Coding Theory and Randomized Sketching for Distributed Optimization

by

Neophytos Charalambides

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Alfred Hero, Co-Chair
Assistant Professor Mert Pilanci; Stanford University, Co-Chair
Associate Professor Mahdi Cheraghchi
Associate Professor Hessam Mahdavifar

Neophytos Charalambides
neochara@umich.edu
ORCID iD: 0000-0002-8528-1467

ACKNOWLEDGEMENTS

It is not easy to express my gratitude towards Professors Hero and Pilanci, since the margin on this page is 'too narrow to contain it', as a famous 17^{th} century mathematician once said.[1] I am also grateful to my committee members, Professors Cheraghchi and Mahdavifar for allowing me to work with them, and taking the time to discuss various problems with me.

I am immensely thankful to my family, specially my parents, grandmother, siblings, Nico and Roger. Additionally, I was fortunate to have interacted with remarkable instructors, mentors, friends, peers and colleagues at Michigan and London; and across the globe, all far too numerous to list. I am also thankful to all the podcast hosts, whom I spent more hours than I should have listening to during lockdowns (... and afterwards). Last and most importantly, I am grateful to Jesus Christ for leading me back to Michigan, and guiding me constantly even when I am not aware nor appreciative of it. As another well-known mathematician famously exclaimed thirty years ago, 'I think I'll stop here'.

[1] This was shamelessly taken from another fantastic mentor I am indebted to [285].

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

ABSTRACT

With the advent of massive datasets, distributed techniques for processing information and carrying out computations are expected to enable exceptional possibilities for engineering, data intensive sciences, and better decision making. Unfortunately, distributive computation networks are prone to straggling servers. In recent years, utilizing coding-theoretic techniques has proven to be a very powerful tool for recovering either exact or approximate computations in the presence of stragglers. Furthermore, existing algorithms for mathematical programming, which are the core component of techniques for processing large datasets, often prove ineffective for scaling to the extent of all available data. Over the past two decades, randomized dimensionality reduction has proven to be a promising tool for approximate computations over large datasets.

These two approaches to dealing with large datasets are referred to as "Coded Computing", and "Randomized Numerical Linear Algebra" or "Sketching". In this dissertation, we introduce new methods for both these areas, extend existing methods, combine techniques from the two areas, and show how sketching algorithms can be utilized to devise coded computing schemes. Additionally, in certain schemes, we provide security through sketching and polynomial codes. The applications we focus on are distributed steepest descent and stochastic steepest descent, matrix-matrix multiplication, graph sparsification, linear regression, matrix inversion and pseudo-inversion, which are vital components to optimization, machine learning, and arguably all engineering disciplines.

CHAPTER I

Introduction

Due to the rapid growth of information sources, today's computing devices face unprecedented volumes of data, and there is an inherent need for overcoming the obstacle of computing over these datasets. In fact, it is estimated that data generation grows four times faster than the world's economy. This is one of the main obstacles which many machine learning algorithms encounter today, and has been coined as "the curse of dimensionality". Two ways of addressing this issue is through *coded computing* and *randomized sketching*. In this dissertation, we study both of these approaches, in combination and separately.

Before elaborating on these two disparate approaches, we recall the pioneering works which motivated the development of *coding and information theory*, and *randomized algorithms*. Source and channel coding are the two main areas of study in *information theory*; which dates back to the works of Claude Shannon [259] and Richard Hamming [139]. Ever since, the ideas that have been developed through the study of codes and information have been used for space exploration, secure and robust communications, data storage and compression, more recently in machine learning, as well as a multitude of other important applications which have made our daily lives significantly easier and more pleasant.

A closely related subject which we also study in this dissertation, is that of *cryptography*, which dates back to Julius Caesar; who used a shift cipher in order to communicate with his generals. In our context, we use techniques and ideas from cryptography in order to securely communicate information through insecure channels, and prevent eavesdroppers and curious workers from recovering the original data. Prior to the 20^{th} century, cryptography was mainly concerned with lexicographic and linguistic patterns, while now it makes extensive use of mathematical tools, including number theory and abstract algebra [27, 82, 102, 247, 258], information theory and statistics [260], and computational complexity [123]. More recently, with the potential

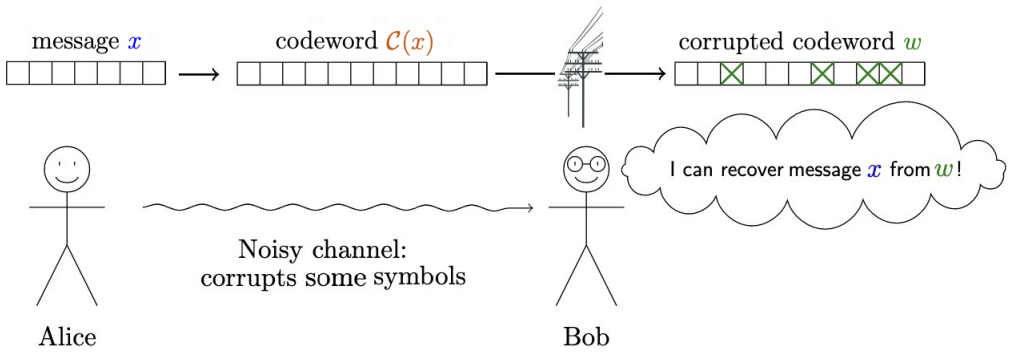


Figure I.1: Idea behind an error-correcting code. Alice delivers an encoded message $\mathcal{C}(x)$ which is transmitted to Bob through a noisy channel. Bob then decodes the corrupted codeword w, to recover the original message x.

advent of quantum computers, researchers have been primarily focusing on lattice-based [9, 229, 244] and isogeny-based [78] schemes for post-quantum cryptography.

Randomization has been used to prove difficult theorems in number theory and combinatorics, *e.g.* the "probabilistic method" introduced by Paul Erdős [103, 104], in addition to constituting the principal arguments underlying Shannon's theory of communication [259]. Randomness has also played a vital role in theoretical computer science, as the simple idea of 'randomly guessing' is surprisingly fruitful in solving otherwise intractable problems. Randomness essentially ensures that something is true about the optimal solution, without knowing it. Some applications benefiting from the use of randomness and randomized algorithms are: primality testing [209, 234], root-finding and factorization [24, 235], cryptography [61, 123], coding theory [194, 264], quantum computing [293], computational geometry and k-means clustering [35, 45, 67], approximation algorithms [121], as well as complexity theory [12, 13, 83, 111]. For further details and a more extensive list, please refer to [208, 210, 218].

1.1 Coded Computing

In recent years, coded computing was introduced in the seminal work of [178]. Though a very simple idea which considered matrix-vector multiplication and data shuffling, it initiated great interest in the information theory community; as well as in many other fields. Coded computing deals with computations over a network comprised of a central server and a certain number of computational worker nodes or servers, who communicate certain subtasks of the desired computation. Such computation networks face what is known as the 'straggler bottleneck', where a delay

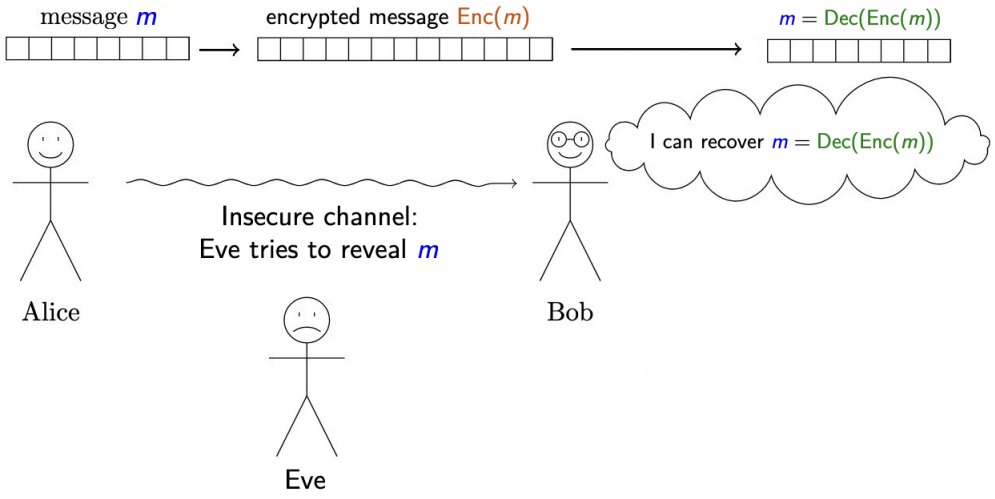


Figure I.2: Idea behind a cryptographic protocol. Alice delivers an encrypted message $\mathsf{Enc}(m)$ which is transmitted to Bob through an insecure channel. Then, an eavesdropper named Eve attempts to intercept the communication and reveal Alice's encrypted message. Fortunately, Eve is not able to do so; as she does not have knowledge of the corresponding decryption function $\mathsf{Dec}(\cdot)$. On the other hand, Bob can recover the original message m, by performing the decryption: $m = \mathsf{Dec}(\mathsf{Enc}(m))$.

in the computation is caused by slow or failing compute nodes. These nodes are referred to as *stragglers*, which in coded computing are associated with erasures or non-responsive workers. The objective of coded computing is to apply an encoding to the data or the computations, so that once a certain fraction of workers deliver their assigned task, the central sever can apply a corresponding decoding step in order to recover the computation; or a high-quality approximation.

An important impediment that hinders deployment of coded computing schemes, is the 'security bottleneck'. This is the potential vulnerability or weak point in the system where security breaches or attacks are more likely to occur. While coded computing itself is primarily focused on improving computational efficiency and reliability, security concerns remain critical. The specific security bottleneck in coded computing can vary depending on the implementation and context. To address these security bottlenecks, it is crucial to follow best practices in secure coding and employ strong encryption techniques. Additionally, ongoing research and developments in secure coded computing systems are necessary to address emerging security challenges. A closely related area is that of multi-party computation [22, 60]. Coded computing and multi-party computation serve different purposes within the domain of *secure* distributed computing, and they differ in their goals and approaches. Coded com-

puting focuses on improving efficiency and reliability, while multi-party computation is primarily concerned with privacy-preserving computations among multiple parties. However, both techniques can be combined to achieve secure and efficient distributed computations. Below, we provide a brief overview of coded computing. For additional information, the reader can refer to [181, 216, 299] and the references therein.

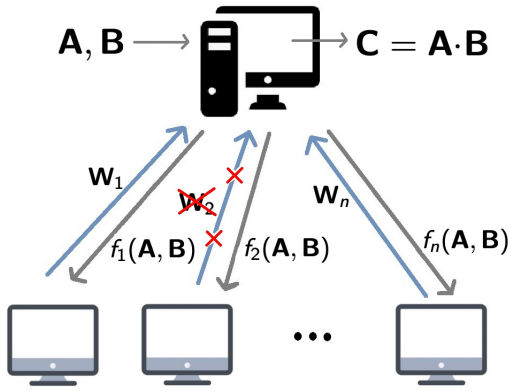


Figure I.3: Schematic of a matrix-matrix multiplication coded computing scheme.

The main computations for which coded computing schemes have been developed are polynomial evaluation, matrix-vector and matrix-matrix multiplication, and gradient computation; referred to as 'gradient coding'. In Figure I.3 we illustrate the steps behind a matrix-matrix multiplication coded computing scheme. The central server has two large matrices $\mathbf{A}$ and $\mathbf{B}$, and its objective is to distributively compute their product $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$. In this example, each of the n worker nodes receive an encoding of $(\mathbf{A}, \mathbf{B})$ through the respective encoding function $f_i(\cdot)$ for $i \in \{1, 2, \ldots, n\}$. The workers then carry out a computation on the encoding they receive, and communicate their computation $\mathbf{W}_i$ back to the central server. As long as the central server receives a fixed fraction of $\{\mathbf{W}_i\}_{i=1}^n$, he or she can recover the product $\mathbf{C}$ after applying a decoding step. In this example, the second worker is regarded as a straggler.

1.2 Randomized Numerical Linear Algebra

A popular way of speeding up computations relating to linear algebra is by first compressing the matrices, and this is the core idea of Randomized Numerical Linear Algebra ; abbreviated as 'RandNLA'. This is also referred to as 'randomized sketching' [238]. It is an interdisciplinary field of study that combines techniques from linear algebra, probability, and random sampling to develop efficient and scalable algorithms

for solving large-scale problems in numerical linear algebra.

Solving linear algebra problems, such as matrix factorization, eigenvalue computation, or solving linear systems, can be computationally expensive. Sketching offers a cheaper randomized approach to tackling these problems, by utilizing randomization and sampling techniques to approximate the solutions with high accuracy; while significantly reducing the computational cost. The core idea behind sketching is to leverage randomization to create structured and well-conditioned matrices that preserve important properties of the original matrices. By applying randomized algorithms to these structured matrices, one can obtain approximate solutions that are statistically close to the true solutions of the original problem. Randomized sketching has found applications in various domains, including machine learning, data analysis, optimization, graph theory, signal processing, and scientific computing [92, 198, 201, 202, 213, 294]. It has proven to be particularly useful in scenarios where traditional algorithms struggle to handle large-scale data, or where computational resources are limited. By exploiting the power of randomization, RandNLA provides a valuable toolbox for solving complex numerical problems efficiently, *e.g.* the libraries "RandBLAS" and "RandLAPACK" have recently been developed and tested in both MATLAB and Python [213].

Some of the initial ideas for RandNLA date back to the work of Johnson and Lindenstrauss [153], who showed that for a fixed set of m points $X \subseteq \mathbb{R}^n$, there exists a projection $f : \mathbb{R}^n \to \mathbb{R}^d$ of the points; for $d < n$, which does not distort the geometry of X, *i.e.* $\|f(x_i) - f(x_j)\|_2^2 \approx \|x_i - x_j\|_2^2$ for all $x_i, x_j \in X$. This result is referred to as the 'JL-lemma'; has found applications in a plethora of different fields over the past four decades, and gives rise to the notion of JL-transforms [1, 5, 7, 8, 75, 76, 110, 113, 157, 187, 188, 205, 225]; which have recently been shown to be optimal [174]. Over the past twenty-five years, this idea has been thoroughly studied through RandNLA in the context of specific problems; including matrix-matrix multiplication, sparsification of graphs, least-squares regression, Newton's method and many more. It is also worth noting that RandNLA is closely related to the area of 'compressed sensing', which considers recovery of information from a randomly compressed high dimensional vector. The objective of compressed sensing is to recover an input vector from corrupted random linear measurements [40, 41, 84]. Furthermore, the 'restricted isometry property' (RIP) is a fundamental concept in compressed sensing, which characterizes the behavior of a measurement or sensing matrix used in compressed sensing scenarios. In contrast to analogous properties encountered in RandNLA (*e.g.* the 'ℓ_2-subspace embedding property'), the RIP is not quantified probabilistically.

Matrices that satisfy the RIP often use random or structured constructions. These matrices play a crucial role in achieving the accuracy and efficiency of compressed sensing applications, and the RIP is a key criterion for their quality [30, 31]. For further information on RandNLA, and algorithms not covered in this dissertation, the reader can refer to [14, 28, 31, 80, 85, 93, 94, 115, 119, 124, 137, 199, 200, 201, 204, 213, 214, 290, 294], as well as [21, 65, 92, 143, 175, 180, 193, 198, 270, 271, 272, 273, 274, 275] for details regarding spectral sparsification of graphs and spectral graph theory.

In Figure I.4 we illustrate a compression of matrix $\mathbf{A} \in \mathbb{R}^{N \times d}$ using the sketching matrix $\mathbf{S} \in \mathbb{R}^{r \times N}$. This results in a compression of the $N \times d$ matrix $\mathbf{A}$ into a smaller matrix $\mathbf{SA}$ having only r rows. Here, we assume that $N \gg d$ and require $N > r > d$. In this example, $\mathbf{A}$ is partitioned into K smaller submatrices $\{\mathbf{A}_i\}_{i=1}^K$, and $\mathbf{S}$ is a sampling matrix (usually with replacement) that samples submatrices $\{\mathbf{A}_i\}_{i=1}^K$ of $\mathbf{A}$ according to a judiciously determined sampling distribution; often with a rescaling of the sampled submatrices. Explicit examples can be found in Appendix 2.2.

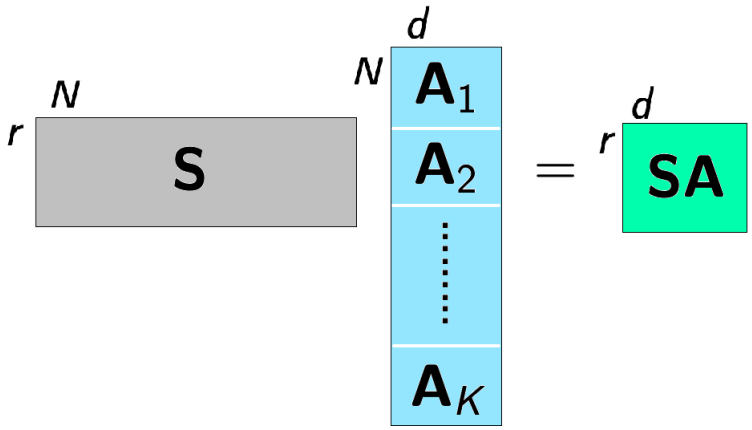


Figure I.4: Sketching of $\mathbf{A}$, so that the 'sketch' $\mathbf{SA}$ of $\mathbf{A}$ is used as a surrogate.

We give an illustration of the main idea behind randomized sketching algorithms in Figure I.5. Here, we are considering a set comprised of N points, and by randomly sampling according to some distribution; we retain r points from the initial dataset. Finally, the sampled points are rescaled according to the sampling distributions and r. This procedure reduces the number of points we are considering from N to r, while still preserving certain geometric properties of the dataset; with high probability. The dimension of the feature space d, is not affected.

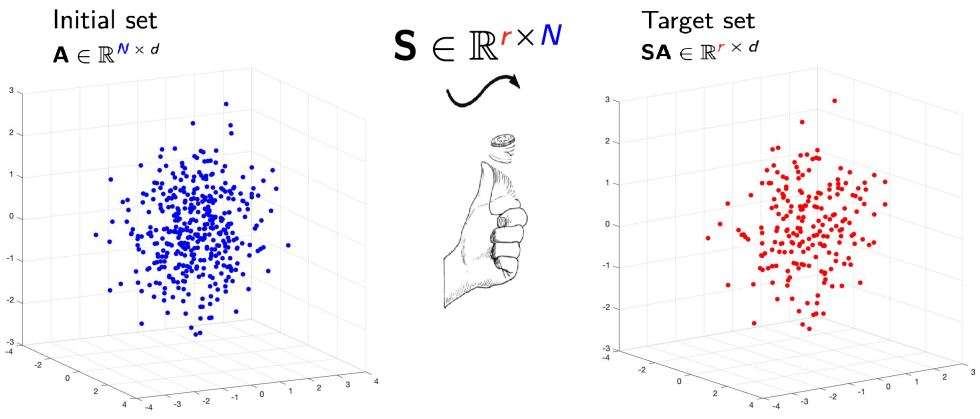


Figure I.5: Dimensionality reduction of a dataset, where $N = 200$ and $r = 100$.

1.3 Dissertation Layout

In this dissertation, we develop and study several techniques for both coded computing and randomized sketching. The main applications we consider are matrix multiplication, graph sparsification, ℓ_2-subspace embedding, steepest descent and the computation of gradients. We also study problems of encryption of information when communicating data in the coded computing paradigm.

Each chapter is self-contained and has its own introduction, so that readers do not need to go through the entire dissertation if they are only interested in one of the five main projects comprising it, and each chapter is accompanied with its own appendix. Worthwhile future work pertaining to the topic of each project, can be found at the end of its dedicated chapter. Below, we list the high level goals and contributions of each chapter. We also include work on graph sparsification by approximate matrix multiplication in Appendix VI.

In Chapters II, III, IV and V, we propose both exact and approximate coded computing schemes. Chapter II focuses on exact schemes through binary repetition codes, for gradient coding and coded matrix multiplication; and Chapter V proposes an approach for recovery of approximate matrix inverses and pseudoinverses. Though the latter proposes approximate schemes, the approximations are a result of the computations and not the coding schemes. In [51], we also showed how one of the matrix-matrix multiplication schemes of Chapter II, can be altered for recovery of accurate matrix inverse and pseudoinverse approximations; by leveraging the approximation algorithms of Chapter V. Furthermore, in Chapter V, the main scheme we propose is secure against potential eavesdroppers.

Chapters III, IV and VI primarily focus on RandNLA, while in Chapters III and IV we propose approximate gradient coding schemes through *iterative block sketching* methods. The primary idea in these chapters is to sample blocks; or pairs of blocks of partitions of the matrices, as in Figure I.4.

In Chapter III we propose iterative sketching by sampling according to what we define as '*block leverage scores*', and in Chapter IV we sample uniformly at random once we apply a random orthonormal matrix to the data matrix $\mathbf{A}$, which "flattens" the block leverage scores of $\mathbf{A}$. Furthermore, we study the security guarantees accompanied by applying a random projection to the data. In Chapter VI we propose a sampling matrix-matrix multiplication algorithm, similar to the $CR-$MM [88, 89, 90], which samples according to a new notion of '*joint leverage scores*', and guarantees a spectral approximation of matrix-matrix multiplication. Lastly, in Chapter VI and Appendix E, we study how one can obtain Laplacian spectral sparsifiers through approximate matrix-matrix multiplication.

- **Chapter II — Generalized Fractional Repetition Codes for Binary Coded Computations**:

  This chapter addresses the gradient coding and coded matrix multiplication problems in distributed optimization and coded computing. We present a numerically stable binary coding method which overcomes the drawbacks of the *Fractional Repetition Coding* gradient coding method proposed by Tandon et al. [279], and can also be leveraged by coded computing networks whose servers are of heterogeneous nature. Specifically, we propose a construction for fractional repetition gradient coding; while ensuring that the generator matrix remains close to perfectly balanced for any set of coded parameters, as well as a low complexity decoding step. The proposed binary encoding avoids operations over the real and complex numbers which are inherently numerically unstable, thereby enabling numerically stable distributed encodings of the partial gradients. We then make connections between gradient coding and coded matrix multiplication. Specifically, we show that any gradient coding scheme can be extended to coded matrix multiplication. Furthermore, we show how the proposed binary gradient coding scheme can be used to construct two different coded matrix multiplication schemes, each achieving different trade-offs.

  - Part of this chapter was presented at the '*2020 IEEE International Symposium on Information Theory*' [47], and the entire chapter is under review for publication to the '*IEEE Transactions on Information Theory*' [48].

- **Chapter III — Gradient Coding through Iterative Block Leverage Score Sampling**:

  We generalize the leverage score sampling sketch for ℓ_2-subspace embeddings, to accommodate sampling subsets of the transformed data, so that the sketching approach is appropriate for distributed settings. This is then used to derive an approximate coded computing approach for first-order methods; known as gradient coding, to accelerate linear regression in the presence of failures in distributed computational networks, *i.e.* stragglers. We replicate the data across the distributed network, to attain the approximation guarantees through the induced sampling distribution. The significance and main contribution of this work, is that it unifies randomized numerical linear algebra with approximate coded computing, while attaining an induced ℓ_2-subspace embedding through uniform sampling. The transition to uniform sampling is done without applying a random projection, as in the case of the subsampled randomized Hadamard transform. Furthermore, by incorporating this technique to coded computing, our scheme is an iterative sketching approach to approximately solving linear regression. We also propose weighting when sketching takes place through sampling with replacement, for further compression.

  – Part of this chapter was presented at the '*2021 SIAM Conference on Applied Linear Algebra*', and the entire chapter is under review for publication to the '*IEEE Transactions on Information Theory*' [56].

- **Chapter IV — Iterative Sketching for Secure Coded Regression**:

  In this work, we propose a method for speeding up linear regression distributively, while ensuring security. We leverage randomized sketching techniques, and improve straggler resilience in asynchronous systems. Specifically, we apply a random orthonormal matrix and then subsample *blocks*, to simultaneously secure the information and reduce the dimension of the regression problem. In our setup, the transformation corresponds to an encoded encryption in an *approximate gradient coding scheme*, and the subsampling corresponds to the responses of the non-straggling workers; in a centralized coded computing network. This results in a distributive *iterative* sketching approach for an ℓ_2-subspace embedding, *i.e.* a new sketch is considered at each iteration. We focus on the special case of the *Subsampled Randomized Hadamard Transform*, which we generalize to block sampling; and discuss how it can be used to secure the data.

– Part of this chapter was presented at the '*2022 IEEE International Symposium on Information Theory*' [49], and we plan on submitting the work of this chapter for publication to the '*IEEE Journal on Selected Areas in Information Theory: Information-Theoretic Methods for Trustworthy and Reliable Machine Learning*' [50].

- **Chapter V — Securely Aggregated Coded Matrix Inversion**:

  Coded computing is a method for mitigating straggling workers in a centralized computing network, by using erasure-coding techniques. Federated learning is a decentralized model for training data distributed across client devices. In this work we propose approximating the inverse of an aggregated data matrix, where the data is generated by clients; similar to the federated learning paradigm, while also being resilient to stragglers. To do so, we propose a coded computing method based on gradient coding. We modify this method so that the coordinator does not access the local data at any point; while the clients access the aggregated matrix in order to complete their tasks. The network we consider is not centrally administrated, and the communications which take place are secure against potential eavesdroppers.

  – Part of this chapter was presented at the '*2023 IEEE Allerton Conference*' [54], and the entire chapter is under review for publication to the '*IEEE Journal on Selected Areas in Information Theory: Dimensions of Channel Coding*' [55].

- **Chapter VI — Approximate Matrix Multiplication by Joint Leverage Score Sampling**:

  A ubiquitous operation in computer science, data science and scientific computing, is matrix multiplication. However, it presents a major computational bottleneck when the matrix dimension is high, as can occur for large data size or feature dimension. A common approach in approximating the product, is to subsample pairs of row vectors from the two matrices, and sum the rank-1 outer-products of the sampled pairs. We propose a sampling distribution based on the leverage scores of the two matrices, and give a characterization of our approximation in terms of the Euclidean norm, analogous to that of a ℓ_2-subspace embedding. This in turn implies a spectral guarantee for $CR-$MM; a similar algorithm which samples according to the ℓ_2-norms of the matrices' row pairs.

We also draw connections to data-oblivious approximate matrix multiplication, and graph sparsification.

  - Part of this chapter was presented at the '*2022 SIAM Conference on Mathematics of Data Science*', and we plan on submitting the work of this chapter for publication to the '15^{th} *Innovations in Theoretical Computer Science*' conference.

- **Appendix E — Graph Sparsification by Approximate Matrix Multiplication**:

  Graphs arising in statistical problems, signal processing, large networks, combinatorial optimization, and data analysis are often dense, which causes both computational and storage bottlenecks. One way of *sparsifying* a *weighted* graph, while sharing the same vertices as the original graph but reducing the number of edges, is through *spectral sparsification*. We study this problem through the perspective of RandNLA. Specifically, we utilize randomized matrix multiplication to give a clean and simple analysis of how sampling according to edge weights gives a spectral approximation to graph Laplacians, without requiring spectral information. Through the $CR-$MM algorithm, we attain a simple and computationally efficient sparsifier whose resulting Laplacian estimate is unbiased and of minimum variance. Furthermore, we define a new notion of *additive spectral sparsifiers*, which has not been considered in the literature.

  - This appendix was presented at the '*2023 IEEE Statistical Signal Processing Workshop*' [46].

