perp, \quad \forall t$. Substituting above from (A.4) we obtain that $z(t+1)^\perp = z(t)^\perp - \delta Q K(t+1)\, g(t)$. Multiplying both sides above with A^TA, and substituting again from (A.4), we obtain that $g(t+1) = g(t) - \delta\, A^TAQ\, K(t+1)\, g(t)$. Upon substituting above from (A.2), we have $g(t+1) = \left(I - \delta A^TAQ\, K_\beta\right)\, g(t) - \delta A^TAQ\widetilde{K}(t+1)\, g(t)$.

Step II: Using triangle inequality above, we obtain that

$$\left\|g(t+1)\right\| \leq \left\|\left(I - \delta A^TAQ\, K_\beta\right)\, g(t)\right\| + \left\|\delta A^TAQ\widetilde{K}(t+1) g(t)\right\|. \tag{A.8}$$

First, we will derive an upper bound on the second term in (A.8). Recall the definition of $\widetilde{K}(t)$ from (A.2). Due to Lemma A.1, for each $j \in \{1, \ldots, d\}$ we obtain that $\left\|\widetilde{k}_j(t)\right\|^2 \leq \rho^{2t} \left\|\widetilde{k}_j(0)\right\|^2$. Since $\left\|\widetilde{K}(t)\right\|_F^2 = \sum_{j=1}^d \left\|\widetilde{k}_j(t)\right\|^2$, from above, we obtain that, $\left\|\widetilde{K}(t)\right\|_F \leq \rho^t \left\|\widetilde{K}(0)\right\|_F$. For any square matrix M, let $\|M\|$ denote the *induced 2-norm* of the matrix. Since $\|M\| \leq \|M\|_F$ [45], from above we get, for all $t \geq 0$, $\left\|\widetilde{K}(t)\right\| \leq \rho^t \left\|\widetilde{K}(0)\right\|_F$. Recall that V is a unitary matrix. Thus, $A^TAQ = V\, \left(SS^\perp\right)\, V^T$. Note that $\left\|A^TAQ\right\| = \lambda_1$ [45]. From the definition of induced

2-norm [45], and substituting from above,

$$\left\|\delta A^T A\, Q\widetilde{K}(t+1)g(t)\right\| \leq \delta\lambda_1 \left\|\widetilde{K}(0)\right\|_F \rho^{t+1}\left\|g(t)\right\|. \tag{A.9}$$

Next, we will upper bound the first term in (A.8). As $A^TA = VSV^T$ where $VV^T = I$, from the definition of K_β we obtain that $K_\beta = V\, Diag\left(\frac{1}{\lambda_1+\beta}, \ldots, \frac{1}{\lambda_d+\beta}\right) V^T$. Also, from the definition of S and $S^\perp$ we have $A^TAQ = V\, Diag(\lambda_1, \ldots, \lambda_r, \underbrace{0, \ldots, 0}_{d-r})\, V^T$. Thus,

$$\left(I - \delta\, A^TAQK_\beta\right) g(t) = V\, Diag(\underbrace{\left(1 - \frac{\delta\lambda_i}{\lambda_i+\beta}\right)}_{i=1,\ldots,r}, \underbrace{1, \ldots, 1}_{d-r}) V^T\, g(t). \tag{A.10}$$

Now, note that from (A.4), $g(t) \in Im\left(A^TA\right)$ where $Im(\cdot)$ denotes the image of a matrix operator. Owing to the fundamental theorem of linear algebra [45], $Im\left(A^TA\right) = \mathcal{N}\left(A^TA\right)^\perp$. Thus, $g(t) \in \mathcal{N}(A^TA)^\perp = span\left\{V_1, \ldots, V_r\right\}$. Recall that the vectors $V_1, \ldots, V_d$, constituting the matrix V, are orthonormal. Therefore,

$$\left\| V\, Diag(\underbrace{\left(1 - \frac{\delta\lambda_i}{\lambda_i+\beta}\right)}_{i=1,\ldots,r}, \underbrace{1, \ldots, 1}_{d-r}) V^T\, g(t) \right\| \leq \max\left\{\left|1 - \frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \ldots, \left|1 - \frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\} \left\|g(t)\right\|, \tag{A.11}$$

where $|\cdot|$ denotes the absolute value. As $\lambda_1 \geq \ldots \geq \lambda_r > 0$ and $\beta > 0$, if $0 < \delta < 2\left(\frac{\lambda_1+\beta}{\lambda_1}\right) = 2\left(1 + \frac{\beta}{\lambda_1}\right)$, then

$$\max\left\{\left|1 - \frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \ldots, \left|1 - \frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\} = \max\left\{\left|1 - \frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \left|1 - \frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\}, \tag{A.12}$$

$$\text{and } \left|1 - \frac{\delta\lambda_i}{\lambda_i+\beta}\right| < 1,\ i = 1, \ldots, r. \tag{A.13}$$

Substituting from (A.12) and (A.11) in (A.10) we obtain that,

$$\left\|\left(I-\delta\, A^T AQK_\beta\right)\, g(t)\right\| \leq \max\left\{\left|1-\frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \left|1-\frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\}\left\|g(t)\right\|. \tag{A.14}$$

Finally, upon substitution from (A.9) and (A.14) in (A.8),

$$\left\|g(t+1)\right\| \leq \max\left\{\left|1-\frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \left|1-\frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\}\left\|g(t)\right\| + \delta\lambda_1\left\|\widetilde{K}(0)\right\|_F \rho^{t+1}\left\|g(t)\right\|, \forall t \geq 0 \tag{A.15}$$

Then, with $\mu = \max\left\{\left|1-\frac{\delta\lambda_1}{\lambda_1+\beta}\right|, \left|1-\frac{\delta\lambda_r}{\lambda_r+\beta}\right|\right\}$, (A.15) and (A.13) prove (2.14). Recall the definition of μ^* from (2.11). Note that [21]

$$\mu \geq \frac{\frac{\lambda_1}{\lambda_1+\beta}-\frac{\lambda_r}{\lambda_r+\beta}}{\frac{\lambda_1}{\lambda_1+\beta}+\frac{\lambda_r}{\lambda_r+\beta}} = \frac{\beta\left(\lambda_1-\lambda_r\right)}{2\lambda_1\lambda_r+\beta\left(\lambda_1+\lambda_r\right)} = \mu^*, \tag{A.16}$$

where the equality $\mu=\mu^*$ holds true, if the value of δ is given by (2.13), which is $\delta = \frac{2}{\frac{\lambda_1}{\lambda_1+\beta}+\frac{\lambda_r}{\lambda_r+\beta}}$. Note that as $\left(\frac{\lambda_r}{\lambda_r+\beta}\right) > 0$, $\frac{2}{\frac{\lambda_1}{\lambda_1+\beta}+\frac{\lambda_r}{\lambda_r+\beta}} < \frac{2}{\frac{\lambda_1}{\lambda_1+\beta}} = 2\left(\frac{\lambda_1+\beta}{\lambda_1}\right)$. Thus, the value of δ in (2.13) satisfies the condition of Theorem 2.1.

Step III: In this step, using the following fact, we prove that $\lim_{t\to\infty}\left\|g(t)\right\| = 0$.

Fact A.1. *Consider an infinite sequence of non-negative values* $\{s_t,\ t=0,1,\ldots\}$ *with* $s_t < s_{t-1},\ \forall t \geq 1,$ *and* $\lim_{t\to\infty} s_t < L$ *where* L *is a positive finite real number. Then, there exists* $0 \leq T' < \infty$ *such that* $s_t < L,\ \forall t > T'$.

In (2.14), let $\alpha_t = \left(\mu + \delta\lambda_1\left\|K(0)-K_\beta\right\|_F \rho^{t+1}\right)$. Note that $\alpha_t \geq 0$ for all $t \geq 0$. Since $\rho < 1$, the sequence $\{\alpha_t\}_{t\geq 0}$ is strictly decreasing, and $\lim_{t\to\infty}\alpha_t = \mu < 1$. Thus, due to Fact A.1, there exists a positive integer τ such that $\alpha_t < 1, \quad \forall t > \tau$. From the recursion (2.14),

we have

$$\left\|g(t+1)\right\| \leq \left(\Pi_{k=\tau+1}^{t}\,\alpha_k\right)\left\|g(\tau+1)\right\|, \quad \forall t > \tau. \tag{A.17}$$

As $\{\alpha_t\}_{t\geq 0}$ is strictly decreasing, (2.14) implies that $\left\|g(\tau+1)\right\| \leq \alpha_\tau \left\|g(\tau)\right\| \leq \alpha_0 \left\|g(\tau)\right\|$. Upon iterating this τ times we get $\left\|g(\tau+1)\right\| \leq \alpha_0^{\tau+1}\left\|g(0)\right\|$. Combining (A.17) and the above we get $\left\|g(t+1)\right\| \leq \left(\Pi_{k=\tau+1}^{t}\,\alpha_k\right)\alpha_0^{\tau+1}\left\|g(0)\right\| \leq \alpha_{\tau+1}^{t+1}\left(\frac{\alpha_0}{\alpha_{\tau+1}}\right)^{\tau+1}\left\|g(0)\right\|$ for all $t > \tau$. Since $\alpha_{\tau+1} < 1$, $\lim_{t\to\infty}\left\|g(t)\right\| = 0$ follows from above.

A.2 Proof of Corollary 2.1

Note that the solution x^*, defined by (2.1), is unique if and only if the matrix A^TA is full-rank, which means, $r = d$. Thus, A^TA is symmetric positive definite and has positive real eigenvalues. In other words, we have $\lambda_1 \geq \ldots \geq \lambda_d > 0$. Moreover, $\left(A^TA\right)^{-1}$ exists, and is also symmetric positive definite. As $r = d$, substituting $\beta = 0$ in (2.11) and (2.12), respectively, we obtain that $\mu^* = 0$ and $\varrho = \frac{\lambda_1 - \lambda_d}{\lambda_1 + \lambda_d}$. Now, (2.14) of Theorem 2.1 implies that, for $\delta = 1$ obtained by substituting $\beta = 0$ in (2.13), $\left\|g(t+1)\right\| \leq \lambda_1 \left\|K(0) - K_\beta\right\|_F \rho^{t+1}\left\|g(t)\right\|, \quad \forall t \geq 0$. Since $\rho < 1$, the above inequality implies that $\lim_{t\to\infty}\frac{\left\|g(t+1)\right\|}{\left\|g(t)\right\|} = 0$.

From the proof of Theorem 2.1, $g(t) = A^TAz(t)$ for each $t \geq 0$. Since $\left(A^TA\right)^{-1}$ exists, we have $z(t) = \left(A^TA\right)^{-1}g(t)$ for each $t \geq 0$. From the definition of induced norm, then we have $\left\|z(t+1)\right\| \leq \left\|\left(A^TA\right)^{-1}\right\|\left\|g(t+1)\right\|$ and $\left\|g(t)\right\| \leq \left\|A^TA\right\|\left\|z(t)\right\|$. The latter implies that $\left\|z(t)\right\| \geq \frac{1}{\left\|A^TA\right\|}\left\|g(t)\right\|$. Then, $\frac{\left\|z(t+1)\right\|}{\left\|z(t)\right\|} \leq \left\|\left(A^TA\right)^{-1}\right\|\left\|A^TA\right\|\frac{\left\|g(t+1)\right\|}{\left\|g(t)\right\|}$. Since $\lim_{t\to\infty}\frac{\left\|g(t+1)\right\|}{\left\|g(t)\right\|} = 0$, from above it follows that $\lim_{t\to\infty}\frac{\left\|z(t+1)\right\|}{\left\|z(t)\right\|} = 0$.

A.3 Proof of Lemma 2.1

Comparing the GD update in (2.4), and that of Algorithm 1 in (2.9), we see that Algorithm 1 with $K(t) = I \quad \forall t \geq 0$ is the GD method. Thus, for GD we define $K_\beta = I$ to which the sequence of matrices $\{K(t)\}$ converges. Now recall the definition of $\widetilde{K}(t)$ from (A.2). For GD, we then have $\widetilde{K}(t) = 0 \quad \forall t \geq 0$. Now we proceed exactly as the proof of Theorem 2.1, and arrive at (A.8) with $\widetilde{K}(t) = 0 \quad \forall t \geq 0$ and $K_\beta = I$. In other words, $\left\|g(t+1)\right\| \leq \left\|\left(I - \delta A^T A Q\right)\, g(t)\right\|, \quad \forall t \geq 0$. Substituting above from eigen-expansion of $A^T A Q$ in Section A.1, $\left\|g(t+1)\right\| \leq \left\| V Diag(\underbrace{(1-\delta\lambda_i)}_{i=1,\ldots,r} \underbrace{1,\ldots,1}_{d-r})\, V^T g(t) \right\|$. Following the argument after (A.10), if $\delta \in \left(0, \frac{2}{\lambda_1}\right)$, from above we have $\left\|g(t+1)\right\| \leq \max\left\{\left|1-\delta\lambda_1\right|, \left|1-\delta\lambda_r\right|\right\} \left\|g(t)\right\|$, and $\left|1-\delta\lambda_i\right| < 1,\ i = 1,\ldots,r$. We define $\max\left\{\left|1-\delta\lambda_1\right|, \left|1-\delta\lambda_r\right|\right\} = \mu$. The smallest possible μ is given by $\mu \geq \frac{\lambda_1 - \lambda_r}{\lambda_1 + \lambda_r}$, which is μ_{GD}. Hence, the proof.

A.4 Proof of Theorem 2.2

We define $\alpha_t = \mu^* + \delta\lambda_1 \left\|K(0) - K_\beta\right\|_F \rho^{t+1}$. As $\beta > 0$ and $\lambda_1 > \lambda_r$, from (2.11) and the definition of μ_{GD} we obtain that $\mu^* < \mu_{GD}$. Since $\rho < 1$, the sequence $\{\alpha_t > 0\}_{t \geq 0}$ is strictly decreasing and $\lim_{t\to\infty} \alpha_t = \mu^* < \mu_{GD}$. Thus, from Fact A.1, there exists a positive integer τ such that $\alpha_t < \mu_{GD}, \forall t > \tau$. From (2.14), for some $\delta > 0$ we have $\left\|g(t+1)\right\| \leq \alpha_t \left\|g(t)\right\|$. Upon iterating the above,

$$\left\|g(t+1)\right\| \leq \left(\Pi_{k=\tau+1}^{t} \alpha_k\right) \left\|g(\tau+1)\right\|, \ \forall t > \tau. \tag{A.18}$$

Since $\{\alpha_t > 0\}_{t\geq 0}$ is strictly decreasing, $\|g(\tau+1)\| \leq \alpha_\tau \|g(\tau)\| \leq \alpha_0 \|g(\tau)\| \leq \alpha_0^{\tau+1}\|g(0)\|$. Upon substituting from above in (A.18), $\|g(t+1)\| \leq \left(\Pi_{k=\tau+1}^{t} \alpha_k\right) \alpha_0^{\tau+1} \|g(0)\|$, $\forall t > \tau$. We define $r_t = \alpha_t/\mu_{GD}$. Thus, $\|g(t+1)\| \leq \left(\Pi_{k=\tau+1}^{t} r_k \mu_{GD}\right) \alpha_0^{\tau+1}\|g(0)\|$. Upon defining $\overline{r} = \underset{t>\tau}{max}\, r_t$, we have $\|g(t+1)\| \leq \left(\Pi_{k=\tau+1}^{t} \overline{r}\mu_{GD}\right) \alpha_0^{\tau+1}\|g(0)\|$. We define $c = \left(\frac{\alpha_0}{\overline{r}\mu_{GD}}\right)^{\tau+1}$. Since $r_t = \frac{\alpha_t}{\mu_{GD}} < 1\, \forall t > \tau$, $\overline{r} = \underset{t>\tau}{max}\, r_t < 1$. Hence, the proof.

A.5 Proof of Theorem 2.3

From (2.18) we have $b^o = b + w_b$, where

$$b^o = \left[(b^{1o})^T, \ldots, (b^{mo})^T\right]^T,\ w_b = \left[(w_b^1)^T, \ldots, (w_b^m)^T\right]^T.$$

From (2.19), we obtain that

$$\mathbb{E}\left[\|w_b\|\right] \leq \mathbb{E}\left[\|w_b\|_1\right] = \sum_{i=1}^{m} \mathbb{E}\left[\left\|w_b^i\right\|_1\right] \leq \eta m. \tag{A.19}$$

Upon substituting from (2.20) in (2.9) we obtain that

$$x(t+1) = x(t) - \delta K(t+1) A^T (Ax(t) - b - w_b). \tag{A.20}$$

Now, consider a point $x^* \in \arg\min_x \sum_{i=1}^{m} F^i(x)$ defined by (2.1). As $\nabla \sum_{i=1}^{m} F^i(x) = A^T(Ax - b)$,

$$A^T(Ax^* - b) = 0_d. \tag{A.21}$$

Recall from (2.21) that $z(t) = x(t) - x^*$ for all t. Subtracting x^* on both sides of (A.20), and using (A.21), we have

$$\begin{aligned} z(t+1) &= \left(I - \delta K(t+1)A^TA\right) z(t) + \delta K(t+1)A^T w_b \\ &\overset{(2.16)}{=} \left(I - \delta K^*A^TA\right) z(t) - \delta \widetilde{K}(t+1)A^TAz(t) + \delta K^*A^T w_b + \delta \widetilde{K}(t+1)A^T w_b. \end{aligned} \tag{A.22}$$

Upon using triangle inequality above and the definition $K^*A^TA = I$ (ref. Section 2.4.1), we obtain that

$$\left\|z(t+1)\right\| \leq (1-\delta)\left\|z(t)\right\| + \delta\left\|\widetilde{K}(t+1)A^TAz(t)\right\| + \delta\left\|K^*A^Tw_b\right\| + \delta\left\|\widetilde{K}(t+1)A^Tw_b\right\|. \tag{A.23}$$

From the definition of induced 2-norm of a matrix [45] and expectation, (A.23) implies that

$$\begin{aligned} \mathbb{E}\left[\left\|z(t+1)\right\|\right] \leq &\left(1 - \delta + \delta\left\|\widetilde{K}(t+1)\right\|\left\|A^TA\right\|\right)\left\|z(t)\right\| \\ &+ \delta\left\|K^*A^T\right\|\mathbb{E}\left[\left\|w_b\right\|\right] + \delta\left\|\widetilde{K}(t+1)\right\|\left\|A^T\right\|\mathbb{E}\left[\left\|w_b\right\|\right]. \end{aligned}$$

Upon substituting above from (A.19) we obtain that

$$\begin{aligned} \mathbb{E}\left[\left\|z(t+1)\right\|\right] \leq &\left(1 - \delta + \delta\left\|\widetilde{K}(t+1)\right\|\left\|A^TA\right\|\right)\left\|z(t)\right\| \\ &+ \delta\left\|K^*A^T\right\|\eta m + \delta\left\|\widetilde{K}(t+1)\right\|\left\|A^T\right\|\eta m. \end{aligned} \tag{A.24}$$

From Lemma 2.2, we have for all $j \in \{1, \ldots, d\}$, $\left\|\widetilde{k}_j(t)\right\|^2 \leq \rho^{2t}\left\|\widetilde{k}_j(0)\right\|^2$, From basic Linear Algebra [45], we know that $\left\|A^TA\right\| = \lambda_1$ and $\left\|A^T\right\| = \sqrt{\lambda_1}$. Upon substituting these in (A.24)

we have

$$\mathbb{E}\left[\left\|z(t+1)\right\|\right] \leq \delta\rho^{t+1}\left\|\widetilde{K}(0)\right\|_F\left(\lambda_1\left\|z(t)\right\| + \eta m\sqrt{\lambda_1}\right) + (1-\delta)\left\|z(t)\right\| + \delta\left\|K^*A^T\right\|\eta m. \tag{A.25}$$

From Singular Value Decomposition [45], $A = USV^T$ where

$$S^T = \left[Diag\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_d}\right), \mathbf{0}_{d\times(\sum_{i=1}^m n_i - d)}\right],$$

and the matrices U, V, respectively, constitutes of left and right orthonormal singular vectors of A. From above, $(A^TA)^{-1}A^T = V(S^TS)^{-1}S^TU^T$. Thus [45], $\left\|(A^TA)^{-1}A^T\right\| = 1/\sqrt{\lambda_d}$. As $K^* = \left(A^TA\right)^{-1}$ (see Section 2.4.1), the above implies that $\left\|K^*A^T\right\| = 1/\sqrt{\lambda_d}$. As discussed in Section 2.4.1, Assumption 2.1 is equivalent to $\lambda_d > 0$. Upon substituting this in (A.25),

$$\mathbb{E}\left[\left\|z(t+1)\right\|\right] \leq \left(1 - \delta + \delta\lambda_1\left\|\widetilde{K}(0)\right\|_F\rho^{t+1}\right)\left\|z(t)\right\| + \delta\eta m\sqrt{\lambda_1}\left\|\widetilde{K}(0)\right\|_F\rho^{t+1} + \delta\eta m\sqrt{1/\lambda_d}.$$

As t is an arbitrary iteration, the above proves (2.22).As $\rho \in [0,1)$ and $\delta \in (0,1]$, there exists $T < \infty$ such that $\left(1 - \delta + \delta\lambda_1\left\|\widetilde{K}(0)\right\|_F\rho^{t+1}\right) < 1$ for all $t \geq T$. Thus, upon retracing (2.22) from t to 0, we have $\lim_{t\to\infty}\mathbb{E}\left[\left\|z(t)\right\|\right] \leq \delta\eta m\sqrt{1/\lambda_d}$.

A.6 Proof of Theorem 2.4

Similar to (A.22) in Section A.5 above, for Algorithm 1 with modifications (2.26)-(2.29), we obtain that

$$z(t+1) = \left(I - \delta K^o(t+1)A^TA\right) z^o(t),\ \forall t. \tag{A.26}$$

Substituting from (2.30) and (2.33) in (A.26), since $K^*A^TA = I$,

$$z(t+1) = \left(1 - \delta - \delta\widetilde{K}^o(t+1)A^TA\right)\left(z(t) + w^x(t)\right).$$

The above implies that

$$\left\|z(t+1)\right\| \leq \left(1 - \delta + \delta\lambda_1\left\|\widetilde{K}^o(t+1)\right\|\right)\left(\left\|z(t)\right\| + \left\|w^x(t)\right\|\right).$$

From Assumption 2.3, since the random variables $\{w^x(t), w_j^k(t), j = 1, \ldots, d\}$ are mutually independent for all t, the above implies that

$$\mathbb{E}_t\left[\left\|z(t+1)\right\|\right] \leq \left(1 - \delta + \delta\lambda_1\mathbb{E}_t\left[\left\|\widetilde{K}^o(t+1)\right\|\right]\right)\left(\left\|z(t)\right\| + \mathbb{E}_t\left[\left\|w^x(t)\right\|\right]\right). \tag{A.27}$$

Define $W^k(t) = \left[w_1^k, \ldots, w_d^k\right]$. From (2.30) then we have $\widetilde{K}^o(t) = \widetilde{K}(t) + W^k(t)$, which implies that

$$\mathbb{E}_t\left[\left\|\widetilde{K}^o(t)\right\|\right] \leq \left\|\widetilde{K}(t)\right\| + \mathbb{E}_t\left[\left\|W^k(t)\right\|\right]. \tag{A.28}$$

By definition of matrix norms [45], $\mathbb{E}_t\left[\left\|W^k(t)\right\|\right] \leq \sqrt{d}\,\mathbb{E}_t\left[\left\|W^k(t)\right\|_1\right] \overset{(2.25)}{\leq} \omega\sqrt{d}$. Then, from (A.28) we get

$$\mathbb{E}_t\left[\left\|\widetilde{K}^o(t)\right\|\right] \leq \left\|\widetilde{K}(t)\right\| + \omega\sqrt{d}. \tag{A.29}$$

Instead of each column $\widetilde{k}_j(t)$ in Lemma 2.2 if we consider $\widetilde{K}(t)$ then following the proof of Lemma 2.2 for $\widetilde{K}_j(t)$, we have a similar result as (2.17): $\left\|\widetilde{K}(t+1)\right\| \leq \rho\left\|\widetilde{K}(t)\right\|$, $\forall t \geq 0$. Upon substituting from above in (A.29) we get

$$\mathbb{E}_t\left[\left\|\widetilde{K}^o(t)\right\|\right] \leq \rho\left\|\widetilde{K}(t-1)\right\| + \omega\sqrt{d} \leq \rho^t\left\|\widetilde{K}(0)\right\| + \omega\sqrt{d}\sum_{i=0}^{t}\rho^i \overset{(2.31)}{=} \left(u(t) - 1 + \delta\right)/(\delta\lambda_1).$$

Upon substituting from above in (A.27) we get

$$\begin{aligned}
&\mathbb{E}_t\left[\left\|z(t+1)\right\|\right] \leq u(t+1)\left(\left\|z(t)\right\| + \mathbb{E}_t\left[\left\|w^x(t)\right\|\right]\right)\\
&\leq u(t+1)\left(\left\|z(t)\right\| + \mathbb{E}_t\left[\left\|w^x(t)\right\|_1\right]\right) \overset{(2.25)}{\leq} u(t+1)\left(\left\|z(t)\right\| + \omega\right)\\
&\leq \Pi_{k=1}^{t+1}u(k)\left\|z(0)\right\| + \left(u(t+1) + u(t+1)u(t) + \ldots + \Pi_{k=1}^{t+1}u(k)\right)\omega.
\end{aligned} \tag{A.30}$$

From (2.33) and (2.25), $\mathbb{E}_t\left[\left\|z^o(t+1)\right\|\right] \leq \mathbb{E}_t\left[\left\|z(t+1)\right\|\right] + \omega$. Substituting from (A.30) in the R.H.S. above proves (2.34).