CHAPTER II

Generalized Fractional Repetition Codes for Binary Coded Computations

2.1 Introduction

The *curse of dimensionality* has been a major impediment to solving large-scale problems, which often require heavy computations. Recently, coding-theoretic ideas have been adopted in order to accommodate such computational tasks in a distributed fashion, under the assumption that *straggler* workers are present [178, 183, 184, 185, 203, 237, 245, 251, 267, 288, 291, 297, 302]. Stragglers are workers whose assigned tasks may never be completed, due to delay or outage, and can significantly increase the overall computation time. When the computational tasks are encoded, *e.g.* by using linear codes, the distributed computations can be protected against erasures.

In this chapter, we first focus on the problem of exact recovery of the gradient in a distributed computing setting. We adopt the framework of [279] where *Gradient Coding* (GC)[1] was proposed for recovery of the gradient when the objective loss function is differentiable and additively separable. Gradient recovery is accomplished by replicating the tasks in a certain way, so as to introduce the redundancy needed to combat the effect of stragglers. The problem of exact recovery of the gradient was studied in several prior works, *e.g.* [47, 134, 186, 223, 239, 279], while the numerical stability issue was studied in [298]. There are also several works involving GC for approximate recovery of the gradient [25, 49, 52, 58, 59, 62, 144, 155, 239, 289, 292]. Also, an idea similar to GC had appeared in [305], though not within a coding-theoretic setting.

We propose a scheme for GC that is numerically stable, which works in fixed point

[1]For brevity, we refer to a *gradient coding scheme* as GCS, a *coded matrix multiplication scheme* as CMMS; and CMMSs for plural.

precision. The proposed scheme avoids floating-point representations and operations, *e.g.* division or multiplication of real or complex numbers. Furthermore, the encoding matrix is binary, simplifying the encoding process. This scheme is also deterministic and does not require generating random numbers. The method is similar in spirit to the *fractional repetition scheme* introduced in [100, 279], where we also drop the strict assumption that $s+1$ divides n, where n is the number of workers and s is the number of stragglers that the scheme tolerates. The main advantage of encoding and decoding real-valued data using binary matrices is that it does not introduce further instability, possibly adding to the computational instability of the associated computation tasks. Such a binary approach was considered in [149] for matrix-vector multiplication. The fact that the encoding matrix is defined over $\{0, 1\}$ allows us to view the encoding as task assignments. This also leads to a more efficient online decoding, which avoids searching through a polynomially large table in the number of workers n, as in the original GCS proposed in [279]. Moreover, our encoding matrix can be understood as a generalization of the *Fractional Repetition Coding* (FRC) scheme from [279] for the case when $(s + 1) \nmid n$, and our decoding algorithm can be used in conjunction with the corresponding GCS, for a low complexity online decoding without constructing a large decoding matrix a priori.

Dropping the aforementioned assumption results in an unbalanced load assignment among the workers. Under the assumption that the workers are *homogeneous*, we allocate the partitions as uniform as possible. To avoid bias when considering workers of *heterogeneous* nature, *i.e.* of different computational power, the allocation of the computational tasks should be done in such a way that all workers have the same expected completion time; as the stragglers are assumed to be uniformly random. We provide an analysis which determines how to appropriately allocate the assignments, so that this objective is met. We note that similar ideas appear in [223, 279], in the context of *partial* and *non-persistent* stragglers; respectively.

The majority of coded computing has focused on fundamental algebraic operations, such as matrix multiplication and polynomial evaluation. Directly adopting these schemes in general optimization problems is often not possible, since the gradient computation may not have any algebraic structure, or can only be evaluated numerically [181]. In this chapter we study the other direction, *i.e.* how to devise *Coded Matrix Multiplication* (CMM) schemes from GC. The key idea is to leverage the additive structure underlying both problems. By a simple modification to the encoding and decoding steps of any exact GCS, we show that the GCS can be utilized for matrix multiplication. In a similar fashion, we can transform any GCS into

a distributive straggler robust addition scheme.

The proposed GC method can be adapted to compute matrix-matrix multiplication in the presence of stragglers; which has gained a lot of attention recently, as well as matrix inverse approximations [51, 54]. The first CMMS was proposed in [178]. Since then, a multitude of CMMSs have been proposed [73, 74, 98, 108, 109, 179, 236, 277, 301, 303, 304], with each of them being advantageous to others in certain aspects. There is also a considerable amount of work on matrix-vector multiplication in distributed computing [18, 97, 133, 149]. Furthermore, numerical stability for matrix multiplication has been studied in [73, 74, 108, 236, 277]. Approximate coded matrix-matrix and matrix-vector schemes have also been devised [53, 112, 151, 146, 250, 280].

We show that any GCS can be extended to a CMMS. The main idea is that the product of two matrices is equal to the sum of the outer-products of their columns and rows, respectively. This property has been utilized in the context of CMM [53, 98, 109] and, to our knowledge, we are the first to connect this to GC. We present two new CMMSs based on the proposed binary GC, each achieving different trade-offs. Since the proposed CMMSs are derived from a GCS, they have properties that differ from other CMM approaches, and do not satisfy the same bounds and thresholds. However, our proposed schemes achieve the optimal trade-off between task allocations and the number of stragglers of GC. They also preserve the desired properties possessed by binary GC methods. For example, there is no need for complex encoding and decoding procedures; and the proposed methods are numerically stable.

Our main contributions are the following:

- Introduction of a *binary* GCS — both in the encoding and decoding, that is resilient to stragglers;
- Elimination of the restrictive assumption $(s+1) \mid n$ in [279];
- We show that the proposed GCS achieves perfect gradient recovery;
- We derive the minimum maximum load over all workers of a binary GCS, for any pair of parameters (s, n);
- We show how the unbalanced assignment, which arises when $(s+1) \nmid n$, can be made optimal when the workers are homogeneous;
- As compared to the original binary scheme [279], we give a more efficient online decoding step;
- We determine the optimal task assignment for heterogeneous workers;
- We show how any GCS can be extended to a CMMS;
- We use our binary GCS to devise two binary CMMSs;

The rest of this chapter is organized as follows. In Section 2.2 we provide a review

of the straggler problem in GC [279]. In Section 2.3 binary GC is introduced, and we describe the conditions for which a close to balanced task allocation needs to meet; when $(s+1) \nmid n$. Our binary encoding and decoding procedures are discussed in Section 2.4. In 2.4.5 we establish the optimality of our GCS, and in 2.4.7 we consider scenarios with heterogeneous workers. The focus is then shifted towards CMM, and in Section 2.5 we show how any GCS can be utilized to devise a CMMS. Then, another CMMS derived from our GCS is presented in 2.5.3. In 2.5.4 we compare and contrast our two CMMSs, and discuss where they have been utilized in other coded computing applications. In Section 2.6 we compare our schemes to prior methods, and draw connections with other areas in information theory. Section 2.7 concludes the chapter.

We also provide appendices with further details on our algorithms, numerical examples, and experimental justification. In Appendix 1.6 we present various applications in which CMM can be utilized in gradient descent iterative algorithms; for Frobenius norm minimization problems. These demonstrate further connections between the CMM and GC problems..

2.2 Preliminaries

2.2.1 Straggler Problem

Consider a single central server that has at its disposal a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \subsetneq \mathbb{R}^p \times \mathbb{R}$ of N samples, where $\mathbf{x}_i$ represents the features and y_i denotes the label of the i^{th} sample. The central server distributes the dataset $\mathcal{D}$ among n workers to facilitate computing the solution of the problem

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \theta) + \mu R(\theta) \right\} \tag{2.1}$$

where $L(\mathcal{D}; \theta) = \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \theta)$ is the empirical loss; for $\ell(\mathbf{x}_i, y_i; \theta)$ a predetermined differentiable loss-function, and $\mu R(\theta)$ is a regularizer. A common approach to solving (2.1) is to employ gradient descent. Even if closed-form solutions exist for (2.1), gradient descent is advantageous for large N.

The central server is assumed to be capable of distributing the dataset appropriately, with a certain level of redundancy, in order to recover the gradient based on the full dataset $\mathcal{D}$. As a first step we partition $\mathcal{D}$ into k disjoint parts $\{\mathcal{D}_j\}_{j=1}^{k}$ each

of size N/k. The gradient is the quantity

$$g = \nabla_\theta L(\mathcal{D}; \theta) = \sum_{j=1}^{k} \nabla_\theta \ell(\mathcal{D}_j; \theta) = \sum_{j=1}^{k} g_j \ . \tag{2.2}$$

We refer to the terms $g_j \coloneqq \nabla_\theta \ell(\mathcal{D}_j; \theta)$ as *partial gradients*. If k does not divide N, one can add auxiliary samples $(\mathbf{0}_{p\times 1}, 0)$ to the dataset $\mathcal{D}$ so that this is met. Since these auxiliary samples would not affect the partial gradients, nor the gradient, without loss of generality, we assume that $k|N$.

In a distributed computing setting each worker node completes its task by returning a certain encoding of its assigned partial gradients. There can be different types of failures that may occur during the computation or the communication process. The worker nodes that fail to complete their tasks and return the outcome to the central server are called *stragglers*. It is assumed that there are s stragglers, thus, the central server only receives $f = n - s$ completed tasks. Let $\mathcal{I} \subsetneq \mathbb{N}_n \coloneqq \{1, \cdots, n\}$ denote the set of indices of the f workers who complete and return their tasks. In practice, the completed tasks may be received at different times. Once *any* set of f tasks is received, the central server should be able to decode the received encoded partial gradients and recover the full gradient g.

2.2.2 Gradient Coding

Gradient coding, proposed in [279], is a procedure comprised of an encoding matrix $\mathbf{B} \in \Sigma^{n\times k}$, and a decoding vector $\mathbf{a}_\mathcal{I} \in \Sigma^n$; determined by $\mathcal{I}$, for Σ the field over which the encoding-decoding takes place. It is commonly assumed that the workers have the same computational power, in which case the same number of tasks is assigned to each of them. We relax this restriction in this chapter, and thus do not need to impose the assumption $(s+1) \mid n$ from [279]. Each row of $\mathbf{B}$ corresponds to an encoding vector, also regarded as a task allocation, and each column corresponds to a data partition $\mathcal{D}_j$.

Each worker node is assigned a number of partial gradients from the partition $\{\mathcal{D}_j\}_{j=1}^k$, indexed by $\mathcal{J}_i \subsetneq \mathbb{N}_k$. The workers are tasked to compute an encoded version of the partial gradients $g_j \in \mathbb{R}^p$ corresponding to their assignments. Let

$$\mathbf{g} \coloneqq \begin{pmatrix} | & | & & | \\ g_1 & g_2 & \dots & g_k \\ | & | & & | \end{pmatrix}^T \in \mathbb{R}^{k\times p} \tag{2.3}$$

denote the matrix whose rows constitute the transposes of the partial gradients. The received encoded partial gradients will be the rows of $\mathbf{Bg} \in \mathbb{R}^{n\times p}$ indexed by $\mathcal{I}$.

The full gradient of the objective (2.1) on $\mathcal{D}$ can be recovered by applying $\mathbf{a}_\mathcal{I}$ which is designed to have a support that is a subset of $\mathcal{I}$

$$g^T = \mathbf{a}_\mathcal{I}^T(\mathbf{Bg}) = \mathbf{1}_{1\times k}\mathbf{g} = \sum_{j=1}^{k} g_j^T, \tag{2.4}$$

provided that the encoding matrix $\mathbf{B}$ satisfies

$$\mathbf{a}_\mathcal{I}^T\mathbf{B} = \mathbf{1}_{1\times k} \tag{2.5}$$

for all $\binom{n}{s}$ possible index sets $\mathcal{I}$. Note that in perfectly balanced schemes, every partition is sent to $s+1$ servers, and each server will receive at least $\frac{k}{n}(s+1)$ distinct partitions. In Section 2.3, we propose a binary design of the encoding matrix $\mathbf{B}$ and decoding vector $\mathbf{a}_\mathcal{I}$. These may then be used for recovering the gradient g at each iteration by the central server.

In [279], a *balanced* assignment is considered, which is the case where all the workers are assigned the same number of tasks. This number is lower bounded by $\frac{k}{n}(s+1)$, *i.e.*

$$\|\mathbf{B}_{(i)}\|_0 \geq \frac{k}{n}(s+1) \quad \text{for all } i \in \mathbb{N}_n \tag{2.6}$$

for $\mathbf{B}_{(i)}$ the i^{th} row of $\mathbf{B}$. When this lower bound is met with equality, the scheme is *maximum distance separable* (MDS). The restriction $(s+1) \mid n$ allows the GC to satisfy this bound, as $\frac{n}{s+1}$ needs to be an integer. In Theorem 2.3.1, we give an analogous lower bound for when $(s+1) \nmid n$.

In general, GC schemes require processing over $s+1$ partitions from each worker in order to tolerate s stragglers. Since these schemes encode over partial gradients computed from unprocessed data, they are applicable to a large class of loss functions, as well as loss functions whose gradient can only be computed numerically, *e.g.* in deep neural networks [181].

We point out that in [182] the partitions sent to each worker are pre-processed, such that the computations at the workers are viewed as evaluating a polynomial at distinct points. This approach is referred to as *Polynomially coded regression*, and only applies to the least squares objective function. The central server computes the gradient by interpolating this polynomial. By working on the encoded data instead, the authors of [182] reduce the threshold on the number of workers that need to

respond.

2.2.3 Notational Conventions

Let $\mathbf{v} \in \mathbb{R}^p$ and $\mathbf{V} \in \mathbb{R}^{p \times q}$ respectively denote an arbitrary vector and matrix of the specified dimensions. The support of $\mathbf{v}$; *i.e.* the index set of elements which are nonzero, is denoted by $\text{supp}(\mathbf{v})$. The number of nonzero elements in $\mathbf{v}$ is denoted by $\text{nnzr}(\mathbf{v})$; *i.e.* $\text{nnzr}(\mathbf{v}) = |\text{supp}(\mathbf{v})|$. We define $\text{nnzr}(\mathbf{V})$ analogously. The row-span of a $\mathbf{V}$ is denoted by $\text{span}(\mathbf{V})$.

The vector Euclidean norm of $\mathbf{v}$ is defined as $\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}^T\mathbf{v}} = \left(\sum_i \mathbf{v}_i^2\right)^{1/2}$, the L^0 norm of $\mathbf{v}$ as $\|\mathbf{v}\|_0 = \text{nnzr}(\mathbf{v})$, and the matrix Frobenius norm of $\mathbf{V}$ as $\|\mathbf{V}\|_F = \sqrt{\text{tr}(\mathbf{V}^T\mathbf{V})} = \left(\sum_i \sum_j \mathbf{V}_{ij}^2\right)^{1/2}$. Also, $\mathbf{V}_{(i)}$ denotes the i^{th} row of $\mathbf{V}$, $\mathbf{V}^{(j)}$ denotes the j^{th} column of $\mathbf{V}$, and the $p \times p$ identity matrix is denoted by $\mathbf{I}_p$. For a row index set $I \subseteq \{1, 2, \ldots, p\}$ of $\mathbf{V}$, the submatrix comprised of the rows indexed by I is $\mathbf{V}_I \in \mathbb{R}^{|I| \times q}$. We denote the set of nonnegative integers by $\mathbb{N}_0 \coloneqq \{0, 1, 2, \ldots\}$, the positive integers up to n by $\mathbb{N}_n \coloneqq \{1, \ldots, n\}$; the positive integers up to n and including 0 by $\mathbb{N}_{0,n} \coloneqq \{0, 1, \ldots, n\}$, and the collection of size q subsets of $\mathbb{N}_n$ by $\mathcal{I}_q^n$; *i.e.* $|\mathcal{I}_q^n| = \binom{n}{q}$. Disjoint unions are represented by $\bigsqcup$; *e.g.* $\mathbb{Z} = \{j : j \text{ is odd}\} \bigsqcup \{j : j \text{ is even}\}$. By $\mathbf{e}_j$ we denote the j^{th} standard basis vector of $\mathbb{R}^n$. The remainder function is denoted by $\text{rem}(\cdot, \cdot)$, *i.e.* for positive integers a and b; $\text{rem}(b, a) = b - a \cdot \lfloor \frac{b}{a} \rfloor$.

In the context of GC, the parameter N is reserved for the number of data samples, each consisting of p features and one label. In the context of CMM, the parameter N denotes the common dimension of the two matrices being multiplied, *i.e.* the number of columns of the first matrix and the number of rows of the second matrix. The integer k denotes the number of partitions of the dataset in GC, and of the matrix or matrices in CMM. With n the number of workers and s the number of stragglers, the number of "blocks" is determined by $\ell = \lfloor \frac{n}{s+1} \rfloor$. The set of indices of the $f = n - s$ non-straggling workers is denoted by $\mathcal{I}$, and is an element of $\mathcal{I}_f^n$. Our encoding matrix is denoted by $\mathbf{B}$, and our decoding vector for the case where a certain $\mathcal{I}$ occurs is denoted by $\mathbf{a}_\mathcal{I}$. In the CMM setting, these are denoted by $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{a}}_\mathcal{I}$, respectively.

2.3 Binary Gradient Coding

In this section we motivate our approach to binary GC. The main idea behind binary schemes is to ensure that the superposition of the corresponding encoding vectors of a certain subset of the workers with non-overlapping assigned tasks, results in the all ones vector. This gives us a simple condition for binary GC, which leads

to a special case of condition 2.5. Additionally, we derive a strict lower bound for the total computational load of any GCS, and formalize what we mean by "close to uniform/balanced" assignments. We incorporate these into an integer program, which we constructively solve through our encoding in Section 2.4. We also derive what the minimum maximum load over all workers of a binary GCS is, for any pair of parameters (s, n).

2.3.1 Binary GC Condition

For our encoding, we have the following simple strategy in order to meet condition (2.5), for any $\mathcal{I} \in \mathcal{I}_f^n$. We divide the workers into $s+1$ subsets, and arrange the data partitions among the workers of each subset in such a way, so that their allocated task of computing and encoding certain partial gradients (corresponding to the arrangement); in each subset partition the entire gradient without overlap within the partition of the workers. As soon as all workers from one of the $s+1$ worker subsets have responded, the gradient g is recoverable. In the worst case, we will have $n-s$ responses, as by the pigeonhole principle; at most s subsets will have exactly one straggler, and the remaining subset will have none. From this, in the case where the workers are partitioned into more than $s+1$ subsets, the scheme could tolerate more stragglers. By this, since we are considering a fixed s, we divide the workers into exactly $s+1$ subsets. Furthermore, the arrangement of data partitions which takes place within each subset of workers, does not matter, as long as every data block/partial gradient is assigned to exactly one worker, *i.e.* there is no overlap. In what follows, we do not consider the degenerate case where a worker is not assigned any partition, *i.e.* $|\operatorname{supp}(\mathbf{B}_{(i)})| \geq 1$ for all $i \in \mathbb{N}_n$. This idea is summarized in Proposition 2.3.1 and Corollary 2.3.1.

Proposition 2.3.1. *Let $\mathbf{B} \in \{0,1\}^{n \times k}$, and partition its rows into $s+1$ nonempty subsets with index sets $\{\mathcal{K}_i\}_{i=0}^s$; i.e. $\bigsqcup_{i=0}^s \mathcal{K}_i = \mathbb{N}_n$. If for all $i \in \mathbb{N}_{0,s}$:*

$$\sum_{j \in \mathcal{K}_i} \mathbf{B}_{(j)} = \mathbf{1}_{1 \times k}, \tag{2.7}$$

then, for any $\mathcal{I} \in \mathcal{I}_f^n$, it follows that $\mathbf{1}_{1\times k} \in \operatorname{span}(\mathbf{B}_{\mathcal{I}})$.

Proof. Fix an arbitrary $\mathcal{I} \subsetneq \mathbb{N}_n$ of size $f = n - s$. By the pigeonhole principle, since only s indices from $\mathbb{N}_n$ are not included in $\mathcal{I}$, we know that at least one of $\{\mathcal{K}_i\}_{i=0}^s$ is contained in $\mathcal{I}$; say $\mathcal{K}_l$. This implies that $\mathbf{B}_{\mathcal{K}_l} \in \{0,1\}^{|\mathcal{K}_l| \times k}$ is a submatrix of

$\mathbf{B}_{\mathcal{I}} \in \{0,1\}^{f\times k}$. By condition (2.7), it follows that $\mathbf{1}_{1\times k} \in \text{span}(\mathbf{B}_{\mathcal{K}_l}) \subsetneq \text{span}(\mathbf{B}_{\mathcal{I}})$, which completes the proof. □

Corollary 2.3.1. *An equivalent formulation of (2.7), is to simultaneously satisfy:*

- $\text{supp}(\mathbf{B}_{(j)}) \bigcap \text{supp}(\mathbf{B}_{(l)}) = \emptyset$ *for all* $j, l \in \mathcal{K}_i$ *where* $j \neq l$
- $\bigcup_{\iota\in\mathcal{K}_i} \text{supp}(\mathbf{B}_{(\iota)}) = \mathbb{N}_k$.

Moreover, the corresponding decoding vector used to meet (2.5) when $\mathcal{K}_i \subsetneq \mathcal{I}$, *is* $\mathbf{a}_{\mathcal{I}} = \sum_{j\in\mathcal{K}_i} \mathbf{e}_j \in \{0,1\}^n$.

Proof. Assume we have a binary GCS with encoding matrix $\mathbf{B}$ and the index set of responsive workers $\mathcal{I}$, for which $\mathcal{K}_i \subsetneq \mathcal{I}$. For a contradiction, assume that for $\mathcal{K}_i$ we have $h \in \text{supp}(\mathbf{B}_{(j)}) \bigcap \text{supp}(\mathbf{B}_{(j)})$. It then follows that in the h^{th} entry of $\sum_{j\in\mathcal{K}_i} \mathbf{B}_{(j)}$, we have an integer greater than 1, which violates (2.7). Furthermore, under the assumption that $\bigcup_{\iota\in\mathcal{K}_i} \text{supp}(\mathbf{B}_{(\iota)}) = \mathbb{N}_k$, it is straightforward that (2.7) holds.

We now need to show that (2.7) implies the two conditions. Under the assumption that (2.7) is true, it follows for each $h \in \mathbb{N}_k$; there is one and only one $j \in \mathcal{K}_i$ for which the h^{th} entry of $\mathbf{B}_{(j)}$ is equal to one. If there were $H > 1$ many such j's or none, then the h^{th} entry of $\sum_{j\in\mathcal{K}_i} \mathbf{B}_{(j)}$ would be H or 0 respectively, contradicting (2.7).

Since we also require the decoding vector $\mathbf{a}_{\mathcal{I}}$ to be binary, we can either add (without rescaling) or ignore the computations of the workers within the given subgroup. Since the objective is to meet (2.5), by (2.7) we simply need to sample and add the corresponding encoding rows of $\mathcal{K}_i$. This is done by the decoding vector $\mathbf{a}_{\mathcal{I}} = \sum_{j\in\mathcal{K}_i} \mathbf{e}_j$. □

The following lemma is a direct generalization of (2.6) from [279, Theorem 1], which considers the balanced case. Our proposed scheme meets the lower bound with equality, which implies a minimized total computational load across the network, *i.e.* $\mathbf{B}$ is as sparse as possible for a GCS that is resilient to s stragglers. We attain a minimal total load balance, while ensuring that we are as balanced as possible when $(s+1) \nmid n$.

Lemma 2.3.1. *The total computational load of any GCS that is resilient to* s *stragglers; is at least* $k \cdot (s+1)$, *i.e.* $\text{nnzr}(\mathbf{B}) \geq k \cdot (s+1)$.

Proof. In order to tolerate s stragglers, each of $\{\mathcal{D}_j\}_{j=1}^k$ needs to be allocated to at least $s+1$ workers, thus $\|\mathbf{B}^{(i)}\|_0 \geq s+1$ for each $i \in \mathbb{N}_k$. Since we have k partitions, it follows that the total load is at least $k \cdot (s+1)$. □

2.3.2 Close to Uniform Assignment Distribution

A drawback of the GCS proposed in Section 2.3 is that the load assignments can have a wide range depending on how small r is compared to $s+1$. This is due to the lighter load assigned to the workers in the remainder block; which is of size r. The uneven workload is the cost we pay for dropping the assumption $(s+1) \mid n$, which does not often hold for a pair of two arbitrary positive integers (s, n)[2]. It is worth mentioning that for small s and a fixed n for which $(s+1) \nmid n$, when considering the original FRC scheme [279], one can easily modify s' and decrease n to n' in order to meet the divisibility condition $(s'+1) \mid n'$.

In order to have a close to close balanced GCS, we wish that the partitioning from Proposition 2.3.1 is done so that $\big||\mathcal{K}_j| - |\mathcal{K}_l|\big| \leq 1$; for all $j, l \in \mathbb{N}_{0,s}$, and $\big|\operatorname{supp}(\mathbf{B}_{(j')}) - \operatorname{supp}(\mathbf{B}_{(l')})\big| \leq 1$ for all $j', l' \in \mathcal{K}_\iota$; for each $\iota \in \mathbb{N}_{0,s}$. We note that in many applications, when a large k is considered, the difference of one between the load of the workers within the same subset $\mathcal{K}_\iota$, can be made insignificant. These integer load differences can be made relatively small, if the work is divided into many small units; *e.g.* by increasing k. In the context of computing gradients, this is often the case when it is taken over a large dataset.

In order to appropriately define a close to balanced assignment, in the case where $(s+1) \nmid n$, we use the following definition of $d_s(\mathbf{B})$; which gives a measure of how far from perfectly balanced the assignments of $\mathbf{B}$; in terms of the bound (2.6).

Definition 2.3.1. *Define* $d_s(\mathbf{B}) \coloneqq \sum_{i=1}^{n} \big|\|\mathbf{B}_{(i)}\|_0 - \frac{k}{n}(s+1)\big|$ *for* $\mathbf{B} \in \mathbb{Z}^{n\times k}$. *This function measures how far the task allocations* $\big\{\|\mathbf{B}_{(i)}\|_0\big\}_{i=1}^{n}$ *are from being* ***uniform****, i.e.* $\|\mathbf{B}_{(i)}\|_0 = \lfloor \frac{k}{n}(s+1) + \frac{1}{2} \rfloor$ *for all* $i \in \mathbb{N}_n$. *Furthermore,* $\{\|\mathbf{B}_{(i)}\|_0\}_{i=1}^{n}$ *is uniform; i.e. all elements are equal, if and only if* $d_s(\mathbf{B}) = 0$.

The objective of our proposed approached is to then give a binary solution to the

[2]For fixed n and random $s \in \{0, \cdots, n-1\}$; we have $(s+1) \mid n$ with probability $\frac{\sigma_0(n)-2}{n}$, where σ_0 is the divisor function of the 0^{th} power.

integer program

$$
\begin{aligned}
\text{(IP)} \qquad \arg\min_{\mathbf{B}\in\mathbb{Z}^{n\times k}} \quad & \big\{d_s(\mathbf{B})\big\} \\
\text{s.t.} \quad & \operatorname{nnzr}(\mathbf{B}) = k\cdot(s+1) \\
& \bigsqcup_{i=0}^{s}\mathcal{K}_i = \mathbb{N}_n \;:\; \big||\mathcal{K}_j| - |\mathcal{K}_l|\big| \le 1,\ \forall j,l\in\mathbb{N}_{0,s} \quad \cdot \\
& \big|\|\mathbf{B}_{(j)}\|_0 - \|\mathbf{B}_{(l)}\|_0\big| \le 1,\ \forall j,l\in\mathcal{K}_i,\ \forall i\in\mathbb{N}_{0,s} \\
& \exists \mathbf{a}_{\mathcal{I}} \text{ w/ } \operatorname{supp}(\mathbf{a}_{\mathcal{I}})\subseteq\mathcal{I};\ \forall\mathcal{I}\in\mathcal{I}_f^n \;:\; \mathbf{a}_{\mathcal{I}}^T\mathbf{B} = \mathbf{1}_{1\times k}
\end{aligned}
$$

whose solutions yield *almost* perfectly balanced GC schemes, with minimum total computational load across the network. The first constraint corresponds to the minimal total load of a GCS that is resilient to s stragglers; from Lemma 2.3.1. In order to have approximately equal cardinality across all partitions of the workers, we impose the second constraint. The third ensures that there is almost perfect balance among workers within the same partition $\mathcal{K}_i$; for each $i \in \mathbb{N}_{0,s}$. Together, the second and third constraints ensure that the maximum load across all workers is minimized, for the parameters n and s. The fourth constraint imposes condition (2.5), to guarantee that $\mathbf{B}$ is a generator matrix of a valid GCS.

2.3.3 Minimum Maximum Load of Workers in a Binary GCS

Next, we show what the minimum maximum load over all workers of a binary GCS is, in order to construct a valid GCS for any pair of parameters (s, n). Let $n = \ell\cdot(s+1)+r$ with $\ell = \lfloor\frac{n}{s+1}\rfloor$. Note that $r \equiv n \bmod (s+1)$. Similarly, let $r = t\cdot\ell+q$; which specifies the Euclidean division of r by l. Therefore, $n = \ell\cdot(s+t+1)+q$. In a particular case, we will also need the parameters specified by the division of n by $(\ell+1)$. Let $n = \lambda\cdot(\ell+1)+\tilde{r}$ (if $\ell = s-r$, then $\lambda = s$). For clarity, we assume that $n = k$, though it is straightforward to adapt our proposed approach to cases with $n \neq k$. To meet this assumption we can incorporate instances of the data point $(\mathbf{0}_{p\times 1}, 0)$ until N' points total are considered; such that $n|N'$, and let $k = n$. To summarize, we have

$$n = \ell\cdot(s+1)+r \qquad 0 \le r < s+1 \tag{2.8}$$

$$r = t\cdot\ell+q \qquad 0 \le q < \ell \tag{2.9}$$

$$n = \lambda\cdot(\ell+1)+\tilde{r} \qquad 0 \le \tilde{r} < \ell+1 \tag{2.10}$$

where all terms are nonnegative integers.

We have already established that in order to tolerate s stragglers; and not more than s in the worst case, the workers need to be partitioned into $s+1$ subsets. In order for each of the subsets to have approximately the same size, for the partitioning $\bigsqcup_{i=0}^{s} \mathcal{K}_i = \mathbb{N}_n$, we assign each $\mathcal{K}_i$ either $\lfloor \frac{n}{s+1} \rfloor$ or $\lceil \frac{n}{s+1} \rceil$ workers, *i.e.* $|\mathcal{K}_i| = \ell$ or $|\mathcal{K}_i| = \ell + 1$ for all $i \in \mathbb{N}_{0,s}$. Note that on average, in partitions comprised of fewer workers, we need to allocate more data partitions to each worker; in order to compensate for the fact that fewer workers are needed to collectively compute all partial gradients $\{g_j\}_{j=1}^{k}$. Consequently, we do not want to assign less that ℓ workers to any partition $\mathcal{K}_i$. This observation corresponds to the second constraint of (IP).

For what will follow, we set the partitions as

$$\mathcal{K}_i = \left\{ i + z \cdot (s+1) : z \in \mathbb{Z}_+ \right\} \bigcap \mathbb{N}_{0,n-1} \tag{2.11}$$

for each $i \in \mathbb{N}_{0,s}$, hence $|\mathcal{K}_\iota| = \ell + 1$ for $\iota \in \{0, 1, \ldots, r-1\}$ and $|\mathcal{K}_j| = \ell$ for $j \in \{r, \ldots, s\}$.[3] By what was discussed above, on average; the workers corresponding to the subsets $\{\mathcal{K}_j\}_{j=r}^{s}$ will be allocated greater loads. This setup leads to the following theorem.

Theorem 2.3.1. *Considering any binary GCS of n workers which tolerates s stragglers, the minimum maximum load of any worker is $s+t+2$. Specifically, given an encoding matrix $\mathbf{B} \in \{0,1\}^{n \times k}$, there always exists an $i \in \mathbb{N}_n$ for which $\|\mathbf{B}_{(i)}\|_0 \geq s+t+1$ when $q = 0$; and $\|\mathbf{B}_{(i)}\|_0 \geq s+t+2$ when $q > 0$.*

Proof. Consider a partition $\mathcal{K}_l$, for which $L = |\mathcal{K}_l|$ such that $L \geq \ell + 1$. By Corollary 2.3.1, it follows that $\sum_{l' \in \mathcal{K}_l} \|\mathbf{B}_{(l')}\|_0 = k$, and the allocation within the subset of workers indexed by $\mathcal{K}_l$ can be done so that each row indexed by $l' \in \mathcal{K}_l$ has support of size $\lfloor \frac{k}{L} \rfloor$ or $\lceil \frac{k}{L} \rceil = \lfloor \frac{k}{L} \rfloor + 1$, for which

$$\left\lfloor \frac{k}{L} \right\rfloor \leq \left\lceil \frac{k}{L} \right\rceil \leq \left\lceil \frac{k}{\ell+1} \right\rceil = \left\lceil \frac{n}{\ell+1} \right\rceil \leq \left\lceil \frac{n}{\ell} \right\rceil \leq s+2 \tag{2.12}$$

i.e. $\|\mathbf{B}_{(l')}\|_0 \leq s+t+2$ for all $l' \in \mathcal{K}_l$.

This shows that it suffices to consider the worker partitions of smaller sizes — subsets $\{\mathcal{K}_j\}_{j=r}^{s}$; which have cardinality $|\mathcal{K}_j| = \ell < \ell + 1$. Without loss of generality, we consider the partitioning (2.11). For this partitioning, the n^{th} worker is now assigned the index 0, *i.e.* $\mathbf{B}_{(0)}$ corresponds to $\mathbf{B}_{(n)}$.

[3] These are precisely the *congruence classes* we will be referring to later.

For a contradiction, assume that $\|\mathbf{B}_{(j')}\|_0 \lneq s+t+2$ for all $j' \in \mathcal{K}_j$, in the case where $q > 0$. It then follows that

$$\sum_{i' \in \mathcal{K}_j} \|\mathbf{B}_{(j')}\|_0 \leq \ell \cdot (s+t+1) = \ell \cdot (s+1) + \ell \cdot t \lneq \ell \cdot (s+1) + \ell \cdot t + q = k \quad (2.13)$$

where the last equality follows from (2.8), (2.9) and $n = k$. Since $q > 0$, we conclude that $\sum_{i' \in \mathcal{K}_j} \|\mathbf{B}_{(j')}\|_0 < k$, which contradicts Corollary 2.3.1. Hence, there is at least one $j' \in \mathcal{K}_j$ for which $\|\mathbf{B}_{(j')}\|_0 \geq s+t+2$.

Similarly, assume that $\|\mathbf{B}_{(j')}\|_0 \lneq s+t+1$ for all $j' \in \mathcal{K}_j$, in the case where $q = 0$ and $\ell > 0$. It then follows that

$$\sum_{i' \in \mathcal{K}_j} \|\mathbf{B}_{(j')}\|_0 \leq \ell \cdot (s+t) \lneq \ell \cdot (s+t) + \ell = \ell \cdot (s+1) + \ell \cdot t = k \quad (2.14)$$

which contradicts Corollary 2.3.1. This completes the proof. □

It is worth noting that in the case where $(s+1) \mid n$; we have $r = t = q = 0$, and Theorem 2.3.1 reduces to bound (2.6) derived in [279] for perfectly balanced schemes.

2.4 Proposed Binary Gradient Coding Scheme

In this section, we present our proposed binary coding scheme, along with its main properties. First, we present the construction of the encoding matrix $\mathbf{B}$; in Subsections 2.4.1, 2.4.2 and 2.4.3. In Subsection 2.4.4 we present an efficient online construction of the corresponding decoding vector $\mathbf{a}_{\mathcal{I}}$. In Subsections 2.4.5 and 2.4.6 we show that our proposed $\mathbf{B}$ and certain variants of it are as close to being uniform as possible; according (IP), and that in the regime $n \geq s^2$ the gap from being perfectly balanced is negligible. Finally, in Subsection 2.4.7 we provide an analysis which determines how to appropriately allocate the assignments when the workers are heterogeneous, so that they have the same expected completion time — this relaxes the cloe so uniform assignment.

2.4.1 Encoding Matrix

The idea is to work with congruence classes $\bmod(s+1)$ on the set of the workers' indices $\mathbb{N}_n$, in such a way that the workers composing a congruence class are roughly assigned the same number of partitions (differing by no more than one), while all partitions appear exactly once in each class. By *congruence class* we simply mean

the set of integers $j \in \mathbb{N}_n$ which are equivalent mod $(s+1)$. The classes are denoted by $\{[i]_{s+1}\}_{i=0}^{s}$. One could use a random assignment once it is decided how many partitions are allocated to each worker. However, in order to get a deterministic encoding matrix, we assign the partitions in "blocks", *i.e.* submatrices consisting of only 1's. With this setup, condition (2.7) becomes

$$\sum_{j \equiv c \bmod (s+1)} \mathbf{B}_{(j)} = \mathbf{1}_{1\times k} \tag{2.15}$$

for each $c \in \mathbb{N}_{0,s}$. As in the proof of Theorem 2.3.1, we reassign the index of the n^{th} worker; so that $\mathbf{B}_{(0)}$ corresponds to $\mathbf{B}_{(n)}$.

In the proposed GCS, the encoding is identical for the classes $\mathfrak{C}_1 \coloneqq \{[i]_{s+1}\}_{i=0}^{r-1}$, and is also identical for the classes $\mathfrak{C}_2 \coloneqq \{[i]_{s+1}\}_{i=r}^{s}$. The objective is to design $\mathbf{B}$ to be as close to a block diagonal matrix as possible, and we do so by ensuring that the difference in the load assignments between any two servers within the same set of classes $\mathfrak{C}_1$ or $\mathfrak{C}_2$; is at most one. We refer to the l disjoint sets of consecutive $s+1$ rows of $\mathbf{B}$ as *blocks*, and the submatrix comprised of the last r rows as the *remainder block*. Note that in total we have $\ell+1$ blocks, including the remainder block, and that each of the first ℓ blocks have workers with indices forming a complete residue system. We will present the task assignments for $\mathfrak{C}_1$ and $\mathfrak{C}_2$ separately. A numerical example where $n = k = 11$ and $s = 3$, is presented in Appendix 1.3.

2.4.2 Repetition Assignment for Classes 0 to $r-1$

In our construction each of the first r residue classes also have an assigned row in the remainder block, such that we could assign r partitions to the last worker of each class in $\mathfrak{C}_1$, and evenly assign $s+1$ to all other workers corresponding to $\mathfrak{C}_1$. Our objective though is to distribute the k tasks among the workers corresponding to the $\ell+1$ blocks as evenly as possible, for the congruence classes corresponding to $\mathfrak{C}_1$, in such a way that *homogeneous* workers have similar loads. By homogeneous we mean the workers have the same computational power, which implies that they exhibit independent and identically distributed statistics for the computation time of similar tasks.

Note that $n = (\ell+1)\cdot s + (\ell + r - s)$. Hence, when $\ell > s - r$, we can assign $s+1$ tasks to each worker in the first $\ell + r - s$ blocks, and s tasks to the workers in the remaining $s+1-r$ blocks. In the case where $\ell \leq s - r$, we assign $\lambda + 1$ tasks to the first $\tilde{r}$ blocks and λ tasks to the remaining $\ell + 1 - \tilde{r}$ blocks. It is worth pointing out

that $\lambda = s$ and $\tilde{r} = 0$ when $\ell = s - r$, which means that every worker corresponding to $\mathfrak{C}_1$ is assigned $\lambda = s$ tasks, as $n = (\ell + 1) \cdot s$.

A pseudocode for this encoding process is presented in Algorithm 10, in Appendix 1.1. For coherence, we index the rows by i starting from 0, and the columns by j starting from 1. We point out that when $\ell > s-r$, we have $\lambda = s$ and $\tilde{r} = \ell+r-s > 0$. In the case where $\ell \leq s - r$, we need to invoke (2.10) which was introduced solely for this purpose, as we need the remainder $\tilde{r}$ to be nonnegative. It follows that Algorithm 10 can be reduced to only include the **else if** statement; eliminating the conditional clause.

2.4.3 Repetition Assignment for Classes r to s

For the workers corresponding to $\mathfrak{C}_2$, we first check if $q = 0$. If this is the case, the n partitions are evenly distributed between these workers, *i.e.* each worker is assigned $(s + t + 1)$ partitions; as $n = \ell \cdot (s + t + 1)$ and here we are only considering ℓ blocks. When $0 < q < r$, we assign $(s + t + 2)$ tasks to each worker of $\mathfrak{C}_2$ in the first q blocks, and $(s + t + 1)$ to the workers in the remaining $\ell - q$ blocks. A pseudocode for the encoding process is provided in Algorithm 11, in Appendix 1.1.

The final step is to *combine* the encodings of the classes $\mathfrak{C}_1$ and $\mathfrak{C}_2$ to get $\mathbf{B}$. That is, we combine the outcomes $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$ of Algorithm 10 and $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$ of Algorithm 11. One could merge the two algorithms into one or run them separately, to get $\mathbf{B} = \tilde{\mathbf{B}}_{\mathfrak{C}_1} + \tilde{\mathbf{B}}_{\mathfrak{C}_2}$.

The encoding matrix $\mathbf{B}$ is also the adjacency matrix of a bipartite graph $G = (\mathcal{L}, \mathcal{R}, \mathcal{E})$, where the vertices $\mathcal{L}$ and $\mathcal{R}$ correspond to the n workers and the k partitions, respectively. We can also vary the number of stragglers s the scheme can tolerate for a fixed n, by trading the sparsity of $\mathbf{B}$. In other words, if $\mathbf{B}$ is designed to tolerate more stragglers, then more overall partial gradients need to be computed. This results in more computations over the network, as $|\operatorname{supp}(\mathbf{B})| = k \cdot (s + 1)$.

2.4.4 Decoding Vector

A drawback of the binary GCS introduced in [279] is that a matrix inversion is required to compute $\mathbf{A} \in \mathbb{R}^{\binom{n}{f} \times n}$; which contains the decoding vectors corresponding to all possible index sets of responsive workers $\mathcal{I} \in \mathcal{I}_f^n$. This matrix needs to be stored and searched through at each iteration of the gradient descent procedure. Searching through $\mathbf{A}$ to find the corresponding $\mathbf{a}_{\mathcal{I}}$ is prohibitive; as it has $\Theta(n^s)$ rows to look through. We propose a more efficient online decoding algorithm in order to mitigate this problem.

The construction of the decoding matrix in [279] when using regular matrix multiplication and inversion, requires $O\left(k^3(k+2n-2s)\right)$ operations to construction the pseudoinverse of a submatrix of $\mathbf{B}$, for each of the possible $\binom{n}{s}$ index sets $\mathcal{I}$. Furthermore, at each iteration of gradient descent, the decoding step requires a search through the rows of $\mathbf{A}$. This comes at a high cost when compared to our online algorithm, which constructs a decoding vector in $O(n+s)$ operations, and does not require any additional storage space.

We point out that a similar decoding approach was developed independently in [58], which focuses on approximating the gradient rather than recovering the exact gradient. The main idea behind the two approaches is that we look at the index set of responsive workers, and then only consider the response of workers with mutually exclusive assigned partitions. While the objective of the scheme in [58] is to form a vector which is close to $\mathbf{1}_{1\times k}$, we guarantee that this vector is attained once $n-s$ workers respond. By (2.4), we can therefore recover the exact gradient.

In the proposed binary GCS there is no rescaling of the partial gradients taking place by encoding through $\mathbf{B}$, as the coefficients are 1 or 0. As a result, the proposed decoding reduces to simply summing a certain subset of the completed encoded tasks, while making sure that each partial gradient is added exactly once. To this end, among any f workers who send back their computed sum of partial gradients, we need to have ℓ workers, $\ell = \frac{n}{s+1} \in \mathbb{Z}_+$ (or $\ell+1$ where $\ell = \lfloor \frac{n}{s+1} \rfloor$, if $(s+1) \nmid n$), who have no common assigned partitions. We elaborate on this in the next paragraph.

If $r=0$, the decoder traverses through the $s+1$ classes consecutively to find one which is a complete residue system (Algorithm 1). This will be used to determine the decoding vector $\mathbf{a}_{\mathcal{I}}$, implied by Corollary 2.3.1. When $r>0$, the decoder first traverses through the last $s+1-r$ congruence classes; checking only the first ℓ blocks. If it cannot find a complete residue system corresponding to returned tasks by non-stragglers, it proceeds to the first r classes; checking also the remainder block. This extra step makes the scheme more efficient. In both cases, by the pigeonhole principle we are guaranteed to have a complete residue system, provided that f completed tasks are received.

The next step is to devise a decoding vector for each of the $\binom{n}{s}$ different straggler scenarios $\mathcal{I}$. We associate the i^{th} complete residue class with a decoding vector $\mathbf{a}_i$ defined as

$$\mathbf{a}_i \coloneqq \sum_{j\in[i]_\ell} \mathbf{e}_j \ \in \{0,1\}^n, \tag{2.16}$$

for $i \in \mathbb{N}_{0,\ell-1}$. Also, note that $\|\mathbf{a}_i\|_0 = \ell+1$ for the decoding vectors corresponding

to the first r classes, and $\|\mathbf{a}_i\|_0 = \ell$ for the remaining classes. In both cases, $\mathbf{a}_{i+1}$ is a cyclic shift of $\mathbf{a}_i$.

At each iteration the gradient is computed once f worker tasks are received. Define the *received indicator-vectors*

$$(\mathrm{rec}_{\mathcal{I}})_i = \begin{cases} 1 & \text{if } i \in \mathcal{I} \\ 0 & \text{if } i \notin \mathcal{I} \end{cases}, \tag{2.17}$$

for each possible $\mathcal{I}$, where $\|\mathrm{rec}_{\mathcal{I}}\|_0 = f$ and $n - \|\mathrm{rec}_{\mathcal{I}}\|_0 = s$. Thus, there is at least one $i \in \mathbb{N}_{0,\ell-1}$ for which $\mathrm{supp}(\mathbf{a}_i) \subsetneq \mathrm{supp}(\mathrm{rec}_{\mathcal{I}})$. If there are multiple $\mathbf{a}_i$'s satisfying this property, any of them can be selected. The pseudocode is presented in Algorithm 1.

Algorithm 1: Decoding Vector $\mathbf{a}_{\mathcal{I}}$

```
Input: received indicator-vector rec_I
Output: decoding vector a_I
for i = s to 0 do
    if (rec_I)_i ≡ 1 then
        l ← i
        if supp(a_l) ⊆ supp(rec_I) then
            a ← a_l                          ▷ a_l is defined in (2.16)
            break
        end
    end
end
return a_I ← a
```

2.4.5 Validity and Optimality of our GCS

Now that we have presented our construction, we provide the accompanying guarantees in terms of validity, and optimality, which motivated our construction.

Theorem 2.4.1. *The proposed encoding-decoding pair* $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$ *satisfy condition* (2.5)*, for any* $\mathcal{I} \in \mathcal{I}_f^n$*. That is,* $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$ *comprise a valid GCS which tolerates up to* s *stragglers.*

Proof. By our construction of $\mathbf{B}$, the rows corresponding to a congruence class are mutually exclusive and their superposition is precisely $\mathbf{1}_{1\times k}$.

By the pigeonhole principle the number of completed encoded tasks that is required at the decoder to guarantee a successful recovery of the gradient, denoted by

ν, is equal to

$$\begin{aligned}\nu &\coloneqq \ell \cdot r + (\ell - 1) \cdot \big[(s+1) - r\big] + 1 \\ &= \ell \cdot (s+1) - s + r = \big[\ell \cdot (s+1) + r\big] - s = n - s \ . \end{aligned} \tag{2.18}$$

Therefore, as long as $\nu = n - s$ many workers respond, there is at least one subset of them whose indices form a complete residue system $\mathrm{mod}(s+1)$. Algorithm 1 determines such an index subset, and constructs a binary vector whose support corresponds to this subset. As a result, $\mathbf{a}_{\mathcal{I}}^T \mathbf{B} = \mathbf{1}_{1\times k}$ for any $\mathcal{I}$ of size $n-s$, and (2.4) is satisfied. □

Note that the total number of task assignments is $k\cdot(s+1)$, for any pair of integers (s, n) where $0 \leq s < n$, as expected. This is the same total load required by the MDS based schemes. Also, our GCS meets the lower bound on total task assignments of $\mathbf{B}$ implied by (2.6) and Lemma 2.3.1

$$\mathrm{nnzr}(\mathbf{B}) = \sum_{i=1}^{n} \|\mathbf{B}_{(i)}\|_0 \geq n \cdot \frac{k}{n}(s+1) = k \cdot (s+1) \tag{2.19}$$

with equality.

It can be observed that the runtime complexity of Algorithm 1 is $O((\ell+1)\cdot(s+1)) = O(n+s)$. This complexity can be slightly reduced by the following modification. The for-loop in Algorithm 1 can be stopped early by only traversing through the classes $0, \cdots, s-1$, and assigning $\mathbf{a}_{\mathcal{I}} \leftarrow \mathbf{a}_s$ if none was selected. This reduces the runtime complexity to $O((\ell+1)\cdot s)$, hence our proposed decoder is significantly faster than the decoding algorithm of [279]. The decoding matrix $\mathbf{A}$ of [279] requires $\binom{n}{s}$ applications of a pseudoinverse for its construction, which makes it impractical for large $\binom{n}{s}$. Once this has been constructed, at each gradient descent iteration it requires an additional decoding step of time complexity $O(f^3)$.

An alternative decoding is to consider a decoding of each of $\mathfrak{C}_1$ and $\mathfrak{C}_2$ separately; in a streaming fashion, and terminate whenever one of the two is completed. This decoding procedure will be especially useful in our first CMMS. Both the CMMS and the alternative decoding will be described in more detail in 2.5.2.

Theorem 2.4.2. *The task allocation through* $\mathbf{B}$ *resulting from Algorithms 10 and 11 is a binary solution to the integer program* (IP).

Theorem 2.4.2 holds for permutations of the columns of $\mathbf{B}$, or a random assignment of tasks per class; as opposed to repeating blocks — as long as all partitions are

present only once in a single worker of each congruence class. The decoding in either of these cases remains the same. Furthermore, the proposed $\mathbf{B}$ can be viewed as an extension of the *cyclic repetition* scheme introduced in [279]. An example of how the allocations can be modified for each congruence class is given in Appendix 1.3. By "valid permutation per congruence class", we mean that a separate permutation is applied to the columns of $\mathbf{B}\big|_{[c]_{s+1}}$; the restriction of $\mathbf{B}$ to the rows corresponding to $[c]_{s+1}$, for each congruence class $c \in \mathbb{N}_{0,s}$.

For the purpose of the applications considered in this chapter, permutations of the rows or columns of $\mathbf{B}$ do not affect the overall performance or guarantees of the proposed GCS; *i.e.* any permutation applied to the encoding matrix of the approach of Algorithms 10 and 11 would have the same result.[4] A permutation of the rows corresponds to permutation of the workers' indices, and a permutation of the columns simply means the workers are assigned different partitions, but since the same number of partitions is allocated to each worker, their total workload remains the same.

Theorem 2.4.3. *A binary encoding matrix $\bar{\mathbf{B}}$ is a valid permutation of the task allocations per congruence class of the encoding matrix $\mathbf{B}$ proposed by Algorithms 10 and 11, for which Algorithm 1 produces a correct decoding vector, i.e. $\mathbf{a}_{\mathcal{I}}^T \bar{\mathbf{B}} = \mathbf{1}_{1\times k}$ for all possible $\mathcal{I} \in \mathcal{I}_f^n$, if:*

1. $\|\bar{\mathbf{B}}_{(i)}\|_0 = \|\mathbf{B}_{(i)}\|_0$
2. $\operatorname{supp}(\bar{\mathbf{B}}_{(i)}) \bigcap \operatorname{supp}(\bar{\mathbf{B}}_{(j)}) = \emptyset \quad$ *if* $\quad i \equiv j \bmod (s+1)$

for $i, j \in \mathbb{N}_n$ distinct. These conditions also imply that $\|\bar{\mathbf{B}}^{(i)}\|_0 = s+1$ for all $i \in \mathbb{N}_k$.

2.4.6 Distribution of Assignments for $n \geq s^2$

Considering the identities (2.8), (2.9) and (2.10), note that for $\ell > r$ we have $t = 0$ and $r = q$. Furthermore, when $\ell = s$ we have $n = s \cdot (s+1) + r \approx s^2$, and in the regime $n \geq s^2$, we can show that t is at most 1. Then, the gap between the heaviest and lightest loads; respectively $s + t + 2$ and s, is at most 3.

Lemma 2.4.1. *Let $n = s^2 + a$ for $a \in \mathbb{N}_0$ and $s < n$. Then, we have $t = 1$ only when $a = s-2, s-1$ or $2s$. Otherwise, $t = 0$.*

Proof. We break up the proof into three cases:

[4] In the case where a permutation is applied to the *rows* of $\mathbf{B}$, the decoding vectors defined in (2.16) should be modified accordingly.

Case $a \in \{0, \cdots, s-3\}$: For $\alpha = s - a \in \{3, 4, \cdots, s\}$:

$$n = s \cdot (s+1) - \alpha = \overbrace{(s-1)}^{\ell} \cdot (s+1) + \overbrace{(s+1-\alpha)}^{r}, \tag{2.20}$$

and $\ell > r$ for any α. Thus, $t = 0$ and $r = q$.

Case $a \in \{s, \cdots, 2s-1\}$: Let $n = s^2 + a = s^2 + (s + \beta)$ for $\beta \in \{0, \cdots, s-1\}$. Then $n = s \cdot (s+1) + \beta$ implies $\ell = s$ and $r = \beta$. Since $r < \ell$, it follows that $t = 0$ and $r = q$.

Case $a \gneq 2s$: The final case to consider is $a = 2s + \gamma$, for $\gamma \in \mathbb{Z}_+$. The resulting parameters are $r = q = \text{rem}\left(\text{rem}\left(\gamma, s+1\right) - 1, s+1\right)$, $\ell = (s^2 + 2s + \gamma - r)/(s+1)$ and $t = 0$.

When $\alpha = 1$ it follows that $r = s$ and $\ell = s - 1$, thus $t = 1$ and $q = 1$. For $\alpha = 2$ we get $r = \ell = s - 1$, hence $t = 1$ and $q = 0$. For both $\alpha = 1$ and $\alpha = 2$; $t = 1$ is a consequence of $r \geq \ell$. In addition, when $\beta = s$ we have $r = \ell = s$; thus $t = 1$ and $q = 0$. □

2.4.7 Task Allocation to Heterogeneous Workers

We now discuss how to allocate the partitions when the workers are of *heterogeneous* nature, such that all workers have the same expected execution time. This analysis may be needed in applications with very discrete, indivisible jobs. In cases where the work can be divided more finely, for any given subset of workers who are to share the total work, one can simply divide the work to be done among the workers in proportion to their computational strengths, to equalize the expected completion time. We present the case where we have two groups of machines, each consisting of the same type. The analysis for more than two groups of machines can be done in a similar fashion.