As $\rho < 1$, by definition of $u(t)$ in (2.31), we obtain that

$$\lim_{t\to\infty} u(t) = 1 - \delta + \delta\frac{\lambda_1\sqrt{d}\omega}{1-\rho} \overset{(2.32)}{=} 1 - \delta + \delta\frac{\omega}{\omega_{bd}} \overset{(2.35)}{<} 1, \tag{A.31}$$

$$\begin{aligned} u(t) - u(t-1) &= \delta\lambda_1\rho^{t-1}\left(\rho\left(\omega\sqrt{d} + \left\|\widetilde{K}(0)\right\|\right) - \left\|\widetilde{K}(0)\right\|\right) \\ &\overset{(2.32)}{=} \delta\lambda_1\rho^{t-1}\left\|\widetilde{K}(0)\right\|\left((\rho/\rho_{bd}) - 1\right), \ \forall t \geq 1. \end{aligned}$$

The above, in conjunction with (2.35), implies that $u(t) < u(t-1), \ \forall t \geq 1$. The limit in (A.31), in conjunction with the fact that $u(t)$ is non-negative for all t, implies that there exists $\tau < \infty$ such that $0 \leq u(t) < 1$ for all $t \geq \tau$. Thus,

$$\lim_{t\to\infty} \Pi_{k=1}^{t} u(k) = 0, \text{ and} \tag{A.32}$$

$$\lim_{t\to\infty} \left(1 + u(t) + \ldots + \Pi_{k=1}^{t} u(k)\right) < \frac{1}{1-u(\tau)}. \tag{A.33}$$

Substituting from (A.32) and (A.33) into (2.34), we obtain that

$$\lim_{t\to\infty} \mathbb{E}_t\left[\left\|z^o(t)\right\|\right] < \omega/(1-u(\tau)). \tag{A.34}$$

Since $\{u(t)\}$ is a strictly decreasing sequence, if (A.34) holds true for some τ satisfying $u(\tau) < 1$ then it also holds true for $\tau+1, \ldots, \infty$. Thus, (A.34) implies that $\lim_{t\to\infty} \mathbb{E}_t\left[\left\|z^o(t)\right\|\right] < \omega/(1-u(\infty))$. Substituting $u(\infty)$ from (A.31) we obtain that $\lim_{t\to\infty} \mathbb{E}_t\left[\left\|z^o(t)\right\|\right] < \frac{\omega}{\delta\left(1-\left(\omega/\omega_{bd}\right)\right)}$. Hence, the proof.

A.7 Proof of Theorem 2.5

A.7.1 Preliminary results

The results below are used in the proof of Theorem 2.5.

(a) Consider an arbitrary iteration $t \geq 0$. Upon taking conditional expectation $\mathbb{E}_{I_t}[\cdot]$ on both sides of (A.60), given the current matrix $K(t)$ and estimate $x(t)$, we have

$$\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right] = \left(I - \alpha\left(\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] + \beta I\right)\right)\widetilde{K}(t) - \alpha\left(\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] - \frac{1}{N}A^T A\right)K_\beta. \tag{A.35}$$

Upon subtracting both sides of (A.60) from that of (A.35) we get

$$\begin{aligned}\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right] - \widetilde{K}(t+1) &= \alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}} - \mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)\left(\widetilde{K}(t) + K_\beta\right) \\ &\overset{(A.2)}{=} \alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}} - \mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)K(t).\end{aligned} \tag{A.36}$$

(b) Now, consider a minimum point $x^* \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^m F^i(x)$ defined by (2.6). From the definition (2.5), the local cost function $F^i(x)$ is convex for each agent $i \in \{1,\ldots,m\}$. Thus, the aggregate cost function $\sum_{i=1}^m F^i(x)$ is also convex. Therefore, $x^* \in X^*$ if and only if [43] $\nabla\sum_{i=1}^m F^i(x^*) = 0_d$, where 0_d denotes the d-dimensional zero vector. As $\nabla\sum_{i=1}^m F^i(x) = A^T(Ax-b)$, x^* is the minimum point if and only if it satisfies

$$A^T(Ax^* - b) = 0_d. \tag{A.37}$$

From the definition of the each agent's stochastic gradient $g^{i_t}(t)$ in (2.38) and the definition of individual cost function's gradient g in (2.37) we get

$$\mathbb{E}_{I_t}\left[g^{\zeta_{t_t}}(t)\right] = \mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] x(t) - \mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T b^{\zeta_{t_t}}\right] = \frac{1}{N}\left(A^T A x(t) - A^T b\right), \quad \text{(A.38)}$$

where the last inequality follows from the definition of the *uniform* random variable ζ_{t_t}. Upon substituting from above and (A.37) in (A.38) we get

$$\mathbb{E}_{I_t}\left[g^{\zeta_{t_t}}(t)\right] = \frac{1}{N} A^T A z(t). \quad \text{(A.39)}$$

The R.H.S. above is the gradient of the objective cost $\frac{1}{m}\sum_{i=1}^{m} F^i$ evaluated at the current estimate $x(t)$ of (2.6), which we denote by $\nabla F(t)$. Thus,

$$\nabla F(t) = \mathbb{E}_{I_t}\left[g^{\zeta_{t_t}}(t)\right] = \frac{1}{N} A^T A z(t). \quad \text{(A.40)}$$

The above equation means that, the stochastic gradient $g^{\zeta_{t_t}}(t)$ at every iteration $t \geq 0$ is an unbiased estimate of the true gradient $\nabla F(t)$ given the current estimate $x(t)$. Noting that $\left\|A^T A\right\| = s_1$ [45], the above implies that

$$\left\|\nabla F(t)\right\| \leq \frac{s_1}{N}\left\|z(t)\right\|. \quad \text{(A.41)}$$

(c) Assumption 2.5, combined with (A.40) and the definition (2.44), implies that the conditional

second moment of the stochastic gradients satisfies, for each iteration $t = 0, 1, \ldots$,

$$\mathbb{E}_{I_t}\left[\left\|g^{\zeta_{t_t}}(t)\right\|^2\right] \leq V_1 + V_G\left\|\nabla F(t)\right\|^2, \tag{A.42}$$

where $V_G = V_2 + 1$. Since

$$\mathbb{E}_{I_t}\left[\left\|g^{\zeta_{t_t}}(t)\right\|^2\right] = \frac{1}{N}\sum_{i=1}^{N}\left\|g^i(t)\right\|^2,$$

(A.42) implies that, for each $i \in \{1, \ldots, N\}$,

$$\left\|g^i(t)\right\|^2 \leq V_1 N + V_G N\left\|\nabla F(t)\right\|^2. \tag{A.43}$$

Thus, there exist two non-negative real scalar values $E_1 \geq \sqrt{V_1 N}$ and $E_2 \geq \sqrt{V_G N}$ such that, for each $i \in \{1, \ldots, N\}$,

$$\left\|g^i(t)\right\| \leq E_1 + E_2\left\|\nabla F(t)\right\|. \tag{A.44}$$

A.7.2 Notations

For the positive valued parameters α, δ, and β, let

$$C_1 = \max_{i=1,\ldots,N} \left\| \left(a^i\right)^T a^i - \frac{1}{N} A^T A \right\|, \tag{A.45}$$

$$\mu = \left(1 - \frac{2\alpha s_d}{N}(1 - \alpha L)\right), \tag{A.46}$$

$$\varrho = \left\| I - \alpha \left(\frac{1}{N} A^T A + \beta I \right) \right\|, \tag{A.47}$$

$$C_4(t) = (V_2 + 1)\frac{s_1^2}{N} \left(dC_3 + \left\|K_\beta\right\|^2 + 2C_2 \left\|K_\beta\right\| \sum_{j=0}^{t} \rho^j + \left\|\widetilde{K}(0)\right\|_F^2 \mu^{t+1} + 2\left\|K_\beta\right\| \left\|\widetilde{K}(0)\right\| \rho^{t+1} \right), \tag{A.48}$$

$$C_5(t) = 2C_1 E_2 \frac{s_1}{N} \left(\left\|K_\beta\right\| + \left\|\widetilde{K}(0)\right\| \varrho^t \right), \tag{A.49}$$

$$C_6(t) = \frac{2s_d}{s_d + N\beta} - 2\frac{s_1}{N} \left\|\widetilde{K}(0)\right\| \varrho^{t+1}, \tag{A.50}$$

$$C_7(t) = 2C_1 E_1 \left(\left\|K_\beta\right\| + \left\|\widetilde{K}(0)\right\| \varrho^t \right), \tag{A.51}$$

$$C_8(t) = C_4(t) + 0.5, \tag{A.52}$$

$$R_3(t) = \delta^2 V_1 N \left(dC_3 + \left\|K_\beta\right\|^2 + 2C_2 \left\|K_\beta\right\| \sum_{j=0}^{t} \rho^j + \left\|\widetilde{K}(0)\right\|_F^2 \mu^{t+1} + 2\left\|K_\beta\right\| \left\|\widetilde{K}(0)\right\| \rho^{t+1} \right), \tag{A.53}$$

$$R_2(t) = R_3(t) + \frac{1}{2}\alpha^2 C_7(t)^2, \tag{A.54}$$

$$\overline{\delta}(t) = \min\{\frac{1}{C_6(t)}, \frac{C_6(t) - \alpha C_5(t)}{C_8(t)}\}. \tag{A.55}$$

A.7.3 Convergence of the pre-conditioner matrix

We present below a result regarding the convergence of the iterative pre-conditioner matrix in Algorithm 2. To present the convergence result of the pre-conditioner matrix $K(t)$, we recall some notation from Section 2.5.3.

- Recall that $a^i \in \mathbb{R}^{1\times d}$ and $b^i \in \mathbb{R}$ respectively denote each row $i \in \{1,\ldots,N\}$ of the collective input matrix A and the collective output vector B.

- For the positive valued parameters α and β, recall from Table 2.2 that

$$C_2 = \alpha \, \frac{1}{N} \sum_{i=1}^{N} \left\| \left(a^i\right)^T a^i - \frac{1}{N} A^T A \right\| \left\| K_\beta \right\| .$$

- For positive values of the parameters α and β, recall from Table 2.2 that

$$\rho = \frac{1}{N} \sum_{i=1}^{N} \left\| I - \alpha \left(\left(a^i\right)^T a^i + \beta I \right) \right\| .$$

- For each iteration $t \geq 0$, let

$$\widetilde{K}(t) = K(t) - K_\beta. \tag{A.56}$$

- Recall that for each $i \in \{1,\ldots,N\}$, Λ_i and λ_i respectively denote the largest and the smallest eigenvalue of the positive semi-definite matrix $\left(a^i\right)^T a^i$. Thus, $\Lambda_i, \lambda_i > 0$.

Lemma A.2. *Consider Algorithm 2 with parameter $\beta > 0$. For each iteration $t \geq 0$, if $0 < \alpha <$*

$\min_{i=1,\ldots,N}\left\{\frac{2}{\Lambda_i+\beta}\right\}$ *then* $\rho < 1$ *and*

$$\mathbb{E}_t\left[\left\|\widetilde{K}(t+1)\right\|\right] \leq \rho^{t+1}\left\|\widetilde{K}(0)\right\| + C_2\sum_{j=0}^{t}\rho^j. \tag{A.57}$$

Proof. Consider an arbitrary iteration $t \geq 0$.

- Recall from Section 2.5.2 that

$$I_t = \{1_t, \ldots m_t\} \cup \{\zeta_t\}.$$

- Recall from Section 2.5.2 that for each iteration $t \geq 0$ and agent $i \in \{1, \ldots, m\}$, $\mathbb{E}_{i_t}[\cdot]$ denotes the conditional expectation of a function the random variable i_t given the current estimate $x(t)$ and the current pre-conditioner $K(t)$. Similarly, for each iteration $t \geq 0$, $\mathbb{E}_{\zeta_t}[\cdot]$ denotes the conditional expectation of a function the random variable ζ_t given the current estimate $x(t)$ and the current pre-conditioner $K(t)$. Recall from Section 2.5.2 that

$$\mathbb{E}_{I_t}[\cdot] = \mathbb{E}_{1_t,\ldots m_t,\zeta_t}(\cdot).$$

- Recall from (2.43) that $\mathbb{E}_t[\cdot]$ denotes the total expectation of a function of the collection of the random variables $\{I_0, \ldots, I_t\}$ given the initial estimate $x(0)$ and initial pre-conditioner matrix $K(0)$. Specifically,

$$\mathbb{E}_t[\cdot] = \mathbb{E}_{I_0,\ldots,I_t}(\cdot).$$

From Step 2 and Step 4 of Algorithm 2, the random variable ζ_{t_t} is *uniformly* distributed in the set $\{1, \ldots, N\}$ that denotes the total number of data points in (A, b). Moreover, $(a^{\zeta_{t_t}}, b^{\zeta_{t_t}})$ is a data point *uniformly* and independently drawn at random from (A, b).

For each iteration $t = 0, 1, \ldots$, upon substituting from (2.39) and (2.40) in (2.41) we have, for each column index $j = 1, \ldots, d$ of the matrix $K(t)$,

$$k_j(t+1) = k_j(t) - \alpha \left(\left(\left(a^{\zeta_{t_t}} \right)^T a^{\zeta_{t_t}} + \beta I \right) k_j(t) - e_j \right). \tag{A.58}$$

Recall the definition of K_β in Table 2.2. Let, $k_{j\beta}$ denote the j-th column of K_β. Then for each column $j = 1, \ldots, d$ of K_β we obtain that

$$\left(\frac{1}{N} A^T A + \beta I \right) k_{j\beta} = e_j. \tag{A.59}$$

Recall the definition (A.56),

$$\widetilde{K}(t) = K(t) - K_\beta, \quad \forall t \geq 0.$$

Let $\widetilde{k}_j(t)$ denote the j-th column of $\widetilde{K}(t)$. Subtracting $k_{j\beta}$ from both sides of (A.58), from the definition of $\widetilde{k}_j(t)$ we have for each $j = 1, \ldots, d$,

$$\begin{aligned}
\widetilde{k}_j(t+1) &= \widetilde{k}_j(t) - \alpha \left(\left(a^{\zeta_{t_t}} \right)^T a^{\zeta_{t_t}} + \beta I \right) \left(\widetilde{k}_j(t) + k_{j\beta} \right) + \alpha e_j \\
&\overset{\text{(A.59)}}{=} \widetilde{k}_j(t) - \alpha \left(\left(a^{\zeta_{t_t}} \right)^T a^{\zeta_{t_t}} + \beta I \right) \left(\widetilde{k}_j(t) + k_{j\beta} \right) + \alpha \left(\frac{1}{N} A^T A + \beta I \right) k_{j\beta} \\
&= \left(I - \alpha \left(\left(a^{\zeta_{t_t}} \right)^T a^{\zeta_{t_t}} + \beta I \right) \right) \widetilde{k}_j(t) - \alpha \left(\left(a^{\zeta_{t_t}} \right)^T a^{\zeta_{t_t}} - \frac{1}{N} A^T A \right) k_{j\beta}.
\end{aligned}$$

Upon horizontally concatenating the columns $\{\widetilde{k}_j(t),\ j=1,\ldots,d\}$, from above we get

$$\widetilde{K}(t+1)=\left(I-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}+\beta I\right)\right)\widetilde{K}(t)-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\frac{1}{N}A^TA\right)K_\beta. \quad \text{(A.60)}$$

Using triangle inequality on the R.H.S. of (A.60) we get

$$\left\|\widetilde{K}(t+1)\right\|\leq\left\|\left(I-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}+\beta I\right)\right)\widetilde{K}(t)\right\|+\alpha\left\|\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\frac{1}{N}A^TA\right)K_\beta\right\|.$$

From the definition of induced 2-norm of matrix [45],

$$\left\|\widetilde{K}(t+1)\right\|\leq\left\|I-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}+\beta I\right)\right\|\left\|\widetilde{K}(t)\right\|+\alpha\left\|\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\frac{1}{N}A^TA\right\|\|K_\beta\|.$$

Upon taking conditional expectation $\mathbb{E}_{I_t}[\cdot]$ on both sides, given the current matrix $K(t)$ and the estimate $x(t)$, we get

$$\begin{aligned}\mathbb{E}_{I_t}\left[\left\|\widetilde{K}(t+1)\right\|\right]\leq&\mathbb{E}_{I_t}\left[\left\|I-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}+\beta I\right)\right\|\right]\left\|\widetilde{K}(t)\right\|\\&+\alpha\mathbb{E}_{I_t}\left[\left\|\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\frac{1}{N}A^TA\right\|\right]\|K_\beta\|. \quad \text{(A.61)}\end{aligned}$$

Since the random variable ζ_{t_t} is *uniformly* distributed in $\{1,\ldots,N\}$, we have

$$\begin{aligned}\mathbb{E}_{I_t}\left[\left\|I-\alpha\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}+\beta I\right)\right\|\right]&=\frac{1}{N}\sum_{i=1}^{N}\left\|I-\alpha\left(\left(a^i\right)^T a^i+\beta I\right)\right\|,\\ \mathbb{E}_{I_t}\left[\left\|\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\frac{1}{n}A^TA\right\|\right]&=\frac{1}{N}\sum_{i=1}^{N}\left\|\left(a^i\right)^T a^i-\frac{1}{N}A^TA\right\|.\end{aligned}$$

Upon substituting from above in (A.61) we obtain that

$$\mathbb{E}_{I_t}\left[\left\|\widetilde{K}(t+1)\right\|\right] \leq \frac{1}{N}\sum_{i=1}^{N}\left\|I-\alpha\left(\left(a^i\right)^T a^i+\beta I\right)\right\|\left\|\widetilde{K}(t)\right\| + \alpha\frac{1}{N}\sum_{i=1}^{N}\left\|\left(a^i\right)^T a^i-\frac{1}{N}A^T A\right\|\left\|K_\beta\right\|. \tag{A.62}$$

Recall from Section A.7.3 that Λ_i and λ_i respectively denote the largest and the smallest eigenvalue of each $\left(a^i\right)^T a^i$. Since $\left(\left(a^i\right)^T a^i+\beta I\right)$ is positive definite for $\beta > 0$, for each value of α satisfying $0<\alpha<\min_{i=1,\ldots,N}\left\{\frac{2}{\Lambda_i+\beta}\right\}$ we have [21]

$$\left\|I-\alpha\left(\left(a^i\right)^T a^i+\beta I\right)\right\| = \max\left\{\left|1-\alpha\left(\Lambda_i+\beta\right)\right|,\left|1-\alpha\left(\lambda_i+\beta\right)\right|\right\}<1,\ i=1,\ldots,N.$$

Using the definitions of C_2 and ρ in (A.62) we then have $\rho<1$ such that

$$\mathbb{E}_{I_t}\left[\left\|\widetilde{K}(t+1)\right\|\right] \leq \rho\left\|\widetilde{K}(t)\right\|+C_2. \tag{A.63}$$

Iterating the above from t to 0, by the law of total expectation we have

$$\mathbb{E}_{t}\left[\left\|\widetilde{K}(t+1)\right\|\right] \leq \rho^{t+1}\left\|\widetilde{K}(0)\right\|+C_2\sum_{j=0}^{t}\rho^j. \tag{A.64}$$

Hence, the proof. □

Lemma A.2 implies that, for sufficiently small value of the parameter α at every iteration, the iterative pre-conditioner matrix $K(t)$ in Algorithm 2 converges *linearly* in expectation to a

neighborhood of the matrix K_β. Since $\rho \in (0,1)$, this neighborhood is characterized from (A.57):

$$\lim_{t\to\infty} \mathbb{E}_t \left[\left\| \widetilde{K}(t) \right\| \right] \leq \frac{C_2}{1-\rho}.$$

From (A.57), smaller value of the parameter ρ implies faster convergence of the pre-conditioner matrix. However, the final error in $K(t)$ is large if ρ is small. Thus, there is a trade-off between the rate of convergence and the final error regarding the convergence of the pre-conditioner matrix in Algorithm 2. From Table 2.2, in the deterministic case, the value of $C_2 = 0$, which means that the sequence of pre-conditioner $\{K(t),\ t = 0, 1, \ldots\}$ converges exactly to K_β in this case.

A.7.4 Proof of the theorem

Here, we formally prove Theorem 2.5.

A.7.4.1 Proof of Part (ii) of Theorem 2.5

Step I: Upon subtracting x^* from both sides of (2.42) and using the definition of $z(t)$ in (2.21), we have

$$z(t+1) = z(t) - \delta\ K(t+1)\ g^{\zeta_{t_t}}(t).$$

The above implies that

$$\left\| z(t+1) \right\|^2 = \left\| z(t) \right\|^2 + \delta^2 \left\| K(t+1) g^{\zeta_{t_t}}(t) \right\|^2 - 2\delta z(t)^T K(t+1) g^{\zeta_{t_t}}(t).$$

Upon taking conditional expectation $\mathbb{E}_{I_t}[\cdot]$ on both sides above, given the current pre-conditioner matrix $K(t)$ and estimate $x(t)$, we have

$$\mathbb{E}_{I_t}\left[\left\|z(t+1)\right\|^2\right] = \left\|z(t)\right\|^2 + \delta^2 \mathbb{E}_{I_t}\left[\left\|K(t+1)g^{\zeta_{t_t}}(t)\right\|^2\right] - 2\delta z(t)^T \mathbb{E}_{I_t}\left[K(t+1)g^{\zeta_{t_t}}(t)\right]. \tag{A.65}$$

Consider the expression $z(t)^T \mathbb{E}_{I_t}\left[K(t+1)g^{\zeta_{t_t}}(t)\right]$ above. Upon substituting from (A.56),

$$\begin{aligned} z(t)^T \mathbb{E}_{I_t}\left[K(t+1)g^{\zeta_{t_t}}(t)\right] &= z(t)^T \mathbb{E}_{I_t}\left[\widetilde{K}(t+1)g^{\zeta_{t_t}}(t)\right] + z(t)^T K_\beta \mathbb{E}_{I_t}\left[g^{\zeta_{t_t}}(t)\right] \\ &\overset{(A.40)}{=} z(t)^T \mathbb{E}_{I_t}\left[\widetilde{K}(t+1)g^{\zeta_{t_t}}(t)\right] + z(t)^T K_\beta \nabla F(t). \end{aligned} \tag{A.66}$$

Consider the following equation

$$\begin{aligned} z(t)^T \mathbb{E}_{I_t}\left[\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\left(\nabla F(t) - g^{\zeta_{t_t}}(t)\right)\right] &= z(t)^T \mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right] \mathbb{E}_{I_t}\left[\left(\nabla F(t) - g^{\zeta_{t_t}}(t)\right)\right] \\ &\overset{(A.40)}{=} 0. \end{aligned}$$

Owing to the above, we have the first expression in the R.H.S. of (A.66) as

$$
\begin{aligned}
&z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)g^{\zeta_{t_t}}(t)\right]\\
=&z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t)-z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\left(\nabla F(t)-g^{\zeta_{t_t}}(t)\right)\right]\\
&+z(t)^T\mathbb{E}_{I_t}\left[\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\left(\nabla F(t)-g^{\zeta_{t_t}}(t)\right)\right]\\
=&z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t)-z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)-\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\right]\nabla F(t)\\
&-z(t)^T\mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]\\
=&z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t)-z(t)^T\mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]. \qquad \text{(A.67)}
\end{aligned}
$$

From (A.36) we have the second expression in the R.H.S. above as

$$
\begin{aligned}
&z(t)^T\mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]\\
&=\alpha z(t)^T\mathbb{E}_{I_t}\left[\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)K(t)g^{\zeta_{t_t}}(t)\right].
\end{aligned}
$$

Applying Cauchy-Schwartz inequality above we have

$$
\begin{aligned}
&z(t)^T\mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]\\
&\leq\alpha\|z(t)\|\left\|\mathbb{E}_{I_t}\left[\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)K(t)g^{\zeta_{t_t}}(t)\right]\right\|.
\end{aligned}
$$

Using Jensen's inequality on the convex function $\|\cdot\|$, from above we have

$$z(t)^T \mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]$$
$$\leq \alpha\|z(t)\|\,\mathbb{E}_{I_t}\left[\left\|\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)K(t)g^{\zeta_{t_t}}(t)\right\|\right]. \tag{A.68}$$

From the definition of induced 2-norm of matrix we have

$$\left\|\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right)K(t)g^{\zeta_{t_t}}(t)\right\|$$
$$\leq\left\|\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}-\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right]\right\|\|K(t)\|\left\|g^{\zeta_{t_t}}(t)\right\| \overset{(A.45)}{\leq} C_1\|K(t)\|\left\|g^{\zeta_{t_t}}(t)\right\|.$$

From the definition of expectation and substituting from above in (A.68) we get

$$z(t)^T\mathbb{E}_{I_t}\left[\left(\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]-\widetilde{K}(t+1)\right)g^{\zeta_{t_t}}(t)\right]\leq \alpha C_1\|z(t)\|\|K(t)\|\,\mathbb{E}_{I_t}\left[\left\|g^{\zeta_{t_t}}(t)\right\|\right]$$
$$\overset{(A.44)}{\leq} \alpha C_1\|z(t)\|\|K(t)\|\left(E_1+E_2\|\nabla F(t)\|\right). \tag{A.69}$$

Upon substituting from (A.66), (A.67) and (A.69) in (A.65) we obtain that

$$\mathbb{E}_{I_t}\left[\|z(t+1)\|^2\right]\leq\|z(t)\|^2+\delta^2\mathbb{E}_{I_t}\left[\left\|K(t+1)g^{\zeta_{t_t}}(t)\right\|^2\right]-2\delta z(t)^T K_\beta\nabla F(t)$$
$$-2\delta z(t)^T\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t)+2\delta\alpha C_1\|z(t)\|\|K(t)\|\left(E_1+E_2\|\nabla F(t)\|\right). \tag{A.70}$$

In the following Steps II-V, we bound the expressions in the R.H.S. above.

Step II: In this step, we bound the second expression $\delta^2 \mathbb{E}_{I_t}\left[\left\|K(t+1)g^{\zeta_{t_t}}(t)\right\|^2\right]$ in the R.H.S. of (A.70). Consider the expression $\mathbb{E}_{I_t}\left[\left\|K(t+1)g^{\zeta_{t_t}}(t)\right\|^2\right]$. Using the definition of induced 2-norm and expectation we have

$$\mathbb{E}_{I_t}\left[\left\|K(t+1)g^{\zeta_{t_t}}(t)\right\|^2\right] \leq \mathbb{E}_{I_t}\left[\left\|K(t+1)\right\|^2\left\|g^{\zeta_{t_t}}(t)\right\|^2\right]$$
$$\overset{(A.43)}{\leq} \left(V_1 N + V_G N\left\|\nabla F(t)\right\|^2\right)\mathbb{E}_{I_t}\left[\left\|K(t+1)\right\|^2\right]. \tag{A.71}$$

Using triangle inequality on induced 2-norm in (A.56) we get

$$\left\|K(t)\right\| \leq \left\|\widetilde{K}(t)\right\| + \left\|K_\beta\right\|, \forall t \geq 0. \tag{A.72}$$

From (A.72) and the definition of expectation,

$$\mathbb{E}_{I_t}\left[\left\|K(t+1)\right\|^2\right] \leq \mathbb{E}_{I_t}\left[\left\|\widetilde{K}(t+1)\right\|^2\right] + \left\|K_\beta\right\|^2 + 2\left\|K_\beta\right\|\mathbb{E}_{I_t}\left[\left\|\widetilde{K}(t+1)\right\|\right]. \tag{A.73}$$

In the rest of this step, we bound the first expression above, which in turn bounds the R.H.S. of (A.71). Note that, the third expression above has already been bounded in (A.63).

For each $i \in \{1, \ldots, N\}$ and each $j \in \{1, \ldots, d\}$, define a function $h_j^i : \mathbb{R}^d \to \mathbb{R}$ such that

$$h_j^i(x) = \frac{1}{2}x^T\left(\left(a^i\right)^T a^i + \beta I\right)x - x^T e_j. \tag{A.74}$$

The gradient of h_j^i is given by

$$\nabla h_j^i(x) = \left(\left(a^i\right)^T a^i + \beta I\right) x - e_j,\ \forall x \in \mathbb{R}^d. \tag{A.75}$$

Note that,

$$\mathbb{E}_{I_t}\left[\nabla h_j^{\zeta_{t_t}}(x)\right] = \left(\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] + \beta I\right) x - e_j = \left(\frac{1}{N}A^T A + \beta I\right) x - e_j,\ \forall x. \tag{A.76}$$

Hence, (A.58) is a stochastic gradient descent update with stepsize α on the objective cost function $H_j : \mathbb{R}^d \to \mathbb{R}$ defined by $H_j(x) = \frac{1}{2}x^T\left(\frac{1}{N}A^T A + \beta I\right)x - x^T e_j$. Under Assumption 1, the matrix $A^T A$ is symmetric positive definite. Equivalently, H_j is $\frac{s_d}{N}$-strongly convex. Also note that, each ∇h_j^i is Lipschitz with a Lipschitz constant $L_i = \left\|\left(a^i\right)^T a^i + \beta I\right\| = \Lambda_i + \beta$.

Upon substituting from (A.59) into (A.76) we have $\mathbb{E}_{I_t}\left[\nabla h_j^{\zeta_{t_t}}(k_{j\beta})\right] = 0_d$, where 0_d denotes the origin of $\mathbb{R}^d$. For each $j = 1, \ldots, d$, we define a quantity

$$\sigma_j^2 = \max_{t\geq 0} \mathbb{E}_{I_t}\left[\left\|\nabla h_j^{\zeta_{t_t}}(k_{j\beta})\right\|^2\right] \tag{A.77}$$

Upon substituting above from (A.75) and (A.59) we get

$$\sigma_j^2 = \max_{t\geq 0} \mathbb{E}_{I_t}\left[\left\|\left(\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}} + \beta I\right) K_\beta e_j - e_j\right\|^2\right] = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\left(a^i\right)^T a^i + \beta I\right) K_\beta e_j - e_j\right\|^2.$$

For each $j = 1, \ldots, d$, let $\widetilde{k}_j(t) = k_j(t) - k_{j\beta}$. Then, if $\alpha < \min\left\{\frac{N}{s_d}, \frac{1}{L}\right\}$, we have for each

$j = 1, \ldots, d$, [227]

$$\mathcal{E}_t \left[\left\| \widetilde{k}_j(t+1) \right\|^2 \right] \leq \mu^{t+1} \left\| \widetilde{k}_j(0) \right\|^2 + C_3. \tag{A.78}$$

where $\mu \in (0, 1)$ (see (A.46)). Then,

$$\left\| \widetilde{K}(t) \right\|^2 \leq \left\| \widetilde{K}(t) \right\|_F^2 = \sum_{j=1}^{d} \left\| \widetilde{k}_j(t) \right\|^2,$$

which implies that

$$\mathbb{E}_{I_t} \left[\left\| \widetilde{K}(t) \right\|^2 \right] \leq \sum_{j=1}^{d} \mathbb{E}_{I_t} \left[\left\| \widetilde{k}_j(t) \right\|^2 \right].$$

Taking the total expectation above and substituting from (A.78) we obtain that

$$\mathbb{E}_t \left[\left\| \widetilde{K}(t+1) \right\|^2 \right] \leq \mu^{t+1} \left\| \widetilde{K}(0) \right\|_F^2 + dC_3. \tag{A.79}$$

Upon substituting from (A.57) and (A.79) in (A.73) we have

$$\mathbb{E}_t \left[\left\| K(t+1) \right\|^2 \right] \leq \mu^{t+1} \left\| \widetilde{K}(0) \right\|_F^2 + dC_3 + \left\| K_\beta \right\|^2 + 2 \left\| K_\beta \right\| \left(\rho^{t+1} \left\| \widetilde{K}(0) \right\| + C_2 \sum_{j=0}^{t} \rho^j \right).$$

Upon substituting from above in (A.71) we obtain that

$$\mathbb{E}_t\left[\left\|K(t+1)g^{i_t}(t)\right\|^2\right] \leq (V_1N + V_GN\left\|\nabla F(t)\right\|^2)(\mu^{t+1}\left\|\widetilde{K}(0)\right\|_F^2 + dC_3 + \left\|K_\beta\right\|^2 + 2\left\|K_\beta\right\|\left(\rho^{t+1}\left\|\widetilde{K}(0)\right\| + C_2\sum_{j=0}^{t}\rho^j\right)). \tag{A.80}$$

Step III: In this step, we bound the the third expression $-z(t)^TK_\beta\nabla F(t)$ in the R.H.S. of (A.70). From eigen value decomposition [45], $A^TA = VSV^T$ where $S = Diag\,(s_1,\ldots,s_d)$, and the matrix V constitutes of orthonormal eigen vectors of A^TA. From above,

$$\left(\frac{1}{N}A^TA + \beta I\right)^{-1}A^TA = VDiag\left(\frac{s_1}{(s_1/N)+\beta},\ldots,\frac{s_d}{(s_d/N)+\beta}\right)V^T.$$

Since $K_\beta = \left(\frac{1}{N}A^TA + \beta I\right)^{-1}$, the eigenvalues of $K_\beta A^TA$ are given by $\left\{\frac{s_i}{(s_i/N)+\beta}|\, i = 1,\ldots,d\right\}$. Thus, from the bound on Rayleigh quotient [45]

$$-z(t)^TK_\beta A^TAz(t) \leq -\lambda_{min}(K_\beta A^TA)\left\|z(t)\right\|^2, \tag{A.81}$$

where $\lambda_{min}(K_\beta A^TA) = \frac{s_d}{(s_d/N)+\beta}$ is the minimum eigen value of $K_\beta A^TA$. Upon substituting from (A.40) in the expression $-z(t)^TK_\beta\nabla F(t)$, we obtain that

$$-z(t)^TK_\beta\nabla F(t) = -\frac{1}{N}z(t)^TK_\beta A^TAz(t) \overset{(A.81)}{\leq} -\frac{1}{N}\lambda_{min}(K_\beta A^TA)\left\|z(t)\right\|^2. \tag{A.82}$$

Step IV: In this step, we bound $\left\|\widetilde{K}(t)\right\|$ which appears in the fifth expression in the R.H.S.

of (A.70). Recall the definition of the *uniform* random variable ζ_{t_t}. Then,

$$\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] = \frac{1}{N}A^T A.$$

Upon substituting from above in (A.35) we have

$$\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right] = \left(I - \alpha\left(\mathbb{E}_{I_t}\left[\left(a^{\zeta_{t_t}}\right)^T a^{\zeta_{t_t}}\right] + \beta I\right)\right)\widetilde{K}(t) = \left(I - \alpha\left(\frac{1}{N}A^T A + \beta I\right)\right)\widetilde{K}(t).$$

Again using the definition of induced matrix 2-norm above, we obtain that

$$\left\|\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\right\| \leq \left\|I - \alpha\left(\frac{1}{N}A^T A + \beta I\right)\right\|\left\|\widetilde{K}(t)\right\|. \tag{A.83}$$

Since $\left(\frac{1}{N}A^T A + \beta I\right)$ is positive definite for $\beta > 0$, for each value of α satisfying $0 < \alpha < \frac{2}{s_1/N+\beta}$ we have $\left\|I - \alpha\left(\frac{1}{N}A^T A + \beta I\right)\right\| = \max\left\{\left|1 - \alpha\left(\frac{s_1}{N} + \beta\right)\right|, \left|1 - \alpha\left(\frac{s_d}{N} + \beta\right)\right|\right\} < 1$ (Fessler, 2020). Using the definition of ϱ (see (A.47)) in (A.83) we then have $\varrho < 1$ such that

$$\left\|\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\right\| \leq \varrho\left\|\widetilde{K}(t)\right\|.$$

Iterating the above from t to 0, the total expectation is given by

$$\left\|\mathcal{E}_t\left[\widetilde{K}(t+1)\right]\right\| \leq \varrho^{t+1}\left\|\widetilde{K}(0)\right\|. \tag{A.84}$$

From the definition of conditional expectation $\mathbb{E}_t[\cdot]$, we have $\mathbb{E}_t\left[\widetilde{K}(t)\right] = \widetilde{K}(t)$. Thus, (A.84)

implies that

$$\left\|\widetilde{K}(t)\right\| \leq \varrho^t \left\|\widetilde{K}(0)\right\|. \tag{A.85}$$

Step V: In this step, we bound the the fourth expression $-z(t)^T \mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t)$ in the R.H.S. of (A.70). Using Cauchy-Schwartz inequality,

$$\begin{aligned} -z(t)^T \mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\nabla F(t) &\leq \left\|z(t)\right\|\left\|\nabla F(t)\right\|\left\|\mathbb{E}_{I_t}\left[\widetilde{K}(t+1)\right]\right\| \\ &\overset{(A.84)}{\leq} \varrho^{t+1}\left\|\widetilde{K}(0)\right\|\left\|z(t)\right\|\left\|\nabla F(t)\right\|. \end{aligned} \tag{A.86}$$

Step VI: In this step, we combine the upper bounds obtained in Step-II to Step-V above to get a bound on the R.H.S. of (A.70) in Step-I. Upon substituting from (A.40), (A.72), (A.80), (A.82), (A.86), (A.85) in (A.70) we obtain that

$$\mathbb{E}_t\left[\left\|z(t+1)\right\|^2\right] \leq \left(1+\delta^2 C_4(t)+\alpha\delta C_5(t)-\delta C_6(t)\right)\left\|z(t)\right\|^2 + \alpha\delta C_7(t)\left\|z(t)\right\| + R_3(t), \tag{A.87}$$

where $C_{4-7}(t)$ and $R_3(t)$ have been defined in (A.48)-(A.53). We apply the AM-GM inequality $2ab \leq (a^2+b^2)$ with $a = \delta\left\|z(t)\right\|$ and $b = \alpha C_7(t)$, which results in

$$\mathbb{E}_t\left[\left\|z(t+1)\right\|^2\right] \leq \left(1+\delta^2(C_4(t)+0.5)+\alpha\delta C_5(t)-\delta C_6(t)\right)\left\|z(t)\right\|^2 + R_3(t) + \frac{1}{2}\alpha^2 C_7(t)^2.$$

From the definitions of $C_8(t)$ in (A.52) and $R_2(t)$ in (A.54), the above can be rewritten as

$$\begin{aligned}\mathbb{E}_t\left[\left\|z(t+1)\right\|^2\right] &\leq \left(1+\delta^2 C_8(t)+\alpha\delta C_5(t)-\delta C_6(t)\right)\left\|z(t)\right\|^2+R_2(t)\\ &= R_1(t)\left\|z(t)\right\|^2+R_2(t).\end{aligned} \tag{A.88}$$

From (A.63), (A.78), and (A.84), we have shown that (A.87), and hence (A.88), hold for any $\delta > 0$ and α satisfying

$$0<\alpha<\min\left\{\frac{N}{s_d},\frac{1}{L},\frac{2}{(s_1/N)+\beta},\frac{2}{\Lambda_i+\beta}|i=1,\ldots,N\right\}.$$

By definition, $L=\max_{i=1,\ldots,N}\{\Lambda_i+\beta\}$. Thus, the above implies that (A.88) holds for

$$0<\alpha<\min\left\{\frac{N}{s_d},\frac{1}{L},\frac{2}{(s_1/N)+\beta}\right\},\ \delta>0. \tag{A.89}$$

Then, (A.88) completes the first part of the proof.

A.7.4.2 Proof of Part (i) of Theorem 2.5

Next, we show that the value of $R_1(t)$ is between 0 and 1 after some finite iteration if the parameter δ satisfies an additional criterion. Note that, the values of $C_5(t)$, $C_6(t)$ and $C_8(t)$ implicitly depend on α, but independent of δ. For any $\alpha > 0$ consider the function $r_\alpha : \mathcal{Z}^+ \to \mathbb{R}$ defined as

$$r_\alpha(t)=\frac{C_6(t)}{\alpha C_5(t)}=\frac{\frac{s_d}{s_d+N\beta}-\frac{s_1}{N}\left\|\widetilde{K}(0)\right\|\varrho^{t+1}}{\alpha C_1 E_2\frac{s_1}{N}\left(\left\|K_\beta\right\|+\left\|\widetilde{K}(0)\right\|\varrho^t\right)}.$$

From the definition, r_α is a strictly increasing function. As $\rho \in (0,1)$, we have the limit of r_α as

$$\lim_{t\to\infty} r_\alpha(t) = \frac{C_6(t)}{\alpha C_5(t)} = \frac{\frac{s_d}{s_d+N\beta}}{\alpha C_1 E_2 \frac{s_1}{N}\left\|K_\beta\right\|} > 0,$$

where the last inequality holds under Assumption 1. Since $r_\alpha(t)$ is strictly increasing with t with a positive limit as $t \to \infty$, there exists $0 \leq T < \infty$ such that $r_\alpha(t) > 0$ for all $t \geq T$. Since the above holds for any $\alpha > 0$, it also holds for the values of α satisfying (A.89). Hence, for any value of the parameter α in (A.89) there exists $T < \infty$ such that

$$C_6(t) - \alpha C_5(t) > 0, \ \forall t \geq T.$$

Now consider any iteration $t \geq T$. If $0 < \delta < \frac{C_6(t)-\alpha C_5(t)}{C_8(t)}$, then we have

$$\delta C_8(t) + \alpha C_5(t) < C_6(t) \implies 1 + \delta^2 C_8(t) + \delta\alpha C_5(t) - \delta C_6(t) < 1.$$

Additionally, if $0 < \delta < \frac{1}{C_6(t)}$, then

$$\delta C_6(t) < 1 \implies 1 - \delta C_6(t) > 0 \implies 1 + \delta^2 C_8(t) + \delta\alpha C_5(t) - \delta C_6(t) > 0.$$

We combine the above two results to conclude the proof. If α satisfies (A.89) then there exists $T < \infty$ such that for any iteration $t \geq T$, if $\delta < \min\{\frac{1}{C_6(t)}, \frac{C_6(t)-\alpha C_5(t)}{C_8(t)}\}$ then

$$\left(1 + \delta^2 C_8(t) + \delta\alpha C_5(t) - \delta C_6(t)\right) \in (0,1).$$

From (A.55), note that the bound on δ above is $\overline{\delta}$. Thus, the proof of the second part of the theorem is complete.

A.7.4.3 Proof of Part (iii) of Theorem 2.5

From Part (ii) of Theorem 2.5, we have $0 < R_1(t) < 1$ for all $t \geq T$. Thus, upon retracing (2.45) from t to 0, we obtain

$$\lim_{t\to\infty} \mathbb{E}_t \left[\left\| z(t+1) \right\|^2 \right] \leq R_2(\infty).$$

Upon substituting from (A.51) in the definition of $R_2(t)$ in (A.54), we get

$$R_2(\infty) = \lim_{t\to\infty} \delta^2 V_1 N \left(dC_3 + \left\| K_\beta \right\|^2 + 2C_2 \left\| K_\beta \right\| \sum_{j=0}^{t} \rho^j + \left\| \widetilde{K}(0) \right\|_F^2 \mu^{t+1} + 2 \left\| K_\beta \right\| \left\| \widetilde{K}(0) \right\| \rho^{t+1} \right)$$
$$+ \frac{1}{2}\alpha^2 \left(2C_1 E_1 \left(\left\| K_\beta \right\| + \left\| \widetilde{K}(0) \right\| \varrho^t \right) \right)^2.$$

From the proof of Part (i) of Theorem 2.5, the values of ρ, μ, ϱ are within the range $(0, 1)$. Then,

$$R_2(\infty) = \delta^2 V_1 N \left(dC_3 + \left\| K_\beta \right\|^2 + \frac{2C_2 \left\| K_\beta \right\|}{1-\rho} + \right) + 2\alpha^2 \left(C_1 E_1 \left\| K_\beta \right\| \right)^2.$$

Hence, the proof.

A.8 Proof of Lemma 3.1

Consider the linear dynamics $\dot{y}(s) = -My(s)$ with initialization $y(0) \in \mathbb{R}^{md}$ and $s \geq 0$. If 0_{md} is the exponentially stable equilibrium point of the above dynamics, then all eigenvalues of

M must have positive real parts. Next, we prove that the above dynamics is exponentially stable.

We let $y^i(s) \in \mathbb{R}^d$ denote the i-th block-row of a vector $y \in \mathbb{R}^{md}$, such that $y(s) = \left[y^1(s)^T, \ldots, y^m(s)^T\right]^T$. Recall the definition of the matrix K^i in (3.5). We define the Lyapunov function candidate $V : \mathbb{R}^{md} \to \mathbb{R}$ such that

$$V(y) = \frac{1}{2}\sum_{i=1}^{m}(y^i)^T \left(K^i\right)^{-1} y^i,\ y \in \mathbb{R}^{md}. \tag{A.90}$$

Clearly, $V(y) > 0$ for any $y \in \mathbb{R}^{md}$ such that $y \neq 0_{md}$, and $V(0_{md}) = 0$. Also, $\lim_{\|y\|\to\infty} V(y) \to \infty$. Upon differentiating both sides of (A.90) with respect to s, along the trajectory of $\dot{y}(s) = -My(s), y(0) \in \mathbb{R}^{md}$ we have

$$\dot{V}(y(s)) = \sum_{i=1}^{m} y^i(s)^T \left(K^i\right)^{-1} \dot{y}^i(s) = -\sum_{i=1}^{m} y^i(s)^T \left(K^i\right)^{-1} M^i y(s).$$

Upon substituting above from (3.9),

$$\dot{V}(y(s)) = -\sum_{i=1}^{m} y^i(s)^T \left((\alpha \left(a^i\right)^T a^i + \left|\mathcal{N}^i\right| I) y^i(s) - \sum_{j\in\mathcal{N}^i} y^j(s) \right).$$

Since the graph $\mathbb{G}$ is undirected, from above we obtain that

$$\dot{V}(y(s)) = -\alpha \sum_{i=1}^{m} \left\| A^i y^i(s) \right\|^2 - \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \left\| y^i(s) - y^j(s) \right\|^2 \leq 0. \tag{A.91}$$

Let $\mathcal{N}(P)$ denote the nullspace of a matrix P: $\mathcal{N}(P) = \left\{v \in \mathbb{R}^{md} : Pv = 0_{md}\right\}$. Define the matrix $A = \left[(A^1)^T, \ldots, (A^m)^T\right]^T$. Assumption 3.1 implies that the matrix A is full-rank. Then, $\mathcal{N}(A) = 0_d$. Under Assumption 3.2, (A.91) implies that $\dot{V}(y(s)) = 0 \iff \exists y_c \in$

$\mathbb{R}^d$ such that $y^i(s) = y_c$, $y^i(s) \in \mathcal{N}(A^i)$, $\forall i \in \{1, \ldots, m\}$. So, $\dot{V}(y(s)) = 0 \iff y^i(s) = 0_d$, $\forall i \in \{1, \ldots, m\} \iff y(s) = 0_{md}$. Hence, we have that $\dot{V}(y(s)) < 0$ for $y(s) \neq 0_{md}$. Then, according to Lyapunov's stability theorem [228], 0_{md} is the exponentially stable equilibrium point of the system $\dot{y}(s) = -My(s)$, $y(0) \in \mathbb{R}^{md}$, and therefore, the eigenvalues of M have positive real parts. Now we consider the expression of M in (A.96). From the argument following (A.97), the eigenvalues of M are real. Thus, M has positive real eigenvalues.

A.9 Proof of Theorem 3.1

Throughout this proof, we assume that, for any $i \in \{1, \ldots, m\}$, $\left(A^i\right)^T A^i$ is not the trivial zero matrix, and $\alpha, \delta > 0$. Consider an arbitrary iteration $t \geq 0$. Define the estimation error at iteration t as

$$z(t) = \left[z^1(t)^T, \ldots, z^m(t)^T\right]^T = \left[(x^1(t) - x^*)^T, \ldots, (x^m(t) - x^*)^T\right]^T. \tag{A.92}$$

Upon substituting above from (3.6) we have $z^i(t+1) = z^i(t) - \alpha\delta K^i \left(A^i\right)^T \left(A^i\, z^i(t) + A^i\, x^* - b^i\right) + \delta K^i \sum_{j \in \mathcal{N}^i} \left(z^j(t) - z^i(t)\right)$. Upon substituting above from the definition of x^* in (3.1),

$$z^i(t+1) = z^i(t) - \alpha\delta K^i \left(A^i\right)^T A^i z^i(t) + \delta K^i \sum_{j \in \mathcal{N}^i} \left(z^j(t) - z^i(t)\right). \tag{A.93}$$

Recall the definition of M from (3.8)-(3.9). Upon substituting from above in (A.92), the estimation error at iteration $(t+1)$ is

$$z(t+1) = (I - \delta M)\, z(t). \tag{A.94}$$

We denote by $Diag(.)$ a block-diagonal matrix of appropriate dimensions, with the arguments denoting the block-diagonal entries in the same order. Recall that L is the *Laplacian* matrix of graph $\mathbb{G}$. We let $\otimes$ denote the Kronecker product of two matrices. For simplicity of notations, we define

$$K = Diag\left(\left\{K^i\right\}_{i=1}^{m}\right), \Gamma = Diag\left(\left\{\left(A^i\right)^T A^i\right\}_{i=1}^{m}\right). \tag{A.95}$$

Upon substituting from (A.95) in the definition of M in (3.8),

$$M = \alpha K\Gamma + K\left(L \otimes I\right). \tag{A.96}$$

For an md-dimensional arbitrary vector $v \neq 0_{md}$, consider the quadratic form of M:

$$\phi(v) = v^T M v = \alpha\, v^T K\Gamma v + v^T K\left(L \otimes I\right) v. \tag{A.97}$$

Since L is symmetric positive semi-definite [75], $L \otimes I$ is also symmetric positive semi-definite. From the definition of K^i in (3.5) and K in (A.95), we have that K is symmetric positive definite. Thus, the matrix product $K\left(L \otimes I\right)$ has non-negative eigenvalues, and $v^T K\left(L \otimes I\right) v \geq 0$. Similarly, since K is symmetric positive definite and Γ is symmetric positive semi-definite, $K\Gamma$ has non-negative eigenvalues, and $v^T K\Gamma v \geq 0$. From Lemma 3.1, the eigenvalues of M are positive. Thus, $\phi(v) > 0$. Therefore, the following three cases are possible.