The two types of workers are denoted by $\mathcal{T}_i$; with a total of τ_i machines, and their expected execution for computing g_j (for equipotent $\mathcal{D}_j$'s) by

$$t_i \coloneqq \mathbb{E}\left[\text{time for } \mathcal{T}_i \text{ to compute } g_j\right], \tag{2.21}$$

for $i \in \{1, 2\}$, where $t_1 \lneq t_2$; *i.e.* machines $\mathcal{T}_1$ are faster. Let $|\mathcal{J}_{\mathcal{T}_i}|$ denote the number

of partitions each worker of $\mathcal{T}_i$ is assigned. The goal is to find $|\mathcal{J}_{\mathcal{T}_1}|$ and $|\mathcal{J}_{\mathcal{T}_2}|$ so that

$$\mathbb{E}\left[\mathcal{T}_1 \text{ compute their task}\right] = \mathbb{E}\left[\mathcal{T}_2 \text{ compute their task}\right], \tag{2.22}$$

implying $t_1 \cdot |\mathcal{J}_{\mathcal{T}_1}| = t_2 \cdot |\mathcal{J}_{\mathcal{T}_2}|$. Hence $|\mathcal{J}_{\mathcal{T}_1}| \gneq |\mathcal{J}_{\mathcal{T}_2}|$, as $t_1 \lneq t_2$. Let $\tau_1 = \frac{\alpha}{\beta} \cdot \tau_2$ for $\frac{\alpha}{\beta} \in \mathbb{Q}_+$ in reduced form. Since $\tau_1 + \tau_2 = n$, it follows that

$$\tau_1 = \frac{\alpha}{\alpha+\beta} n \qquad \text{and} \qquad \tau_2 = \frac{\beta}{\alpha}\tau_1 = \frac{\beta}{\alpha+\beta} n. \tag{2.23}$$

To simplify the presentation of the task assignments, we assume $(s+1) \mid n$. If $(s+1) \nmid n$, one can follow a similar approach to that presented in Subsection 2.4.1 to obtain a close to uniform task allocation; while *approximately* satisfying (2.22).

The main idea is to fully partition the data across the workers, such that each congruence class is comprised of roughly $\frac{\alpha}{\alpha+\beta} \cdot \frac{k}{s+1}$ workers of type $\mathcal{T}_1$, and $\frac{\beta}{\alpha+\beta} \cdot \frac{k}{s+1}$ workers of type $\mathcal{T}_2$. We want $\frac{\tau_1+\tau_2}{s+1} = \frac{n}{s+1}$ many workers for each congruence class, and

$$|\mathcal{J}_{\mathcal{T}_1}| \cdot \frac{\tau_1}{s+1} + |\mathcal{J}_{\mathcal{T}_2}| \cdot \frac{\tau_2}{s+1} = k \tag{2.24}$$

partitions to be assigned to each class. That is, the dataset $\mathcal{D}$ is completely distributed across each congruence class, and our GCS is designed accordingly.

Putting everything together, the following conditions determine $|\mathcal{J}_{\mathcal{T}_1}|$ and $|\mathcal{J}_{\mathcal{T}_2}|$

(i) $t_1 \cdot |\mathcal{J}_{\mathcal{T}_1}| = t_2 \cdot |\mathcal{J}_{\mathcal{T}_2}| \quad \iff \quad |\mathcal{J}_{\mathcal{T}_2}| = \frac{t_1}{t_2} \cdot |\mathcal{J}_{\mathcal{T}_2}|$

(ii) $|\mathcal{J}_{\mathcal{T}_1}| \cdot \tau_1 + |\mathcal{J}_{\mathcal{T}_2}| \cdot \tau_2 = (s+1) \cdot k$

(iii) $\tau_2 = \frac{\beta}{\alpha} \cdot \tau_1 \quad \iff \quad \tau_1 = \frac{\alpha}{\beta} \cdot \tau_2.$

By substituting (iii) into (ii) to solve for $|\mathcal{J}_{\mathcal{T}_2}|$, and then plugging it into (i) to solve for $|\mathcal{J}_{\mathcal{T}_1}|$, we get

$$|\mathcal{J}_{\mathcal{T}_1}| = (s+1) \cdot k \cdot \left(\frac{\alpha t_2}{\alpha t_2 + \beta t_1}\right) \cdot \frac{1}{\tau_1} \tag{2.25}$$

$$|\mathcal{J}_{\mathcal{T}_2}| = (s+1) \cdot k \cdot \left(\frac{\beta t_1}{\alpha t_2 + \beta t_1}\right) \cdot \frac{1}{\tau_2} \tag{2.26}$$

which we round to get appropriate numbers of assignments.

This framework may be generalized to any number of different groups of machines. Under the same assumptions, for $\mathcal{T}_1, \cdots, \mathcal{T}_m$ different groups with $t_i \lneq t_{i+1}$ for all $i \in \mathbb{N}_{m-1}$:

(i) $t_1 \cdot |\mathcal{J}_{\mathcal{T}_1}| = t_2 \cdot |\mathcal{J}_{\mathcal{T}_2}| = \cdots = t_m \cdot |\mathcal{J}_{\mathcal{T}_m}|$

(ii) $|\mathcal{J}_{\mathcal{T}_1}| \cdot \tau_1 + | + |\mathcal{J}_{\mathcal{T}_2}| \cdot \tau_2 + \cdots + |\mathcal{J}_{\mathcal{T}_m}| \cdot \tau_m = (s+1) \cdot k$

(iii) $\tau_1 = \frac{\alpha_2}{\beta_2} \cdot \tau_2 = \cdots = \frac{\alpha_m}{\beta_m} \cdot \tau_m$, for $\frac{\alpha_{i+1}}{\beta_{i+1}} \in \mathbb{Q}_+$

need to be met. This gives us a system of $2(m-1)+1 = 2m-1$ equations with m unknowns $\{|\mathcal{J}_{\mathcal{T}_j}|\}_{j=1}^m$, which is solvable.

2.5 Binary Coded Matrix Multiplication Schemes

Multiplication of two matrices is one of the most common operations. Coded matrix multiplication is a principled framework for providing redundancy in distributed networks, to guarantee recovery in the presence of stragglers [179]. As in GC, each worker is requested to carry out some computation and encode it; before sending it back to the central server. In this section we first show how *any* GCS can be used to devise a CMMS, and then present two different schemes based on our binary GCS. Like the fractional repetition scheme, our two schemes resemble replication codes. For simplicity in presentation, throughout this section we assume that $k|N$.

The two CMMSs have applications beyond matrix multiplication, which we discuss in Subsection 2.5.4. Regarding matrix multiplication, the schemes have different trade-offs in terms of communication, storage, and computational operations, required by each worker. Depending on the application and the resources available, one may even be easier to implement compared to the other.

As pointed out in [182], despite recent advancements in distributed gradient computations, schemes under parameters (s, n) have not been developed which have a *recovery threshold* (*i.e.* the worst case minimum number of workers that need to respond in order to recover the full gradient) less than $f = n - s$. On the other hand, many CMMSs exhibit considerably better recovery thresholds — the optimum asymptotic recovery threshold of $\mu\nu$ for $1/\mu$ and $1/\nu$ respectively the fraction of A and B stored by each worker; was achieved through Polynomial Codes [303].

Improving the recovery threshold comes at the cost of trading encoding and decoding complexities, storage, restrictions on how the matrices are partitioned, and numerical stability. The two schemes we propose have a recovery threshold of $f = n-s$, though do not suffer from any of the aforementioned drawbacks. For simplicity in presentation, we assume that $N = \ell \cdot k$ for $\ell \in \mathbb{Z}_+$ and N the effective dimension; which implies that we have a balanced assignment. When this is not the case, the analysis carried out in 2.4.1 can be applied.

2.5.1 CMM-1 — **Outer-Product Representation**

Consider a single central server node that has at its disposal the matrices $A \in \mathbb{R}^{L\times N}$ and $B \in \mathbb{R}^{N\times M}$, and it can distribute submatrices of A and B among n workers; to compute their product $C = AB$ in an accelerated manner. One way of computing C is to leverage the fact that

$$C = \sum_{i=1}^{N} A^{(i)} B_{(i)} \tag{2.27}$$

which has also been used in [109, 98]. Recall that $A^{(i)}$ denotes the i^{th} column of A, and $B_{(i)}$ the i^{th} row of B, as specified in Subsection 2.2.3. This makes the process parallelizable. To make use of this outer-product representation, we partition A and B each into k disjoint submatrices consisting of $\tau = N/k$ columns and rows respectively, which we denote by $\tilde{A}_j \in \mathbb{R}^{L\times\tau}$ and $\tilde{B}_j \in \mathbb{R}^{\tau\times M}$ for $j = 1, \cdots, k$. That is

$$A = \begin{bmatrix} \tilde{A}_1 & \cdots & \tilde{A}_k \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} \tilde{B}_1^T & \cdots & \tilde{B}_k^T \end{bmatrix}^T. \tag{2.28}$$

The central server is then capable of distributing the pairs $(\tilde{A}_j, \tilde{B}_j)$ appropriately, with a certain level of redundancy, in order to recover C

$$C = \sum_{j}^{k} \tilde{A}_j \tilde{B}_j \ . \tag{2.29}$$

Define $X_j \coloneqq \tilde{A}_j \tilde{B}_j \in \mathbb{R}^{L\times M}$ for all j, and the matrix

$$\mathbf{X} \coloneqq \begin{bmatrix} X_1^T \mid \cdots \mid X_k^T \end{bmatrix}^T \in \mathbb{R}^{kL\times M}, \tag{2.30}$$

similar to how $\mathbf{g}$ was defined (2.3) in Section 2.2. Recall that the main idea behind GC is to construct the pair of encoding matrix $\mathbf{B}$ and decoding vector $\mathbf{a}_{\mathcal{I}}$, such that $\mathbf{a}_{\mathcal{I}}^T \mathbf{B} = \mathbf{1}_{1\times k}$ for all $\binom{n}{s}$ possible index sets $\mathcal{I}$. A CMMS can be devised by the pair $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$. The matrix product $C = AB$ is described as:

$$C = (\overbrace{\tilde{\mathbf{a}}_{\mathcal{I}}^T}^{\tilde{\mathbf{a}}_{\mathcal{I}}^T} \otimes \mathbf{I}_L) \cdot ((\overbrace{\tilde{\mathbf{B}}}^{\tilde{\mathbf{B}}} \otimes \mathbf{I}_L) \cdot \mathbf{X}) = (\overbrace{\underbrace{\mathbf{1}_{1\times k}}_{\mathbf{a}_{\mathcal{I}}^T\mathbf{B}} \otimes \mathbf{I}_L}^{\tilde{\mathbf{a}}_{\mathcal{I}}^T\tilde{\mathbf{B}}}) \cdot \mathbf{X} = \sum_{j=1}^{k} X_j, \tag{2.31}$$

where $\tilde{\mathbf{B}} \in \mathbb{C}^{nL\times kL}$ is now the encoding matrix for the CMM, and $\tilde{\mathbf{a}}_{\mathcal{I}} \in \mathbb{C}^{nL\times L}$ is the *decoding matrix* corresponding to the non-straggler index set $\mathcal{I}$. Expression (2.31) is

analogous to (2.4).

Theorem 2.5.1. *Any GCS can be extended to a coded matrix multiplication or addition scheme.*

Proof. Consider a GCS $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$, for which $\mathbf{a}_{\mathcal{I}}^T \mathbf{B} = \mathbf{1}_{1\times k}$. By (2.31) it follows that

$$\tilde{\mathbf{a}}_{\mathcal{I}}^T(\tilde{\mathbf{B}}\mathbf{X}) = \sum_{j=1}^{k} X_j = C \ . \tag{2.32}$$

Therefore, a CMM method $(\tilde{\mathbf{B}}, \tilde{\mathbf{a}}_{\mathcal{I}})$ is obtained.

For matrix addition, we simply construct $\mathbf{X}$ by augmenting the k equi-sized matrices we want to add; instead of the products $\{\tilde{A}_j\tilde{B}_j\}_{j=1}^k$ in (2.30), and we obtain a coded matrix addition scheme. □

Let $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$ be the encoding-decoding GC pair from Section 2.3. In Theorem 2.5.1, the resulting pair $(\tilde{\mathbf{B}}, \tilde{\mathbf{a}}_{\mathcal{I}})$ is a CMMS whose encoding matrix $\tilde{\mathbf{B}} = \mathbf{B} \otimes \mathbf{I}_L$ represents the partition pairs $(\tilde{A}_j, \tilde{B}_j)$ as the columns of $\tilde{\mathbf{B}}$; and its rows represent the n workers. That is, the worker corresponding to the i^{th} row of $\mathbf{B}$ receives the partition pairs corresponding to $\mathcal{J}_i = \text{supp}(\mathbf{B}_{(i)})$, and is asked to send back the summation of the outer-products

$$C_j \coloneqq \sum_{j\in\mathcal{J}_i} \tilde{A}_j\tilde{B}_j = \sum_{j\in\mathcal{J}_i} X_j \ . \tag{2.33}$$

The decoding matrix $\tilde{\mathbf{a}}_{\mathcal{I}}^T = \mathbf{a}_{\mathcal{I}}^T \otimes \mathbf{I}_L$ only involves the computations of a complete residue system associated with the received workers, which are determined by $\text{supp}(\mathbf{a}_{\mathcal{I}})$.

The communication cost per worker which along with the storage required at the central server are the major drawbacks of this approach. Each worker will have to send back a matrix of size $L \times M$, and in the worst case, the central server will need to store $k \cdot (s+1)$ matrices of this size before it can recover C. The computation cost per worker is equivalent to that of multiplying two matrices, of dimensions corresponding to the block pairs. An alternative CMM decoding process overcomes the storage issue at the central server, which is described next.

2.5.2 Decoding as a Streaming Process

In the case where $(s+1)|n$, we can use a streaming process for the recovery of C. In this process, we only retain a single computation corresponding to each of the blocks of the encoding matrix $\mathbf{B}$; where $\mathbf{B}$ is now a block diagonal matrix with $\ell = \frac{n}{s+1}$

diagonal blocks of the form $\mathbf{1}_{(s+1)\times\lfloor k/\ell \rfloor}$ or $\mathbf{1}_{(s+1)\times\lceil k/\ell \rceil}$. The process terminates once a single worker from each block has responded. The pseudocode for this procedure is given in Algorithm 2.

Algorithm 2: Decoding in a Streaming Fashion

Input: computations C_j sequentially
Output: product C
Initialize: $C = \mathbf{0}_{L\times N}$, and $R = \emptyset$ the index set of the received workers' blocks
while $|R| < \ell$ **do**
  receive computation C_j ▷ $j \in \mathbb{N}_n$
  $\hat{\ell} \leftarrow \lceil j/(s+1) \rceil$ ▷ block index of the j^{th} worker
  if $\hat{\ell} \notin R$ **then**
    $C \leftarrow C + C_j$
    $R \leftarrow R \cup \{\hat{\ell}\}$
  end
end

The benefit of this approach, compared to the decoding $\tilde{\mathbf{a}}_{\mathcal{I}}^T = \mathbf{a}_{\mathcal{I}}^T \otimes \mathbf{I}_L$ for $\mathbf{a}_{\mathcal{I}}$ from Algorithm 1, is that the central server will never need to store more than double the entries of the product C. In the case where $(s+1) \nmid n$, we can do the exact same process by simply breaking the problem into two subroutines, one dealing with the workers whose indices correspond to the first r congruence classes $\mathrm{mod}(s+1)$, and the other with the workers corresponding to the remaining $s+1-r$ congruence classes. That is, we will work with $\mathbf{B}_{\mathfrak{C}_1}$ for $\ell+1$ blocks; and $\mathbf{B}_{\mathfrak{C}_2}$ for ℓ blocks separately. We carry out Algorithm 2 in parallel for the two cases, and terminate whenever one of the two has computed C. Now, the central server will need to store a total number of entries no more than twice the size of matrix C. This decoding procedure can be done analogously for GCS. An example with further details is provided in 1.3.1.

Algorithm 2 takes into account *which* workers have responded up to a certain instance, rather than only the *total number* of workers which have responded. The recovery threshold in the worst case is $n-s$, matching that of our previous decoding procedure. On average though, considering all possible index sets $\mathcal{I}$ of responsive workers which correspond to a valid decoding according to Algorithm 2, less workers than the worst case of $n-s$ need to respond.

If $(s+1) \nmid n$, the worst case occurs when all workers corresponding to ℓ blocks of $\mathbf{B}_{\mathfrak{C}_1}$ and $\ell-1$ blocks of $\mathbf{B}_{\mathfrak{C}_2}$ respond, along with only one worker from either of the two remaining blocks. By (2.18), the total number of responsive workers is $n-s$. In the best case, we need a single worker corresponding to each block of $\mathbf{B}_{\mathfrak{C}_2}$ to respond,

i.e. $\ell = \lfloor \frac{n}{s+1} \rfloor$ responsive workers. Similarly, if $(s+1) \mid n$, in the best case we require $\ell = \frac{n}{s+1}$ workers to respond, and in the worst case $n - (\ell - 1) \cdot (s+1) + 1 = n - s$ many workers.

2.5.3 CMM-2 — **Augmentation of Submatrices**

In a system where the main limitation is the communication load which can be handled from the workers to the central server; as well as storage of the computed task at the worker nodes, CMM-1 is not ideal, even with the more efficient decoding process. Next, we discuss an alternative CMMS which is superior in these aspects.

In contrast to the partitioning (2.28) of CMM-1, in this scheme we partition A along its rows and B along its columns, as was done for the Polynomial codes in [303], *i.e.*

$$A = \begin{bmatrix} \bar{A}_1^T & \cdots & \bar{A}_{k_1}^T \end{bmatrix}^T \quad \text{and} \quad B = \begin{bmatrix} \bar{B}_1 & \cdots & \bar{B}_{k_2} \end{bmatrix}, \tag{2.34}$$

where $\bar{A}_j \in \mathbb{R}^{\frac{L}{k_1} \times N}$ and $\bar{B}_j \in \mathbb{R}^{N \times \frac{M}{k_2}}$. Each worker computes the product of a submatrix of A with a submatrix of B, and then the central server augments the received computations accordingly.

For coherence, we let $k = k_1 k_2$ for $k_1, k_2 \in \mathbb{Z}_+$. To simplify the presentation of our scheme, we consider the case where $k_1 | L$, $k_2 | M$ and $(s+1) \mid k$; *i.e.* $\mathrm{S} = L/k_1 \in \mathbb{Z}_+$ and $\mathrm{T} = M/k_2 \in \mathbb{Z}_+$, and similar to our GCS that $n = k$. The product C of the two matrices under this partitioning is equal to

$$\begin{pmatrix} \boxed{\bar{A}_1\bar{B}_1} & \boxed{\bar{A}_1\bar{B}_2} & \cdots & \boxed{\bar{A}_1\bar{B}_{k_2-1}} & \boxed{\bar{A}_1\bar{B}_{k_2}} \\ \boxed{\bar{A}_2\bar{B}_1} & \ddots & & & \boxed{\bar{A}_2\bar{B}_{k_2}} \\ \vdots & & \ddots & & \vdots \\ \boxed{\bar{A}_{k_1-1}\bar{B}_1} & & & \ddots & \boxed{\bar{A}_{k_1-1}\bar{B}_{k_2}} \\ \boxed{\bar{A}_{k_1}\bar{B}_1} & \boxed{\bar{A}_{k_1}\bar{B}_2} & \cdots & \boxed{\bar{A}_{k_1}\bar{B}_{k_2-1}} & \boxed{\bar{A}_{k_1}\bar{B}_{k_2}} \end{pmatrix}$$

where we denote each product submatrix by $\bar{C}_{i,j} \coloneqq \bar{A}_i \bar{B}_j \in \mathbb{R}^{\mathrm{S} \times \mathrm{T}}$, and each block row column by

$$\bar{C}_i = \begin{bmatrix} \bar{C}_{i,1} & \cdots & \bar{C}_{i,k_2} \end{bmatrix} \in \mathbb{R}^{\mathrm{S} \times M}; \text{ for each } i \in \mathbb{N}_{k_1}. \tag{2.35}$$

The product submatrices can be ordered in terms of the indices i and j, *e.g.* through the bijection $\phi : \mathbb{N}_{k_1} \times \mathbb{N}_{k_2} \to \mathbb{N}_k$ defined as $\phi : (i,j) \mapsto (i-1)k_2 + j$. Since $k = k_1 k_2$, each $\bar{C}_{i,j}$ corresponds to one of k distinct subtasks which need to be retrieved.

As mentioned, the main benefit of CMM-2 when compared to CMM-1, is that the communication load between the i^{th} worker and the central server drops by a factor of $k/|\mathcal{J}_i|$; when considering equipotent partitions of A and of B. Therefore, if Algorithm 1 were to be used for the decoding step, the central server would also require much less temporary storage. The workers on the other hand, need to store the entire matrix A.

The idea behind both the decoding Algorithms 1 and 2 work, under a slight modification which we explain. In the proposed GCS we dealt with vector addition, and in our first CMMS; with matrix addition. Now, we focus on submatrices of the final product, which is common in CMM [178, 303]. Our decoding vector $\mathbf{a}_{\mathcal{I}}$ will be the same, but the way we apply it is different. If Algorithm 1 were to be used, every worker corresponding to the same congruence class $c \in \mathbb{N}_{0,s}$ communicates back the same set of computations $\{\bar{C}_{i,j}\}_{\phi(i,j)\in\mathcal{J}_{[c]_{s+1}}}$, which sets are distinct for each congruence class. Hence, whenever a complete residue system, in terms of the workers indices, is received, then the central server will have in its possession all the computations $\{\bar{C}_{i,j} : i \in \mathbb{N}_{k_1}, j \in \mathbb{N}_{k_2}\}$. These computations are then rearranged in order to recover C.

If Algorithm 2 were to be used, the same idea holds. The central server waits until at least one corresponding worker from each block, from one of the two matrices $\mathbf{B}_{\mathfrak{C}_1}$ or $\mathbf{B}_{\mathfrak{C}_2}$ has responded. Formally, we want a scheme $(\tilde{\mathbf{B}}, \tilde{\mathbf{a}}_{[I]_{s+1}})$ such that $\tilde{\mathbf{a}}^T_{[I]_{s+1}}\tilde{\mathbf{B}} = \mathbf{I}_{k_1 M}$ (note that $k\mathrm{T} = k_1 k_2 \cdot M/k_2 = k_1 M$) for any $\mathcal{I} \in \mathcal{I}^n_f$, where $[I]_{s+1}$ is the congruence class of the complete residue system present in $\mathcal{I}$. This is analogous to the GC condition $\mathbf{a}_{\mathcal{I}}\mathbf{B} = \mathbf{1}_{k\times 1}$. To summarize, the encoding process is

$$\overbrace{\left(\mathbf{I}_{k_1k_2} \otimes \mathbf{1}_{(s+1)\times 1} \otimes \mathbf{I}_{\mathrm{T}}\right)}^{\tilde{\mathbf{B}}\in\{0,1\}^{k\mathrm{T}(s+1)\times k\mathrm{T}}} \cdot \overbrace{\begin{bmatrix} \bar{C}^T_1 \\ \vdots \\ \bar{C}^T_{k_1} \end{bmatrix}}^{\bar{\mathbf{C}}^T\in\mathbb{R}^{k\mathrm{T}\times \mathrm{S}}} = \tilde{\mathbf{B}} \cdot \begin{bmatrix} \bar{C}^T_{1,1} \\ \bar{C}^T_{1,2} \\ \vdots \\ \bar{C}^T_{1,k_2} \\ \vdots \\ \bar{C}^T_{k_1,k_2} \end{bmatrix} = \begin{bmatrix} \bar{C}^T_{1,1} \\ \vdots \\ \bar{C}^T_{1,1} \\ \vdots \\ \bar{C}^T_{k_1,k_2} \\ \vdots \\ \bar{C}^T_{k_1,k_2} \end{bmatrix}, \tag{2.36}$$

where the transpose of each submatrix $\bar{C}_{i,j}$ appears $s+1$ times along the rows of the encoding $\tilde{\mathbf{B}}\bar{\mathbf{C}}^T \in \mathbb{R}^{(s+1)k_1M\times \mathrm{S}}$, each corresponding to one of the $s+1$ potential workers that are asked to compute $\bar{C}_{i,j}$. This encoding is analogous to the encoding $\mathbf{Bg}$ in our GCS, where the partial gradients $\{g_l\}_{l=1}^k$ correspond to the submatrices

$\{\bar{C}_{i,j} : i \in \mathbb{N}_{k_1}, j \in \mathbb{N}_{k_2}\}$. Furthermore, $\tilde{\mathbf{B}}$ reveals the task allocation which is applied to the pairs $\{(\bar{A}_i, \bar{B}_j) : i \in \mathbb{N}_{k_1}, j \in \mathbb{N}_{k_2}\}$. Alternatively, the matrix $\tilde{\mathbf{B}}$ is constructed as described in Algorithm 3.

Algorithm 3: Encoding Matrix $\tilde{\mathbf{B}}$ — CMM-2

Input: parameters k_1, k_2, s and $\mathrm{T} = M/k_2$ ▷ assume $n = k = k_1 k_2$
Output: $\tilde{\mathbf{B}} \in \{0,1\}^{k\mathrm{T}(s+1)\times k\mathrm{T}}$
Initialize: $\mathbf{B} = \mathbf{0}_{k(s+1)\times k}$
for $j = 1$ *to* k **do**
  $\mathbf{B}[(j-1)\cdot(s+1)+1 : j\cdot(s+1), j] = \mathbf{1}_{(s+1)\times 1}$
end
return $\tilde{\mathbf{B}} \leftarrow \mathbf{B} \otimes \mathbf{I}_{\mathrm{T}}$

For the decoding matrix $\tilde{\mathbf{a}}_{[I]_{s+1}}$ constructed by Algorithm 4, we use Algorithm 1 as a subroutine in order to determine which congruence class I of worker indices forms a complete residue system, in the inner **if** statement. We could directly use the indicator-vector $\mathrm{rec}_{\mathcal{I}}$, though working with $\mathbf{a}_{\mathcal{I}}$ is preferred. This also reveals how the decoding is similar to that of our GCS. By our assumptions, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{a}}_{[I]_{s+1}}$ are both of size $k\mathrm{T}(s+1) \times k\mathrm{T}$; where $k\mathrm{T} = k_1 M$. As was previously mentioned, a rearrangement on $\bar{\mathbf{C}}^T$ needs to take place to finally recover C. In the special case where $k_1 = 1$, a rearrangement is not necessary.

Algorithm 4: Decoding Matrix $\tilde{\mathbf{a}}_{[I]_{s+1}}$ — CMM-2

Input: decoding vector $\mathbf{a}_{\mathcal{I}}$, by invoking Algorithm 1, and design parameters k_1, k, s and $\mathrm{T} = M/k_1$
Output: $\tilde{\mathbf{a}}_{[I]_{s+1}} \in \{0,1\}^{k\mathrm{T}(s+1)\times k\mathrm{T}}$
Initialize: $\tilde{\mathbf{a}}_{\mathcal{I}} = \mathbf{0}_{k(s+1)\times k}$
for $j = 1$ *to* $s+1$ **do**
  if $(\mathbf{a}_{\mathcal{I}})_j \equiv 1$ **then**
    $I \leftarrow j-1$
  end
end
for $j = 1$ *to* k **do**
  $\tilde{\mathbf{a}}_{\mathcal{I}}[(j-1)\cdot(s+1)+(I+1), j] = 1$
end
return $\tilde{\mathbf{a}}_{[I]_{s+1}} \leftarrow \tilde{\mathbf{a}}_{\mathcal{I}} \otimes \mathbf{I}_{\mathrm{T}}$

Similar to GC where $\mathbf{a}_{\mathcal{I}}$ is constructed such that $\mathbf{a}_{\mathcal{I}}^T \mathbf{B} = \mathbf{1}_{1\times k}$, we constructed $\tilde{\mathbf{a}}_{[I]_{s+1}}$ to satisfy $\tilde{\mathbf{a}}_{[I]_{s+1}}^T \tilde{\mathbf{B}} = \mathbf{I}_{k_1 M}$. By this, the above pair $(\tilde{\mathbf{a}}_{[I]_{s+1}}, \tilde{\mathbf{B}}_{\mathcal{I}})$ yields

$$\tilde{\mathbf{a}}_{[I]_{s+1}}^T (\tilde{\mathbf{B}}_{\mathcal{I}} \bar{\mathbf{C}}^T) = \mathbf{I}_{k_1 M} \bar{\mathbf{C}}^T = \bar{\mathbf{C}}^T, \tag{2.37}$$

which implies that our CMM construction works as expected. In appendix 1.2 we provide the analysis for the case where $k_1 = 1$, as a simpler version of CMM-2. A numerical example is depicted in Appendix 1.3.2, for parameters $k_1 = 1$ and $k_2 = 8$.