Case-I: $v^T K\Gamma v = 0$. Since $\phi(v) > 0$, we have

$$v^T K\left(L \otimes I\right) v > 0 \iff v^T K\left(L \otimes I\right) v \geq \underline{\lambda}_{\min}(K\left(L \otimes I\right))\|v\|^2$$
$$\overset{(A.97)}{\Longrightarrow} \phi(v) \geq \underline{\lambda}_{\min}(K\left(L \otimes I\right))\|v\|^2 .$$

Case-II: $v^T K\left(L \otimes I\right) v = 0$. Since $\phi(v) > 0$, we have

$$v^T K\Gamma v > 0 \iff v^T K\Gamma v \geq \underline{\lambda}_{\min}(K\Gamma)\|v\|^2 \overset{(A.97)}{\Longrightarrow} \phi(v) \geq \alpha\, \underline{\lambda}_{\min}(K\Gamma)\|v\|^2 .$$

Case-III: $v^T K\Gamma v > 0$, $v^T K\left(L \otimes I\right) v > 0$. From (A.97),

$$\phi(v) \geq \left(\alpha\, \underline{\lambda}_{\min}(K\Gamma) + \underline{\lambda}_{\min}(K\left(L \otimes I\right))\right)\|v\|^2 .$$

Thus, for any $v \neq 0_{md}$, we obtain that $\phi(v) \geq \min\left\{\underline{\lambda}_{\min}(K\left(L \otimes I\right)),\, \alpha\, \underline{\lambda}_{\min}(K\Gamma)\right\}\left\|v^2\right\|$, which implies

$$\lambda_{\min}(M) \geq \min\left\{\underline{\lambda}_{\min}(K\left(L \otimes I\right)),\, \alpha\, \underline{\lambda}_{\min}(K\Gamma)\right\}. \tag{A.98}$$

From (A.97), we further have that

$$\lambda_{\max}(M) \leq \lambda_{\max}(K\left(L \otimes I\right)) + \alpha\, \lambda_{\max}(K\Gamma). \tag{A.99}$$

Next, we compute the eigenvalues in the R.H.S. of (A.98)-(A.99).

Let $\lambda_1^i \geq \ldots \geq \lambda_d^i \geq 0$ denote the eigenvalues of $\left(A^i\right)^T A^i$. Let $S^i = Diag(\lambda_1^i, \ldots, \lambda_d^i)$,

and let the matrix U^i consists of the orthonormal eigenvectors $[U_1^i, \ldots, U_d^i]$ such that $\left(A^i\right)^T A^i U_j^i = \lambda_j^i U_j^i$. Note that $(U^i)^T U^i = I$, and $\left(A^i\right)^T A^i = U^i S^i (U^i)^T$. From (3.5), then we obtain

$$K^i = U^i Diag\left(\{\frac{1}{\alpha\lambda_j^i + |\mathcal{N}^i|}\}_{j=1}^d\right)(U^i)^T, \tag{A.100}$$

$$K^i\left(A^i\right)^T A^i = U^i Diag\left(\{\frac{\lambda_j^i}{\alpha\lambda_j^i + |\mathcal{N}^i|}\}_{j=1}^d\right)(U^i)^T. \tag{A.101}$$

From (A.95), $K\Gamma = Diag\left[K^1 A^T A 1, \ldots, K^m \left(A^m\right)^T A^m\right]$. So, the smallest non-zero eigenvalue of $K\Gamma$ is $\min_{i=1,\ldots,m} \underline{\lambda}_{\min}(K^i \left(A^i\right)^T A^i)$. From (A.101),

$$\underline{\lambda}_{\min}(K\Gamma) = \min_{i=1,\ldots,m} \frac{\underline{\lambda}^i}{\alpha\underline{\lambda}^i + |\mathcal{N}^i|}. \tag{A.102}$$

Similarly, the largest eigenvalue of $K\Gamma$ is

$$\lambda_{\max}(K\Gamma) = \max_{i=1,\ldots,m} \frac{\overline{\lambda}^i}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}. \tag{A.103}$$

Since K is block diagonal with positive eigenvalues and $(L \otimes I)$ is symmetric positive semi-definite, we have [229, 230]

$$\underline{\lambda}_{\min}(K\,(L \otimes I)) \geq \lambda_{\min}(K)\,\underline{\lambda}_{\min}\,(L \otimes I)\,, \tag{A.104}$$

$$\lambda_{\max}(K\,(L \otimes I)) \leq \lambda_{\max}(K)\,\lambda_{\max}\,(L \otimes I)\,. \tag{A.105}$$

From (A.95) and the decomposition of K^i in (A.100), $\lambda_{\min}(K) = \min_{i=1,\ldots,m} \frac{1}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}$, $\lambda_{\max}(K) = \max_{i=1,\ldots,m} \frac{1}{|\mathcal{N}^i|}$. Recall that, $\overline{\lambda}_L$ and $\underline{\lambda}_L$ respectively denote the largest and the smallest non-zero

eigenvalues of L. Then, $\underline{\lambda}_{\min}(L \otimes I) = \underline{\lambda}_L$, $\lambda_{\max}(L \otimes I) = \overline{\lambda}_L$. Upon substituting from above in (A.104)-(A.105), we get

$$\underline{\lambda}_{\min}(K(L \otimes I)) \geq \min_{i=1,\ldots,m} \frac{1}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|} \underline{\lambda}_L, \tag{A.106}$$

$$\lambda_{\max}(K(L \otimes I)) \leq \max_{i=1,\ldots,m} \frac{1}{|\mathcal{N}^i|} \overline{\lambda}_L. \tag{A.107}$$

Upon substituting from (A.102)-(A.103) and (A.106)-(A.107) in (A.98)-(A.99), we obtain that the condition number of M is bounded as

$$\kappa(M) = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)} \leq \frac{\max_{i=1,\ldots,m} \frac{\overline{\lambda}_L}{|\mathcal{N}^i|} + \alpha \max_{i=1,\ldots,m} \frac{\overline{\lambda}^i}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}}{\min\left\{\min_{i=1,\ldots,m} \frac{\underline{\lambda}_L}{\alpha\overline{\lambda}^i + |\mathcal{N}^i|}, \alpha \min_{i=1,\ldots,m} \frac{\underline{\lambda}^i}{\alpha\underline{\lambda}^i + |\mathcal{N}^i|}\right\}}.$$

Substituting above from (3.12), $\kappa(M) \leq \kappa_M$. Since $\phi(v) = v^T M v > 0$, M has at least one positive eigenvalue. Under the condition of the theorem, $\delta < \frac{2}{\lambda_{\max}(M)}$. Then, from (A.94) it follows that there exists ρ such that (3.13) holds and $\frac{\kappa(M)-1}{\kappa(M)+1} \leq \rho < 1$ [21]. Since $\kappa(M) \leq \kappa_M$, the proof is complete.

A.10 Proof of Theorem 3.2

Consider an arbitrary iteration $t \geq 0$. Define the estimation error at iteration t as $z_o(t) = \left[z_o^1(t)^T, \ldots, z_o^m(t)^T\right]^T = \left[(x_o^1(t) - x^*)^T, \ldots, (x_o^m(t) - x^*)^T\right]^T$. Upon substituting above from (3.14) and (A.92),

$$z_o(t) = z(t) + \left[\zeta^1(t)^T, \ldots, \zeta^m(t)^T\right]^T = z(t) + \zeta(t). \tag{A.108}$$

Proceeding in the same way as (A.94) in Section A.9, from the estimate update equation (3.15) we have $z(t+1) = (I - \delta M)\, z_o(t)$. Upon substituting above from (A.108), $z(t+1) = (I - \delta M)\, z(t) + (I - \delta M)\, \zeta(t)$. Using triangle inequality and the definition of induced 2-norm of matrix, from above we get

$$\left\|z(t+1)\right\| \leq \|I - \delta M\| \left\|z(t)\right\| + \|I - \delta M\| \left\|\zeta(t)\right\|. \tag{A.109}$$

Define, $\rho = \|I - \delta M\|$. Under Assumption 3.1 and $\alpha > 0$, from Lemma 3.1 we have that $\lambda_{\min}(M) > 0$. Since $0 < \delta < \frac{2}{\lambda_{\max}(M)}$ and $\lambda_{\min}(M) > 0$, we have $\frac{\kappa_M - 1}{\kappa_M + 1} \leq \rho < 1$ [21]. Under Assumption 3.3, from the definition of $\zeta(t)$ in Section 3.2.2,

$$\mathbb{E}_{\zeta_t}\left[\left\|\zeta(t)\right\|\right] \leq \mathbb{E}_{\zeta_t}\left[\left\|\zeta(t)\right\|_1\right] = \sum_{i=1}^{m} \mathbb{E}_{\zeta_t}\left[\left\|\zeta^i(t)\right\|_1\right] \leq m\omega. \tag{A.110}$$

Upon taking expectation with respect to $\zeta(t)$ on (A.109) and substituting from above, we obtain that $\mathbb{E}_{\zeta_t}\left[\left\|z(t+1)\right\|\right] \leq \rho\left\|z(t)\right\| + \rho\, m\, \omega$. Upon iterating the above from t to 0, the total expectation is obtained as $\mathbb{E}_t\left[\left\|z(t+1)\right\|\right] \leq \rho^{t+1}\left\|z(0)\right\| + m\,\omega \sum_{i=1}^{t+1} \rho^i$. Upon substituting above from (A.108) and (A.110) proves (3.18). Since $0 \leq \rho < 1$, taking limit as $t \to \infty$ on both sides of (3.18) proves the theorem.

A.11 Proof of Theorem 3.3

Consider any solution x^* of (3.22) and an arbitrary iteration $t \geq 0$. Recall the definition of estimation error at iteration t from (A.92). We define the matrix $A = \left[(A^1)^T, \ldots, (A^m)^T\right]^T$. When (3.22) has multiple solutions, the matrix A (equivalently $A^T A$) has a non-zero nullspace.

Let $\mathcal{N}(A^TA)$ denote the nullspace of matrix A^TA and $\mathcal{N}(A^TA)^\perp$ denote the orthogonal vector space of $\mathcal{N}(A^TA)$. Due to the fundamental theorem of linear algebra [45], $\mathbb{R}^n = \mathcal{N}(A^TA) \oplus \mathcal{N}(A^TA)^\perp$. Therefore, for each $t \geq 0$, we can decompose vector $z^i(t)$ into two orthogonal vectors $z(t)^\perp \in \mathcal{N}(A^TA)^\perp$ and $z^i(t)^\mathcal{N} \in \mathcal{N}(A^TA)$. Specifically, $z^i(t) = z^i(t)^\mathcal{N} + z^i(t)^\perp$. Below, we separately analyze the convergence $z(t)^\perp$ and $z^i(t)^\mathcal{N}$, and combine them subsequently.

Consider the eigen-decomposition of $A^TA = VSV^T$, where V consists of the orthonormal eigenvectors $[V_1, \ldots, V_d]$ and S is the diagonal matrix of the eigenvalues of A^TA. Let the rank of A^TA be denoted by r. We define $S^\perp = Diag(\underbrace{1, \ldots, 1}_{r}, \underbrace{0, \ldots, 0}_{d-r})$. Let $span\{V_1, \ldots, V_r\} = \left\{\sum_{i=1}^r u_i V_i : u_i \in \mathbb{R},\ \forall i\right\}$. As $V_1, \ldots, V_d$ are orthogonal,

$$\mathcal{N}(A^TA)^\perp = span\{V_1, \ldots, V_r\}, \tag{A.111}$$

$$\mathcal{N}(A^TA) = span\{V_{r+1}, \ldots, V_d\}. \tag{A.112}$$

Define a *projection matrix* $Q = VS^\perp V^T$. Note that for a vector $v \in \mathbb{R}^d$, due to the fundamental lemma of linear algebra, the vectors Qv and $(v - Qv)$ belong to the orthogonal vector spaces $\mathcal{N}(A^TA)^\perp$ and $\mathcal{N}(A^TA)$, respectively. Thus, $z(t)^\perp = Qz(t)$. Similarly, the projection matrix to

$\mathcal{N}(A^TA)$ is $P = I - Q = VDiag(\underbrace{0,\ldots,0}_{r}, \underbrace{1,\ldots,1}_{d-r})V^T$. Therefore,

$$z^i(t+1)^\perp = Q\, z^i(t+1) \overset{(A.93)}{=} z^i(t)^\perp - \alpha\delta QK^i \left(A^i\right)^T A^i z^i(t) + \delta QK^i \sum_{j\in\mathcal{N}^i} \left(z^j(t) - z^i(t)\right), \tag{A.113}$$

$$z^i(t+1)^\mathcal{N} = P\, z^i(t+1) \overset{(A.93)}{=} z^i(t)^\mathcal{N} - \alpha\delta PK^i \left(A^i\right)^T A^i z^i(t) + \delta PK^i \sum_{j\in\mathcal{N}^i} \left(z^j(t) - z^i(t)\right). \tag{A.114}$$

Consider a vector $w \in \mathcal{N}(A^TA)$. Since $\mathcal{N}(A^TA) = \cap_{i=1}^m \mathcal{N}(\left(A^i\right)^T A^i)$, $w \in \mathcal{N}(\left(A^i\right)^T A^i) \forall i$. Recall the eigen-decomposition of $K^i \left(A^i\right)^T A^i$ from (A.101). Let the eigenvalues of $K^i \left(A^i\right)^T A^i$ be denoted by $\overline{s}_j$, i.e., $\overline{s}_j = \frac{\lambda_j^i}{\alpha\lambda_j^i + |\mathcal{N}^i|}$. Let the rank of $\left(A^i\right)^T A^i$ be denoted by r_i. From (A.101) and definition of P, $PK^i \left(A^i\right)^T A^i = \sum_{l=r+1}^d V_l V_l^T \sum_{j=1}^{r_i} \overline{s}_j U_j^i U_j^{iT}$. Since $\mathcal{N}(\left(A^i\right)^T A^i)^\perp = span\left\{U_1^i, \ldots, U_{r_i}^i\right\}$ and $\mathcal{N}(A^TA) = \cap_{i=1}^m \mathcal{N}(\left(A^i\right)^T A^i)$, we have that $span\left\{U_1^i, \ldots, U_{r_i}^i\right\} \subseteq \mathcal{N}(A^TA)^\perp$. So, from (A.112), $V_l^T U_j^i = 0$ for each $j = 1, \ldots, r_i$ and each $l = r+1, \ldots, d$, and hence, $PK^i \left(A^i\right)^T A^i = 0_{d\times d}$. Since $P = I - Q$, we have $QK^i \left(A^i\right)^T A^i = K^i \left(A^i\right)^T A^i$. From (A.100) and definition of P, $PK^i = \sum_{l=r+1}^d V_l V_l^T \sum_{j=1}^d \frac{1}{\alpha\lambda_j^i + |\mathcal{N}^i|} U_j^i U_j^{iT}$. Now, $\lambda_j^i = 0$ for each $j = r_i + 1, \ldots, d$. From above, $PK^i = \frac{1}{|\mathcal{N}^i|} \sum_{l=r+1}^d V_l V_l^T \sum_{j=r_i+1}^d U_j^i U_j^{iT}$. Since $\sum_{j=r_i+1}^d U_j^i U_j^{iT}$ is the projection matrix to $\mathcal{N}(\left(A^i\right)^T A^i)$ and $\mathcal{N}(A^TA) \subseteq \mathcal{N}(\left(A^i\right)^T A^i)$, from (A.112) we have $PK^i = \frac{1}{|\mathcal{N}^i|} \sum_{l=r+1}^d V_l V_l^T = \frac{1}{|\mathcal{N}^i|} P$. Then, for any vector $v \in \mathcal{N}(A^TA)^\perp$ we have $PK^i v = \frac{1}{|\mathcal{N}^i|} Pv = 0_d$. And, $QK^i v = (I - P)K^i v = K^i v$. Consider any vector $w \in \mathcal{N}(A^TA)$. Then, $QK^i w = K^i w - PK^i w = K^i w - \frac{1}{|\mathcal{N}^i|} Pw = (K^i - \frac{1}{|\mathcal{N}^i|} I)w$. So, from (A.100), $QK^i w = \sum_{j=1}^{r_i} \frac{-\alpha\lambda_j^i}{(\alpha\lambda_j^i + |\mathcal{N}^i|)|\mathcal{N}^i|} U_j^i U_j^{iT} w$. Now, $w \in \mathcal{N}(\left(A^i\right)^T A^i) = span\left\{U_{r_i+1}^i, \ldots, U_d^i\right\}$. Thus,

$QK^i w = 0_d$. Above we have shown that

$$PK^i \left(A^i\right)^T A^i = O_{d\times d}, QK^i \left(A^i\right)^T A^i = K^i \left(A^i\right)^T A^i,$$
$$QK^i v = K^i v, PK^i v = QK^i w = 0_d, PK^i = \frac{P}{|\mathcal{N}^i|}. \tag{A.115}$$

Upon substituting from (A.115) in (A.113)-(A.114),

$$z^i(t+1)^\perp = z^i(t)^\perp - \alpha\delta QK^i \left(A^i\right)^T A^i z^i(t)^\perp + \delta QK^i \sum_{j\in\mathcal{N}^i} \left(z^j(t)^\perp - z^i(t)^\perp\right),$$

$z^i(t+1)^{\mathcal{N}} = z^i(t)^{\mathcal{N}} + \frac{\delta P}{|\mathcal{N}^i|} \sum_{j\in\mathcal{N}^i} \left(z^j(t)^{\mathcal{N}} - z^i(t)^{\mathcal{N}}\right) = z^i(t)^{\mathcal{N}} + \frac{\delta}{|\mathcal{N}^i|} \sum_{j\in\mathcal{N}^i} \left(z^j(t)^{\mathcal{N}} - z^i(t)^{\mathcal{N}}\right)$.

Define

$$\overline{Q} = Diag(Q,\ldots,Q), z(t)^\perp = \left[z^1(t)^{\perp T},\ldots,z^m(t)^{\perp T}\right]^T, z(t)^{\mathcal{N}} = \left[z^1(t)^{\mathcal{N}T},\ldots,z^m(t)^{\mathcal{N}T}\right]^T.$$

Upon substituting from above in (A.92), $z(t+1)^\perp = \left(I - \delta\overline{Q}M\right) z(t)^\perp = (I - \delta M)\, z(t)^\perp$, and $z(t+1)^{\mathcal{N}} = \left(I - \delta M^{\mathcal{N}}\right) z(t)^{\mathcal{N}}$. Considering the Lyapunov function $V(y) = \frac{1}{2}\sum_{i=1}^m (y^i)^T \left|\mathcal{N}^i\right| y^i$ for $y \in \mathbb{R}^{md}$, if $\delta < \frac{2}{\lambda_{max}(M^{\mathcal{N}})}$, it follows that $z^i(t)^{\mathcal{N}}$ reaches consensus to some vector $z^*(t)^{\mathcal{N}}$. Moreover [21], there exists $\rho_1 > 0$ such that $\frac{\kappa(M^{\mathcal{N}})-1}{\kappa(M^{\mathcal{N}})+1} \leq \rho_1 < 1$ and $\left\|z(t+1)^{\mathcal{N}}\right\| \leq \rho_1 \left\|z(t)^{\mathcal{N}}\right\|$, for all $t \geq 0$. Without Assumption 3.1, from the argument following (A.97) we have $\phi(v) \geq 0$, and hence, the eigenvalues of M are non-negative. From (A.115) and the definition of M^i in (3.9), if $w \in \mathcal{N}(A^T A)$ then $w \in \mathcal{N}(QM^i)$. Thus, using (A.115), the quadratic form of $\overline{Q}M$ is $v^{\perp T} M v^\perp$ for any $v \in \mathbb{R}^{md}$. Moreover, since $v^{\mathcal{N}} \in \mathcal{N}(\overline{Q}M)$ and $v^\perp$ is orthogonal to $v^{\mathcal{N}}$, we have $v^\perp \notin \mathcal{N}(\overline{Q}M)$. Then, the quadratic form of $\overline{Q}M$ is non-zero. So, the quadratic form of

$\overline{Q}M$ is positive. In the proof of Theorem 3.1, Assumption 3.1 has only been used to show that $\phi(v) > 0$. Thus, without Assumption 3.1, we follow the rest of the proof of Theorem 3.1 exactly, and obtain the following. If $\delta < \frac{2}{\lambda_{\max}(M)}$, then there exists ρ_2 such that $\frac{\kappa(M)-1}{\kappa(M)+1} \leq \rho_2 < 1$ and $\left\|z(t+1)^{\perp}\right\| \leq \rho_2 \left\|z(t)^{\perp}\right\|, \forall t \geq 0$. We define $\rho = \max\{\rho_1, \rho_2\}$. Since $z(t) = z(t)^{\mathcal{N}} + z(t)^{\perp}$ and $z(t)^{\mathcal{N}T} z(t)^{\perp} = 0$, we get $\left\|z(t+1)\right\|^2 \leq \rho^2 \left\|z(t)\right\|^2$, if $\delta < \min\{\frac{2}{\lambda_{\max}(M)}, \frac{2}{\lambda_{\max}(M^{\mathcal{N}})}\} = \frac{2}{\lambda_{\max}(M)}$. Moreover, $\rho \geq \max\{\frac{\kappa(M^{\mathcal{N}})-1}{\kappa(M^{\mathcal{N}})+1}, \frac{\kappa(M)-1}{\kappa(M)+1}\} = \frac{\max\{\kappa_M, \kappa_{\mathcal{N}}\}-1}{\max\{\kappa_M, \kappa_{\mathcal{N}}\}+1}$ and $\rho < 1$.

Finally, we find the limit point of $x^i(t)$. Since $\{z^i(t)^{\mathcal{N}},\ i = 1, \ldots, m\}$ reach consensus in $\mathcal{N}(A^T A)$, $\{x^i(t)^{\mathcal{N}},\ i = 1, \ldots, m\}$ also reach consensus to some vector in $\mathcal{N}(A^T A)$, which we denote by $x^*_{\mathcal{N}}$. Since $z^i(t)^{\perp}$ converges to 0_d, $x^i(t)^{\perp}$ converges to $x^{*\perp}$. Therefore, $x^i(t) = x^i(t)^{\perp} + x^i(t)^{\mathcal{N}}$ converges to $x^{*\perp} + x^*_{\mathcal{N}}$. Since $x^*_{\mathcal{N}} \in \mathcal{N}(A^T A)$, we have $A\left(x^{*\perp} + x^*_{\mathcal{N}}\right) = Ax^{*\perp}$. Since $Ax^{*\mathcal{N}} = 0$, we get $A\left(x^{*\perp} + x^*_{\mathcal{N}}\right) = Ax^{*\perp} + Ax^{*\mathcal{N}} = Ax^* = b$. So, each $x^i(t)$ converges to the same solution of (3.22). Hence, the proof.

A.12 Proof of Theorem 3.4

We follow the proof of Lemma 3.1 presented in Appendix A.8: below, we show that 0_{md} is the exponentially stable equilibrium point of $\dot{y}(s) = -My(s)$, $y(0) \in \mathbb{R}^{md}$, and therefore, the eigenvalues of M have positive real parts. Then, for small enough step-size $\delta > 0$, the estimation error $z(t)$ in (A.94) linearly converges to 0_{md}.

Under Assumption 3.4, there exists a vector $\sigma = [\sigma_1, \ldots, \sigma_m]^T \in \mathbb{R}^m$ with $\sum_{i=1}^m \sigma_i = 1$ and $\sigma_i > 0$ such that $\sigma^T L = 0_{1\times d}$ [54]. Define $\Phi = Diag(\sigma_1, \ldots, \sigma_m)$. Consider the Lyapunov function candidate $V : \mathbb{R}^{md} \to \mathbb{R}$ such that $V(y) = \frac{1}{2} y^T (\Phi \otimes I) K^{-1} y$ for any $y \in \mathbb{R}^{md}$. Since $\Phi \otimes I$ is diagonal matrix, V is positive definite. Moreover, $\dot{V}(y) = -y^T(\Phi \otimes$

$I)K^{-1}My$. Upon substituting above from (3.8) and using properties of Kronecker product, $\dot{V}(y) = -\alpha \sum_{i=1}^{m} \sigma_i \left\| A^i y^i \right\|^2 - y^T(\Phi L \otimes I)y$. Now,

$$y^T(\Phi L \otimes I)y = \frac{1}{2} y^T \left((\Phi L \otimes I) + (\Phi L \otimes I)^T \right) y = \frac{1}{2} y^T \left((\Phi L + L^T \Phi) \otimes I \right) y.$$

Since $\sigma^T L = 0_d^T$, we have $1_d^T(\Phi L + L^T\Phi) = 0_d^T$ and $(\Phi L + L^T\Phi)1_d = 0_d$. So, $(\Phi L + L^T\Phi)$ is a symmetric Laplacian associated with an undirected graph [54]. Then, $\dot{V}(y) \leq 0$. Since $(\Phi L + L^T\Phi)$ is Laplacian associated with an undirected graph, under Assumption 3.1, $\dot{V}(y) = 0 \iff y^i \in \mathcal{N}(\left(A^i\right)^T A^i), \exists y_c \in \mathbb{R}^d$ such that $y^i(s) = y_c \; \forall i \iff y = 0_{md}$. Thus, $\dot{V}(y) < 0$ for $y \neq 0_{md}$. Hence, the proof.

A.13 Proof of Lemma 3.2

For any $i \in [m]$,

$$(\lambda, v) \text{ is an Eigen-pair of } \overline{K}^i \iff (K^i + \eta I)^{-1}(K^i - \eta I)v = \lambda v$$

$$\iff (K^i - \eta I)v = (K^i + \eta I)\lambda v \text{ [So} \lambda \neq 1 \text{, otherwise } v = 0 \text{ is an Eigenvector of } \overline{K}^i. \text{]}$$

$$\iff K^i v = \frac{1+\lambda}{1-\lambda}\eta v \iff (\frac{1+\lambda}{1-\lambda}\eta, v) \text{ is an Eigen-pair of } K^i.$$

We have $K^i = k(N^i)^{-1}$, which can easily be verified by substituting this expression in the Lyapunov equation (3.42) which has a unique solution. Then,

$$\lambda[K^i] = \frac{k}{\lambda[N^i]} = \frac{k}{\lambda[(A^i)^T A^i] + |\mathcal{N}^i|} > 0 \iff \frac{1+\lambda}{1-\lambda}\eta > 0 \iff |\lambda| < 1.$$

Therefore, $\overline{K}^i$ is Schur and the claim follows.

A.14 Proof of Lemma 3.3

We begin with the necessary definitions (ref. [72]):

$$\overline{r}_{ij} = r_{ij} - x^*, \tag{A.116}$$

$$\overline{x}^i = x^i - x^*. \tag{A.117}$$

$$\overline{S}^i = (1/2)\left\|\overline{x}^i\right\|^2, \tag{A.118}$$

$$V^{ij}(t) = \frac{1}{2}\int_{t-\tau^{ij}}^{t}\left\|\vec{s}_{ij}(y) - \frac{1}{\sqrt{2\eta}}\eta x^*\right\|^2 dy + \frac{1}{2}\int_{t-\tau^{ji}}^{t}\left\|\vec{s}_{ji}(y) - \frac{1}{\sqrt{2\eta}}\eta x^*\right\|^2 dy, \tag{A.119}$$

$$V = \sum_{i\in[m]}\overline{S}^i + \sum_{(i,j)\in\mathcal{E}} V^{ij}. \tag{A.120}$$

As the proof is long, we outline the steps as follows:

1. Define a suitable time-dependent Lyapunov function candidate V (ref. (A.120)) for the combined agent dynamics, where the overall Lyapunov function is contributed by storage functionals $\overline{S}^i$ for individual agents and V^{ij} for their links.

2. The time derivative of V is shown to be non-positive, and consensus is established between the agents using extension on LaSalle's principle.

3. Finally, asymptotic convergence is established by showing that the solution set of the time derivative of the Lyapunov function V being identically zero only consists of each agent asymptotically reaching the desired solution x^*.

From (3.48) and (A.117),

$$\dot{\overline{x}}^i = \dot{x}^i = v^i - K^i \phi^i(x^i). \tag{A.121}$$

From (A.118), $\overline{S}^i$ is "positive definite" and

$$\dot{\overline{S}}^i = (\overline{x}^i)^T v^i - (\overline{x}^i)^T K^i (A^i)^T A^i (x^i - x^*). \tag{A.122}$$

From (3.44), (A.116) and (A.117),

$$v^{ij} = K^i(\overline{r}_{ij} - \overline{x}^i),\ v^i = \sum_{j \in \mathcal{N}^i} K^i(\overline{r}_{ij} - \overline{x}^i).$$

Substituting them into (A.122),

$$\begin{aligned} \dot{\overline{S}}^i =& (\overline{x}^i)^T \sum_{j \in \mathcal{N}^i} K^i(\overline{r}_{ij} - \overline{x}^i) - (\overline{x}^i)^T K^i (A^i)^T A^i \overline{x}^i \\ =& \sum_{j \in \mathcal{N}^i} [(\overline{r}_{ij})^T v^{ij} - (\overline{x}^i - \overline{r}_{ij})^T K^i (\overline{x}^i - \overline{r}_{ij})] - (\overline{x}^i)^T K^i (A^i)^T A^i \overline{x}^i. \end{aligned} \tag{A.123}$$

We use the following facts (ref. proof of Lemma 5 in [72]):

$$V^{ij}(t) \geq 0 \quad \forall t \tag{A.124}$$

$$\text{and } \dot{v}^{ij}(t) = -(v^{ij})^T \overline{r}_{ij} - (v^{ji})^T \overline{r}_{ji}. \tag{A.125}$$

From (A.123) and (A.125) it follows that,

$$\dot{V} = -\sum_{i\in[m]}\sum_{j\in\mathcal{N}^i}(\overline{x}^i - \overline{r}_{ij})^T K^i (\overline{x}^i - \overline{r}_{ij}) \sum_{i\in[m]} (\overline{x}^i)^T K^i (A^i)^T A^i \overline{x}^i \leq 0 \tag{A.126}$$

So, $x^i \in \mathcal{L}_\infty \; \forall i$. From (3.44)-(3.47),

$$r_{ij}(t) = (\overline{K}^i)^2 r_{ij}(t - \tau^{ij} - \tau^{ji}) + \beta^{ij}(t), \tag{A.127}$$

where $\beta^{ij}(t)$ are linear functions of x^i and x^j at times t, $t-\tau^{ij}$, $t-\tau^{ji}$, $t-\tau^{ij}-\tau^{ji}$ (ref. [72]). Also $\beta^{ij}(t)$ are bounded linear, because $x^i \in \mathcal{L}_\infty$. Lemma 3.2 states that Eigenvalues of $\overline{K}^i \; \forall i$ are within unit circle, if $\eta > 0$. Then, (A.127) is a stable difference equation with bounded inputs, which means $r_{ij} \in \mathcal{L}_\infty$. So, extension of LaSalle's principle for time delay systems is applicable. Now,

$$\dot{V} \equiv 0 \iff \overline{x}^i = \overline{r}_{ij} \, \forall i, \, \forall j \in \mathcal{N}^i, \sum_{i\in[m]} (\overline{x}^i)^T K^i (A^i)^T A^i \overline{x}^i = 0. \tag{A.128}$$

Thus, LaSalle's principle implies that

$$\forall i, \; x^i \to r_{ij} \text{ as } t \to \infty \, \forall j \in \mathcal{N}^i. \tag{A.129}$$

From (3.44)-(3.47) one can see that,

$$\begin{aligned}\eta r_{ij}(t) =& \sqrt{2\eta}\vec{s}_{ji}(t-\tau^{ji}) - K^i(r_{ij}(t) - x^i(t)) \\ =& -K^j(r_{ji}(t-\tau^{ji}) - x^j(t-\tau^{ji})) + \eta r_{ji}(t-\tau^{ji}) - K^i(r_{ij}(t) - x^i(t)).\end{aligned} \tag{A.130}$$

From (A.129) and (A.130), for every $i, \forall j \in \mathcal{N}^i$,

$$\lim_{t\to\infty} r_{ij}(t) = \lim_{t\to\infty} r_{ji}(t-\tau^{ji}) = \lim_{t\to\infty} r_{ji}(t) \implies \lim_{t\to\infty} x^i(t) = \lim_{t\to\infty} x^j(t).$$

Thus consensus is achieved asymptotically. Then, from (A.128) we have

$$\dot{V} \equiv 0 \implies \lim_{t\to\infty} x^i(t) = \lim_{t\to\infty} x(t)\, \forall i, \lim_{t\to\infty} \sum_{i\in[m]} (\overline{x}(t))^T K^i (A^i)^T A^i \overline{x}(t) = 0. \tag{A.131}$$

for some $x : \mathbb{R} \to \mathbb{R}^d$ and $\overline{x} := x - x^*$. So, the only thing left to be shown is

$$\lim_{t\to\infty} \sum_{i\in[m]} (\overline{x}(t))^T K^i (A^i)^T A^i \overline{x}(t) = 0 \implies \lim_{t\to\infty} \overline{x}(t) = 0.$$

So it is sufficient to show that,

$$\sum_{i\in[m]} K^i (A^i)^T A^i \succ 0. \tag{A.132}$$

We have

$$K^i \in S^d_{++}, (A^i)^T A^i \in S^d_+ \, \forall i \implies \sum_{i\in[m]} K^i (A^i)^T A^i \succeq 0$$

For each i, consider the Eigen decomposition $(A^i)^T A^i = U^i \Lambda^i (U^i)^T$, where U^i is the Eigenvector matrix of $(A^i)^T A^i$ and Λ^i is a diagonal matrix with the Eigenvalues of $(A^i)^T A^i$ in the diagonal. Then,

$$N^i = |\mathcal{N}^i| I + (A^i)^T A^i = U^i (|\mathcal{N}^i| I + \Lambda^i)(U^i)^T.$$

We have $K^i = k(N^i)^{-1}$, which can easily be verified by substituting in the Lyapunov equation (3.42) which has a unique solution. Then,

$$K^i (A^i)^T A^i = k U^i diag\{\frac{\lambda_{ik}}{|\mathcal{N}^i| + \lambda_{ik}}\}_{k=1}^d (U^i)^T,$$

where λ_{ik} are the Eigenvalues of $(A^i)^T A^i \in S^d_+$, and $\lambda_{ik} = 0$ if the k^{th} column of U^i is in $\mathcal{N}(A^i)$. This also means $K^i (A^i)^T A^i \in S^d_+$.

Consider any $x \in \mathbb{R}^n, x \neq 0$. Now $\exists j^* \in [m]$ s.t. $x \notin \mathcal{N}(A^{j^*})$. Otherwise, $x \in \mathcal{N}(A^i)\ \forall i$

which implies $x \in \mathcal{N}(A)$. From Assumption 3.1, $\mathcal{N}(A) = 0$ which is a contradiction. Now,

$$x^T K^{j^*} (A^{j^*})^T A^{j^*} x = k \sum_{k=1}^{d} \frac{\lambda_{j^*k}}{|\mathcal{N}^{j^*}| + \lambda_{j^*k}} \left\| x^T u_{j^*k} \right\|^2 . \tag{A.133}$$

We consider the nontrivial case where for each i, A^i is not the zero matrix. Then, a positive eigenvalue exists, because $(A^{j^*})^T A^{j^*} \in S_+^d$ and all of its eigenvalues cannot be zero. Define $l = n - \mathcal{N}((A^{j^*})^T A^{j^*})$. Then $\exists l \geq 1$ such that $\lambda_{j^*k} > 0$, $k = 1, ..., l$ (possibly by rearrangement of indices) and the eigenvectors $\{u_{j^*k}\}_{k=1}^{l} \not\subset \mathcal{N}((A^{j^*})^T A^{j^*})$ and $\{u_{j^*k}\}_{k=l+1}^{d} = \mathcal{N}((A^{j^*})^T A^{j^*})$. Then,

$$x^T K^{j^*} (A^{j^*})^T A^{j^*} x = 0 \iff x \perp span\{u_{j*k}\}_{k=1}^{l} \iff x \in span\{u_{j*k}\}_{k=l+1}^{d}$$

$$\iff x \in \mathcal{N}((A^{j^*})^T A^{j^*})$$

Therefore, $x^T K^{j^*} (A^{j^*})^T A^{j^*} x > 0$ and the claim follows from (A.132).

A.15 Proof of Theorem 3.5

For every $i \in [m]$, consider the system (3.48) with the definitions (3.45)-(3.47). Then, Lemma 3.3 implies that $x^i \to x^* \ \forall i$ as $t \to \infty$. So it is enough to show that, there exists $\vec{s}_{ij}$ $\forall j \in \mathcal{N}^i$ satisfying definitions (3.45)-(3.47) such that the dynamics in (3.43) is same as the dynamics (3.48). Let

$$\vec{s}_{ij}(t) = \sqrt{\frac{\eta}{2}} x^i(t) \, \forall j \in \mathcal{N}^i. \tag{A.134}$$

From (3.47) and the scattering variables chosen as above,

$$\overleftarrow{s}_{ij}(t) = \vec{s}_{ji}(t - \tau^{ji}) = \sqrt{\frac{\eta}{2}} x^j(t - \tau^{ji}). \tag{A.135}$$

From (3.44), (3.45) and (A.135) it can be seen that,

$$\eta x^j(t-\tau^{ji}) = K^i(r_{ij}(t) - x^i(t)) + \eta r_{ij}(t) \implies r_{ij}(t) = (K^i + \eta I)^{-1}(\eta x^j(t-\tau^{ji}) + K^i x^i(t)).$$

By the above equation and (3.44),

$$\begin{aligned} v^{ij}(t) =& K^i[(K^i + \eta I)^{-1}(\eta x^j(t-\tau^{ji}) + K^i x^i(t)) - x^i(t)] \\ =& \eta K^i(K^i + \eta I)^{-1}(x^j(t-\tau^{ji}) - x^i(t)), \end{aligned} \tag{A.136}$$

where the last equality follow from the fact

$$(K^i + \eta I)^{-1}K^i - I = (K^i + \eta I)^{-1}K^i - (K^i + \eta I)^{-1}(K^i + \eta I) = -\eta(K^i + \eta I)^{-1}.$$

Equation (A.136) substituted in (3.48) leads to (3.43), and the choice of variables in (A.134) satisfy definitions (3.45)-(3.47). Therefore, the proof is complete, following Lemma 3.3.

A.16 Proof of Lemma 4.1

Recall the definition of K^* from Section 4.2.2. For each iteration $t \geq 0$, define the matrix

$$\widetilde{K}(t) = K(t) - K^*. \tag{A.137}$$

Consider an arbitrary iteration $t \geq 0$. Upon substituting from (4.3) and (4.5) in the definition (A.137),

$$\begin{aligned} \widetilde{K}(t+1) =& K(t) - \alpha(t)\left(\left(\nabla^2 f(x(t)) + \beta I\right)K(t) - I\right) - K^* \\ =& K(t) - K^* - \alpha(t)\left(\left(\nabla^2 f(x(t)) + \beta I\right)K(t) - \left(\nabla^2 f(x(t)) + \beta I\right)K^*\right) \\ & - \alpha(t)\left(\left(\nabla^2 f(x(t)) + \beta I\right)K^* - \left(\nabla^2 f(x^*) + \beta I\right)\left(\nabla^2 f(x^*) + \beta I\right)^{-1}\right). \end{aligned}$$

Upon substituting above from (A.137) and the definition of K^*,

$$\widetilde{K}(t+1) = \left(I - \alpha(t)\left(\nabla^2 f(x(t)) + \beta I\right)\right)\widetilde{K}(t) - \alpha(t)\left(\nabla^2 f(x(t)) - \nabla^2 f(x^*)\right)K^*.$$

Applying Cauchy-Schwartz inequality above,

$$\left\|\widetilde{K}(t+1)\right\| \leq \left\|I-\alpha(t)\left(\nabla^2 f(x(t))+\beta I\right)\right\|\left\|\widetilde{K}(t)\right\| + \alpha(t)\left\|\nabla^2 f(x(t))-\nabla^2 f(x^*)\right\|\|K^*\|. \tag{A.138}$$

Under Assumption 4.2, each local cost function f^i is convex, implying that the aggregate cost function $f = \sum_{i=1}^m f^i$ is convex. Thus, $\left(\nabla^2 f(x(t))+\beta I\right)$ is positive definite for $\beta > 0$. It follows that, if $\alpha(t) < \frac{1}{\lambda_{max}\left[\nabla^2 f(x(t))\right]+\beta}$ then $\rho(t) = \left\|I-\alpha(t)\left(\nabla^2 f(x(t))+\beta I\right)\right\| < 1$. This proves the lemma.

A.17 Proof of Theorem 4.1

Before presenting the proof, we define the following notation. Recall the definition of the minimum points X^* in (4.1). For a minimum points $x^* \in X^*$ and each iteration $t \geq 0$, define the estimation error

$$z(t) = x(t) - x^*. \tag{A.139}$$

Step 1: In this step, we derive an upper bound of $\left\|\widetilde{K}(t+1)\right\|$, provided with the iterations from 0 to t. In this step, we use (A.138) from the proof of Lemma 4.1. Consider an arbitrary iteration $t \geq 0$. From Section 4.2.2, recall that η denotes the induced 2-norm of the matrix K^*. Then,

$$\eta = \|K^*\| = \left\|\left(\nabla^2 f(x^*)+\beta I\right)^{-1}\right\| = \frac{1}{\lambda_{min}\left[\nabla^2 f(x^*)\right]+\beta}. \tag{A.140}$$

Under Assumption 4.2 and 4.5, $\nabla^2 f(x^*)$ is positive definite. Then, $\lambda_{min}\left[\nabla^2 f(x^*)\right] > 0$, and $\eta < \frac{1}{\beta}$. From Section 4.2.2, recall the definition $\rho = \sup_{t\geq 0}\rho(t)$. From Lemma 4.1 we have

$0 \leq \rho < 1$. Upon substituting from (A.140) and the definition of ρ in (A.138),

$$\left\|\widetilde{K}(t+1)\right\| \leq \rho \left\|\widetilde{K}(t)\right\| + \eta\, \alpha(t) \left\|\nabla^2 f(x(t)) - \nabla^2 f(x^*)\right\|. \tag{A.141}$$

Under Assumption 4.4, $\left\|\nabla^2 f(x(t)) - \nabla^2 f(x^*)\right\| \leq \gamma \left\|x(t) - x^*\right\|$. Upon substituting above from the definition (A.139), $\left\|\nabla^2 f(x(t)) - \nabla^2 f(x^*)\right\| \leq \gamma \left\|z(t)\right\|$. Upon substituting from above in (A.141),

$$\left\|\widetilde{K}(t+1)\right\| \leq \rho \left\|\widetilde{K}(t)\right\| + \eta\gamma\, \alpha(t) \left\|z(t)\right\|.$$

Iterating the above from t to 0 we have

$$\left\|\widetilde{K}(t+1)\right\| \leq \rho^{t+1} \left\|\widetilde{K}(0)\right\| + \eta\gamma\, \alpha(t) \left(\left\|z(t)\right\| + \rho \left\|z(t-1)\right\| + \ldots + \rho^t \left\|z(0)\right\|\right). \tag{A.142}$$

Step 2: In this step, we derive an upper bound on the estimation error $\left\|z(t+1)\right\|$, provided iteration t. Upon substituting from (4.4) in (A.139), with the parameter $\delta = 1$,

$$z(t+1) = z(t) - K(t) \sum_{i=1}^{m} g^i(t).$$

Upon substituting above from (4.2) and the definition of aggregate cost function $f = \sum_{i=1}^{m} f^i$,

$$z(t+1) = z(t) - K(t) \nabla f(x(t)).$$

Upon substituting above from the definition of $\widetilde{K}(t)$ in (A.137),

$$\begin{aligned} z(t+1) =& z(t) - K^* \nabla f(x(t)) - \widetilde{K}(t) \nabla f(t) \\ =& - K^* \left(\nabla f(x(t)) - (K^*)^{-1} z(t)\right) - \widetilde{K}(t) \nabla f(x(t)). \end{aligned}$$

Since x^* is a minimum point of the aggregate cost function f in (4.1), the first order necessary condition states that $\nabla f(x^*) = 0_d$. Here, 0_d denotes the d-dimensional zero vector. Thus, the

above equation can be rewritten as

$$z(t+1) = -K^* \left(\nabla f(x(t)) - \nabla f(x^*) - (K^*)^{-1} z(t) \right) - \widetilde{K}(t) \nabla f(x(t)). \tag{A.143}$$

Using the fundamental theorem of calculus [22],

$$\nabla f(x(t)) - \nabla f(x^*) = (x(t) - x^*) \int_0^1 \nabla^2 f(yx(t) + (1-y)x^*) dy.$$

From above and the definition of $K^* = \left(\nabla^2 f(x^*) + \beta I \right)^{-1}$ (see Section 4.2.2), we have

$$\begin{aligned}
&\nabla f(x(t)) - \nabla f(x^*) - (K^*)^{-1} z(t) \\
&= (x(t) - x^*) \int_0^1 \nabla^2 f(yx(t) + (1-y)x^*) dy - \left(\nabla^2 f(x^*) + \beta I \right) z(t) \\
&= (x(t) - x^*) \int_0^1 \left(\nabla^2 f(yx(t) + (1-y)x^*) - \nabla^2 f(x^*) \right) dy - \beta z(t).
\end{aligned}$$

From (A.139), recall that $z(t) = x(t) - x^*$. Thus, from above we obtain that

$$\begin{aligned}
&\left\| \nabla f(x(t)) - \nabla f(x^*) - (K^*)^{-1} z(t) \right\| \\
&\leq \left\| z(t) \right\| \int_0^1 \left\| \nabla^2 f(yx(t) + (1-y)x^*) - \nabla^2 f(x^*) \right\| dy + \beta \left\| z(t) \right\|.
\end{aligned}$$

Under Assumption 4.4, $\left\| \nabla^2 f(yx(t) + (1-y)x^*) - \nabla^2 f(x^*) \right\| \leq \gamma(1-y) \left\| z(t) \right\|$. Thus,

$$\begin{aligned}
\left\| \nabla f(x(t)) - \nabla f(x^*) - (K^*)^{-1} z(t) \right\| &\leq \gamma \left\| z(t) \right\|^2 \int_0^1 (1-y) dy + \beta \left\| z(t) \right\| \\
&= \frac{\gamma}{2} \left\| z(t) \right\|^2 + \beta \left\| z(t) \right\|.
\end{aligned}$$

Upon using Cauchy-Schwartz inequality in (A.156), and then substituting from above, we obtain

$$\left\| z(t+1) \right\| \leq \eta \frac{\gamma}{2} \left\| z(t) \right\|^2 + \eta\beta \left\| z(t) \right\| + \left\| \widetilde{K}(t) \right\| \left\| \nabla f(x(t)) \right\|.$$

Since $\nabla f(x^*) = 0_d$, the above can be rewritten as

$$\left\| z(t+1) \right\| \leq \eta \frac{\gamma}{2} \left\| z(t) \right\|^2 + \eta\beta \left\| z(t) \right\| + \left\| \widetilde{K}(t) \right\| \left\| \nabla f(x(t)) - \nabla f(x^*) \right\|.$$

Upon using the Lipschitz property (4.6) in above,

$$\left\|z(t+1)\right\| \leq \eta\frac{\gamma}{2}\left\|z(t)\right\|^2 + \eta\beta\left\|z(t)\right\| + l\left\|\widetilde{K}(t)\right\|\left\|z(t)\right\|. \tag{A.144}$$

Upon substituting above from (A.142) in Step 1, we have

$$\begin{aligned}\left\|z(t+1)\right\| &\leq \eta\frac{\gamma}{2}\left\|z(t)\right\|^2 + \eta\beta\left\|z(t)\right\| \\ &+ l\left\|z(t)\right\|\left(\rho^t\left\|\widetilde{K}(0)\right\| + \eta\gamma\,\alpha(t-1)\left(\left\|z(t-1)\right\| + \rho\left\|z(t-2)\right\| + \ldots + \rho^{t-1}\left\|z(0)\right\|\right)\right).\end{aligned} \tag{A.145}$$

Step 3: In this final step of the proof, we prove (4.10) which is a direct result of the following claim.

Claim A.1. *Given the conditions of Theorem 4.1, for each iteration $t \geq 0$ the following statement holds true:*

$$\left\|z(t+1)\right\| \leq \frac{1}{\mu}\left\|z(t)\right\| \quad \textit{and} \quad \left\|z(t)\right\| < \frac{1}{\mu\eta\gamma}. \tag{A.146}$$

Proof. We proof the aforementioned claim by the principle of induction. First, we show that the claim is true for iteration $t = 0$. At $t = 0$, from (A.144) we have

$$\left\|z(1)\right\| \leq \left\|z(0)\right\|\left(\eta\frac{\gamma}{2}\left\|z(0)\right\| + \eta\beta + l\left\|\widetilde{K}(0)\right\|\right).$$

Upon substituting above from the condition (4.8),

$$\left\|z(1)\right\| \leq \frac{1}{2\mu}\left\|z(0)\right\|.$$

Since $\mu > 1$, from above we have

$$\left\|z(1)\right\| < \frac{1}{\mu}\left\|z(0)\right\|.$$

Moreover, (4.8) implies that $\left\|z(0)\right\| < \frac{1}{\mu\eta\gamma}$. Thus, the claim holds for $t = 0$. Now, we assume

that the claim is true for the iterations from 0 to t. We need to show that the claim is also true for the iteration $t+1$. From the above assumption we have

$$\big\|z(t+1)\big\| \leq \frac{1}{\mu}\big\|z(t)\big\| \leq \frac{1}{\mu^2}\big\|z(t-1)\big\| \leq \ldots \leq \frac{1}{\mu^{t+1}}\big\|z(0)\big\| < \frac{1}{\mu^{t+1}}\frac{1}{\mu\eta\gamma}. \tag{A.147}$$

Since $\mu > 1$, the above implies that $\big\|z(t+1)\big\| < \frac{1}{\mu\eta\gamma}$. Now, consider the expression

$$\Big(\big\|z(t)\big\| + \rho\big\|z(t)\big\| + \ldots + \rho^t\big\|z(0)\big\|\Big)$$

in the R.H.S. of (A.145) for iteration $t+1$. Upon substituting from (A.147),

$$\big\|z(t)\big\| + \rho\big\|z(t)\big\| + \ldots + \rho^t\big\|z(0)\big\| \leq \big\|z(0)\big\|\left(\frac{1}{\mu^t} + \frac{\rho}{\mu^{t-1}} + \ldots + \rho^t\right) = \big\|z(0)\big\|\,\frac{1-(\mu\rho)^{t+1}}{\mu^t(1-\mu\rho)}.$$

From the condition of Theorem 4.1 we have that $\mu\rho < 1$. Upon substituting from above in (A.145) for iteration $t+1$,

$$\big\|z(t+2)\big\| \leq \big\|z(t+1)\big\|\left(\eta\frac{\gamma}{2}\big\|z(t+1)\big\| + \eta\beta + l\rho^{t+1}\Big\|\widetilde{K}(0)\Big\| + \eta\gamma l\;\alpha(t)\big\|z(0)\big\|\,\frac{1-(\mu\rho)^{t+1}}{\mu^t(1-\mu\rho)}\right). \tag{A.148}$$

From Step 1 we have $\rho < 1$. Thus, $l\rho^{t+1}\Big\|\widetilde{K}(0)\Big\| < l\Big\|\widetilde{K}(0)\Big\|$. Additionally, if $\alpha(t) < \frac{\mu^t(1-\mu\rho)}{2l(1-(\mu\rho)^{t+1}}$, then $\eta\gamma l\;\alpha(t)\big\|z(0)\big\|\,\frac{1-(\mu\rho)^{t+1}}{\mu^t(1-\mu\rho)} < \eta\frac{\gamma}{2}\big\|z(0)\big\|$. Using condition (4.8) then we have

$$\eta\beta + l\rho^{t+1}\Big\|\widetilde{K}(0)\Big\| + \eta\gamma l\;\alpha(t)\big\|z(0)\big\|\,\frac{1-(\mu\rho)^{t+1}}{\mu^t(1-\mu\rho)} < \frac{1}{2\mu}. \tag{A.149}$$

We have shown above that $\big\|z(t+1)\big\| < \frac{1}{\mu\eta\gamma}$, which implies that $\eta\frac{\gamma}{2}\big\|z(t+1)\big\| < \frac{1}{2\mu}$. Upon substituting from above and (A.149) in (A.148),

$$\big\|z(t+2)\big\| \leq \frac{1}{\mu}\big\|z(t+1)\big\|.$$

Thus, the claim holds for iteration $t+1$. Due to the principle of induction, the proof of the claim is complete. □

Equation (4.10) directly follows from the aforementioned claim, and concludes the proof of Theorem 4.1.