2.5.4 Comparison between CMM-1 and CMM-2

We present the trade-offs, in terms of communication, storage, and computational operations required by each worker for our CMMSs in Table II.1. Each scheme may have different uses in practice, in which one is preferable to the other. In certain applications, one may even be easier to implement compared to the other. Also, depending on the limitations and the parameters of the system employing the matrix-matrix multiplications and the underlying application, a different CMMS may be more suitable. Both CMM-1 and CMM-2 have been applied in other coded computing schemes, in which the other cannot be utilized. The approach of CMM-1 was used for approximate matrix-multiplication [53], and the special case of CMM-2 where $k_1 = 1$; for matrix inversion [51]. We briefly explain these applications, after comparing the trade-offs of the two schemes.

Trade-Offs Between Our Two CMM Schemes			
	Communication	Computation	Storage
CMM-1	LM	$LMN/k \cdot \lvert\mathcal{J}_i\rvert$	$\frac{N}{k}(L+M) \cdot \lvert\mathcal{J}_i\rvert$
CMM-2	$\frac{LM}{k_1 k_2} \cdot \lvert\mathcal{J}_i\rvert$	$\frac{LMN}{k_1 k_2} \cdot \lvert\mathcal{J}_i\rvert$	$N\left(\frac{L}{k_1}+\frac{M}{k_2}\right) \cdot \lvert\mathcal{J}_i\rvert$

Table II.1: Comparison of the communication, computation and storage required by the workers in each of our schemes.

For a fair comparison between the two schemes, let us assume that the same number of jobs $|\mathcal{J}_i|$ is assigned to every worker across both CMM-1 and CMM-2. In terms of communication; if the bandwidth is limited, CMM-2 is preferred, as each worker only needs to send a fraction of the LM matrix symbols to the central server; since $|\mathcal{J}_i|/(k_1 k_2) < 1$. In terms of computation, the total number of floating-point operations carried out locally by the workers, is the same in the two schemes, when the total number of subtasks (k and $k_1 k_2$ respectively) are equal. The preferred scheme therefore depends on how parameters k_1 and k_2 are selected for CMM-2; relative to k for CMM-1, and vice versa. In terms of storage, a similar comparison holds, *e.g.* if we set $k_1 = k_2 = k$; the workers in both schemes require the same amount of local storage.

Theorem 2.5.1 and CMM-1 were incorporated in a *weighted CMM* approximation scheme [53]. The idea behind the weighting is that each outer-product matrix, which is requested to be computed, is scaled by an integer factor corresponding to an importance sampling distribution on the submatrix pairs $\{(\tilde{A}_j, \tilde{B}_j)\}_{j=1}^{k}$. The fact that the workers in CMM-1 compute the outer-products of column-row submatrix pairs, permits us to combine this approach with CR-multiplication; a randomized technique which produces a low-rank approximate product of minimum variance [88, 89, 90], as both CMM-1 and CR-multiplication leverage (2.33). This procedure resulted in an approximate CMM with reduced storage and number of operations at the workers.

The approach of CMM-2 was used in [51], as a basis to approximate the inverse and pseudoinverse of a matrix in the presence of stragglers. The analogy which takes place is that instead of the products $A\bar{B}_i$, the workers in the matrix inversion scheme communicate back

$$\left[\hat{B}^{((s+1)i+1)} \ \cdots \ \hat{B}^{((s+1)(i+1))}\right] \in \mathbb{R}^{N\times\frac{N}{k}}, \tag{2.38}$$

for $i = 0, 1, \cdots, k-1$ and $k = N/(s+1)$. Matrix $\hat{B}$ is the approximation of the inverse of $A \in \mathbb{R}^{N\times N}$, *i.e.* $A\hat{B} \approx \mathbf{I}_N$, whose columns are estimated by the workers; who approximate the solutions to the regression problems

$$\hat{B}^{(l)} = \arg\min_{\mathbf{b}\in\mathbb{R}^N} \left\{\|A\mathbf{b} - \mathbf{e}_l\|_2^2\right\}, \tag{2.39}$$

for each of the columns of $\hat{B}$ requested by them, by using an iterative method of their choice.

2.6 Comparison to Prior Works

In this section we briefly review some related work, which we compare our schemes to. We review a polynomial based GC, and three polynomial CMM approaches. We compare and contrast each of the CMM approaches to one of ours (CMM-1, CMM-2, and the special case of CMM-2 presented in Appendix 1.2), which are in fact the closest line of work we could find to each of our proposed schemes. Generally speaking, the communication, storage and computation required by our CMMSs is the same with the respective one we compare it to.

The advantage of the CMM polynomial schemes we will discuss is in terms of the recovery threshold. These schemes achieve a better threshold, as they encode linear

combinations of submatrices of one or both the matrices, and then carry out the computation on the encoded submatrices, from which a fraction of all the assigned tasks they then decode to retrieve the matrix product. As in GC, in our CMMSs we first carry out the computations and then encode them locally at the worker nodes, *e.g.* (2.32) and (2.36). Our CMMSs meet the optimal recovery threshold known for GC, as this is met by the underlying GCS; which we proposed. However, our schemes are superior in terms of encoding and decoding complexity. Furthermore, since the encodings are binary matrices consisting only of 0's and 1's, they introduce no numerical instability nor rounding errors.

We also draw connections with weighted GC, distributed storage systems, and LDPC codes.

2.6.1 Reed-Solomon Scheme and Weighted Gradient Coding

First, we compare our proposed GCS with the one introduced in [134], which provides improvements in terms of the decoding complexity to [279] and, to the best of authors' knowledge, is the first work to consider constructing the decoding vector $\mathbf{a}_{\mathcal{I}}$ online; instead of the matrix comprised of all possible $\mathbf{a}_{\mathcal{I}}$ decoding vectors.

The main idea in [134] is to use balanced Reed-Solomon codes [135], which are evaluation polynomial codes. Each column of the encoding matrix $\mathbf{B}$ corresponds to a partition $\mathcal{D}_i$ and is associated with a polynomial that evaluates to zero at the respective workers who have not been assigned that partition part. The properties of balanced Reed-Solomon codes imply the decomposition $\mathbf{B}_{\mathcal{I}} = \mathbf{G}_{\mathcal{I}}\mathbf{T}$, where $\mathbf{G}_{\mathcal{I}}$ is a Vandermonde matrix over certain roots of unity, and the entries of $\mathbf{T}$ corresponds to the coefficients of polynomials; constructed such that their constant term is 1, *i.e.* $\mathbf{T}_{(1)} = \mathbf{1}_{1\times k}$. The decoding vector $\mathbf{a}_{\mathcal{I}}^T$ is the first row of $\mathbf{G}_{\mathcal{I}}^{-1}$, for which $\mathbf{a}_{\mathcal{I}}^T\mathbf{G}_{\mathcal{I}} = \mathbf{e}_1^T$. A direct consequence of this is that $\mathbf{a}_{\mathcal{I}}^T\mathbf{B}_{\mathcal{I}} = \mathbf{e}_1^T\mathbf{T} = \mathbf{T}_{(1)}$, thus $\mathbf{a}_{\mathcal{I}}^T(\mathbf{B}_{\mathcal{I}}\mathbf{g}) = g^T$.

A drawback of this construction is that it works over the complex numbers and requires an inversion, which introduces numerical instability. Each decoding vector $\mathbf{a}_{\mathcal{I}}$ can be computed in time $O((n-s)^2)$, while the decoding vector proposed in this chapter is constructed in time $O(n+s)$.

The Reed-Solomon based scheme was also used as a basis for *weighted gradient coding* [52]. The idea behind the weighting is similar to that of the weighted CMMS [53] described in 2.5.4. In *weighted GC* the goal is not to recover the sum of the partial gradients, but a weighted linear combination according to some predetermined weight vector $\mathbf{w} \in \mathbb{Z}_+^{1\times k}$. This has many applications in signal processing and statistics. Note also that our proposed binary gradient code $(\mathbf{B}, \mathbf{a}_{\mathcal{I}})$ can be extended to a weighted

scheme $(\tilde{\mathbf{B}}, \mathbf{a}_{\mathcal{I}})$ by the simple modification of $\mathbf{B}$:

$$\tilde{\mathbf{B}}_{ij} = \begin{cases} \mathbf{w}_j & \text{if } \mathbf{B}_{ij} = 1 \\ 0 & \text{if } \mathbf{B}_{ij} = 0 \end{cases} . \tag{2.40}$$

2.6.2 CMM MatDot Codes

The proposed CMM-1 described in 2.5.1 is in nature, close to the "MatDot Codes" from [98, 109], which work with the rank-τ outer-products. In the MatDot procedure, a polynomial of the submatrices $\tilde{A}_i$ and $\tilde{B}_i$ is evaluated, *i.e.*

$$p_A(x) = \sum_{j=1}^{k} \tilde{A}_j x^{j-1} \quad \text{and} \quad p_B(x) = \sum_{j=1}^{k} \tilde{B}_j x^{k-j}, \tag{2.41}$$

over arbitrary distinct elements $x_1, \cdots, x_n$ of a finite field $\mathbb{F}_q$ for some $q > n$. The ι^{th} worker receives the evaluations $p_A(x_\iota)$ and $p_B(x_\iota)$, *i.e.* the evaluations of the polynomials at the evaluation point corresponding to the worker; x_ι. Each worker is the requested to communicate back the computation $P(x_i) = p_A(x_i)p_B(x_i)$, which is a polynomial of degree $2(k-2)$. The sum of all the outer-products is the coefficient of x^{k-1} of the polynomial $p_A(x)p_B(x)$. Once any $2k-1$ evaluations of the polynomial $P(x)$ on distinct points are received, polynomial interpolation of Reed-Solomon decoding can be applied in order to retrieve the product AB [109, 98].

The MatDot procedure in [98, 109] is described for A and B both being square matrices of size $N \times N$, though there is no reason why it should not work for non-square matrices. The overall encoding complexity for each worker if both matrices considered are squares; is $O(N^2 n)$. The overall decoding complexity per worker is $O(k \log^2 k)$ for each entry [170], thus; the overall decoding complexity is $O(N^2 k \log^2 k)$.

The communication cost per worker of the MatDot scheme is the same as CMM-1. A drawback is the storage required at the central server, which was overcome for our scheme through the alternative decoding process of Algorithm 2.

2.6.3 CMM Polynomial Codes

The "Polynomial Codes" proposed in [303] follow a similar approach in terms of computational tasks and concatenation as the proposed CMM-2, though a different encoding and decoding procedure is considered. The Polynomial Codes CMMS partitions both the matrices, A into k_2 submatrices across its rows; and B into k_2 submatrices across its columns. The encodings which take place are similar to those

of MatDot Codes, and the workers are requested to compute the product between an encoding of the submatrices A and of the submatrices B. Once k_1k_2 workers respond, the decoding involves the inversion of a Vandermonde matrix; which is not numerically stable, in order to retrieve the submatrices of C each of size $\frac{L}{k_1} \times \frac{M}{k_2}$, which are then concatenated.

A restriction of the Polynomial Codes, which cannot be altered if we want to have a recovery threshold of k_1k_2, is the fact that the resulting products of the encoded submatrices all need to have the same size, therefore requiring that $k_1|L$ and $k_2|M$. From the analysis we carried for homogeneous workers of our GCS, the partitions of A and B for our CMM-2 do not all need to have the same number of columns.

An extension of the Polynomial codes in [304] does the same augmentation argument after the decoding step, though their encodings take place over submatrices of A and B, where the partitions are carried out for both matrices across the rows and columns.

2.6.4 Connection to Distributed Storage Systems

A central theme of this chapter was relaxing the condition that $(s+1) \mid n$, in order to design a GCS for a pair of integers (s, n) where $0 \leq s < n$ for which $(s+1) \nmid n$. The main idea behind this condition is that the k partitions can be appropriately grouped together and the workers will all get the same number of partitions. This is what is referred to as uniform, defined in Definition 2.3.1. If $d_s(\mathbf{B}) = 0$, the assignment is balanced, according to the terminology in [279]. The arithmetic is easier to work with under this assumption, which is also why our construction results in a block diagonal matrix $\mathbf{B}$; when $(s+1) \mid n$. This is a permutation of the rows of the fractional repetition scheme from [279].

The aforementioned assumption prevails in the construction of *distributed storage systems* as well, which use similar techniques, including replication and block coding to provide reliability. Specifically, in the design of *locally repairable codes* (LRCs) [227]. We relate these two applications; of GC and distributed storage, by indicating how this assumption translates from LRCs to GC schemes, by discussing an analog of [227, Remark 2] in this context. The reader is referred to [227] and the references therein for further details on LRCs.

Remark 2.6.1. *Observe that when $(s+1) \mid n$, we can partition the set of k disjoint parts of $\mathcal{D}$ into $\frac{n}{s+1}$ disjoint $(s+1)$-groups. The method used in [279] to prove the lower bound $\|\mathbf{B}_{(i)}\|_0 \geq \frac{k}{n}(s+1)$ of [279, Theorem 1], relies on the fact that at least*

one encoding of all of the $(s+1)$-groups need be collected, in order for the scheme to be resilient to any s stragglers. The construction of their code meets this bound with equality, as does ours; under the given assumption. That is, the constructions gives an achievability proof for the case of $(s+1) \mid n$. Furthermore, we show that the pair-wise disjoint parts $\mathcal{D}$ is one of the possibly many arrangements of repair groups that leads to optimal constructions (Theorem 2.4.2), as we can rearrange any of the allocation of the parts among each congruence class of workers. This is done in such a way that each partition is allocated to exactly one worker per class.

2.6.5 Connection to LDPC Codes

In terms of error-correcting codes, Theorem 2.4.3 suggests an analogy between permutations of the task allocations per congruence class $\bar{\mathbf{B}}$ of our encoding matrix $\mathbf{B}$, and parity check matrices of LDPC codes. Specifically, $\bar{\mathbf{B}}$ matches the definition of an irregular LDPC parity check matrix $\mathbf{H}$ [116, 196, 246], since along the columns of $\bar{\mathbf{B}}$ and $\mathbf{B}$ we have a balanced load, and along the rows we have an unbalanced load when $(s+1) \nmid n$. That is, $\bar{\mathbf{B}}$ suggested by our construction and Theorem 2.4.3 is a valid $\mathbf{H}$, with the additional constraint that any two rows corresponding to the same congruence class $\mathrm{mod}(s+1)$ have disjoint supports. It is intriguing to see what can be inferred from the constructions and theory of our binary encoding to that of LDPC codes and vice versa, and if further connections can be established.

2.7 Conclusion and Future Work

In this chapter, we introduced a binary GCS for distributed optimization. The main advantages of our code design is that (i) it provides numerically stable computations of the gradient, (ii) it removes the limiting assumption $(s+1) \mid n$, and (iii) it has an improved and more tractable decoding stage compared to that of the first GCS proposed in [279]. We provided an analysis of the proposed design, and showed that it is optimal in terms of the minimizing function d_s defined in Definition 2.3.1. Both homogeneous and heterogeneous workers were considered. It is worth noting that more recent work also considers this direction [269], though their focus is on improving the recovery threshold. We then presented two CMM approaches; as extensions of our binary GCS.

There are several interesting directions for future work. We have seen that the proposed schemes accommodate various matrix operations. It would be interesting to see what other operations they can accommodate, in order to devise exact and

approximate straggler resilient coded computing schemes. Another direction is to incorporate privacy and security into our schemes. A third direction, is to further explore the connections between coded computations and codes for distributed storage systems. Specifically, it would be worthwhile to explore the connections between the proposed GCS, the GCS of [155], and the distributed storage systems of [100], which we briefly described in Subsection 2.6.4.

CHAPTER III

Gradient Coding through Iterative Block Leverage Score Sampling

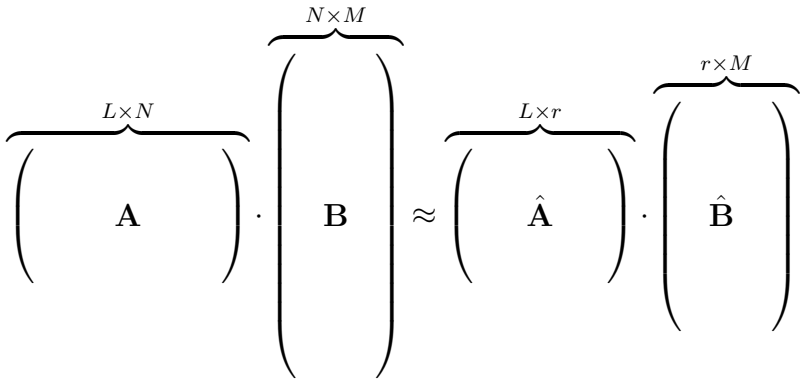
3.1 Introduction

In this chapter we bridge two disjoint areas, to accelerate first-order methods distributively, while focusing on linear regression. Specifically, we propose a framework in which *Randomized Numerical Linear Algebra* (RandNLA) sampling algorithms can be used to devise *Coded Computing* (CC) schemes. We focus on the task of ℓ_2-*subspace embedding* (ℓ_2-s.e.); through leverage score sampling, and distributed gradient computation; which is referred to as *gradient coding* (GC).

Traditional numerical linear algebra algorithms are deterministic. For example, inverting a full-rank matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ requires $O(N^3)$ arithmetic operations by performing Gaussian elimination, as does naive matrix multiplication. The fastest known algorithm which multiplies two $N \times N$ matrices, requires $O(N^\omega)$ operations; for $\omega < 2.373$ [11, 230]. Other important problems are computing the determinant, singular and eigenvalue decompositions, SVD, QR and Cholesky factorizations.

Although these deterministic algorithms run in polynomial time and are numerically stable, their exponents make them prohibitive for many applications in scientific computing and machine learning, when N is in the order of millions or billions [200, 202]. To circumvent this issue, one can perform these algorithms on a significantly smaller approximation. Specifically, for a matrix $\mathbf{S} \in \mathbb{R}^{r\times N}$ with $r \ll N$, we apply the deterministic algorithm on the surrogate $\hat{\mathbf{A}} = \mathbf{SA} \in \mathbb{R}^{r\times d}$. The matrix $\mathbf{S}$ is referred to as a "dimension-reduction" or a "sketching" matrix, and $\hat{\mathbf{A}}$ is a "sketch" of $\mathbf{A}$, which contains as much information about $\mathbf{A}$ as possible. For instance, when multiplying $\mathbf{A} \in \mathbb{R}^{L\times N}$ and $\mathbf{B} \in \mathbb{R}^{N\times M}$, we apply a carefully chosen $\mathbf{S} \in \mathbb{R}^{r\times N}$ on

each to get



for $\hat{\mathbf{A}} = \mathbf{A}\mathbf{S}^\top$ and $\hat{\mathbf{B}} = \mathbf{S}\mathbf{B}$. Thus, naive matrix multiplication now requires $O(LMr)$ operations; instead of $O(LMN)$. Such approaches have been motivated by the Johnson-Lindenstrauss lemma [153], and require low complexity.

A multitude of other problems, such as k-means clustering [35, 45, 67] and computing the SVD of a matrix [86, 87, 89, 90], make use of this idea; in order to accelerate computing accurate approximate solutions. We refer the reader to the following monographs and comprehensive surveys on the rich development of RandNLA [93, 137, 200, 201, 214, 290, 294], an interdisciplinary field that exploits randomization as a computational resource; to develop improved algorithms for large-scale linear algebra problems.

The problem of ℓ_2-s.e.; a form of spectral approximation of a matrix, has been extensively studied in RandNLA. The main techniques for constructing appropriate sketching ℓ_2-s.e. matrices, are performing a random projection or row-sampling. Well-known choices of $\mathbf{S}$ for reducing the effective dimension N to r include: i) *Gaussian projection*; for a matrix $\mathbf{\Theta} \sim \mathcal{N}(0,1)$ define $\mathbf{S} = \frac{1}{\sqrt{r}}\mathbf{\Theta}$, ii) *leverage score sampling*; sample with replacement r rows from the matrix according to its normalized leverage score distribution and rescale them appropriately, iii) *Subsampled Hadamard Transform* (SRHT); apply a Hadamard transform and a random signature matrix to judiciously make the leverage scores approximately uniform and then follow similar steps to the leverage score sampling procedure.

In this chapter, we first generalize ii) to appropriately sample *submatrices* instead of rows to attain a ℓ_2-s.e guarantee. We refer to such approaches as *block sampling*. Throughout this paper, sampling is done with replacement (w.r.). Sampling blocks has been explored in "block-iterative methods" for solving systems of linear equations [101, 131, 215, 240]. Our motivation in dealing with blocks rather than individual vectors, is the availability to invoke results that can be used to characterize the

approximations of distributed computing networks, to speed up first-order methods, as sampling individual rows/columns is prohibitive in real-world environments. This in turn leads to an *iterative sketching* approach, which has been well studied in terms of second-order methods [232, 173, 231]. By iterative sketching, we refer to an iterative algorithm which uses a new sketch $\mathbf{S}_{[s]}$ at each iteration. The scenario where a single sketch $\mathbf{S}$ is applied before the iterative process, is referred to as the "sketch-and-solve paradigm" [255].

Second, we propose a general framework which incorporates our sketching algorithm into a CC approach. This framework accommodates a central class of sketching algorithms, that of importance (block) sampling algorithms (*e.g.* CUR decomposition [221], CR-multiplication [53]). Coded computing is a novel computing paradigm that utilizes coding theory to effectively inject and leverage data and computation redundancy to mitigate errors and slow or non-responsive servers; known as *stragglers*, among other fundamental bottlenecks, in large-scale distributed computing. In our setting, the straggling effect is due to computations being communicated over *erasure channels*, whose erasure probability follows a probability distribution which is central to the CC probabilistic model. The seminal work of [178] which first introduced CC, focused on exact matrix-vector multiplication and data shuffling. More recent works deal with recovering good approximations, while some have utilized techniques from RandNLA; *e.g.* [20, 52, 53, 127, 128, 147]. Our results are presented in terms of the standard CC probabilistic model proposed in [178], though they extend to any computing network comprised of a central server and computing nodes, referred to as *servers*.

To mitigate stragglers, we appropriately encode and replicate the data blocks, which leads to accurate CC estimates. In contrast to previous works which simply replicate each computational task or data block the same number of times [305, 279, 47, 48], we replicate blocks according to their *block leverage scores*. Consequently, this induces a non-uniform sampling distribution in the aforementioned CC model; which is an approximation to the normalized block leverage scores. A drawback of using RandNLA techniques is that exact computations are not recoverable, though our method does not require a decoding step, a task of high complexity and a prevalent bottleneck in CC. For more details on the various directions of CC, the reader is referred to the monographs [181, 216].

The central idea of our approach is that non-uniform importance sampling can be emulated, by replicating tasks across the network's servers, who communicate through an erasure channel. The tasks' computation times follow a runtime distribution [178],

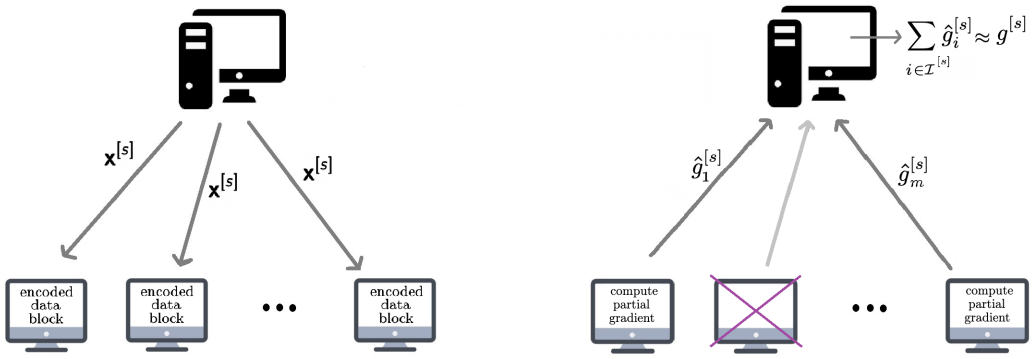


Figure III.1: Schematic of our approximate GC scheme, at iteration s. Each server has an encoded block of data, of which they compute the gradient once they receive the updated parameters $\mathbf{x}^{[s]}$. The central server then aggregates a subset of all the gradients $\{\hat{g}_j^{[s]}\}_{j=1}^m$, indexed by $\mathcal{I}^{[s]}$, to approximate the gradient $g^{[s]}$. At each iteration we expect a different index set $\mathcal{I}^{[s]}$, which leads to iterative sketching.

which along with a prespecified gradient transmission "*ending time*" T, determine the number of replications per task across the network. Though similar ideas have been proposed [52, 53, 127, 128]; where sketching has been incorporated into CC, this is the first time redundancy is introduced through RandNLA; as opposed to compression, to obtain approximation guarantees. In terms of CC, though uniform replication is a very powerful technique, it does not capture the relevance between the information of the dataset. We capture such information through replication and rescaling according to the block leverage scores. By then allowing *uniform* sampling of these blocks, we attain a spectral approximation. In the CC setting this then corresponds to an iterative ℓ_2-s.e. sketching method. The shortcoming of this approach, which is the cost we pay for guaranteeing a spectral approximation through uniform sampling, is that we expect to require a large amount of servers; when the underlying sampling distribution is non-uniform.

In Appendix 2.5 we discuss how further compression can be attained by introducing *weighting*, while guaranteeing the same results when first and second order methods are used for linear regression in the sketch-and-solve paradigm (Proposition 2.5.1 and Corollary 2.5.1). We also show that in terms of the expected reduced dimension, we have minimal benefit when the block leverage scores distribution is uniform (Theorem 2.5.1). This further justifies the fact that sharper decays in leverage scores lead to more accurate algorithms[226].

All completed jobs that are received by the central server are aggregated to get the

final gradient approximation, at each iteration of the descent method being carried out. Thus, unlike most CC schemes, ours does not store completed jobs which will not be accounted for. Our method sacrifices accuracy for speed of computation, and the inaccuracy is quantified in terms of the resulting ℓ_2-s.e. (Theorem 3.3.1). Specifically, the computations of the responsive servers will correspond to sampled *block* computation tasks of our proposed generalization to leverage score sampling, summarized in Algorithm 5. Approximate coded computations is a current interest in information-theory, as it is conceivable that data dependent approximation schemes could lead to faster inexact but accurate solutions, at a lower computational cost [181].

To summarize, our main contributions are: 1) propose *block* leverage score sketching, to accommodate block sampling for ℓ_2-s.e., 2) provide theoretical guarantees for the algorithm's performance, 3) show the significance of weighting; when our *weighted* sketching algorithms are is applied in iterative first and second order methods, 4) propose *expansion networks*; which use the sampling distribution to determine how to replicate and distribute blocks in the CC framework — this unifies the disciplines of RandNLA and CC — where replication and uniform sampling (without a random projection) result in a spectral approximation, 5) show how expansion networks are used for approximate distributed *steepest descent* (SD); and approach the optimal solution with unbiased gradient estimators in a similar manner to *batch stochastic steepest descent* (SSD), 6) experimental justification on the performance of our algorithm on artificial datasets with non-uniform induced sampling distributions.

The chapter is organized as follows. In 3.2 we present the notation which will used, and review necessary background. In 3.3.1 we present related works, in terms of CC. In 3.3.2 we present our sketching algorithm and its theoretical approximation guarantees. In 3.3.3 and 3.3.4 we give a framework for which our algorithm; as well as potentially other importance sampling algorithms, can be used to devise CC schemes. This is where we introduce redundancy through RandNLA, which has not been done before. In 3.3.5 we summarize our GC scheme, and in 3.3.6 we give a brief synopsis of our main results and tie everything together. In 3.3.7 we show how our scheme relates to *approximate GC*. We conclude with experimental evaluations in 3.4 on fabricated data with highly non-uniform underlying sampling distributions, to convey the maximum benefit of what we propose.

3.2 Notation and Background

We denote $\mathbb{N}_n \coloneqq \{1, 2, \cdots, n\}$, and $X_{\{n\}} = \{X_i\}_{i=1}^n$; where X could be replaced by any variable. We use $\mathbf{A}, \mathbf{B}$ to denote real matrices, $\mathbf{b}, \mathbf{x}$ real column vectors, $\mathbf{I}_n$ the $n \times n$ identity matrix, $\mathbf{0}_{n\times m}$ and $\mathbf{1}_{n\times m}$ respectively the $n \times m$ all zeros and all ones matrices, and by $\mathbf{e}_i$ the standard basis column vector whose dimension will be clear from the context. The largest eigenvalue of a matrix $\mathbf{M}$, is denoted by $\lambda_1(\mathbf{M})$. By $\mathbf{A}_{(i)}$ we denote the i^{th} row of $\mathbf{A}$, by $\mathbf{A}^{(j)}$ its j^{th} column, by $\mathbf{A}_{ij}$ the value of $\mathbf{A}$'s entry in position (i, j), and by $\mathbf{x}_i$ the i^{th} element of $\mathbf{x}$. The rounding function to the nearest integer is expressed by $\lfloor\cdot\rceil$, *i.e.* $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$ for $a \in \mathbb{R}$. Disjoint unions are represented by $\bigsqcup$; *e.g.* $\mathbb{Z} = \{j : j \text{ is odd}\} \bigsqcup \{j : j \text{ is even}\}$, and we define $\biguplus$ as the addition of multisets; *e.g.* $\{1, 2, 3\} \biguplus \{3, 4\} = \{1, 2, 3, 3, 4\}$. The diagonal matrix with real entries $a_{\{n\}}$ is expressed as $\operatorname{diag}\big(a_{\{n\}}\big)$.

We partition vectors and matrices across their rows into K submatrices, in a way that no submatrix differs from another by more than one row. For simplicity, we assume that K divides the number of rows N. That is, for a ℓ_2-s.e. of $\mathbf{A} \in \mathbb{R}^{N\times d}$ with target $\mathbf{b} \in \mathbb{R}^N$, we assume $K \mid N$ and the "size of each partition" is $\tau = N/K$.[1] We partition $\mathbf{A}, \mathbf{b}$ across their rows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^\top & \cdots & \mathbf{A}_K^\top \end{bmatrix}^\top \qquad \text{and} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1^\top & \cdots & \mathbf{b}_K^\top \end{bmatrix}^\top \tag{3.1}$$

where $\mathbf{A}_i \in \mathbb{R}^{\tau\times d}$ and $\mathbf{b}_i \in \mathbb{R}^\tau$ for all $i \in \mathbb{N}_K$. Partitions, are referred to as *blocks*. Throughout the paper we consider the case where $N \gg d$. For $\mathbf{A}$ full-rank, its SVD is $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{N\times d}$ is its reduced left orthonormal basis.

Matrix $\mathbf{A}$ represents a dataset $\mathcal{D}$ of N samples with d features, and $\mathbf{b}$ the corresponding labels of the data points. The partitioning (3.1) corresponds to K sub-datasets $\mathcal{D}_{\{K\}}$, *i.e.* $\mathcal{D} = \bigsqcup_{j=1}^K \mathcal{D}_j$. Our results are presented in terms of an arbitrary partition $\mathbb{N}_N = \bigsqcup_{\iota=1}^K \mathcal{K}_\iota$, for $\mathbb{N}_N$ the index set of the rows of $\mathbf{A}$ and $\mathbf{b}$. The index subsets $\mathcal{K}_{\{K\}}$ indicate which data samples are in each sub-dataset. By $\mathbf{A}_{(\mathcal{K}_\iota)}$, we denote the submatrix of $\mathbf{A}$ comprised of the rows indexed by $\mathcal{K}_\iota$. That is, for $\mathbf{I}_{(\mathcal{K}_\iota)}$ the restriction of $\mathbf{I}_N$ to only include its rows corresponding to $\mathcal{K}_\iota$, we have $\mathbf{A}_{(\mathcal{K}_\iota)} = \mathbf{I}_{(\mathcal{K}_\iota)} \cdot \mathbf{A}$. By $\mathcal{K}_\iota^i$, we indicate that the ι^{th} block was sampled at trial i, *i.e.* the superscript i indicates the sampling trial and the subscript $\iota \in \mathbb{N}_K$ which block was sampled at

[1] If $K \nmid N$, we appropriately append zero vectors/entries until this is met. It is not required that all blocks have the same size, though we discuss this case to simplify the presentation. One can easily extend our results to blocks of varying sizes, and use the analysis from [47] to determine the optimal size of each partition.

that trial. Aslo, by $j(i)$ we denote the index of the submatrix which was sampled at the i^{th} sampling trial, *i.e.* $\mathcal{K}_{j(i)} = \mathcal{K}^i_{j(i)}$. The complement of $\mathcal{K}_\iota$ is denoted by $\bar{\mathcal{K}}$; *i.e.* $\bar{\mathcal{K}}_\iota = \mathbb{N}_N \backslash \mathcal{K}_\iota$, for which $\mathbf{U}^\top_{(\mathcal{K}_\iota)}\mathbf{U}_{(\mathcal{K}_\iota)} = \mathbf{I}_d - \mathbf{U}^\top_{(\bar{\mathcal{K}}_\iota)}\mathbf{U}_{(\bar{\mathcal{K}}_\iota)}$.

Sketching matrices are represented by $\mathbf{S}$ and $\widetilde{\mathbf{S}}_{[s]}$. The script $[s]$ indexes an iteration $s = 0, 1, 2, \ldots$ which we drop when it is clear from the context. We will be reducing dimension N to r, *i.e.* $\mathbf{S} \in \mathbb{R}^{r\times N}$. Sampling matrices are denoted by $\mathbf{\Omega} \in \{0,1\}^{r\times N}$, and diagonal rescaling matrices by $\mathbf{D} \in \mathbb{R}^{N\times N}$.

Approximate block sampling distributions to $\Pi_{\{K\}}$ are denoted by $\tilde{\Pi}_{\{K\}}$, and the distributions induced through expansion networks by $\bar{\Pi}_{\{K\}}$. We quantify the difference between distributions $\Pi_{\{K\}}$ and $\tilde{\Pi}_{\{K\}}$ by the distortion metric $d_{\Pi,\tilde{\Pi}} \coloneqq \frac{1}{K}\sum_{i=1}^K |\Pi_i - \tilde{\Pi}_i|$, which is the ℓ_1 distortion between $\Pi_{\{K\}}$ and $\tilde{\Pi}_{\{K\}}$, *e.g.* [145].

3.2.1 Least Squares Approximation

Least squares approximation is a technique to find an approximate solution to a system of linear equations that has no exact solution, and has found applications in many fields [96]. Consider the system $\mathbf{Ax} = \mathbf{b}$, for which we want to find an approximation to the best-fitted

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \Big\{ L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}) \coloneqq \|\mathbf{Ax} - \mathbf{b}\|_2^2 \Big\}, \tag{3.2}$$

which objective function L_{ls} has gradient

$$g^{[s]} = \nabla_{\mathbf{x}} L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[s]}) = 2\mathbf{A}^\top(\mathbf{Ax}^{[s]} - \mathbf{b}). \tag{3.3}$$

We refer to the gradient of the block pair $(\mathbf{A}_i, \mathbf{b}_i)$ from (3.1) as the i^{th} *partial gradient*; $g_i^{[s]} = \nabla_{\mathbf{x}} L_{ls}(\mathbf{A}_i, \mathbf{b}_i; \mathbf{x}^{[s]})$. Existing exact methods find a solution vector $\mathbf{x}^\star$ in $O(Nd^2)$ time, where $\mathbf{x}^\star = \mathbf{A}^\dagger\mathbf{b}$. In Subsection 3.3.5 we focus on approximating the optimal solution $\mathbf{x}^\star$ by using our methods, via distributive SD/SSD and iterative sketching. What we present also accommodates regularizers of the form $\lambda\|\mathbf{x}\|_2^2$, though to simplify our expressions, we only consider (3.2).

3.2.2 Steepest Descent

When considering a minimization problem with a convex differentiable objective function $L\colon \mathbb{R}^d \to \mathbb{R}$, we select an initial $\mathbf{x}^{[0]} \in \mathbb{R}^d$ and repeat at iteration $s+1$: $\mathbf{x}^{[s+1]} \leftarrow \mathbf{x}^{[s]} - \xi_s \cdot \nabla_{\mathbf{x}} L(\mathbf{x}^{[s]})$; for $s = 0, 1, 2, \ldots$, until a prespecified termination criterion is met. The parameter $\xi_s > 0$ is the corresponding step-size, which may be

adaptive or fixed. To guarantee convergence of L_{ls}, one can select $\xi_s = 2/\sigma_{\max}(\mathbf{A})^2$ for all iterations, though this is too conservative.

3.2.3 Leverage Scores

Many sampling algorithms select data points according to their *leverage scores* [91, 195]. The leverage scores of $\mathbf{A}$ measure the extent to which the vectors of its orthonormal basis $\mathbf{U}$ are correlated with the standard basis, and define the key structural non-uniformity that must be dealt with when developing fast randomized matrix algorithms; as they characterize the importance of the data points. Leverage scores defined as $\ell_i \coloneqq \|\mathbf{U}_{(i)}\|_2^2$, and are agnostic to any particular basis, as they are equal to the diagonal entries of the projection matrix $P_{\mathbf{A}} = \mathbf{A}\mathbf{A}^\dagger = \mathbf{U}\mathbf{U}^\top$. The *normalized leverage scores* of $\mathbf{A}$ are

$$\pi_i \coloneqq \left\|\mathbf{U}_{(i)}\right\|_2^2 / \|\mathbf{U}\|_F^2 = \left\|\mathbf{U}_{(i)}\right\|_2^2 / d \qquad \text{for each } i \in \mathbb{N}_N, \tag{3.4}$$

and $\pi_{\{N\}}$ form a sampling probability distribution; as $\sum_{i=1}^N \pi_i = 1$ and $\pi_i \geqslant 0$ for all i. This induced distribution has proven to be useful in linear regression [91, 106, 294, 201].

The normalized *block leverage scores*, introduced independently in [52, 221], are the sum of the normalized leverage scores of the subset of rows constituting the block. Analogous to (3.4), considering the partitioning of $\mathcal{D}$ according to $\mathcal{K}_{\{K\}}$, the *normalized block leverage scores* of $\mathbf{A}$ are defined as

$$\Pi_l \coloneqq \left\|\mathbf{U}_{(\mathcal{K}_l)}\right\|_F^2 / \|\mathbf{U}\|_F^2 = \left\|\mathbf{U}_{(\mathcal{K}_l)}\right\|_F^2 / d = \sum_{j \in \mathcal{K}_l} \pi_j \quad \text{for each } l \in \mathbb{N}_K. \tag{3.5}$$

A related notion is that of the *Frobenius block scores*, which in the case of a partitioning as in (3.1); are $\|\mathbf{A}_\iota\|_F^2$ for each $\iota \in \mathbb{N}_K$, which scores have been used for CR-multiplication [89, 90]. In our context, the block leverage scores of $\mathbf{A}$; are the Frobenius block scores of $\mathbf{U}$.

A drawback of using leverage scores, is that calculating them requires $O(Nd^2)$ time. To alleviate this, one can instead settle for relative-error approximations which can be approximated much faster, *e.g.* [91] does so in $O(Nd \log N)$ time. In particular, we can consider approximate normalized scores $\tilde{\Pi}_{\{K\}}$ where $\tilde{\Pi}_i \geqslant \beta \Pi_i$ for all i, for some misestimation factor $\beta \in (0, 1]$. Since $\Pi_{\{K\}}$ and $\tilde{\Pi}_{\{K\}}$ are identical if and only if $\beta = 1$, a higher β implies the approximate distribution is more accurate.

3.2.4 Subspace Embedding

Our approach to approximating (3.2), is to apply a ℓ_2-s.e sketching matrix $\mathbf{S} \in \mathbb{R}^{r\times N}$ on $\mathbf{A}$. Recall that $\mathbf{S} \in \mathbb{R}^{r\times N}$ is a $(1\pm\epsilon)$ ℓ_2-*subspace embedding* of the column-space of $\mathbf{A}$, if

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \leqslant \|\mathbf{S}\mathbf{A}\mathbf{x}\|_2^2 \leqslant (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \tag{3.6}$$

for all $\mathbf{x} \in \mathbb{R}^d$, w.h.p. [294]. Notice that such an $\mathbf{S}$, is also a $(1\pm\epsilon)$ ℓ_2-s.e. of $\mathbf{U}$, as $\{\mathbf{A}\mathbf{x} : \mathbf{x} \in \mathbb{R}^d\} = \{\mathbf{U}\mathbf{y} : \mathbf{y} \in \mathbb{R}^d\}$. This implies that (3.6) is equivalent to

$$(1-\epsilon)\|\mathbf{y}\|_2^2 = (1-\epsilon)\|\mathbf{U}\mathbf{y}\|_2^2 \leqslant \|\mathbf{S}\mathbf{U}\mathbf{y}\|_2^2 \leqslant (1+\epsilon)\|\mathbf{U}\mathbf{y}\|_2^2 = (1+\epsilon)\|\mathbf{y}\|_2^2 \tag{3.7}$$

for all $\mathbf{y} \in \mathbb{R}^d$. The upper and lower bounds on $\|\mathbf{S}\mathbf{U}\mathbf{y}\|_2^2$ respectively imply

$$\mathbf{y}^\top\big((\mathbf{S}\mathbf{U})^\top\mathbf{S}\mathbf{U} - \mathbf{I}_d\big)\mathbf{y} \leqslant \epsilon\|\mathbf{y}\|_2^2 \quad \text{and} \quad \mathbf{y}^\top\big(\mathbf{I}_d - (\mathbf{S}\mathbf{U})^\top\mathbf{S}\mathbf{U}\big)\mathbf{y} \leqslant \epsilon\|\mathbf{y}\|_2^2$$

thus, a simplified condition for a ℓ_2-s.e. of $\mathbf{A}$ is

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^\top\mathbf{S}^\top\mathbf{S}\mathbf{U}\|_2 \leqslant \epsilon\right] \geqslant 1-\delta \tag{3.8}$$

for a small $\delta \geqslant 0$.

For the overdetermined system $\mathbf{A}\mathbf{x} = \mathbf{b}$, we require $r > d$, and in the sketch-and-solve paradigm the objective is to determine an $\hat{\mathbf{x}}$ which satisfies

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}^\star - \mathbf{b}\|_2 \leqslant \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2 \leqslant (1+\epsilon)\|\mathbf{A}\mathbf{x}^\star - \mathbf{b}\|_2, \tag{3.9}$$

where $\hat{\mathbf{x}}$ is an approximate solution to the modified least squares problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^d}{\arg\min}\Big\{L_{\mathbf{S}}(\mathbf{S},\mathbf{A},\mathbf{b};\mathbf{x}) \coloneqq \|\mathbf{S}(\mathbf{A}\mathbf{x}-\mathbf{b})\|_2^2\Big\}. \tag{3.10}$$

If (3.8) is met, we get w.h.p. the approximation characterizations:

1. $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2 \leqslant \frac{1+\epsilon}{1-\epsilon}\|\mathbf{A}\mathbf{x}^\star - \mathbf{b}\|_2 \leqslant (1+O(\epsilon))\|\mathbf{A}\mathbf{x}^\star - \mathbf{b}\|_2$
2. $\|\mathbf{A}(\mathbf{x}^\star - \hat{\mathbf{x}})\|_2 \leqslant \epsilon\|(\mathbf{I}_N - \mathbf{U}\mathbf{U}^\top)\mathbf{b}\|_2 = \epsilon\|\mathbf{b}^\perp\|_2$

where $\mathbf{b}^\perp = \mathbf{b} - \mathbf{A}\mathbf{x}^\star$ is orthogonal to the column span of $\mathbf{A}$, *i.e.* $\mathbf{A}^\top\mathbf{b}^\perp = \mathbf{0}_{d\times 1}$.

3.2.5 Coded Computing Probabilistic Model

In GC (Figure III.1), there is a central server who shares the K disjoint subsets $\mathcal{D}_{\{K\}}$ of $\mathcal{D}$ among m *homogeneous*[2] servers, to facilitate computing the solution of minimization problems with differentiable additively separable objective functions, *e.g.* (3.2):

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 = \sum_{j=1}^{K} L_{ls}(\mathcal{D}_j;\mathbf{x}) \right\}.$$

Since the objective function $L_{ls}(\mathbf{A},\mathbf{b};\mathbf{x})$ is additively separable, it follows that $g^{[s]} = \sum_{j=1}^{K} g_j^{[s]}$. The objective function's gradient is updated in a distributed manner; while only requiring q servers to respond, *i.e.* it is robust to $m-q$ stragglers. This is achieved through an encoding of the computed partial gradients by the servers, and a decoding step once q servers have sent back their encoded computation.

We consider the probabilistic computational model introduced in [178], which is the standard CC paradigm; and is central to our framework. This model assumes the existence of a *mother runtime distribution* $F(t)$, with a corresponding probability density function $f(t)$. Let T_0 be the time it takes a single machine to complete its computation, and define $F(t) \coloneqq \Pr[T_0 \leqslant t]$. We further assume that the runtime distribution of the subtasks, with random amount of completion time T^i, are a scaled distribution of $F(t)$. That is, when all servers have a computational task of size τ, computing a τ/N-fraction of the overall computation; follows the runtime distribution $\tilde{F}(t) \coloneqq F(t\tau/N) = \Pr[T^i \leqslant t]$. In this chapter, we view the computations as being communicated to the central server over erasure channels, where the l^{th} server W_l has an erasure probability[3]

$$\phi(t) \coloneqq 1-\tilde{F}(t) = 1-\Pr\big[W_l \text{ responds by time } t\big], \tag{3.11}$$

i.e. the probability that W_l is a straggler at time t is $\phi(t)$. All servers have the same erasure probability, as we are assuming they are homogeneous.

In our setting, there are two hyperparameters required for determining an expansion network. First, one needs to determine a time instance $t \leftarrow T$ after which the central server will stop receiving servers' computations.[4] This may be decided by

[2]This means that they have the same computational power, independent and identically distributed statistics for the computing time of similar tasks; and expected response time.

[3]This is also known as the *survival function*: $\phi(t) = \int_t^\infty \big(1-\tilde{f}(u)\big)du = 1-\tilde{F}(t) = \Pr[T^i > t]$, for $\tilde{f}(t)$ the PDF corresponding to $\tilde{F}(t)$. The function $\phi(t)$ is monotonically decreasing.

[4]By $t \leftarrow T$, we mean T is a realization of the time variable t.

factors such as the system's limitations, number of servers, or an upper bound on the desired waiting time for the servers to respond. At time T, according to $\tilde{F}(t)$, the central server receives roughly $q(T) \coloneqq \lfloor \tilde{F}(T) \cdot m \rfloor$ server computations. We refer to the prespecified time instance T after which the central server stops receiving computations; as the "*ending time*". If the sketching procedure of the proposed sketching algorithm were to be carried out by a single server, there would be no benefit in setting T such that $q(T)\tau > N$, as the exact calculation could have taken place in the same amount of time. In distributed networks though there is no control over which servers respond, and it is not a major concern if $q(T)\tau$ is slightly over N; as we still accelerate the computation. The trade-off between accuracy and waiting time t is captured in Theorem 3.3.1, for $q \leftarrow q(t)$ sampling trials. The second hyperparameter we need in order to design an expansion network, is the block size τ; which is determined by K the number of partitions (3.1). Together, $q(T)$ and τ determine the ideal number of servers needed for our framework to perfectly emulate sampling according to the datas' block leverage scores $\Pi_{\{K\}}$.

3.3 Coded Computing from RandNLA

In this section, we first present our *block* leverage score sampling algorithm, which is more practical and can be carried out more efficiently than its vector-wise counterpart. Our ℓ_2-s.e. result is presented in Theorem 3.3.1. By setting $\tau = 1$ and for $\beta = 1$, we get a known result for (exact) leverage score sampling.

In Subsection 3.3.3 we incorporate our block sampling algorithm into the CC probabilistic model described above, in which we leverage task redundancy to mitigate stragglers. Specifically, we show how to replicate computational tasks among the servers, under the integer constraints imposed by the physical system and the desired waiting time; to approximate the gradient at each iteration, in a way that emulates the sampling procedure of the sketch presented in Algorithm 5. In Subsection 3.3.4 we further elaborate on when a perfect emulation is possible, and how emulated block leverage score sampling can be improved when it cannot be done perfectly; through the proposed networks. In Subsection 3.3.5 we present our GC approach, and relate it to SD and SSD; which in turn implies convergence guarantees with appropriate step-sizes. Furthermore, at each iteration we have a different induced sketch, hence our procedure lies under the framework of iterative sketching. Specifically, we obtain gradients of multiple sketches of the data $\left(\widetilde{\mathbf{S}}_{[1]}\mathbf{A}, \widetilde{\mathbf{S}}_{[2]}\mathbf{A}, \ldots, \widetilde{\mathbf{S}}_{[n]}\mathbf{A}\right)$ and iteratively refine the solution, where n can be chosen logarithmic in N. A schematic of our

approach is provided in Figure III.2, and in Appendix 2.2 we provide a concrete example of the induced sketching matrices resulting from the iterative process.

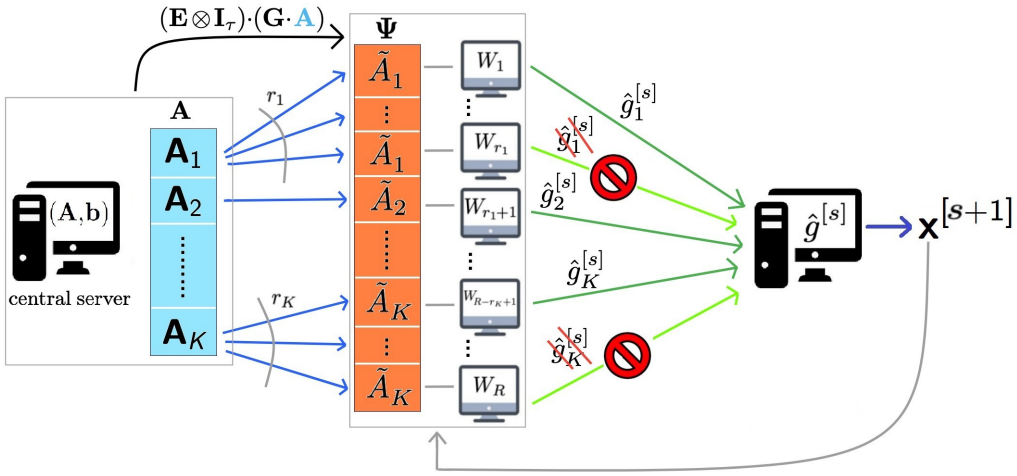


Figure III.2: Illustration of our GC approach, at iteration $s+1$. The blocks of $\mathbf{A}$ (and $\mathbf{b}$) are encoded through $\mathbf{G}$ and then replicated through $\mathbf{E}\otimes\mathbf{I}_\tau$, where each block of the resulting $\mathbf{\Psi}$ is given to a single server. At this iteration, servers W_{r_1} and W_R are stragglers, and their computations are not received. The central server determines the estimate $\hat{g}^{[s]}$, and then shares $\mathbf{x}^{[s+1]}$ with all the servers. The resulting estimate is the gradient of the induced sketch, *i.e.* $\hat{g}^{[s]} = \nabla_{\mathbf{x}} L_{\mathbf{S}}(\widetilde{\mathbf{S}}_{[s]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[s]})$.

3.3.1 Related Work

Related works [112, 127, 128, 146] have utilized similar ideas to the GC approach we present. The paper titled "Anytime Coding for Distributed Computation" [112] proposes replicating subtasks according to the job, while [127] and [146] incorporate sketching into CC. It is worth noting that even though we focus on gradient methods in this chapter; our approach also applies to second-order methods, as well as approximate matrix products through the CR-multiplication algorithm [53, 86, 88]. We briefly discuss this in Section 3.5.

The work of [112] deals with matrix-vector multiplication. Similar to our work, they also replicate the computational tasks a certain number of times; and stop the process at a prespecified instance. Here, the computation $\mathbf{Ax}$ for $\mathbf{A} \in \mathbb{R}^{N\times N}$ and $\mathbf{x} \in \mathbb{R}^N$ is broken up into C different tasks; prioritizing the smaller computations. The m servers are split up into c groups, which are asked to compute one of the tasks $\mathbf{y}_j = \left(\sum_{i\in\mathcal{J}_j} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top\right)\mathbf{x}$, for $\mathbf{A} = \sum_{l=1}^{N} \sigma_l \mathbf{u}_l \mathbf{v}_l^\top$ the SVD representation of $\mathbf{A}$. Each

task $\mathbf{y}_j$ is computed by the servers of the respective group, and $\mathbb{N}_N = \bigsqcup_{j=1}^{s} \mathcal{J}_j$ is a disjoint partitioning of the rank-1 outer-products of the SVD representation. The size of the j^{th} task is $|\mathcal{J}_j| = p_j$, which in our work is determined by the normalized block scores. The scores in our proposed schemes are motivated and justified by RandNLA, in contrast to the selection of the sizes p_j which is not discussed in [112]. Furthermore, the scheme of [112] requires a separate maximum distance separable code for each job $\mathbf{y}_j$; thus requiring multiple decodings, while we do not require a decoding step. Another drawback of [112] is that an integer program is set up to determine the optimal ending time, which the authors do not solve, while we determine a scheme for any desired ending time. Lastly, we note that the ℓ_2-s.e. approximation guarantee of our method, depends on the ending time T.

In terms of sketching and RandNLA, the works of [146] and [128] utilize the *Count-Sketch* [57]; which relies on hashing. In "CodedSketch" [146], Count-Sketches are incorporated into the design of a variant of the improved "Entangled Polynomial Code" [304], to combine approximate matrix multiplication with straggler tolerance. The code approximates the submatrix blocks $\{\mathbf{C}_{i,j} : (i,j) \in \mathbb{N}_{k_1} \times \mathbb{N}_{k_2}\}$ of the final product matrix $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ (matrix $\mathbf{C}$ is partitioned k_1 times across its rows, and k_2 times across its columns), with an accuracy that depends on $\|\mathbf{C}_{i',j'}\|_F$ for all $(i',j') \in \mathbb{N}_{k_1} \times \mathbb{N}_{k_2}$. This prevents it from being applicable to applications that require accuracy guarantees without oracle knowledge of the outcome of each submatrix of the matrix product $\mathbf{C}$. This approach permits each block of $\mathbf{C}$ to be approximately recovered, if a subset of the servers complete their tasks.