A.18 Proof of Theorem 4.2

For this proof, we borrow the results (A.142) and (A.144) from the proof of Theorem 4.1.

Under Assumption 4.6, $\nabla^2 f(x(t))$ is positive definite for $x(t) \in \mathcal{D}$. So, Assumption 4.6 is a special case of Assumption 4.5. Moreover, since $\nabla^2 f(x(t))$ is positive definite for $x(t) \in \mathcal{D}$, Theorem 4.1 is applicable with $\beta = 0$ in this case. To prove (4.11), consider (A.144) from the proof of Theorem 4.1. Upon taking limits on both sides as $t \to \infty$ and substituting $\beta = 0$, we get

$$\lim_{t\to\infty} \frac{\left\|z(t+1)\right\|}{\left\|z(t)\right\|} \leq \lim_{t\to\infty} \left(\eta\frac{\gamma}{2}\left\|z(t)\right\| + l\left\|\widetilde{K}(t)\right\|\right). \tag{A.150}$$

Since $\mu > 1$, from (4.10) of Theorem 4.1 we have that $\lim_{t\to\infty}\left\|z(t)\right\| = 0$. Since $\rho < 1$ and $\lim_{t\to\infty}\left\|z(t)\right\| = 0$, from (A.142) we have $\lim_{t\to\infty}\left\|\widetilde{K}(t)\right\| = 0$. Upon substituting them in (A.150) above, we obtain (4.11). Hence, the proof.

A.19 Proof of Theorem 5.2

The time-derivative of f along the trajectories $x(t)$ of (5.23) is given by

$$\dot{f}(x(t)) = \sum_{i=1}^{d} \nabla_i f(x(t))\dot{x}_i(t) = -\sum_{i=1}^{d} \frac{1}{\alpha_g(t)} \frac{\lambda_7 \nabla_i f(x(t))\mu_i(t) + \lambda_8 \left\|\nabla_i f(x(t))\right\|^2}{\nu_i(t)^c}. \tag{A.151}$$

Upon multiplying both sides above with $\alpha_g(t)$, followed by integrating both sides w.r.t. time from 0 to t and substituting from (5.20) we have

$$\int_0^t \alpha_g(s)\dot{f}(x(s))ds = -\sum_{i=1}^{d}\int_0^t \frac{\lambda_7\mu_i(s)\dot{\mu}_i(s)}{\lambda_2\nu_i(s)^c}ds - \sum_{i=1}^{d}\int_0^t \frac{\lambda_7\lambda_1\mu_i(s)^2}{\lambda_2\nu_i(s)^c}ds - \sum_{i=1}^{d}\int_0^t \frac{\lambda_8\left\|\nabla_i f(x(s))\right\|^2}{\nu_i(s)^c}ds. \tag{A.152}$$

Integrating by parts we have the first term on R.H.S. as

$$\int_0^t \frac{\mu_i(s)\dot{\mu}_i(s)}{\nu_i(s)^c}ds = \left[\frac{\mu_i(s)^2}{2\nu_i(s)^c}\right]_0^t + \frac{c}{2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\dot{\nu}_i(s)ds.$$

Upon substituting above from (5.22), and using that $\mu(0) = 0_d$,

$$\int_0^t \frac{\mu_i(s)\dot{\mu}_i(s)}{\nu_i(s)^c}ds = \frac{\mu_i(t)^2}{2\nu_i(t)^c} + \frac{c\lambda_4}{2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\zeta_i(s)ds - \frac{c\lambda_5}{2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c}ds + \frac{c\lambda_6}{2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\psi(\nabla_i f(x(s)),\mu_i(s))ds.$$

Upon substituting above in (A.152) we obtain that

$$\begin{aligned}\int_0^t \alpha_g(s)\dot{f}(x(s))ds = &-\sum_{i=1}^{d}\frac{\lambda_7}{2\lambda_2}\mu_i(t)^2\nu_i(t)^{-c} - \sum_{i=1}^{d}\frac{c\lambda_7\lambda_4}{2\lambda_2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\zeta_i(s)ds\\ &-\sum_{i=1}^{d}\frac{\lambda_7}{\lambda_2}\left(\lambda_1 - \frac{c\lambda_5}{2}\right)\int_0^t \mu_i(s)^2\nu_i(s)^{-c}ds\\ &-\sum_{i=1}^{d}\frac{c\lambda_7\lambda_6}{2\lambda_2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\psi(\nabla_i f(x(s)),\mu_i(s))ds\\ &-\sum_{i=1}^{d}\int_0^t \lambda_8\left\|\nabla_i f(x(s))\right\|^2\nu_i(s)^{-c}ds.\end{aligned} \tag{A.153}$$

We consider the possible cases $\lambda_7 = 0$ and $\lambda_7 > 0$. First we consider $\lambda_7 > 0$. We define, $\gamma_1 = 1 - \lambda_2$ and $\gamma_2 = 1 - \lambda_6$. Upon differentiating both sides of (5.19) w.r.t t we get $\dot{\alpha_g}(t) = \frac{c\gamma_2^{t+1}(1-\gamma_1^{t+1})\log\gamma_2 - \gamma_1^{t+1}(1-\gamma_2^{t+1})\log\gamma_1}{(1-\gamma_2^{t+1})^{1.5}}$. So we have

$$\dot{\alpha_g}(t) < 0 \iff \left(\frac{\gamma_2}{\gamma_1}\right)^{t+1}\frac{1-\gamma_1^{t+1}}{1-\gamma_2^{t+1}} > \frac{1}{c}\frac{\log\gamma_1}{\log\gamma_2}. \tag{A.154}$$

From the condition $\lambda_2 > \lambda_6$, we have $1 > \gamma_2 > \gamma_1 > 0$. Then, $\left(\frac{\gamma_2}{\gamma_1}\right)^{t+1}$ and $\frac{1-\gamma_1^{t+1}}{1-\gamma_2^{t+1}}$ are, respectively, increasing and decreasing functions of t. Since $1 > \gamma_2 > \gamma_1 > 0$, we have $\lim_{t\to\infty}\left(\frac{\gamma_2}{\gamma_1}\right)^t \to \infty$ and $\lim_{t\to\infty}\frac{1-\gamma_1^t}{1-\gamma_2^t} = 1$. Thus, $\left(\frac{\gamma_2}{\gamma_1}\right)^t \frac{1-\gamma_1^t}{1-\gamma_2^t}$ is increasing in $t \geq T'$ for some $T' < \infty$. Then, there exists $T \in [T', \infty)$ such that (A.154) holds for all $t \geq T$. Integrating by parts we rewrite the L.H.S. in (A.153),

$$\int_0^t \alpha_g(s)\dot{f}(x(s))ds = \left[\alpha_g(s)f(x(s))\right]_0^t - \int_0^t \dot{\alpha_g}(s)f(x(s))ds.$$

Upon substituting from above in (A.153), for $t \geq T$,

$$\begin{aligned}
&\alpha_g(t)f(x(t)) + \sum_{i=1}^d \frac{\lambda_7}{2\lambda_2}\mu_i(t)^2\nu_i(t)^{-c} = \alpha_g(0)f(x(0)) + \int_0^T \dot{\alpha_g}(s)f(x(s))ds + \int_T^t \dot{\alpha_g}(s)f(x(s))ds \\
&- \sum_{i=1}^d \frac{c\lambda_7\lambda_4}{2\lambda_2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\zeta_i(s)ds - \sum_{i=1}^d \frac{\lambda_7}{\lambda_2}\left(\lambda_1 - \frac{c\lambda_5}{2}\right)\int_0^t \mu_i(s)^2\nu_i(s)^{-c}ds \\
&- \sum_{i=1}^d \frac{c\lambda_7\lambda_6}{2\lambda_2}\int_0^t \mu_i(s)^2\nu_i(s)^{-c-1}\psi(\nabla_i f(x(s)), \mu_i(s))ds - \sum_{i=1}^d \int_0^t \lambda_8 \left\|\nabla_i f(x(s))\right\|^2 \nu_i(s)^{-c}ds.
\end{aligned} \tag{A.155}$$

For each $i \in \{1, \ldots, d\}$, consider the state-space system described by (5.21)-(5.22), with the state vector $[\zeta_i, \nu_i]^T \in \mathbb{R}^2$. From (5.24), $\lambda_4 \geq 0$. Here, we consider the possible cases: $\lambda_4 > 0$ and $\lambda_4 = 0$.

- $\lambda_4 > 0$: Define the state matrix $A = \begin{bmatrix} -\lambda_3 & \lambda_3 \\ \lambda_4 & -\lambda_5 \end{bmatrix}$. Since $\zeta(0) = 0_d$, the states are given by the solution

$$\begin{bmatrix} \zeta_i(t) \\ \nu_i(t) \end{bmatrix} = e^{At}\begin{bmatrix} 0 \\ \nu_i(0) \end{bmatrix} + \int_0^t e^{A(t-s)}\begin{bmatrix} 0 \\ \lambda_6 \end{bmatrix}\psi(\nabla_i f(x(s)), \mu_i(s))ds.$$

Upon computing the state-transition matrix $\phi(t) = e^{At}$ and substituting above, we obtain

that

$$\zeta_i(t) = \phi_{12}(t-1)\nu_i(1) + \lambda_2 \int_1^t \phi_{12}(t-s)\psi(\nabla_i f(x(s)), \mu_i(s))ds, \tag{A.156}$$

$$\nu_i(t) = \phi_{22}(t-1)\nu_i(1) + \lambda_2 \int_1^t \phi_{22}(t-s)\psi(\nabla_i f(x(s)), \mu_i(s))ds, \tag{A.157}$$

where $\phi_{12}(t) = \lambda_3 e^{-\frac{\lambda_3+\lambda_5}{2}t} \frac{e^{\frac{p}{2}t} - e^{-\frac{p}{2}t}}{p}$, $\phi_{22}(t) = e^{-\frac{\lambda_3+\lambda_5}{2}t} \frac{e^{\frac{p}{2}t}(p-\lambda_5+\lambda_3) + e^{-\frac{p}{2}t}(p+\lambda_5-\lambda_3)}{2p}$, and $p = \sqrt{(\lambda_3-\lambda_5)^2 + 4\lambda_3\lambda_4}$. Since $\lambda_3, \lambda_6 > 0$ (ref. (5.24)), $\nu_i(0) > 0$, $\psi \geq 0$, and $p > 0$, from (A.156) we have $\zeta_i(t) > 0$. Since $\lambda_3, \lambda_4 > 0$, we have $p = \sqrt{(\lambda_3-\lambda_5)^2 + 4\lambda_3\lambda_4} > |\lambda_5 - \lambda_3|$, which implies that $\phi_{22}(t) > 0$. From above and $\nu_i(0) > 0$, we have $\nu_i(t) > 0$.

- $\lambda_4 = 0$: From (5.22), $\dot{\nu}_i(t) = -\lambda_5\nu_i(t) + \lambda_6\psi(\nabla_i f(x(t)), \mu_i(t))$. The solution of this ODE is $\nu_i(t) = e^{-\lambda_5 t}\nu_i(0) + \lambda_6 \int_0^t e^{-\lambda_5(t-s)}\psi(\nabla_i f(x(s)), \mu_i(s))ds$, which implies that $\nu_i(t) > 0$.

Since $\zeta_i(t) > 0$ and $\psi \geq 0$, due to (5.24) and (A.154), the R.H.S. in (A.155) is decreasing in $t \geq T$. Then, the L.H.S. in (A.155) is also decreasing in $t \geq T$. Since $\mu_i(t)$ and $\nu_i(t)$ are continuous and $\nu_i(t) > 0$, $\frac{\lambda_7}{2\lambda_2}\mu_i(t)^2\nu_i(t)^{-c}$ is continuous. Also, $\alpha_g(t)f(x(t))$ is continuous. Thus, considering the compact interval $[0, T]$, $\alpha_g(T)f(x(T)) + \sum_{i=1}^d \frac{\lambda_7}{2\lambda_2}\mu_i(T)^2\nu_i(T)^{-c} =: M_T$ is finite. Since the L.H.S. in (A.155) is decreasing in $t \geq T$, we have the L.H.S. in (A.155) bounded above by M_T for all $t \geq T$.

From the R.H.S. of (A.155) then we have that $\mu_i(t)^2\nu_i(t)^{-c-1}\psi(\nabla_i f(x(t)), \mu_i(t))$ and $\mu_i(t)^2\nu_i(t)^{-c}$ are integrable w.r.t. t and bounded. Moreover, if $\lambda_8 > 0$, we also have $\left\|\nabla_i f(x(t))\right\|^2$ $\nu_i(t)^{-c}$ integrable and bounded. It implies that, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded unless $\mu_i(t) = 0$ or $\nu_i(t) = \infty$. To show that $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded even if $\mu_i(t) = 0$ or $\nu_i(t) = \infty$, we consider the possible cases: $\lambda_8 = 0$ and $\lambda_8 > 0$.

- $\lambda_8 = 0$: From (5.23), either of the conditions $\mu_i(t) = 0$ and $\nu_i(t) = \infty$ implies that

$\dot{x}_i(t) = 0$ and, hence, $\frac{d}{dt}\nabla_i f(x(t)) = 0$. Due to continuity of $\nabla_i f$, we then have $\nabla_i f(x(t))$ is bounded. Upon solving (5.20), $\mu_i(t) = \lambda_2 \int_0^t e^{-\lambda_1(t-s)}\nabla_i f(x(s)ds$, and hence, $\mu_i(t)$ is bounded. From Assumption 5.3.A, $\nabla_i f(x(t))$ and $\mu_i(t)$ being bounded implies that $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded.

- $\lambda_8 > 0$: In this case, $\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c}$ is bounded. Thus, $\nabla_i f(x(t))$ is bounded unless $\nu_i(t) = \infty$, in which case $\dot{x}_i(t) = 0$. Following the argument in the previous case, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded.

Thus, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded, and therefore, Assumption 5.3.B implies that $\nabla_i f(x(t))$ is bounded.

To show that $\nu_i(t)$ is bounded, we consider the possible cases: $\lambda_4 > 0$ and $\lambda_4 = 0$.

- $\lambda_4 > 0$: We can rewrite ϕ_{22} as $\phi_{22}(t) = \frac{e^{-\frac{\lambda_3+\lambda_5-p}{2}t}(p-\lambda_5+\lambda_3)+e^{-\frac{\lambda_3+\lambda_5+p}{2}t}(p+\lambda_5-\lambda_3)}{2p}$. Since $p = \sqrt{(\lambda_3-\lambda_5)^2+4\lambda_3\lambda_4} \leq \lambda_3+\lambda_5$ for $\lambda_5 \geq \lambda_4$ (see (5.24)) and $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded, it follows from (A.157) that $\nu_i(t)$ is bounded.

- $\lambda_4 = 0$: In this case, $\nu_i(t) = e^{-\lambda_5 t}\nu_i(0) + \lambda_6 \int_0^t e^{-\lambda_5(t-s)}\psi(\nabla_i f(x(s)), \mu_i(s))ds$. The above implies that $\nu_i(t)$ is bounded as $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded.

Earlier, we have shown that $\mu_i(t)$ is bounded and $\nu_i(t) > 0$. From (5.23) then we have, $\dot{x}_i(t)$ is bounded. From (5.20), $\mu_i(t) = 0$ implies that $\dot{\mu}_i(t) = \lambda_1\nabla_i f(x(t))$. Thus, $\mu_i(t)$ can be zero only at isolated points t. Otherwise, for some $h > 0$ there exists an interval $(t-h, t+h)$ such that $\mu_i(s) = 0$ for all $s \in (t-h, t+h)$. In that case, $\dot{\mu}_i(s) = 0$ for all $s \in (t-h, t+h)$. Since $\dot{\mu}_i(s) = \lambda_1\nabla_i f(x(s))$ for all $s \in (t-h, t+h)$, we then have $\nabla_i f(x(s)) = 0$ for all $s \in (t-h, t+h)$, which proves the theorem.

We have shown above that $\mu_i(t) = 0$ only at isolated points and $\nu_i(t)$ is bounded. So, $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c}}$ is bounded except at isolated points. Since $\mu_i(t)^2\nu_i(t)^{-c}$ is integrable and $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c}}$ is bounded except at isolated points, we have $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c}}$ is integrable. Since $\nu_i(t)$ is bounded and $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c}}$ is integrable, we have $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c-1}}$ integrable. Now, we apply Cauchy-Schwartz inequality on the functions $\mu_i(t)\nu_i(t)^{(-c-1)/2}\psi(\nabla_i f(x(t)), \mu_i(t))^{0.5}$ and $\frac{1}{\mu_i(t)\nu_i(t)^{(-c-1)/2}}$. Since we have $\mu_i(t)^2\nu_i(t)^{-c-1}\psi(\nabla_i f(x(t)), \mu_i(t))$ and $\frac{1}{\mu_i(t)^2\nu_i(t)^{-c-1}}$ integrable, the Cauchy-Schwartz inequality implies that $\psi(\nabla_i f(x(t)), \mu_i(t))^{0.5}$ is integrable. Since $\psi(\nabla_i f(x(t)), \mu_i(t))^{0.5}$ is bounded and integrable, it is also square-integrable. Thus, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is integrable.

Now, the time-derivative of ψ along the trajectory $x_i(t)$ is

$$\frac{d}{dt}\psi(\nabla_i f(x(t)), \mu_i(t)) = \nabla\psi(\nabla_i f(x(t)), \mu_i(t))^T \begin{bmatrix} \left[\nabla^2 f(x(t))\right]_i \dot{x}(t) \\ \dot{\mu}_i(t) \end{bmatrix}.$$

We have shown that $\nabla_i f(x(t))$ and $\mu_i(t)$ are bounded. Then, according to Assumption 5.3.A, $\nabla\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded. From Assumption 3.2, $[\nabla^2 f(x(t))]_i$ is bounded. We have shown that $\dot{x}_i(t)$ is bounded. Since $\nabla_i f(x(t))$ and $\mu_i(t)$ are bounded, (5.20) implies that $\dot{\mu}_i(t)$ is bounded. Then, from the above equation we have $\frac{d}{dt}\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded. Thus, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is uniformly continuous w.r.t t.

We have shown that, for each $i \in \{1, \ldots, d\}$, $\psi(\nabla_i f(x(t)), \mu_i(t))$ is integrable and uniformly continuous. From Barbalat's lemma [157] it follows that $\lim_{t\to\infty} \psi(\nabla_i f(x(t)), \mu_i(t)) = 0$. Then, according to Assumption 5.3.C, $\lim_{t\to\infty} \nabla_i f(x(t)) = 0$.

Next, we consider the case $\lambda_7 = 0$. In this case, $\alpha_g(t) = 1$ (see (5.19)). Also, from (5.24), $\lambda_8 > 0$. For $\lambda_7 = 0$, the argument in the paragraph following (A.155) still holds, and we obtain

that $\nu_i(t) > 0$. Upon substituting $\lambda_7 = 0$ and $\alpha_g(t) = 1$ in (A.153),

$$f(x(t)) = f(x(0)) - \sum_{i=1}^{d} \int_0^t \lambda_8 \left\|\nabla_i f(x(s))\right\|^2 \nu_i(s)^{-c} ds.$$

Then, $f(x(t))$ is decreasing in t. From Assumption 3.1, f is bounded below. Thus, $\lim_{t\to\infty} f(x(t))$ is finite. So, the above equation implies that $\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c}$ is integrable. So, from the previous argument for the case $\lambda_8 > 0$, $\nabla_i f(x(t))$ is bounded. From (5.23) then we have, $\dot{x}_i(t)$ is bounded. Upon solving (5.20), $\mu_i(t) = \lambda_2 \int_0^t e^{-\lambda_1(t-s)} \nabla_i f(x(s) ds$, and hence, $\mu_i(t)$ is bounded. Now,

$$\begin{aligned}
&\frac{d}{dt}\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c} \\
&= -c\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c-1}\dot{\nu}_i(t) + 2\nabla_i f(x(t)) \left[\nabla^2 f(x(t))\right]_i \dot{x}(t)\nu_i(t)^{-c}. \qquad \text{(A.158)}
\end{aligned}$$

From Assumption 3.2, $[\nabla^2 f(x(t))]_i$ is bounded. From Assumption 5.3.A, $\nabla_i f(x(t))$ and $\mu_i(t)$ being bounded implies that $\psi(\nabla_i f(x(t)), \mu_i(t))$ is bounded. From the previous arguments for the cases $\lambda_4 = 0$ and $\lambda_4 > 0$, it follows that $\nu_i(t)$ and $\dot{\nu}_i(t)$ are bounded. So, (A.158) implies that $\frac{d}{dt}\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c}$ is bounded. Then, $\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c}$ is uniformly continuous. Again, we apply Barbalat's lemma [157] and obtain $\lim_{t\to\infty}\left\|\nabla_i f(x(t))\right\|^2 \nu_i(t)^{-c} = 0$. Since $\nu_i(t)$ is bounded, we get $\lim_{t\to\infty}\left\|\nabla_i f(x(t))\right\| = 0$. The proof is complete.

A.20 Proof of Theorem 5.3

The time-derivative of f along the trajectories of (5.29) is

$$\dot{f}(x(t)) = -\sum_{i=1}^{d} \nabla_i f(x(t)) \frac{\gamma}{h(t)} \mu_i(t)\eta_i(t).$$

Upon substituting above from (5.26), $\dot{f}(x(t)) = -\frac{\gamma}{\lambda_1 h(t)} \sum_{i=1}^{d} \left(\dot{\mu}_i(t) + \lambda_1 \mu_i(t)\right) \mu_i(t)\eta_i(t)$. Integrating with respect to (w.r.t.) time s from 0 to t,

$$\int_0^t h(s)\dot{f}(x(s))ds = -\frac{\gamma}{\lambda_1}\sum_{i=1}^{d}\int_0^t \dot{\mu}_i(s)\mu_i(s)\eta_i(s)ds - \gamma\sum_{i=1}^{d}\int_0^t \mu_i(s)^2\eta_i(s)ds. \tag{A.159}$$

Integrating by parts we have the first term on R.H.S. as $\int_0^t \dot{\mu}_i(s)\mu_i(s)\eta_i(s)ds = \left[\frac{1}{2}\mu_i(s)^2\eta_i(s)\right]_0^t - \int_0^t \frac{1}{2}\mu_i(s)^2\dot{\eta}_i(s)ds$ Upon substituting from above in (A.159) and using $\mu(0) = 0_d$, $\int_0^t h(s)\dot{f}(x(s))ds = -\frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t) + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\int_0^t \mu_i(s)^2\dot{\eta}_i(s)ds - \gamma\sum_{i=1}^{d}\int_0^t \mu_i(s)^2\eta_i(s)ds$. For each $t \geq 0$, we partition the set $\{s : 0 \leq s \leq t\}$ into $L_i(t) = \{s : 0 \leq s \leq t, \eta_i(s) = \frac{\eta_l(s)}{\alpha}\}$, $U_i(t) = \{s : 0 \leq s \leq t, \eta_i(s) = \frac{\eta_u(s)}{\alpha}\}$, and $M_i(t) = \{s : 0 \leq s \leq t, \eta_i(s) = \frac{1}{\sqrt{\nu_i(s)}}\}$. Upon rewriting the above equation and substituting from (5.27),

$$\begin{aligned}
&\int_0^t h(s)\dot{f}(x(s))ds + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t) = \frac{\gamma\lambda_2}{4\lambda_1}\sum_{i=1}^{d}\int_{M_i(t)}\frac{\mu_i(s)^2}{\sqrt{\nu_i(s)}}ds - \gamma\sum_{i=1}^{d}\int_0^t \mu_i(s)^2\eta_i(s)ds \\
&- \frac{\gamma\lambda_2}{4\lambda_1}\sum_{i=1}^{d}\int_{M_i(t)}\mu_i(s)^2\nu_i(s)^{-1.5}\left\|\nabla_i f(x(s))\right\|^2 \\
&+ \frac{\gamma}{2\alpha\lambda_1}\sum_{i=1}^{d}\left(\int_{L_i(t)}\mu_i(s)^2\dot{\eta}_l(s)ds + \int_{U_i(t)}\mu_i(s)^2\dot{\eta}_u(s)ds\right) \\
&= -\gamma\left(1 - \frac{\lambda_2}{4\lambda_1}\right)\sum_{i=1}^{d}\int_{M_i(t)}\frac{\mu_i(s)^2}{\sqrt{\nu_i(s)}}ds - \gamma\sum_{i=1}^{d}\int_{L_i(t)}\mu_i(s)^2\left(\eta_i(s) - \frac{\dot{\eta}_l(s)}{2\alpha\lambda_1}\right)ds \\
&- \gamma\sum_{i=1}^{d}\int_{U_i(t)}\mu_i(s)^2\eta_i(s)ds + \sum_{i=1}^{d}\int_{U_i(t)}\frac{\gamma\mu_i(s)^2\dot{\eta}_u(s)}{2\alpha\lambda_1}ds \\
&- \frac{\gamma\lambda_2}{4\lambda_1}\sum_{i=1}^{d}\int_{M_i(t)}\mu_i(s)^2\nu_i(s)^{-1.5}\left\|\nabla_i f(x(s))\right\|^2.
\end{aligned} \tag{A.160}$$

We let $r(t)$ denote the R.H.S. in (A.160). By (5.28) and Assumption 5.5, we have $0 < \frac{L}{\alpha} \leq \eta_i(t) \leq \frac{R}{\alpha}$ for $t \geq 0$. Under Assumption 5.4, η_u is non-increasing and $\dot{\eta}_l(t) \leq 2\alpha\lambda_1\eta_l(t)$ for $t \geq T_2$. So, under the condition $\lambda_2 < 4\lambda_1$, $r(t)$ is decreasing for $t \geq T_2$. Integrating by parts we rewrite the L.H.S. in (A.160) as $\int_0^t h(s)\dot{f}(x(s))ds = \left[h(s)f(x(s))\right]_0^t - \int_0^t \dot{h}(s)f(x(s))ds$.