In "OverSketch" [128], redundancy is introduced through additional Count-Sketches, to mitigate the effect of stragglers in distributed matrix multiplication. In particular, the count-sketches $\breve{\mathbf{A}} = \mathbf{AS}$ and $\breve{\mathbf{B}} = \mathbf{S}^\top\mathbf{B}$ of the two inputs $\mathbf{A}$ and $\mathbf{B}$ are computed, and are partitioned into submatrices of size $b \times b$. The $b \times b$ submatrices of the final product $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ are then approximately calculated, by multiplying the corresponding row-block of $\breve{\mathbf{A}}$ and column-block of $\breve{\mathbf{B}}$; each of which is done by one server. The "OverSketch" idea has also been extended to distributed Newton Sketching [127] for convex optimization problems.

3.3.2 Block Leverage Score Sampling

In the leverage score sketch [91, 106, 195, 201, 294] we sample w.r. r rows according $\pi_{\{N\}}$ (3.2), and then rescale each sampled row by $1/\sqrt{r\pi_i}$. Instead, we sample w.r. q blocks from (3.1) according to $\Pi_{\{K\}}$ (3.5), and rescale them by $1/\sqrt{q\Pi_i}$. The pseudocode of the *block leverage score sketch* is given in Algorithm 5, where we consider

an approximate distribution $\tilde{\Pi}_{\{K\}}$ such that $\tilde{\Pi}_i \geqslant \beta \Pi_i$ for all i, for $\beta \in (0,1]$ a dependent loss in accuracy [91, 95, 201]. The spectral guarantee of the sketching matrix $\widetilde{\mathbf{S}}$ of Algorithm 5 is presented in Theorem 3.3.1. Iterative sketching in our distributed GC approach through Algorithm 5, corresponds to selecting a new sampling matrix $\widetilde{\mathbf{\Omega}}^{[s]}$ at each iteration through the servers' responses, *i.e.* $\widetilde{\mathbf{S}}_{[s]} = \widetilde{\mathbf{D}} \cdot \widetilde{\mathbf{\Omega}}^{[s]}$ for each s.

Algorithm 5: Block Leverage Score Sketch

Input: $\mathbf{A} \in \mathbb{R}^{N \times d}$, $\tau = \frac{N}{K}$, $q = \frac{r}{\tau} > \frac{d}{\tau}$
Output: $\widetilde{\mathbf{S}} \in \mathbb{R}^{r \times N}$, $\widehat{\mathbf{A}} \in \mathbb{R}^{r \times d}$
Initialize: $\mathbf{\Omega} = \mathbf{0}_{q \times K}$, $\mathbf{D} = \mathbf{0}_{q \times q}$
Compute: (approximate) distribution $\tilde{\Pi}_{\{K\}}$ (3.5)
for $j = 1$ *to* q **do**
  sample w.r. i_j from $\mathbb{N}_K$, according to $\tilde{\Pi}_{\{K\}}$
  $\mathbf{\Omega}_{j,i_j} = 1$ ▷ equivalently $\mathbf{\Omega}_{(j)} = \mathbf{e}_{i_j}^\top$
  $\mathbf{D}_{j,j} = \sqrt{\frac{\tau}{r\tilde{\Pi}_{i_j}}} = \sqrt{\frac{1}{q\tilde{\Pi}_{i_j}}}$
end
$\widetilde{\mathbf{\Omega}} \leftarrow \mathbf{\Omega} \otimes \mathbf{I}_\tau$
$\widetilde{\mathbf{D}} \leftarrow \mathbf{D} \otimes \mathbf{I}_\tau$
$\widetilde{\mathbf{S}} \leftarrow \widetilde{\mathbf{D}} \cdot \widetilde{\mathbf{\Omega}}$ ▷ $\widetilde{\mathbf{S}} = (\mathbf{D} \cdot \mathbf{\Omega}) \otimes \mathbf{I}_\tau$
$\widehat{\mathbf{A}} \leftarrow \widetilde{\mathbf{S}} \cdot \mathbf{A}$

Theorem 3.3.1. *The sketching matrix* $\widetilde{\mathbf{S}}$ *of Algorithm 5 is a* $(1 \pm \epsilon)$ ℓ_2*-s.e of* $\mathbf{A}$*, according to* (3.8)*. Specifically, for* $\delta > 0$ *and* $q = \Theta\left(\frac{d}{\tau} \log\left(2d/\delta\right)/(\beta\epsilon^2)\right)$ *we get*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^\top \widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}} \mathbf{U}\|_2 \leqslant \epsilon\right] \geqslant 1 - \delta.$$

Proof. The main tool is a matrix Chernoff bound [294, Fact 1]. We define random matrices corresponding to the sampling process and bound their norm and variance, in order to apply the aforementioned Chernoff bound. The complete proof can be found in Appendix 2.1. □

The importance of Theorem 3.3.1 extends beyond leverage score sampling. Specifically, one can apply a random projection to "flatten" the block leverage scores; *i.e.* they are all approximately equal, and then sample w.r. uniformly at random. This is the main idea behind the analysis of the SRHT [5, 6]. The trade-off between such algorithms and Algorithm 5, is computing the leverage scores explicitly vs. applying a random projection. Such sketching approaches which do not directly utilize the data, are referred to as "data oblivious sketches", and are better positioned for handling

high velocity streams; as well as highly unstructured and arbitrarily distributed data [211]. Multiplying the data by random matrix spreads the information in the rows of the matrix, such that all rows are of equal importance; and the new matrix is "incoherent". In Appendix 2.3, we show when Algorithm 5 and the corresponding block sampling counterpart of the SRHT [49] achieve the same asymptotic guarantees, for the same number of sampling trials q.

Next, we provide a sub-optimality result for non-iterative sketching of the block leverage score sketch for ordinary least squares:

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ L_{\mathbf{S}}(\widetilde{\mathbf{S}}, \mathbf{A}, \mathbf{b}; \mathbf{x}) \coloneqq \|\widetilde{\mathbf{S}}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2 \right\} . \tag{3.12}$$

which follows from the results of [231]. Specifically, following the proof of [231, Theorem 1]; which is based on a reduction from statistical minimax theory combined with information-theoretic bounds and an application of Fano's inequality, we simply need to upper bound $\left\|\mathbb{E}\big[\widetilde{\mathbf{S}}^\top \big(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^\top\big)^{-1}\widetilde{\mathbf{S}}\big]\right\|_2$.

Corollary 3.3.1. *For any full-rank data matrix $\mathbf{A} \in \mathbb{R}^{N\times d}$ with a noisy observation model $\mathbf{b} = \mathbf{A}\mathbf{x}^\bullet + \mathbf{w}$ where $\mathbf{w} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_N)$, the optimal least squares solution $\mathbf{x}^\star$ of (3.2); has prediction error $\mathbb{E}\big[\|\mathbf{A}(\mathbf{x}^\bullet - \mathbf{x}^\star)\|_2^2\big] \lesssim \frac{\sigma^2 d}{N}$. On the other hand, [231, Theorem 1] implies that any estimate $\tilde{\mathbf{x}}$ based on the sketched system $(\widetilde{\mathbf{S}}\mathbf{A}, \widetilde{\mathbf{S}}\mathbf{b})$ produced from Algorithm 5 with sampling probabilities $\Pi_{\{K\}}$, has a prediction error lower bound of*

$$\mathbb{E}\big[\|\mathbf{A}(\mathbf{x}^\bullet - \tilde{\mathbf{x}})\|_2^2\big] \gtrsim \frac{\sigma^2 d}{\min\{r, N\}} . \tag{3.13}$$

Even though Corollary 3.3.1 considers sampling according to the exact block leverage scores, its proof can be modified to accommodate approximate sampling also. Additionally, the above corollary holds for constrained least squares, though we do not explicitly state it; as it is not a focus of the work presented in this chapter. From (3.13), it is clear that for a smaller r with $r < N$; we get a less efficient sketch and approximation $\tilde{\mathbf{x}}$, though when considering a higher r which approaches N; we get an improvement in the accuracy of $\tilde{\mathbf{x}}$ at the cost of a higher computation and computational load in our resulting GC scheme.

3.3.3 Expansion Networks

The framework we propose emulates the sampling w.r. procedure of Algorithm 5, in distributed CC environments. Even though we focus on ℓ_2-s.e. and descent meth-

ods in this chapter, the proposed framework applies to any matrix algorithm which utilizes importance sampling with replacement. In contrast to other CC schemes in which RandNLA was used to *compress* the network; *e.g.* [52, 53, 250], here the networks are *expanded* according to $\Pi_{\{K\}}$ — the computations corresponding to the blocks are replicated through the servers; proportional to $\Pi_{\{K\}}$. It is unlikely that we can exactly emulate this distribution, as the number of replications per task need to be integers. Instead, we mimic the exact probabilities with an *induced* distribution $\bar{\Pi}_{\{K\}}$ through *expansion networks*, which are determined by $\tilde{F}(t)$ at a prespecified $t \leftarrow T$; after which the central server stops receiving computations for that iteration.

We propose the minimization problem (3.17), whose approximate solution $\hat{r}_{\{K\}}$ (3.18) suggests the number of replicas r_i of each block in our expansion network. We note that (3.17) is a surrogate to the integer program (3.20), whose solution can achieve an accurate realizable distribution $\bar{\Pi}_{\{K\}}$ to $\Pi_{\{K\}}$ through the distributed network, by appropriately replicating the blocks. Unfortunately, the integer program (3.20) is not always solvable. Nonetheless, when we have an approximation to (3.17) or (3.20), through *uniform* sampling we can minimize w.h.p. the ℓ_2-s.e. condition for $\mathbf{A}$ (3.8), up to a small error, given the integer constraints imposed by the physical system — $r_i \in \mathbb{Z}_+$ for all i and $R = \sum_{l=1}^{K} r_l$ such that $R \approx m$. In the CC context, we want $m = R$; *i.e.* the total number of replicated blocks is equal to the number of servers. Next, we describe the desired induced distribution $\bar{\Pi}_{\{K\}}$, in order to set up (3.17).

Assume w.l.o.g. that $\Pi_j \leqslant \Pi_{j+1}$ for all $j \in \mathbb{N}_{K-1}$, thus $r_j \leqslant r_{j+1}$. The sampling distribution through the expansion network translates to

$$\bar{\Pi}_i \coloneqq \Pr\left[\text{the } i^{th} \text{ block is sampled}\right] = r_i/R \approx \Pi_i \tag{3.14}$$

for all $i \in \mathbb{N}_K$ and $R = \sum_{l=1}^{K} r_l$. Our objective is to determine $r_{\{K\}}$ such that $\bar{\Pi}_i \approx \Pi_i$ for all i. Furthermore, for an erasure probability determined by $\phi(t)$ at a specified time t (3.11), the probability that the computation corresponding to the i^{th} block is sampled w.r. through the erasure channels at time t is

$$\Pr\left[\text{sample the } i^{th} \text{ block through the channels}\right] = 1 - \phi(t)^{\rho_i(t)}, \tag{3.15}$$

for some $\rho_i(t) \in \mathbb{R}_{>0}$,[5] and the network emulates the sampling distribution $\Pi_{\{K\}}$ exactly when

$$\Pi_i = 1 - \phi(t)^{\rho_i(t)} \quad \text{for all } i. \tag{3.16}$$

[5]To be realizable, through replications, we need $\rho_i(t) \in \mathbb{Z}_+$.

The replications which take place can be interpreted as the task allocation through a directed bipartite graph $G = (\mathcal{L}, \mathcal{R}, \mathcal{E})$, where $\mathcal{L}$ and $\mathcal{R}$ correspond to the K *encoded* partitions $\tilde{A}_{\{K\}}$ and m servers respectively, where $\deg(x_i) = r_i$ for all $x_i \in \mathcal{L}$ and $\deg(y_j) = 1$ for all $y_j \in \mathcal{R}$; with $\{x_i, y_j\} \in \mathcal{E}$ only if the j^{th} server W_j is assigned $\tilde{A}_i$.

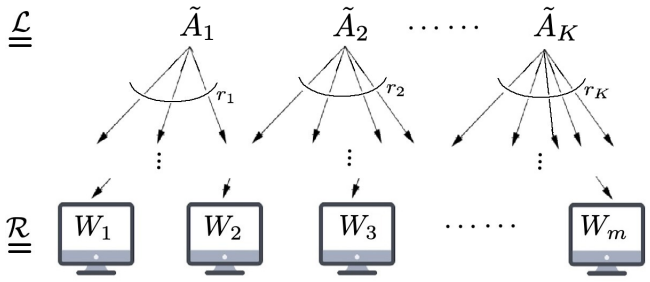


Figure III.3: Depiction of an expansion network, as a bipartite graph, for $m = \sum_{l=1}^{K} r_l$.

Our goal is to determine $r_{\{K\}}$, which minimize the error in the emulated distribution $\bar{\Pi}_{\{K\}}$. Under the assumption that we have an integer number of replicas per block, from (3.15) and (3.16) we deduce that $\Pi_i \approx 1 - \phi(t)^{r_i}$ for $r_i \in \mathbb{Z}_+$, which lead us to the minimization problem[6]

$$\arg \min_{r_{\{K\}} \subsetneq \mathbb{Z}_+} \left\{ \Delta_{\Pi,\bar{\Pi}} \coloneqq \frac{1}{K} \sum_{i=1}^{K} \left| \Pi_i - \left(1 - \phi(t)^{r_i}\right) \right| \right\} = \tag{3.17}$$
$$= \arg \left\{ \sum_{i=1}^{K} \min_{r_i \in \mathbb{Z}_+} \left\{ \left| \Pi_i - \left(1 - \phi(t)^{r_i}\right) \right| \right\} \right\} .$$

By combining (3.14) and (3.15), we then solve for the approximate replications $\hat{r}_{\{K\}}$ at time t:

$$\bar{\Pi}_i \approx \Pi_i = 1 - \phi(t)^{\rho_i(t)} \quad \Longrightarrow \quad \hat{r}_i = \left\lfloor \frac{\log(1 - \Pi_i)}{\log(\phi(t))} \right\rceil = \lfloor \rho_i(t) \rceil \,, \tag{3.18}$$

which result in the induced distribution $\bar{\Pi}_i = \hat{r}_i / \hat{R}$, for $\hat{R} \coloneqq \sum_{l=1}^{K} \hat{r}_l$. In our context, we also require that $\hat{R} \approx m$.

Ideally, the above procedure would result in replication numbers $\hat{r}_{\{K\}}$ for which $\hat{R} = m$. This though is unlikely to occur, as $\Pi_{\{K\}}$ and R are determined by the data, and m is a physical limitation. There are several practical ways to work around this.

[6]Note that $\Delta_{\Pi,\bar{\Pi}} \equiv d_{\Pi,\tilde{\Pi}}$, where $\tilde{\Pi}_i = 1 - \phi(t)^{r_i}$ for all $i \in \mathbb{N}_K$. For our proposed distribution $\bar{\Pi}_{\{K\}}$, we may have $d_{\Pi,\bar{\Pi}} \neq \Delta_{\Pi,\bar{\Pi}}$.

One approach is to redefine $\hat{r}_{\{K\}}$ to $\tilde{r}_{\{K\}}$ by $\tilde{r}_i = \hat{r}_i \pm \alpha_i$ for α_i small integers such that $\sum_{l=1}^{K} \tilde{r}_l = m$ and $\sum_{l=1}^{K} |\Pi_l - \tilde{r}_l/m|$ is minimal. If $m \gg \hat{R}$ for a large enough τ, we can set the number of replicas to be $\tilde{r}_i \approx \left\lfloor m/\hat{R} \right\rceil \cdot \hat{r}_i$. Furthermore, the block size τ can be selected such that $\hat{R}$ is approximately equal to the system's parameter m. We focus on the issue of having $\hat{R} \approx m$ in Subsection 3.3.4.

Lemma 3.3.1. *The approximation $\hat{r}_{\{K\}}$ according to (3.18) of the minimization problem (3.17) at time t, satisfies*

$$\Delta_{\Pi,\bar{\Pi}} \leqslant \left(1 - \sqrt{\phi(t)}\right) \cdot \left(\sum_{l=1}^{K} \phi(t)^{\min\limits_{i \in \mathbb{N}_K} \{\hat{r}_i, \rho_i(t)\}}\right).$$

Proof. We break the proof into the cases where we round $\rho_i(t)$ to both the closest integers above and below. In either case, we know that $\big(\rho_i(t) - \hat{r}_i(t)\big) \in [-1/2, 1/2]$, for each $i \in \mathbb{N}_K$. Denote the respective individual summands of $\Delta_{\Pi,\bar{\Pi}}$ by Δ_i. In the case where $r_i = \lfloor \rho_i(t) \rfloor$, we have $\rho_i(t) = \hat{r}_i + \eta$ for $\eta \in [0, 1/2]$, hence

$$\begin{aligned}
\Delta_i &= \left|\left(1 - \phi(t)^{\rho_i(t)}\right) - \left(1 - \phi(t)^{\hat{r}_i}\right)\right| \\
&= \left|\phi(t)^{\hat{r}_i} - \phi(t)^{\rho_i(t)}\right| \\
&= \left|\phi(t)^{\hat{r}_i} - \phi(t)^{r_i + \eta}\right| \\
&= \left|\phi(t)^{\hat{r}_i} \cdot \left(1 - \phi(t)^{\eta}\right)\right| \\
&\leqslant \left|\phi(t)^{\hat{r}_i} \cdot \left(1 - \phi(t)^{1/2}\right)\right| \\
&= \phi(t)^{\hat{r}_i} \cdot \left(1 - \sqrt{\phi(t)}\right).
\end{aligned}$$

Similarly, in the case where $r_i = \lceil \rho_i(t) \rceil$, we have $\hat{r}_i = \rho_i(t) + \eta$ for $\eta \in [0, 1/2]$; and

$$\Delta_i \leqslant \phi(t)^{\rho_i(t)} \cdot \left(1 - \sqrt{\phi(t)}\right).$$

Considering all summands, it follows that

$$\Delta_{\Pi,\bar{\Pi}} = \sum_{l=1}^{K} \Delta_l \leqslant \sum_{l=1}^{K} \left(\phi(t)^{\min\limits_{i \in \mathbb{N}_K} \{\hat{r}_i, \rho_i(t)\}} \cdot \left(1 - \sqrt{\phi(t)}\right)\right).$$

□

We note that all terms involved in the upper bound of Lemma 3.3.1 are positive and strictly less that one. Furthermore, for a larger t we have a smaller $\phi(t)$, while

for a smaller t we have a smaller $\hat{r}_i$ for each i. This bound further corroborates the importance of the hyperparameter t and the distribution $F(t)$, in designing expansion networks.

The replication of blocks which takes place, can be described through a corresponding "*expansion matrix*":

$$\widetilde{\mathbf{E}} = \mathbf{E} \otimes \mathbf{I}_\tau = \begin{pmatrix} \mathbf{1}_{r_1 \times 1} & & & \\ & \mathbf{1}_{r_2 \times 1} & & \\ & & \ddots & \\ & & & \mathbf{1}_{r_K \times 1} \end{pmatrix} \otimes \mathbf{I}_\tau \in \{0,1\}^{R\tau \times K\tau} \tag{3.19}$$

where $\mathbf{E} \in \{0,1\}^{R \times K}$ is the adjacency matrix of the bipartite graph G (up to a permutation of the rows/server indices). It follows that $(\widetilde{\mathbf{E}} \cdot \mathbf{A}, \widetilde{\mathbf{E}} \cdot \mathbf{b})$ are comprised of replicated blocks of the partitioning in (3.1), with replications according to $r_{\{K\}}$.

For the proposed networks, the multiplicative misestimation factor in Theorem 3.3.1 is $\beta_{\bar{\Pi}} = \min_{i \in \mathbb{N}_K}\{\Pi_i / \bar{\Pi}_i\} \leqslant 1$. In the case where $\tilde{R} = \sum_{l=1}^K \tilde{r}_l > m$ and $\tilde{\Pi}_i \coloneqq \tilde{r}_i / \tilde{R}$, Algorithm 6 takes $\tilde{r}_{\{K\}}$ as an input and determines $r_{\{K\}}$ such that $R = \sum_{l=1}^K r_l = m$. The updated distribution $\bar{\Pi}_{\{K\}}$ where $\bar{\Pi}_i = r_i / R$ for each i, also has a more accurate misestimation factor; *i.e.* $\beta_{\bar{\Pi}} > \beta_{\tilde{\Pi}}$. To establish sampling guarantees in relation to $d_{\Pi,\bar{\Pi}}$, one would need to invoke an additive approximation error to the scores, *i.e.* $\tilde{\Pi}_i \leqslant \Pi_i + \epsilon$ for all i where $\epsilon \geqslant 0$ is a small constant [68, 91]. In our distributed networks, the additive error would be $\epsilon_{\bar{\Pi}} = \max_{i \in \mathbb{N}_K}\{|\Pi_i - \bar{\Pi}_i|\}$.

3.3.4 Optimal Induced Distributions

Recall that (3.17) is a surrogate to the integer program

$$r^\star_{\{K\}} = \arg \min_{\substack{r_1,\ldots,r_K \in \mathbb{Z}_+ \\ R = \sum_{l=1}^K r_l}} \left\{ d_{\Pi,\Pi^\star} = \frac{1}{K} \sum_{i=1}^K \big| \Pi_i - \overbrace{r_i / R}^{\Pi^\star_i} \big| \right\} \tag{3.20}$$

for $R \approx m$ the total number of servers. Potential solutions $r^\star_{\{K\}}$ can achieve the closest realizable distribution to $\Pi_{\{K\}}$ through expansion networks. Similar to $\Delta_{\Pi,\bar{\Pi}}$ from (3.17), the distortion metric $d_{\Pi,\Pi^\star}$ is a measure of closeness between the distributions $\Pi_{\{K\}}$ and $\bar{\Pi}_{\{K\}}$; under the network imposed constraints. In the case where $\Pi_{\{K\}} \subsetneq (0,1) \backslash \mathbb{Q}_+$; *i.e.* $\Pi_{\{K\}}$ are not necessarily all rational, the integer constraints of the physical network may deem exactly emulating $\Pi_{\{K\}}$ impossible. The integer program (3.20) cannot be solved exactly when $\Pi_{\{K\}} \subsetneq (0,1) \backslash \mathbb{Q}_+$; as we can always get finer

approximations, *e.g.* through a continued fraction approximation. This is specific to the ending time T when considering erasures over the communication channels according to (3.11), which T we do not include in (3.20); in order to simplify notation. Furthermore, (3.20) can also be considered for centralized distributed settings which differ from the system model proposed in [178]. We note that solvers to (3.20) exist, when R is fixed and we remove the constraint $R = \sum_{l=1}^{K} r_l$. The proof of Corollary 3.3.2 is a constructive solution of (3.20) when $\Pi_{\{K\}} \subsetneq [0,1] \cap \mathbb{Q}_+$, in which case perfect emulation is possible.

Proposition 3.3.1. *A perfect emulation occurs when $d_{\Pi,\Pi^\star} = 0$. This is possible if and only if $\Pi_i \in [0,1] \cap \mathbb{Q}_+$ for all i and the denominators of $\Pi_{\{K\}}$ in reduced form are factors of R; i.e. $R \cdot \Pi_i \in \mathbb{Z}_+$.*

Proof. If $d_{\Pi,\Pi^\star} = 0$, then $\Pi_i = \Pi_i^\star$ for all $i \in \mathbb{N}_K$, thus $\Pi_{\{K\}}$ and $\Pi_{\{K\}}^\star$ are the same sampling distributions.

For the reverse direction, assume that for all i we have $\Pi_i = a_i/b_i$ for coprime integers $a_i, b_i \in \mathbb{Z}_+$, and that $R = \mu_i b_i$ for some $\mu_i \in \mathbb{Z}_+$; thus $R \cdot \Pi_i = \mu_i a_i \in \mathbb{Z}_+$. Let $r_i = \mu_i a_i$. It follows that $\Pi_i^\star = \frac{r_i}{R} = \frac{\mu_i a_i}{\mu_i b_i} = \frac{a_i}{b_i} = \Pi_i$ for all i, hence $d_{\Pi,\Pi^\star} = 0$. Now, assume for a contradiction that there is a $j \in \mathbb{N}_K$ for which $\Pi_j \in (0,1) \backslash \mathbb{Q}_+$. Then, by definition, Π_j cannot be expressed as a fraction r_j/R for $r_j, R \in \mathbb{Z}_+$, thus $d_{\Pi,\Pi^\star} \geqslant |\Pi_j - \Pi_j^\star| > 0$. □

Corollary 3.3.2. *When $\Pi_{\{K\}} \subsetneq [0,1] \cap \mathbb{Q}_+$, we can solve (3.20), so that $d_{\Pi,\Pi^\star} = 0$.*

Proof. From Proposition 3.3.1, the smallest R in order to attain $r_{\{K\}}$ for which $d_{\Pi,\Pi^\star} = 0$ when considering $\Pi_i = a_i/b_i$ in reduced form, is the *least common multiple* $R = \operatorname{lcm}(b_1, \ldots, b_K)$. For each $i \in \mathbb{N}_K$, we then have $R = \mu_i b_i$ for $\mu_i \in \mathbb{Z}_+$, and $r_i = \mu_i a_i$. Hence $\Pi_i^\star = \frac{r_i}{R} = \frac{\mu_i a_i}{\mu_i b_i} = \frac{a_i}{b_i} = \Pi_i$, for which $d_{\Pi,\Pi^\star} = 0$. □

Lemma 3.3.2. *If for a set of integers $\tilde{r}_{\{K\}}$; we have $\tilde{R} = \sum_{l=1}^{K} \tilde{r}_l$, $m = \tilde{R}$, and $\tilde{\Pi}_i = \tilde{r}_i/\tilde{R}$ for all $i \in \mathbb{N}_K$, then:*

$$\frac{1}{m} \cdot \min_{i \in \mathbb{N}_K} \left\{ \left\lfloor |m \cdot \Pi_i - \tilde{r}_i| \right\rfloor \right\} \leqslant d_{\Pi,\tilde{\Pi}} \leqslant \frac{1}{m} \cdot \max_{i \in \mathbb{N}_K} \left\{ \left\lceil |m \cdot \Pi_i - \tilde{r}_i| \right\rceil \right\}. \tag{3.21}$$

Proof. Let $\tilde{d}_i = |\Pi_i - \tilde{\Pi}_i| = |\Pi_i - \tilde{r}_i/m|$ for each i, hence

$$\tilde{r}_L \coloneqq \frac{1}{m} \cdot \min_{i \in \mathbb{N}_K} \left\{ \left\lfloor |m \cdot \Pi_i - \tilde{r}_i| \right\rfloor \right\} \leqslant \tilde{d}_i \leqslant \frac{1}{m} \cdot \max_{i \in \mathbb{N}_K} \left\{ \left\lceil |m \cdot \Pi_i - \tilde{r}_i| \right\rceil \right\} =: \tilde{r}_U$$

for all $i \in \mathbb{N}_K$. By rescaling the sum over all $\tilde{d}_{\{K\}}$ by $1/K$, we get

$$\tilde{r}_L = \frac{K \cdot \tilde{r}_L}{K} \leqslant \frac{1}{K} \cdot \sum_{i=1}^{K} \tilde{d}_i \leqslant \frac{K \cdot \tilde{r}_U}{K} = \tilde{r}_U$$

which completes the proof. □

Next, we give a simple approximation to (3.20), for when we do not consider the erasure channel characterization through (3.16); nor an ending time T. Given $\Pi_{\{K\}}$, the replication numbers are $\tilde{r}_i = \lfloor \Pi_i / \Pi_1 \rceil$ and $\tilde{R} = \sum_{l=1}^{K} \tilde{r}_l$. Further note that for more accurate approximations, we can select an integer $\nu > 1$ and take $\tilde{r}_i = \lfloor \nu \cdot \Pi_i / \Pi_1 \rceil$. From the proof of Corollary 3.3.2, it follows that if $\Pi_{\{K\}} \subsetneq [0,1] \cap \mathbb{Q}_+$ and $\nu = \mu_1 a_1$, we get $\tilde{r}_i = \Pi_i / \Pi_1 = \mu_i a_i \in \mathbb{Z}_+$, which solves (3.16). The drawback of designing an expansion network with this solution, is that as ν increases; $\tilde{R}$ also increases.