Under Assumption 5.3, we define $T = \max(T_1, T_2)$. Upon substituting from above in (A.160), for $t \geq T$,

$$h(t)f(x(t)) + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t)$$
$$= h(0)f(x(0)) + \int_0^T \dot{h}(s)f(x(s))ds + \int_T^t \dot{h}(s)f(x(s))ds + r(t). \tag{A.161}$$

From (A.161), since $r(t)$ is decreasing and $h(t)$ non-increasing in $t \geq T$, we have $h(t)f(x(t)) + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t)$ is decreasing for $t \geq T$. Since $h(t)f(x(t))+\frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t)$ is continuous, considering the compact interval $[0, T]$, $h(T)f(x(T)) + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(T)^2\eta_i(T)$ is finite. Then, $h(t)f(x(t)) + \frac{\gamma}{2\lambda_1}\sum_{i=1}^{d}\mu_i(t)^2\eta_i(t)$ is bounded for $t \geq T$. So, $h(t)f(x(t))$ is bounded for $t \geq T$. From (A.159) and (A.161), it follows that $\dot{\mu}_i(t)\mu_i(t)\eta_i(t)$ and $\mu_i(t)^2\eta_i(t)$ are bounded and integrable. So, their superposition $\big(\dot{\mu}_i(t)\mu_i(t)\eta_i(t) + \lambda_1\mu_i(t)^2\eta_i(t)\big)$ is integrable. Upon substituting $\dot{\mu}_i(t)$ above from (5.26), $\lambda_1\nabla_i f(x(t))\mu_i(t)\eta_i(t)$ is integrable.

From (5.26) and (5.29), $\mu_i(t) = 0$ implies $\dot{\mu}_i(t) = \lambda_1\nabla_i f(x(t))$ and $\dot{x}_i(t) = 0$. So, $\mu_i(t)$ can be zero only at isolated points t. Otherwise, for some $h > 0$ there exists an interval $(t-h, t+h)$ such that $\mu_i(s) = 0$ for all $s \in (t-h, t+h)$. In that case, $\dot{\mu}_i(s) = 0$ for all $s \in (t-h, t+h)$. Since $\dot{\mu}_i(s) = \lambda_1\nabla_i f(x(s))$ for all $s \in (t-h, t+h)$, we then have $\nabla_i f(x(s)) = 0$ for all $s \in (t-h, t+h)$, which proves the theorem. We also have $\eta_i(t) > 0$. Thus, $\mu_i(t)^2\eta_i(t)$ can be zero only at isolated points. Since $\nabla_i f(x(t))\mu_i(t)\eta_i(t)$ and $\mu_i(t)^2\eta_i(t)$ are integrable and $\mu_i(t)^2\eta_i(t)$ can be zero only at isolated points, $\frac{\nabla_i f(x(t))\mu_i(t)\eta_i(t)}{\mu_i(t)^2\eta_i(t)} = \frac{\nabla_i f(x(t))}{\mu_i(t)}$ is integrable. Since $\frac{\nabla_i f(x(t))}{\mu_i(t)}$ is integrable and $\mu_i(t)$ can be zero only at isolated points, $\frac{\nabla_i f(x(t))}{\mu_i(t)^2}$ is integrable. Similarly, since $\mu_i(t)^2\eta_i(t)$ is integrable and $\eta_i(t) > 0$, $\mu_i(t)^2$ is integrable. Since $\frac{\nabla_i f(x(t))}{\mu_i(t)^2}$ and $\mu_i(t)^2$ are integrable, Cauchy-Schwartz inequality on $\frac{\left|\nabla_i f(x(t))\right|^{0.5}}{\left|\mu_i(t)\right|}$ and $\left|\mu_i(t)\right|$ implies that $\left|\nabla_i f(x(t))\right|^{0.5}$ is integrable. So, $\left\|\nabla_i f(x(t))\right\|^2$ is integrable for each $i \in \{1, \ldots, d\}$. Thus,

$\left\|\nabla f(x(t)\right\|^2$ is integrable.

Since $\left\|\nabla f(x(t)\right\|^2$ is integrable, $\nabla f(x(t)$ is bounded. Since $\mu_i(t)^2\eta_i(t)$ is bounded and $0 < \frac{L}{\alpha} \leq \eta_i(t) \leq \frac{R}{\alpha}$, $\mu_i(t)$ and $\eta_i(t)$ are bounded. From (5.29), then $\dot{x}(t)$ is bounded. Now, $\frac{d}{dt}\left\|\nabla f(x(t))\right\|^2 = 2\nabla f(x(t))^T\nabla^2 f(x(t))\dot{x}(t)$. We have shown that $\nabla f(x(t))$ and $\dot{x}(t)$ are bounded. From Assumption 5.2, we have all the entries in $\nabla^2 f(x(t))$ bounded. Then, from the above equation we have $\frac{d}{dt}\left\|\nabla f(x(t))\right\|^2$ bounded, and so, $\left\|\nabla f(x(t))\right\|^2$ is uniformly continuous.

We have shown that, $\left\|\nabla f(x(t))\right\|^2$ is integrable and uniformly continuous. From Barbalat's lemma [156] it follows that $\lim_{t\to\infty}\left\|\nabla f(x(t))\right\|^2 = 0$. The proof is complete.

A.21 Proof of Theorem 5.4

The time-derivative of f along the trajectories $x(t)$ of (5.34) is

$$\dot{f}(x(t)) = -\sum_{i=1}^{d}\frac{\gamma_1\nabla_i f(x(t))\mu_i(t)}{\alpha_2(t)\sqrt{\nu_i(t)}} - \sum_{i=1}^{d}\frac{\gamma_2\left\|\nabla_i f(x(t))\right\|^2}{\alpha_1(t)\sqrt{\nu_i(t)}}.$$

Upon integrating both sides above with respect to (w.r.t.) time s from 0 to t and substituting $\nabla_i f(x(s))$ from (5.32),

$$\int_0^t \alpha_2(s)\dot{f}(x(s))ds = -\sum_{i=1}^{d}\int_0^t \frac{\gamma_2\alpha_2(s)\left\|\nabla_i f(x(s))\right\|^2}{\alpha_1(s)\sqrt{\nu_i(s)}}ds$$
$$-\sum_{i=1}^{d}\int_0^t \frac{\gamma_1\mu_i(s)\dot{\mu}_i(s)}{\lambda_1\sqrt{\nu_i(s)}}ds - \sum_{i=1}^{d}\int_0^t \frac{\gamma_1\mu_i(s)^2}{\sqrt{\nu_i(s)}}ds.$$

Since $\lambda_1 \in (0,1)$, from the definitions (5.30)-(5.31) we get $\alpha_2(t) \geq \alpha_1(t)$. From above then we have

$$\int_0^t \alpha_2(s)\dot{f}(x(s))ds \leq -\sum_{i=1}^{d}\int_0^t \frac{\gamma_2\left\|\nabla_i f(x(s))\right\|^2}{\sqrt{\nu_i(s)}}ds$$
$$-\sum_{i=1}^{d}\int_0^t \frac{\gamma_1\mu_i(s)\dot{\mu}_i(s)}{\lambda_1\sqrt{\nu_i(s)}}ds - \sum_{i=1}^{d}\int_0^t \frac{\gamma_1\mu_i(s)^2}{\sqrt{\nu_i(s)}}ds. \tag{A.162}$$

Integrating by parts the second term on R.H.S. above and substituting $\dot{\nu}_i(s)$ from (5.33),

$$\int_0^t \alpha_2(s)\dot{f}(x(s))ds \leq -\sum_{i=1}^{d}\gamma_1\left(1-\frac{\lambda_2}{4\lambda_1}\right)\int_0^t \frac{\mu_i(s)^2}{\sqrt{\nu_i(s)}}ds$$
$$-\sum_{i=1}^{d}\frac{\gamma_1\mu_i(t)^2}{2\lambda_1\sqrt{\nu_i(t)}}-\sum_{i=1}^{d}\gamma_2\int_0^t \left\|\nabla_i f(x(s))\right\|^2\nu_i(s)^{-0.5}ds$$
$$-\sum_{i=1}^{d}\frac{\gamma_1\lambda_2}{4\lambda_1}\int_0^t \mu_i(s)^2\nu_i(s)^{-1.5}\left\|\nabla_i f(x(s))\right\|^2 ds. \tag{A.163}$$

We define, $a=1-\lambda_1$ and $b=1-\lambda_2$. Upon differentiating both sides of (5.31) w.r.t t we get

$$\dot{\alpha_2}(t)<0 \iff \left(\frac{b}{a}\right)^{t+1}\frac{1-a^{t+2}}{1-b^{t+1}} > 2a\frac{\log a}{\log b}. \tag{A.164}$$

From the condition $\lambda_1 > \lambda_2$, we have $1>b>a>0$. Then, $\left(\frac{b}{a}\right)^{t+1}$ and $\frac{1-a^{t+2}}{1-b^{t+1}}$ are, respectively, increasing and decreasing functions. Since $1>b>a>0$, we have $\lim_{t\to\infty}\left(\frac{b}{a}\right)^{t+1}\to\infty$ and $\lim_{t\to\infty}\frac{1-a^{t+2}}{1-b^{t+1}}=1$. Thus, $\left(\frac{b}{a}\right)^{t+1}\frac{1-a^{t+2}}{1-b^{t+1}}$ is an increasing function of t. Then, there exists $T<\infty$ such that (A.164) holds for all $t\geq T$. Integrating by parts the L.H.S. in (A.163), for $t\geq T$ we have

$$\alpha_2(t)f(x(t))+\sum_{i=1}^{d}\frac{\gamma_1\mu_i(t)^2}{2\lambda_1\sqrt{\nu_i(t)}}-\int_T^t \dot{\alpha}_2(s)f(x(s))ds$$
$$+\sum_{i=1}^{d}\int_0^t\left(\left(1-\frac{\lambda_2}{4\lambda_1}\right)\frac{\gamma_1\mu_i(s)^2}{\sqrt{\nu_i(s)}}+\frac{\gamma_2\left\|\nabla_i f(x(s))\right\|^2}{\sqrt{\nu_i(s)}}\right)ds$$
$$+\sum_{i=1}^{d}\frac{\gamma_1\lambda_2}{4\lambda_1}\int_0^t \mu_i(s)^2\nu_i(s)^{-1.5}\left\|\nabla_i f(x(s))\right\|^2 ds$$
$$\leq \alpha_2(0)f(x(0))+\int_0^T \dot{\alpha}_2(s)f(x(s))ds. \tag{A.165}$$

Since $\dot{\alpha}_2(t)<0$ for $t\geq T$ and $\lambda_2<\lambda_1$, from above we have the L.H.S. in (A.165) is bounded. Then, $\mu_i(t)^2\nu_i(t)^{-1.5}\left\|\nabla_i f(x(t))\right\|^2$, $\mu_i(t)^2\nu_i(t)^{-0.5}$, and $\left\|\nabla_i f(x(t))\right\|^2\nu_i(t)^{-0.5}$ are

integrable w.r.t. t and bounded. It implies that, $\left\|\nabla_i f(x(t))\right\|$ is bounded unless $\nu_i(t) = \infty$. From (5.34), $\nu_i(t) = \infty$ implies that $\dot{x}_i(t) = 0$ and, hence, $\frac{d}{dt}\nabla_i f(x(t)) = 0$. Due to continuity of $\nabla_i f$ and $\left\|\nabla f(x(0))\right\| < \infty$, we then have $\left\|\nabla_i f(x(t))\right\|$ is bounded for all t. Integrating both sides of (5.32)-(5.33), $\mu_i(t) = \lambda_1 \int_0^t e^{-\lambda_1(t-s)}\nabla_i f(x(s))ds$ and $\nu_i(t) = \lambda_2 \int_0^t e^{-\lambda_2(t-s)}\left\|\nabla_i f(x(s))\right\|^2 ds + \nu_i(0)e^{-\lambda_2 t}$. Since $\left\|\nabla f(x(t))\right\|$ is bounded and $\lambda_1, \lambda_2 > 0$, the above equations imply that $\mu_i(t)$ and $\nu_i(t)$ are bounded. Moreover, $\nu_i(t) > 0$ as $\nu_i(0) > 0$. From (5.34), then $\dot{x}(t)$ is bounded.

Now, $\frac{d}{dt}\frac{\left\|\nabla_i f(x(t))\right\|^2}{\sqrt{\nu_i(t)}} = \frac{2\nabla_i f(x(t))\left[\nabla^2 f(x(t))\right]_i \dot{x}(t)}{\sqrt{\nu_i(t)}} - \frac{\left\|\nabla_i f(x(t))\right\|^2 \dot{\nu}_i(t)}{2\nu_i(t)^{1.5}}$. From Assumption 5.2, $[\nabla^2 f(x(t))]_i$ is bounded. Since $\left\|\nabla f(x(t))\right\|$ and $\nu_i(t)$ are bounded, (5.33) implies that $\dot{\nu}_i(t)$ is bounded. We have also shown that $\nu_i(t) > 0$ and $\dot{x}_i(t)$ is bounded. So, the above equation implies that $\frac{d}{dt}\frac{\left\|\nabla_i f(x(t))\right\|^2}{\sqrt{\nu_i(t)}}$ is bounded. Then, $\frac{\left\|\nabla_i f(x(t))\right\|^2}{\sqrt{\nu_i(t)}}$ is uniformly continuous. We apply Barbalat's lemma [156] and obtain $\lim_{t\to\infty}\frac{\left\|\nabla_i f(x(t))\right\|^2}{\sqrt{\nu_i(t)}} = 0$. Since $\nu_i(t)$ is bounded, we get $\lim_{t\to\infty}\left\|\nabla_i f(x(t))\right\| = 0$. The proof is complete.

A.22 Proof of Theorem 5.5

We utilize the proof of Theorem 5.2. Note that, instead of specifying the expression of bias correction $\alpha_g(t)$, Theorem 5.2 also holds under the additional condition that there exists $T < \infty$ such that $\alpha_g(t)$ in (5.19) is decreasing for $t \geq T$.

We define $b = 1 - \lambda_2$. From (5.35), the time-derivative of $\rho(t)$ is

$$\dot{\rho}(t) = -2b^{t+1}\frac{1 - b^{t+1} - \log(1/b) - t\log(1/b)}{(1-b^{t+1})^2}.$$

Since $b \in (0,1)$, the above implies that there exists $T_1 < \infty$ such that $\dot{\rho}(t) > 0$ for $t \geq T_1$. From the definition (5.36) then $\dot{r}(t) > 0$ for $t \geq T_1$. Since $\dot{\rho}(t) > 0$ for $t \geq T_1$, from (5.37)

we have that there exists $T_2 \in [T_1, \infty)$ such that $\dot{x}_i(t) = -\gamma \frac{r(t)}{\alpha_1(t)} \frac{\mu_i(t)}{\sqrt{\nu_i(t)}}$ for $t \geq T_2$. Upon differentiating (5.30) w.r.t. t and following the argument after (A.154), we have that there exists $T_3 < \infty$ such that $\dot{\alpha}_1(t) < 0$ for $t \geq T_3$. Thus, there exists $T = \max\{T_2, T_3\}$ such that $\frac{\alpha_1(t)}{r(t)}$ is decreasing for $t \geq T$. Due to continuity, considering the finite interval $[0, T]$, we have that $\nabla f(x(t))$, $\mu(t)$, $\nu(t)$, and $x(t)$ are bounded for $t \in [0, T]$.

Upon comparing with the state-space model (5.13)-(5.15), the argument in the above paragraph implies that, for all $t \geq T$, the ODEs (5.32), (5.33), and (5.37) are similar to Adam if we replace $\alpha(t)$ in (5.12) with $\frac{\alpha_1(t)}{r(t)}$. Now, we have shown that $\frac{\alpha_1(t)}{r(t)}$ is decreasing for $t \geq T$, $\big\{\nabla f(x(t)), \mu(t), \nu(t), x(t) : 0 \leq t \leq T\big\}$ is bounded, and Theorem 5.2 holds under the additional condition that there exists $T < \infty$ such that $\alpha_g(t)$ in (5.19) is decreasing for $t \geq T$. Thus, following the proof of Theorem 5.2 we have that $\lim_{t\to\infty} \nabla f(x(t)) = 0_d$. The proof is complete.

A.23 Proof of Theorem 5.6

We consider the non-trivial case $\nabla f(x(t)) \neq 0_d$. We define, $q_1 = \frac{p_1}{p_1 - 1}$ and $q_2 = \frac{p_2}{p_2 - 1}$. The time-derivative of f along the trajectories $x(t)$ of (5.38) is $\dot{f}(x(t)) = \nabla f(x(t))^T \dot{x}(t) = -c_1 \big\|\nabla f(x(t))\big\|^{q_1} - c_2 \big\|\nabla f(x(t))\big\|^{q_2}$. Upon integrating both sides above w.r.t. time s from 0 to t, $f(x(t)) + c_1 \int_0^t \big\|\nabla f(x(s))\big\|^{q_1} ds + c_2 \int_0^t \big\|\nabla f(x(s))\big\|^{q_2} ds = f(x(0))$. Since $f(x(t)) > 0$ and $c_1, c_2 > 0$, the above implies that $\big\|\nabla f(x(t))\big\|^{q_2}$ is integrable and bounded. Since $p_2 \in (1, 2)$, from the definition of q_2 we obtain $q_2 > 2$. Since $\big\|\nabla f(x(t))\big\|^{q_2}$ is bounded and $q_2 > 2$, $\big\|\nabla f(x(t))\big\|$ and $\big\|\nabla f(x(t))\big\|^{q_2 - 2}$ are bounded.

From triangle inequality on the R.H.S. of (5.38),

$$\big\|\dot{x}(t)\big\| \leq c_1 \big\|\nabla f(x(t))\big\|^{\frac{1}{p_1 - 1}} + c_2 \big\|\nabla f(x(t))\big\|^{\frac{1}{p_2 - 1}}.$$

Since $\left\|\nabla f(x(t))\right\|$ is bounded and $p_1, p_2 > 1$, we have $\dot{x}(t)$ is bounded.

The time-derivative of $\left\|\nabla f(x(t))\right\|^{q_2}$ along the trajectories $x(t)$ of (5.38) is $\frac{d}{dt}\left\|\nabla f(x(t))\right\|^{q_2} = q_2\left\|\nabla f(x(t))\right\|^{q_2-2}\nabla f(x(t))^T\nabla^2 f(x(t))\dot{x}(t)$. We have shown that $\left\|\nabla f(x(t))\right\|, \left\|\nabla f(x(t))\right\|^{q_2-2}$, and $\dot{x}(t)$ are bounded. From Assumption 5.2, $\nabla^2 f(x(t))$ is bounded. Then, the above equation implies that $\frac{d}{dt}\left\|\nabla f(x(t))\right\|^{q_2}$ is bounded. So, $\left\|\nabla f(x(t))\right\|^{q_2}$ is uniformly continuous.

Since $\left\|\nabla f(x(t))\right\|^{q_2}$ is integrable and uniformly continuous, from Barbalat's lemma [156] $\lim_{t\to\infty}\left\|\nabla f(x(t))\right\|^{q_2} = 0$. Since $q_2 > 0$, the proof is complete.

Bibliography

[1] Harold W Sorenson. Least-squares estimation: from Gauss to Kalman. *IEEE spectrum*, 7(7):63–68, 1970.

[2] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[3] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[4] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[5] Navid Azizan-Ruhi, Farshad Lahouti, Amir Salman Avestimehr, and Babak Hassibi. Distributed solution of large-scale linear systems via accelerated projection-based consensus. *IEEE Transactions on Signal Processing*, 67(14):3806–3817, 2019.

[6] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. Iterative pre-conditioning to expedite the gradient-descent method. In *2020 American Control Conference (ACC)*, pages 3977–3982, 2020.

[7] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. Robustness of iteratively pre-conditioned gradient-descent method: The case of distributed linear regression problem. *IEEE Control Systems Letters*, 5(6):2180–2185, 2020.

[8] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. Accelerating distributed SGD for linear regression using iterative pre-conditioning. In *Learning for Dynamics and Control*, pages 447–458. PMLR, 2021.

[9] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. Iterative pre-conditioning for expediting the distributed gradient-descent method: The case of linear least-squares problem. *Automatica*, 137:110095, 2022.

[10] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. On distributed solution of ill-conditioned system of linear equations under communication delays. In *2019 Sixth Indian Control Conference (ICC)*, pages 413–418. IEEE, 2019.

[11] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. On pre-conditioning of decentralized gradient-descent when solving a system of linear equations. *IEEE Transactions on Control of Network Systems*, 9(2):811–822, 2022.

[12] Kushal Chakrabarti and Nikhil Chopra. Generalized AdaGrad (G-AdaGrad) and Adam: A state-space perspective. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 1496–1501, 2021.

[13] Kushal Chakrabarti and Nikhil Chopra. Analysis and synthesis of adaptive gradient algorithms in machine learning: The case of AdaBound and MAdamSSM. In *2022 61st IEEE Conference on Decision and Control (CDC)*, 2022. (accepted).

[14] Kushal Chakrabarti and Nikhil Chopra. A state-space perspective on the expedited gradient methods: Nadam, RAdam, and rescaled gradient flow. In *2022 Eighth Indian Control Conference (ICC)*. IEEE, 2022. (accepted).

[15] Kushal Chakrabarti, Amrit S. Bedi, Fikadu T. Dagefu, Jeffrey N. Twigg, and Nikhil Chopra. Fast distributed beamforming without receiver feedback. In *56th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022. (accepted).

[16] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *Journal of Machine Learning Research*, 18:1–42, 2017.

[17] Yinchu Zhu and Jelena Bradic. Linear hypothesis testing in dense high-dimensional linear models. *Journal of the American Statistical Association*, 113(524):1583–1600, 2018.

[18] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008.

[19] John Canny, Shi Zhong, Scott Gaffney, Chad Brower, Pavel Berkhin, and George H John. Method and system for generating a linear machine learning model for predicting online user input actions, January 29 2013. US Patent 8,364,627.

[20] Sanjukta Bhowmick, Victor Eijkhout, Yoav Freund, Erika Fuentes, and David Keyes. Application of machine learning to the selection of sparse linear solvers. *Int. J. High Perf. Comput. Appl*, 2006.

[21] Jeffrey A Fessler. Image reconstruction: Algorithms and analysis. `http://web.eecs.umich.edu/~fessler/book/c-opt.pdf`, 2020. [Online book draft; accessed 17-February-2020].

[22] Carl T Kelley. *Iterative methods for optimization*. SIAM, 1999.

[23] Y Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Sov. Math. Doklady*, 27(2):372–376, 1983.

[24] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[25] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[26] Mahyar Fazlyab, Alejandro Ribeiro, Manfred Morari, and Victor M Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

[27] Bernard Gold and Charles M Rader. Effects of quantization noise in digital filters. In *Proceedings of the April 26-28, 1966, Spring joint computer conference*, pages 213–219, 1966.

[28] Jordan L Holi and J-N Hwang. Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*, 42(3):281–290, 1993.

[29] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

[30] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[31] Othmane Sebbouh, Ch Dossal, and Aude Rondepierre. Convergence rates of damped inertial dynamics under geometric conditions and perturbations. *SIAM Journal on Optimization*, 30(3):1850–1877, 2020.

[32] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[33] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[36] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[37] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[38] Timothy Dozat. Incorporating nesterov momentum into adam. In *International Conference on Learning Representations Workshops*, 2016.

[39] Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. *The Journal of Machine Learning Research*, 17(1):5312–5354, 2016.

[40] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[41] Daniel Liberzon. Notes for ECE 517: Nonlinear and adaptive control, 2016.

[42] Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 71. SIAM, 2000.

[43] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[44] Kushal Chakrabarti, Nirupam Gupta, and Nikhil Chopra. Iterative pre-conditioning for expediting the gradient-descent method: The distributed linear least-squares problem. *arXiv preprint arXiv:2008.02856*, 2020.

[45] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[46] Dheeru Dua and Casey Graff. UCI machine learning repository. `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`, 2017. University of California, Irvine, School of Information and Computer Sciences.

[47] MNIST in CSV. `https://www.kaggle.com/oddrationale/mnist-in-csv`, 2018. Accessed: 19-September-2020.

[48] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012.

[49] Tao Yang, Jemin George, Jiahu Qin, Xinlei Yi, and Junfeng Wu. Distributed least squares solver for network linear equations. *Automatica*, 113:108798, 2020.

[50] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

[51] Ji Liu, Shaoshuai Mou, and A Stephen Morse. Asynchronous distributed algorithms for solving linear algebraic equations. *IEEE Transactions on Automatic Control*, 63(2):372–385, 2017.

[52] Keyou You, Shiji Song, and Roberto Tempo. A networked parallel algorithm for solving linear algebraic equations. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1727–1732. IEEE, 2016.

[53] Xuan Wang, Jingqiu Zhou, Shaoshuai Mou, and Martin J Corless. A distributed algorithm for least squares solutions. *IEEE Transactions on Automatic Control*, 2019.

[54] Yu Tang and Jie Mei. Distributed algorithms for solving a linear equation under a directed graph. In *2018 37th Chinese Control Conference (CCC)*, pages 7193–7198. IEEE, 2018.

[55] Peng Wang, Wei Ren, and Zhisheng Duan. Distributed algorithm to solve a system of linear equations with unique or multiple solutions from arbitrary initializations. *IEEE Transactions on Control of Network Systems*, 6(1):82–93, 2018.

[56] Guodong Shi, Brian DO Anderson, and Uwe Helmke. Network flows that solve linear equations. *IEEE Transactions on Automatic Control*, 62(6):2659–2674, 2016.

[57] Brian DO Anderson, Shaoshuai Mou, A Stephen Morse, and Uwe Helmke. Decentralized gradient algorithm for solution of a linear equation. *Numerical Algebra, Control & Optimization*, 6(3):319, 2016.

[58] Jingqiu Zhou, Xuan Wang, Shaoshuai Mou, and Brian DO Anderson. Finite-time distributed linear equation solver for solutions with minimum l_1-norm. *IEEE Transactions on Automatic Control*, 65(4):1691–1696, 2019.

[59] Lili Wang and A Stephen Morse. A distributed observer for a time-invariant linear system. *IEEE Transactions on Automatic Control*, 63(7):2123–2130, 2017.

[60] Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[61] Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

[62] S Sh Alaviani and Nicola Elia. A distributed algorithm for solving linear algebraic equations over random networks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 83–88. IEEE, 2018.

[63] Tianyu Wu, Kun Yuan, Qing Ling, Wotao Yin, and Ali H Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2017.

[64] Shaoshuai Mou, Ji Liu, and A Stephen Morse. A distributed algorithm for solving a linear algebraic equation. *IEEE Transactions on Automatic Control*, 60(11):2863–2878, 2015.

[65] Ji Liu, Xiaobin Gao, and Tamer Başar. A communication-efficient distributed algorithm for solving linear algebraic equations. In *2014 7th International Conference on NETwork Games, COntrol and OPtimization (NetGCoop)*, pages 62–69. IEEE, 2014.

[66] Xiaobin Gao, Ji Liu, and Tamer Başar. Stochastic communication-efficient distributed algorithms for solving linear algebraic equations. In *2016 IEEE Conference on Control Applications (CCA)*, pages 380–385. IEEE, 2016.

[67] Xuan Wang, Shaoshuai Mou, and Dengfeng Sun. Improvement of a distributed algorithm for solving linear equations. *IEEE Transactions on Industrial Electronics*, 64(4):3113–3117, 2016.

[68] Yang Liu, Youcheng Lou, Brian Anderson, and Guodong Shi. Network flows that solve least squares for linear equations. *arXiv preprint arXiv:1808.04140*, 2018.

[69] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

[70] Konstantinos I Tsianos and Michael G Rabbat. Distributed dual averaging for convex optimization under communication delays. In *2012 American Control Conference (ACC)*, pages 1067–1072. IEEE, 2012.

[71] Håkan Terelius, Ufuk Topcu, and Richard M Murray. Decentralized multi-agent optimization via dual decomposition. *IFAC proceedings volumes*, 44(1):11245–11251, 2011.

[72] Takeshi Hatanaka, Nikhil Chopra, Takayuki Ishizaki, and Na Li. Passivity-based distributed optimization with communication delays using PI consensus algorithm. *IEEE Transactions on Automatic Control*, 63(12):4421–4428, 2018.

[73] Reza Olfati-Saber. Distributed Kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498. IEEE, 2007.

[74] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

[75] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001.

[76] Howard E Bell. Gershgorin's theorem and the zeros of polynomials. *The American Mathematical Monthly*, 72(3):292–295, 1965.

[77] Shinkyu Park and Nuno C Martins. Design of distributed lti observers for state omniscience. *IEEE Transactions on Automatic Control*, 62(2):561–576, 2016.

[78] Lili Wang, A Stephen Morse, Daniel Fullmer, and Ji Liu. A hybrid observer for a distributed linear system with a changing neighbor graph. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1024–1029. IEEE, 2017.

[79] Romeo Ortega, Emmanuel Nuno, and Alexey Bobtsov. Distributed observers for LTI systems with finite convergence time: A parameter estimation-based approach. *IEEE Transactions on Automatic Control*, 2020.

[80] Joao P Hespanha. *Linear systems theory*. Princeton university press, 2018.

[81] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems*, volume 33, pages 18795–18806, 2020.

[82] Qianqian Tong, Guannan Liang, and Jinbo Bi. Calibrating the adaptive learning rate to improve convergence of ADAM. *arXiv preprint arXiv:1908.00700*, 2019.

[83] Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Advances in neural information processing systems*, 27:2510–2518, 2014.

[84] Youzuo Lin, Daniel O'Malley, and Velimir Valentinov Vesselinov. A computationally efficient Levenberg-Marquardt algorithm and its application to hydrogeologic inverse modeling. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2016.

[85] Jarosław Bilski, Bartosz Kowalczyk, Alina Marchlewska, and Jacek M Zurada. Local Levenberg-Marquardt algorithm for learning feedforward neural networks. *Journal of Artificial Intelligence and Soft Computing Research*, 10, 2020.

[86] André L Tits. Notes for ENEE 664: Optimal control, 2011.

[87] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[88] Gonglin Yuan, Zengxin Wei, and Yuning Yang. The global convergence of the Polak–Ribière–Polyak conjugate gradient algorithm under inexact line search for nonconvex functions. *Journal of Computational and Applied Mathematics*, 362:262–275, 2019.

[89] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[90] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-Newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[91] Thomas O'Leary-Roseberry, Nick Alger, and Omar Ghattas. Inexact Newton methods for stochastic nonconvex optimization with applications to neural network training. *arXiv preprint arXiv:1905.06738*, 2019.

[92] Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1):35–70, 2020.

[93] Quoc Tran-Dinh, Nhan Pham, and Lam Nguyen. Stochastic gauss-newton algorithms for nonconvex compositional optimization. In *International Conference on Machine Learning*, pages 9572–9582. PMLR, 2020.

[94] Brian Bullins, Kshitij Patel, Ohad Shamir, Nathan Srebro, and Blake E Woodworth. A stochastic Newton algorithm for distributed convex optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[95] Xuezhe Ma. APOLLO: An adaptive parameter-wise diagonal quasi-newton method for nonconvex stochastic optimization. *arXiv preprint arXiv:2009.13586*, 2020.

[96] Satish Balay, Kris Buschelman, Victor Eijkhout, William D Gropp, Dinesh Kaushik, M Knepley, Lois Curfman McInnes, Barry F Smith, and Hong Zhang. Petsc users manual technical report anl-95/11-revision 2.1. 3, argonne national laboratory, 2002. *Google Scholar*, 2002.

[97] Tjalling J Ypma. The effect of rounding errors on Newton-like methods. *IMA Journal of Numerical Analysis*, 3(1):109–118, 1983.

[98] John E Dennis and Homer F Walker. Inaccuracy in quasi-Newton methods: Local improvement theorems. In *Mathematical Programming at Oberwolfach II*, pages 70–85. Springer, 1984.

[99] Tony Doungho Choi and Carl T Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162, 2000.

[100] Albert S Berahas, Richard H Byrd, and Jorge Nocedal. Derivative-free optimization of noisy functions via quasi-Newton methods. *SIAM Journal on Optimization*, 29(2):965–993, 2019.

[101] Hao-Jun M Shi, Yuchen Xie, Richard Byrd, and Jorge Nocedal. A noise-tolerant quasi-Newton algorithm for unconstrained optimization. *SIAM Journal on Optimization*, 32(1):29–55, 2022.

[102] James Lucas, Juhan Bae, Michael R Zhang, Stanislav Fort, Richard Zemel, and Roger Grosse. Analyzing monotonic linear interpolation in neural network loss landscapes. *arXiv preprint arXiv:2104.11044*, 2021.

[103] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *International Conference on Machine Learning*, pages 343–351. PMLR, 2013.

[104] Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610*, 2019.

[105] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

[106] Yeming Wen, Kevin Luk, Maxime Gazeau, Guodong Zhang, Harris Chan, and Jimmy Ba. An empirical study of large-batch stochastic gradient descent with structured covariance noise. *arXiv e-prints*, pages arXiv–1902, 2019.

[107] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate Bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

[108] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

[109] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, 2020.

[110] Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[111] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[112] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[113] Liangchen Luo, Wenhao Huang, Qi Zeng, Zaiqing Nie, and Xu Sun. Learning personalized end-to-end goal-oriented dialog. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6794–6801, 2019.

[114] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[115] John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[116] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[117] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[118] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[119] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[120] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019.

[121] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[122] Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. *arXiv preprint arXiv:2002.03432*, 2020.

[123] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413. PMLR, 2018.

[124] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[125] Manzil Zaheer, Sashank J. Reddi, Devendra Singh Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *NeurIPS*, pages 9815–9825, 2018.

[126] Chen Zhu, Yu Cheng, Zhe Gan, Furong Huang, Jingjing Liu, and Tom Goldstein. MaxVA: Fast adaptation of step sizes by maximizing observed variance of gradients. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 628–643. Springer, 2021.

[127] Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, 2017.

[128] Lichao Mou, Lorenzo Bruzzone, and Xiao Xiang Zhu. Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2):924–935, 2018.

[129] Lichao Mou, Yuansheng Hua, and Xiao Xiang Zhu. A relation-augmented fully convolutional network for semantic segmentation in aerial scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12416–12425, 2019.

[130] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[131] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[132] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.

[133] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, 2020.

[134] Jizong Peng, Ping Wang, Christian Desrosiers, and Marco Pedersoli. Self-paced contrastive learning for semi-supervised medical image segmentation with meta-labels. *Advances in Neural Information Processing Systems*, 34, 2021.

[135] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

[136] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

[137] Xiaoxia Wu, Rachel Ward, and Léon Bottou. WNGrad: Learn the learning rate in gradient descent. *arXiv preprint arXiv:1803.02865*, 2018.

[138] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686. PMLR, 2019.

[139] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

[140] Anas Barakat and Pascal Bianchi. Convergence rates of a momentum algorithm with bounded adaptive step size for nonconvex optimization. In *Asian Conference on Machine Learning*, pages 225–240. PMLR, 2020.

[141] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019.

[142] Anas Barakat and Pascal Bianchi. Convergence and dynamical behavior of the ADAM algorithm for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 31(1):244–274, 2021.

[143] Dimitri Bertsekas. *Convex optimization algorithms*. Athena Scientific, 2015.

[144] Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *2017 American Control Conference (ACC)*, pages 3114–3119. IEEE, 2017.

[145] Bin Hu and Laurent Lessard. Dissipativity theory for Nesterov's accelerated method. In *International Conference on Machine Learning*, pages 1549–1557. PMLR, 2017.

[146] AC Wilson, B Recht, and MI Jordan. A Lyapunov analysis of momentum methods in optimization (2016). *arXiv preprint arXiv:1611.02635*, 2016.

[147] Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

[148] Guilherme França, Daniel P Robinson, and René Vidal. Gradient flows and proximal splitting methods: A unified view on accelerated and stochastic optimization. *Physical Review E*, 103(5):053304, 2021.

[149] Bin Shi, Simon S Du, Michael I Jordan, and Weijie J Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, pages 1–70, 2021.

[150] Miguel Vaquero, Pol Mestres, and Jorge Cortes. Resource-aware discretization of accelerated optimization flows: the heavy-ball dynamics case. *IEEE Transactions on Automatic Control*, 2022.

[151] Pedro Savarese. On the convergence of AdaBound and its connection to SGD. *arXiv preprint arXiv:1908.04457*, 2019.

[152] Jinlan Liu, Jun Kong, Dongpo Xu, Miao Qi, and Yinghua Lu. Convergence analysis of AdaBound with relaxed bound functions for non-convex optimization. *Neural Networks*, 145:300–307, 2022.

[153] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8828–8836, 2021.

[154] Param Budhraja, Mayank Baranwal, Kunal Garg, and Ashish Hota. Breaking the convergence barrier: Optimization via fixed-time convergent flows. *arXiv preprint arXiv:2112.01363*, 2021.

[155] Ashia C Wilson, Lester Mackey, and Andre Wibisono. Accelerating rescaled gradient descent: Fast optimization of smooth functions. *Advances in Neural Information Processing Systems*, 32, 2019.

[156] Hassan K Khalil. *Nonlinear systems*. Pearson New York, 2002.

[157] I Barbalat. Systemes d'équations différentielles d'oscillations non linéaires. *Rev. Math. Pures Appl*, 4(2):267–270, 1959.

[158] Hesameddin Mohammadi, Samantha Samuelson, and Mihailo R Jovanovic. Transient growth of accelerated optimization algorithms. *IEEE Transactions on Automatic Control*, 2022.

[159] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[160] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[161] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[162] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[163] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

[164] Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. `https://web.stanford.edu/~jurafsky/slp3/ed3book_sep212021.pdf`. [Third edition draft; 21-September-2021].

[165] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[166] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, and Chee Yen Leow. Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions. *IEEE Communications Surveys & Tutorials*, 19(4):2092–2116, 2017.

[167] Keyvan Zarifi, Sofiène Affes, and Ali Ghrayeb. Collaborative null-steering beamforming for uniformly distributed wireless sensor networks. *IEEE Transactions on Signal Processing*, 58(3):1889–1903, 2009.

[168] N NikAbdMalik, Mazlina Esa, and Nurul Mu'azzah Abdul Latiff. Least-square collaborative beamforming linear array for steering capability in green wireless sensor networks. *Journal of Electronic Science and Technology*, 14(2):118–125, 2016.

[169] Shahab Farazi, Kim Chinkidjakarn, and D Richard Brown. Simultaneous distributed beamforming and nullforming with adaptive positioning. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 129–132. IEEE, 2016.

[170] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, Chee Yen Leow, and Mohd Fairus Mohd Yusof. Beampatten optimization in distributed beamforming using multiobjective and metaheuristic method. In *2014 IEEE Symposium on Wireless Technology and Applications (ISWTA)*, pages 86–91. IEEE, 2014.

[171] Nikolaos Chatzipanagiotis, Yupeng Liu, Athina Petropulu, and Michael M Zavlanos. Distributed cooperative beamforming in multi-source multi-destination clustered systems. *IEEE Transactions on Signal Processing*, 62(23):6105–6117, 2014.

[172] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, and Chee Yen Leow. Psogsa-explore: A new hybrid metaheuristic approach for beampattern optimization in collaborative beamforming. *Applied Soft Computing*, 30:229–237, 2015.

[173] D Richard Brown, Upamanyu Madhow, Patrick Bidigare, and Soura Dasgupta. Receiver-coordinated distributed transmit nullforming with channel state uncertainty. In *2012 46th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2012.

[174] Lei Yu, Yinsheng Wei, and Wei Liu. Adaptive beamforming based on nonuniform linear arrays with enhanced degrees of freedom. In *TENCON 2015-2015 IEEE Region 10 Conference*, pages 1–5. IEEE, 2015.

[175] Yongsheng Fan, Yuanping Zhou, Donglin He, and Wenlong Xia. Fast transmit beamforming with distributed antennas. *IEEE Antennas and Wireless Propagation Letters*, 16:121–124, 2016.

[176] Sairam Goguri, Ben Peiffer, Raghu Mudumbai, and Soura Dasgupta. A class of scalable feedback algorithms for beam and null-forming from distributed arrays. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 1447–1451. IEEE, 2016.

[177] Justin S Kong, Fikadu T Dagefu, and Brian M Sadler. Distributed adaptive beamforming and nullforming for secure wireless communications, March 31 2022. US Patent App. 17/205,355.

[178] Barry D Van Veen and Kevin M Buckley. Beamforming: A versatile approach to spatial filtering. *IEEE assp magazine*, 5(2):4–24, 1988.

[179] Amy Kumar, Raghuraman Mudumbai, Soura Dasgupta, Upamanyu Madhow, and D Richard Brown. Distributed mimo multicast with protected receivers: A scalable algorithm for joint beamforming and nullforming. *IEEE Transactions on Wireless Communications*, 16(1):512–525, 2016.

[180] Jemin George, Anjaly Parayil, Cemal Tugrul Yilmaz, Bethany L Allik, He Bai, and Aranya Chakrabortty. Multi-agent coordination for distributed transmit beamforming. In *2020 American Control Conference (ACC)*, pages 144–149. IEEE, 2020.

[181] Jemin George, Cemal Tugrul Yilmaz, Anjaly Parayil, and Aranya Chakrabortty. A model-free approach to distributed transmit beamforming. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5170–5174. IEEE, 2020.

[182] Anjaly Parayil, Amrit Singh Bedi, and Alec Koppel. Joint position and beamforming control via alternating nonlinear least-squares with a hierarchical gamma prior. In *2021 American Control Conference (ACC)*, pages 3513–3518. IEEE, 2021.

[183] Tzanis Anevlavis, Jonathan Bunton, Anjaly Parayil, Jemin George, and Paulo Tabuada. To beam or not to beam? beamforming with submodularity-inspired group sparsity. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 390–395. IEEE, 2020.

[184] Arjun Muralidharan and Yasamin Mostofi. Energy optimal distributed beamforming using unmanned vehicles. *IEEE Transactions on Control of Network Systems*, 5(4):1529–1540, 2017.

[185] Samer Hanna, Enes Krijestorac, and Danijela Cabric. Destination-feedback free distributed transmit beamforming using guided directionality. *arXiv preprint arXiv:2108.01837*, 2021.

[186] Costas Kravaris, Juergen Hahn, and Yunfei Chu. Advances and selected recent developments in state and parameter estimation. *Computers & chemical engineering*, 51:111–123, 2013.

[187] Nikolaos Kazantzis and Costas Kravaris. Discrete-time nonlinear observer design using functional equations. *Systems & Control Letters*, 42(2):81–94, 2001.

[188] Arthur J Krener and MingQing Xiao. Nonlinear observer design in the Siegel domain. *SIAM Journal on Control and Optimization*, 41(3):932–953, 2002.

[189] Alessandro Astolfi and Laurent Praly. Global complete observability and output-to-state stability imply the existence of a globally convergent observer. *Mathematics of Control, Signals and Systems*, 18(1):32–65, 2006.

[190] Nikolas Kazantzis. Map invariance and the state reconstruction problem for nonlinear discrete-time systems. *European Journal of Control*, 15(2):105–119, 2009.

[191] Jeff S Shamma and Kuang-Yang Tu. Approximate set-valued observers for nonlinear systems. *IEEE Transactions on Automatic Control*, 42(5):648–658, 1997.

[192] Masoud Abbaszadeh and Horacio J Marquez. Robust H∞ observer design for sampled-data Lipschitz nonlinear systems with exact and Euler approximate models. *Automatica*, 44(3):799–806, 2008.

[193] Iasson Karafyllis and Costas Kravaris. From continuous-time design to sampled-data design of observers. *IEEE Transactions on Automatic Control*, 54(9):2169–2174, 2009.

[194] Madiha Nadri, Hassan Hammouri, and Rafael Mota Grajales. Observer design for uniformly observable systems with sampled measurements. *IEEE Transactions on Automatic Control*, 58(3):757–762, 2012.

[195] Vincent Andrieu, Madiha Nadri, Ulysse Serres, and Jean-Claude Vivalda. Self-triggered continuous–discrete observer with updated sampling period. *Automatica*, 62:106–113, 2015.

[196] Daoyuan Zhang, Yanjun Shen, Jun Mei, and Zhi-Hong Guan. Sampled-data observer design for a class of nonlinear systems with delayed measurements. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 2242–2246. IEEE, 2016.

[197] Hossein Beikzadeh and Horacio J Marquez. Input-to-error stable observer for nonlinear sampled-data systems with application to one-sided Lipschitz systems. *Automatica*, 67:1–7, 2016.

[198] Aleksandar Haber, Ferenc Molnar, and Adilson E Motter. State observation and sensor selection for nonlinear networks. *IEEE Transactions on Control of Network Systems*, 5(2):694–708, 2017.

[199] Tarek Ahmed-Ali, Iasson Karafyllis, and Fouad Giri. Sampled-data observers for delay systems. *IFAC-PapersOnLine*, 53(2):5901–5908, 2020.

[200] Laura Menini, Corrado Possieri, and Antonio Tornambè. Design of high-gain observers based on sampled measurements via the interval arithmetic. *Automatica*, 131:109741, 2021.

[201] Saeed Kashefi and Majid Hajatipour. New optimal observer design for a class of nonlinear systems based on approximation. *International Journal of Dynamics and Control*, pages 1–12, 2022.

[202] J Gauthier and G Bornard. Observability for any $u(t)$ of a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 26(4):922–926, 1981.

[203] Antonio Tornambè. High-gain observers for non-linear systems. *International Journal of Systems Science*, 23(9):1475–1489, 1992.

[204] G Ciccarella, M Dalla Mora, and A Germani. A robust observer for discrete time nonlinear systems. *Systems & Control Letters*, 24(4):291–300, 1995.

[205] Hassan K Khalil and Laurent Praly. High-gain observers in nonlinear feedback control. *International Journal of Robust and Nonlinear Control*, 24(6):993–1015, 2014.

[206] Laura Menini, Corrado Possieri, and Antonio Tornambe. A "practical" observer for nonlinear systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 3015–3020. IEEE, 2017.

[207] Hyungbo Shim and Daniel Liberzon. Nonlinear observers robust to measurement disturbances in an ISS sense. *IEEE Transactions on Automatic Control*, 61(1):48–61, 2015.

[208] Christopher V Rao, James B Rawlings, and David Q Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Transactions on Automatic Control*, 48(2):246–258, 2003.

[209] Angelo Alessandri, Marco Baglietto, and Giorgio Battistelli. Moving-horizon state estimation for nonlinear discrete-time systems: New stability results and approximation schemes. *Automatica*, 44(7):1753–1765, 2008.

[210] Angelo Alessandri, Marco Baglietto, Giorgio Battistelli, and Victor Zavala. Advances in moving horizon estimation for nonlinear systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5681–5688. IEEE, 2010.

[211] Angelo Alessandri and Mauro Gaggero. Fast moving horizon state estimation for discrete-time systems using single and multi iteration descent methods. *IEEE Transactions on Automatic Control*, 62(9):4499–4511, 2017.

[212] PE Moraal and Jessy W Grizzle. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 40(3):395–404, 1995.

[213] Emrah Bıyık and Murat Arcak. A hybrid redesign of Newton observers in the absence of an exact discrete-time model. *Systems & Control Letters*, 55(6):429–436, 2006.

[214] E Biyik and Murat Arcak. Hybrid Newton observer design using the inexact Newton method and GMRES. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.

[215] Shigeru Hanba. Numerical nonlinear observers using pseudo-Newton-type solvers. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(17):1592–1606, 2008.

[216] Filippo Cacace, Valerio Cusimano, and Alfredo Germani. An efficient approach to the design of observers for continuous-time systems with discrete-time measurements. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7549–7554. IEEE, 2011.

[217] Dina Shona Laila and Alessandro Astolfi. Sampled-data observer design for a class of nonlinear systems with applications. In *Proc. 17th International Symposium on Mathematical Theory of Networks and Systems*, pages 715–722, 2006.

[218] Ramine Nikoukhah. A new methodology for observer design and implementation. *IEEE Transactions on Automatic Control*, 43(2):229–234, 1998.

[219] Daniele Carnevale, Sergio Galeani, Mario Sassano, and Alessandro Astolfi. Nonlinear observer design techniques with observability functions. In *52nd IEEE Conference on Decision and Control*, pages 31–36. IEEE, 2013.

[220] Daniele Astolfi and Corrado Possieri. Design of local observers for autonomous nonlinear systems not in observability canonical form. *Automatica*, 103:443–449, 2019.

[221] Salim Ibrir. LPV approach to continuous and discrete nonlinear observer design. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8206–8211. IEEE, 2009.

[222] Arthur Gelb et al. *Applied optimal estimation*. MIT press, 1974.

[223] J Boyle and J Wen. Newton descent observer for nonlinear discrete-time systems. Technical report, KAPL (Knolls Atomic Power Laboratory (KAPL), Niskayuna, NY), 2005.

[224] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[225] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[226] Zhengjie Yang, Wei Bao, Dong Yuan, Nguyen H Tran, and Albert Y Zomaya. Federated learning with nesterov accelerated gradient. *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[227] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

[228] Hassan K Khalil. *Nonlinear systems; 3rd ed.* Prentice-Hall, Upper Saddle River, NJ, 2002.

[229] Rajendra Bhatia. Linear algebra to quantum cohomology: the story of Alfred Horn's inequalities. *The American Mathematical Monthly*, 108(4):289–318, 2001.

[230] Alexander A. Klyachko. Random walks on symmetric spaces and inequalities for matrix spectra. *Linear Algebra and its Applications*, 319(1):37–59, 2000.