To drop the constraint $R = \sum_{l=1}^{K} r_l$ of (3.20) and the assumption that $m = R$, we give a procedure in Algorithm 6 for determining $r_{\{K\}}$ from a given set $\tilde{r}_{\{K\}}$ (*e.g.* those proposed in (3.18)) to get the induced distribution $\{\bar{\Pi}_i = r_i/m\}_{i=1}^{K}$; where $m = \sum_{l=1}^{K} r_l$. In Algorithm 6, $\chi = 1$ and $\chi = 0$ indicate whether $\tilde{R} > m$ or $\tilde{R} < m$ respectively.

Remark 3.3.1. *The objective of Algorithm 6 is to reduce the upper bound of (3.21) when $m < \tilde{R}$, while guaranteeing that $\sum_{l=1}^{K} r_l = m$. In practice, the more concerning and limiting case is when $m < \tilde{R}$. The bottleneck of Algorithm 6 is retrieving the index j in the while loop, which takes $O(K)$ time. In order to reduce the number of instances we solve (⧫), we ensure that we only reduce the replica numbers $\hat{r}_{\{K\}}$ in the case where $R > m$; and increase them when $R < m$, by the inner **if** statement. Moreover, this is carried out once before sharing the replicated blocks. The more practical and realistic case is when $R > m$, as we can get a closer approximation with a greater R; and $\operatorname{lcm}(b_1, \ldots, b_K)$ will likely be large when $\Pi_{\{K\}} \subsetneq [0,1] \cap \mathbb{Q}_+$. We note that the integers $\hat{r}_{\{K\}}$ of (3.18) and their sum $\hat{R}$ are a byproduct of the prespecified ending time $t \leftarrow T$, the mother runtime distribution $F(t)$, and the block size τ, which can be selected so that $\hat{R} > m$.*

Proposition 3.3.2. *Assume we are given $\tilde{r}_{\{K\}}$ (not necessarily according to (3.18)), for which $m < \tilde{R} = \sum_{l=1}^{K} \tilde{r}_l$. Denote by $\tilde{\Pi}_{\{K\}}$ the corresponding sampling distribution $\left\{\tilde{\Pi}_i = \tilde{r}_i / \tilde{R}\right\}_{i=1}^{K}$, for which $d_{\Pi,\tilde{\Pi}} \leqslant \tilde{\epsilon}$. Then, the output $r_{\{K\}}$ of Algorithm 6 produces an induced distribution $\{\bar{\Pi}_i = r_i/m\}_{i=1}^{K}$ which satisfies $d_{\Pi,\bar{\Pi}} \lesssim d_{\Pi,\tilde{\Pi}} \leqslant \tilde{\epsilon}$.*

Algorithm 6: Determine $r_{\{K\}}$ from $\tilde{r}_{\{K\}}$

Input: m, $\Pi_{\{K\}}$, $\tilde{r}_{\{K\}}$, $\tilde{R} = \sum_{i=1}^{K} \tilde{r}_i$
Output: replication numbers $r_{\{K\}}$
Initialize: $\tilde{d}_{\{K\}} = \left\{\tilde{d}_i \coloneqq \Pi_i - \tilde{r}_i/m\right\}_{i=1}^{K}$, $r_{\{K\}} = \tilde{r}_{\{K\}}$, $R = \tilde{R}$, $\chi = 1$, $\tilde{j} = 0$
if $R < m$ **then**
  $\chi \leftarrow 0$ ▷ ▷ χ indicates: $\tilde{R} \geqslant m$ or $\tilde{R} < m$
  $\tilde{d}_{\{K\}} \leftarrow \left\{-\tilde{d}_i\right\}_{i=1}^{K}$
end
while $R \neq m$ **do**
  $j \leftarrow \arg\min_{i \in \mathbb{N}_K} \left\{\tilde{d}_{\{K\}}\right\}$ (⧫)
  if $(-1)^{\chi+1} \cdot \left(\Pi_j - \frac{1}{m}\left(r_j + (-1)^{\chi}\right)\right) > 0$ ***and*** $\tilde{j} \equiv j$ **then**
    $\tilde{d}_j \leftarrow 1$
  end
  else
    $r_j \leftarrow r_j + (-1)^{\chi}$
    $R \leftarrow R + (-1)^{\chi}$
    $\tilde{d}_j \leftarrow (-1)^{\chi+1} \cdot (\Pi_j - r_j/m)$
  end
  $\tilde{j} \leftarrow j$
end

Proof. The case where $\tilde{R} > m$, corresponds to $\chi = 1$, in which case we Algorithm 6 returns $r_{\{K\}}$ for which $r_i \leqslant \tilde{r}_i$ for all $i \in \mathbb{N}_K$. In this case, the optimization problem (⧫) assigns to j a partition index for which $\Pi_j < r_j/m$.[7] The **if** statement guarantees that we did not produce an r_j for which $\Pi_j > r_j/m$; when we previously previously $\Pi_j < \tilde{r}_j/\tilde{R}$ (or $\Pi_j < r_j/R$ after a reassignment of $\tilde{r}_j$). Along with the fact that $\frac{r_j - 1}{R-1} < \frac{r_j}{R}$, it follows that for the updated difference d'_j:

$$\left|d'_j\right| = \left|\Pi_j - \frac{r_j - 1}{R-1}\right| = \frac{r_j - 1}{R-1} - \Pi_j \lneq \frac{r_j}{R} - \Pi_j = \left|\Pi_j - \frac{r_j}{R}\right| = \left|\tilde{d}_j\right|,$$

i.e. at each iteration of the **else** statement, we decrease the distortion metric, thus $d_{\Pi,\bar{\Pi}} \lneq d_{\Pi,\tilde{\Pi}} \leqslant \tilde{\epsilon}$.

The **else** statement is carried out $\tilde{R} - m$ times, producing $r_{\{K\}}$ for which $\sum_{l=1}^{K} r_l = \tilde{R} - (\tilde{R} - m) = m$, hence the normalizing integer for the distribution $\bar{\Pi}_{\{K\}}$ is $R = m$. □

Remark 3.3.2. *To summarize, the objective is to determine replicas* (3.20)*, in order*

[7] Since $\tilde{d}_j = \Pi_j - r_j/m < 0$, we have $\Pi_j < r_j/m$.

to emulate block leverage score sampling of Algorithm 5 through erasure channels; at an ending time T. By Proposition 3.3.1, this is not always possible. Instead, we give an estimate solution to (3.17) through $\hat{r}_{\{K\}}$ in (3.18). To get a valid sampling distribution in the CC framework, we normalize the replica numbers $\hat{r}_{\{K\}}$ by their sum $\hat{R}$. Furthermore, we propose Algorithm 6 which takes estimates $\hat{r}_{\{K\}}$ and modifies them to return $r_{\{K\}}$ for which $m = \sum_{l=1}^{K} r_l$, and improves the induced approximate block leverage score distribution when $\hat{R} > m$.

3.3.5 GC through Leverage Score Sampling

Next, we derive the server computations of our GC scheme, so that the central server retrieves the gradient of $\|\widetilde{\mathbf{S}}_{[s]}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2$; for $\widetilde{\mathbf{S}}_{[s]}$ according to Algorithm 5 at iteration s, to iteratively approximate (3.2). Furthermore, we show that with a diminishing step-size, our updates $\mathbf{x}^{[s]}$ converge in expectation to the optimal solution $\mathbf{x}^\star$.

The blocks of our leverage score sampling procedure are those of the encoded data $\tilde{\mathbf{A}} \coloneqq \mathbf{G} \cdot \mathbf{A}$ and $\tilde{\mathbf{b}} \coloneqq \mathbf{G} \cdot \mathbf{b}$, for

$$\mathbf{G} = \operatorname{diag}\left(\left\{1/\sqrt{q\bar{\Pi}_i}\right\}_{i=1}^{K}\right) \otimes \mathbf{I}_\tau \in \mathbb{R}_{\geqslant 0}^{N \times N}. \tag{3.22}$$

Specifically, the encoding carried out by the central server corresponds to the rescaling through $\mathbf{G}$. We partition both $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{b}}$ across their rows analogous to (3.1):

$$\tilde{\mathbf{A}} = \mathbf{G} \cdot \mathbf{A} = \begin{bmatrix} \tilde{A}_1^\top & \cdots & \tilde{A}_K^\top \end{bmatrix}^\top \quad \text{and} \quad \tilde{\mathbf{b}} = \mathbf{G} \cdot \mathbf{b} = \begin{bmatrix} \tilde{b}_1^\top & \cdots & \tilde{b}_K^\top \end{bmatrix}^\top \tag{3.23}$$

where $\tilde{A}_i \in \mathbb{R}^{\tau \times d}$ and $\tilde{b}_i \in \mathbb{R}^\tau$ for all $i \in \mathbb{N}_K$. Furthermore, all the data across the expansion network after the scalar encoding, is contained in aggregated '*expanded and encoded matrix-vector pairs*' $(\vec{\Psi}, \vec{\psi}) \coloneqq (\widetilde{\mathbf{E}} \cdot \tilde{\mathbf{A}}, \widetilde{\mathbf{E}} \cdot \tilde{\mathbf{b}}) \in \mathbb{R}^{R\tau \times d} \times \mathbb{R}^{R\tau}$ (Figure III.2). For the encoded objective function $L_{\mathbf{G}}(\mathbf{x}) \coloneqq \|\mathbf{G}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2$, we have:

(i) $L_{\mathbf{G}}(\mathbf{x}) = \mathbf{x}^\top \left(\sum_{i=1}^{K} \tilde{A}_i^\top \tilde{A}_i\right) \mathbf{x} + \left(\sum_{i=1}^{K} (\tilde{b}_i^\top - 2\mathbf{x}^\top \tilde{A}_i^\top) \tilde{b}_i\right)$

(ii) $\nabla_{\mathbf{x}} L_{\mathbf{G}}(\mathbf{x}) = 2 \sum_{i=1}^{K} \tilde{A}_i^\top \left(\tilde{A}_i \mathbf{x} - \tilde{b}_i\right)$

(iii) $\nabla_{\mathbf{x}} L_{\mathbf{G}}(x_{\mathbf{G}}^\star) = 0 \implies x_{\mathbf{G}}^\star = \left(\sum_{i=1}^{K} \tilde{A}_i^\top \tilde{A}_i\right)^{-1} \cdot \left(\sum_{i=1}^{K} \tilde{A}_i^\top \tilde{b}_i\right)$.

We make use of (ii) to approximate the gradient distributively. Each server is provided with a partition $\tilde{\mathcal{D}}_j = (\tilde{A}_j, \tilde{b}_j)$, and computes the respective summand of

the gradient from (ii), which is the encoded partial gradient on $\mathcal{D}_j = (\mathbf{A}_j, \mathbf{b}_j)$:

$$\hat{g}_i^{[s]} \coloneqq \nabla_{\mathbf{x}} L_{ls}(\tilde{\mathcal{D}}_i; \mathbf{x}^{[s]}) = \nabla_{\mathbf{x}} L_{ls}\left(1/\sqrt{q\bar{\Pi}_i} \cdot \mathbf{A}_i, 1/\sqrt{q\bar{\Pi}_i} \cdot \mathbf{b}_i; \mathbf{x}^{[s]}\right) = \frac{1}{q\bar{\Pi}_i} \cdot g_i^{[s]} \ . \quad (3.24)$$

Once a server computes its assigned partial gradient, it sends it back to the central server. When the central server receives q responses, it sums them in order to obtain the approximate gradient $\hat{g}^{[s]}$.

Denote the index multiset corresponding to the encoded pairs $(\tilde{A}_j, \tilde{b}_j)$ of the received computations at iteration s with $\mathcal{S}^{[s]}$, for which $|\mathcal{S}^{[s]}| = q$. The vector parameter's update is then $\mathbf{x}^{[s+1]} = \mathbf{x}^{[s]} - \xi_s \cdot \hat{g}^{[s]}$, where

$$\hat{g}^{[s]} = \sum_{i \in \mathcal{S}^{[s]}} \nabla_{\mathbf{x}} L_{ls}(\tilde{\mathcal{D}}_j; \mathbf{x}^{[s]}) = 2 \sum_{i \in \mathcal{S}^{[s]}} \tilde{A}_i^\top \left(\tilde{A}_i \mathbf{x}^{[s]} - \tilde{b}_i\right) \quad (3.25)$$

and ξ_s is an appropriate step-size. In the case where q is not determined a priori or varies at each iteration, the scaling corresponding to $1/\sqrt{q}$ in the encoding through $\mathbf{G}$ could be done by the central server; after that iteration's computations are aggregated. We consider the case where q is the same for all iterations.

Next, we present the guarantees of our proposed GC scheme, which rely on Algorithm 5. The procedure we outlined above along with the following results, show how RandNLA can be utilized to devise efficient approximate GC schemes.

Remark 3.3.3. *By sampling $q = q(T)$ blocks at each iteration, and performing the approximate gradient update (3.25), one obtains a SSD version of the encoded linear system $\mathbf{G} \cdot (\mathbf{A}\mathbf{x}) = \mathbf{G} \cdot \mathbf{b}$. This follows from the fact that different servers are expected to respond faster at each iteration, as we assume that they are homogeneous and have the same expected response time.*

In Remark 3.3.3, the application of $\widetilde{\mathbf{\Omega}}^{[s]}$ (in Algorithm 5) has a direct correspondence to the index set $\mathcal{I}^{[s]}$ of the $q(T)$ non-stragglers of iteration s, which can be viewed as drawing $\mathcal{I}^{[s]}$ uniformly at random from $\{\mathcal{I} \subseteq \mathbb{N}_m : |\mathcal{I}| = q(T)\}$. This is what induces a first-order stochastic *iterative* sketching method for (3.2) through the proposed GC scheme, and removes the bias towards the samples which would be selected through the single $\widetilde{\mathbf{\Omega}}^{[1]}$; in the sketch-and-solve paradigm. Specifically, we do not solve the modified problem (3.10) which only accounts for a reduced dimension determined at the beginning of the iterative process, whose approximate solution is $\mathbf{x}_{\mathbf{G}}^\star \approx (\widetilde{\mathbf{S}}\mathbf{A})^\dagger(\widetilde{\mathbf{S}}\mathbf{b})$,[8] Instead, we consider a different reduced linear system

[8]Since $\widetilde{\mathbf{S}} \in \mathbb{R}^{r \times N}$ for $r < N$; we have $\widetilde{\mathbf{S}}^\dagger \widetilde{\mathbf{S}} \neq \mathbf{I}_N$, hence $(\widetilde{\mathbf{S}}\mathbf{A})^\dagger(\widetilde{\mathbf{S}}\mathbf{b}) \neq \mathbf{A}^\dagger \mathbf{b}$.

$\widetilde{\mathbf{S}}_{[s]} \cdot \left(\mathbf{A}\mathbf{x}^{[s]}\right) = \widetilde{\mathbf{S}}_{[s]} \cdot \mathbf{b}$ at each iteration. This further justifies the result of Corollary 3.3.1. This benefit of iterative sketching is validated numerically in Section 3.4.

Theorem 3.3.2. *The proposed GC scheme based on the block leverage score sketch results in a SSD procedure for $L_{ls}(\mathbf{\Psi}, \vec{\psi}; \mathbf{x})$. Furthermore, at each iteration it produces an unbiased estimator of (3.3), i.e. $\mathbb{E}[\hat{g}^{[s]}] = g^{[s]}$.*

Proof. The computations of the q fastest servers indexed by $\mathcal{I}^{[s]}$ (which corresponds to $\widetilde{\mathbf{\Omega}}^{[s]}$), are added to produce $\hat{g}^{[s]}$, and the sampling of Algorithm 5 is according to $\bar{\Pi}_{\{K\}}$. By Remark 3.3.3, it follows that each $\mathcal{I}^{[s]}$ has equal chance of occurring, which is precisely the stochastic step of SSD, *i.e.* each group of q encoded block pairs has an equal chance of being selected.

Since the servers are homogeneous and respond independently of each other, it follows that at iteration s; each $\hat{g}_i$ is received with probability $\bar{\Pi}_i$. Therefore

$$\begin{aligned}\mathbb{E}\left[\hat{g}^{[s]}\right] &= \mathbb{E}\left[\sum_{i\in\mathcal{I}^{[s]}} \hat{g}_i^{[s]}\right] = \sum_{i\in\mathcal{I}^{[s]}} \mathbb{E}\left[\hat{g}_i^{[s]}\right] = \sum_{i\in\mathcal{I}^{[s]}} \sum_{j=1}^{K} \bar{\Pi}_j \cdot \hat{g}_j^{[s]} \\ &= q \cdot \sum_{j=1}^{K} \bar{\Pi}_j \cdot \hat{g}_j^{[s]} \overset{\flat}{=} q \cdot \sum_{j=1}^{K} \bar{\Pi}_j \cdot \frac{1}{q\bar{\Pi}_j} \cdot g_j^{[s]} = \sum_{j=1}^{K} g_j^{[s]} = g^{[s]}\end{aligned}$$

where in $\flat$ we invoked (3.24). □

Lemma 3.3.3. *The optimal solution of the modified least squares problem $L_{ls}(\mathbf{\Psi}, \vec{\psi}; \mathbf{x})$, is equal to the optimal solution $\mathbf{x}^\star$ of (3.2).*

Proof. Note that the modified objective function $L_{ls}(\mathbf{\Psi}, \vec{\psi}; \mathbf{x})$ is $\|\widetilde{\mathbf{E}}\mathbf{G} \cdot (\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2$. Denote its optimal solution by $x^\star \in \mathbb{R}^d$. Further note that $\widetilde{\mathbf{E}}$ is comprised of $\tau \times \tau$ identity matrices in such a way that it is full-rank, and $\mathbf{G}$ corresponds to a rescaling of these $\mathbf{I}_\tau$ matrices, thus $\widetilde{\mathbf{E}}\mathbf{G}$ is also full-rank. It then follows that

$$x^\star = \left((\mathbf{E}\mathbf{G}) \cdot \mathbf{A}\right)^\dagger \cdot \left((\mathbf{E}\mathbf{G}) \cdot \mathbf{b}\right) = \mathbf{A}^\dagger \cdot \left((\mathbf{E}\mathbf{G})^\dagger \cdot (\mathbf{E}\mathbf{G})\right) \cdot \mathbf{b} = \mathbf{A}^\dagger \cdot \mathbf{I}_N \cdot \mathbf{b} = \mathbf{x}^\star.$$

□

The crucial aspect of our expansion network (incorporated in Theorem 3.3.2), which allowed us to use block leverage score sampling in the proposed GC scheme, is that uniform sampling of $L_{ls}(\mathbf{\Psi}, \vec{\psi}; \mathbf{x}^{[s]})$ is $\beta_{\bar{\Pi}}$-approximately equivalent to block sampling of $L_{ls}(\tilde{\mathbf{A}}, \tilde{\mathbf{b}}; \mathbf{x}^{[s]})$ according to the block leverage scores of $\mathbf{A}$. Since the two objective functions are differentiable and additively separable, the resulting gradients

are equal, under the assumption that we use the same $\mathbf{x}^{[s]}$ and sampled index set $\mathcal{S}^{[s]}$.[9] As previously mentioned, the main drawback is that in certain cases we need significantly more servers to accurately emulate $\Pi_{\{K\}}$.

The significance of Theorem 3.3.2, is that our distributed approach guarantees well-known established SD and SSD results which assume that the approximate gradient is an unbiased estimator, *e.g.* [257, Chapter 14]. Even though we are not guaranteed a descent at every iteration (*i.e.* we could have $L_{ls}(\mathcal{D};\mathbf{x}^{[s+1]}) > L_{ls}(\mathcal{D};\mathbf{x}^{[s]})$ or $\|\mathbf{x}^{[s+1]}-\mathbf{x}^{\star}\|_2^2 > \|\mathbf{x}^{[s]}-\mathbf{x}^{\star}\|_2^2$), stochastic descent methods are more common in practice when dealing with large datasets, as empirically they outperform regular SD. This is also confirmed in our experiments.

3.3.6 Convergence to $\mathbf{x}^{\star}$

Next, we give a summary of our main results thus far, and explain how together they imply convergence of our approach in expectation, to iteratively solves $L_{ls}(\mathbf{\Psi},\vec{\psi};\mathbf{x}^{[s]})$. Moreover, the contraction of our method is quantified in Appendix 2.4.

Firstly, as summarized in Remark 3.3.2, we mimic block leverage score sampling w.r. of $(\mathbf{A},\mathbf{b})$ (from $L_{ls}(\mathbf{A},\mathbf{b};\mathbf{x}^{[s]})$) through uniform sampling, by approximately solving (3.17) through the implication of (3.18) (Lemma 3.3.1). This is done implicitly by communicating computations over erasure channels. Secondly, by Theorem 3.3.1 we know that the proposed block leverage score sketching matrices satisfy (3.8); where the approximate sampling distribution $\bar{\Pi}_{\{K\}}$ is determined through the proposed expansion network associated with $\Pi_{\{K\}}$. Hence, at each iteration, we approach a solution $\hat{\mathbf{x}}^{[s]}$ of the induced sketched system $\widetilde{\mathbf{S}}_{[s]}\cdot\left(\mathbf{A}\mathbf{x}^{[s]}\right)=\widetilde{\mathbf{S}}_{[s]}\cdot\mathbf{b}$, which $\widetilde{\mathbf{S}}_{[s]}$ satisfies (3.9) with overwhelming probability. Thirdly, by Theorem 3.3.2 and Lemma 3.3.3, with a diminishing step-size ξ_s, our updates $\mathbf{x}^{[s]}$ converge to $\mathbf{x}^{\star}$ in expectation, at a rate of $O(1/\sqrt{s}+r/s)$ [38, 79]. A synopsis is given below:

$$\left\{\begin{array}{c}L_{\mathbf{S}}(\widetilde{\mathbf{S}}_{[s]},\mathbf{A},\mathbf{b};\mathbf{x})\text{ sol'ns}\\ \text{satisfy (3.8) and (3.9)}\end{array}\right\}\xleftarrow{\text{3.3.1, 3.3.2}}\left\{\begin{array}{c}\text{Solve } L_{ls}(\mathbf{\Psi},\vec{\psi};\mathbf{x}^{[s]})\\ \text{through 'sketched-GC'}\end{array}\right\}\xrightarrow{\text{3.3.2, 3.3.3}}\left\{\begin{array}{c}\text{With a diminishing } \xi_s\text{:}\\ \lim\mathbb{E}[\mathbf{x}^{[s]}]\to\mathbf{x}^{\star}\end{array}\right\}.$$

3.3.7 Approximate GC from ℓ_2-s.e.

In conventional GC, the objective is to construct an encoding matrix $\mathbf{G}\in\mathbb{R}^{m\times K}$ and decoding vectors $\mathbf{a}_{\mathcal{I}}\in\mathbb{R}^{1\times q}$, such that $\mathbf{a}_{\mathcal{I}}\mathbf{G}_{(\mathcal{I})}=\vec{\mathbf{1}}$ for any set of non-straggling

[9] The index set of the sampled blocks from $L_{ls}(\mathbf{\Psi},\vec{\psi};\mathbf{x}^{[s]})$, corresponds to an index *multiset* of the sampled blocks from $L_{ls}(\tilde{\mathbf{A}},\tilde{\mathbf{b}};\mathbf{x}^{[s]})$, as in the latter we are considering sampling *with replacement*.

servers $\mathcal{I}$. It follows that the optimal decoding vector for a set $\mathcal{I}$ of size q in *approximate* GC [59, 155, 253] is

$$\mathbf{a}_{\mathcal{I}}^{\star} = \arg\min_{\mathbf{a}\in\mathbb{R}^{1\times q}} \left\{ \|\mathbf{a}\mathbf{G}_{(\mathcal{I})} - \vec{\mathbf{1}}\|_2^2 \right\} \qquad \Longrightarrow \qquad \mathbf{a}_{\mathcal{I}}^{\star} = \vec{\mathbf{1}}\mathbf{G}_{(\mathcal{I})}^{\dagger} \tag{3.26}$$

for $\mathbf{G}_{(\mathcal{I})}^{\dagger} = \left(\mathbf{G}_{(\mathcal{I})}^{\top}\mathbf{G}_{(\mathcal{I})}\right)^{-1}\mathbf{G}_{(\mathcal{I})}^{\top} \in \mathbb{R}^{K\times q}$.

Proposition 3.3.3. *The error in the approximated gradient $\grave{g}^{[s]}$ of an optimal approximate linear regression GC scheme $(\mathbf{G}, \mathbf{a}_{\mathcal{I}}^{\star})$, satisfies*

$$\left\|g^{[s]} - \grave{g}^{[s]}\right\|_2 \leqslant 2\sqrt{K} \cdot \mathrm{err}(\mathbf{G}_{(\mathcal{I})}) \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2, \tag{3.27}$$

for $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) \coloneqq \left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2$.

Proof. Consider the optimal decoding vector of an approximate GC scheme $\mathbf{a}_{\mathcal{I}}^{\star}$ (3.27). In the case where $q \geqslant K$, it follows that $\mathbf{a}_{\mathcal{I}}^{\star} = \vec{\mathbf{1}}\mathbf{G}_{(\mathcal{I})}^{\dagger}$.

Let $\mathbf{g}^{[s]}$ be the matrix comprised of the transposed exact partial gradients at iteration s, *i.e.*

$$\mathbf{g}^{[s]} \coloneqq \begin{pmatrix} g_1^{[s]} & g_2^{[s]} & \dots & g_K^{[s]} \end{pmatrix}^{\top} \in \mathbb{R}^{K\times d}\ .$$

Then, for a GC encoding-decoding pair $(\mathbf{G}, \mathbf{a}_{\mathcal{I}})$ satisfying $\mathbf{a}_{\mathcal{I}}\mathbf{G}_{(\mathcal{I})} = \vec{\mathbf{1}}$ for any $\mathcal{I}$, it follows that

$$\mathbf{a}_{\mathcal{I}}\left(\mathbf{G}_{(\mathcal{I})}\mathbf{g}^{[s]}\right) = \vec{\mathbf{1}}\mathbf{g}^{[s]} = \sum_{j=1}^{K}\left(g_j^{[s]}\right)^{\top} = \left(g^{[s]}\right)^{\top}\ .$$

Hence, the gradient can be recovered exactly. Considering an optimal approximate scheme $(\mathbf{G}, \mathbf{a}_{\mathcal{I}}^{\star})$ which recovers the gradient estimate $\grave{g}^{[s]} = \left(\mathbf{a}_{\mathcal{I}}^{\star}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}$, the error in the gradient approximation is

$$\begin{aligned}
\left\|g^{[s]} - \grave{g}^{[s]}\right\|_2 &= \left\|\left(\vec{\mathbf{1}} - \mathbf{a}_{\mathcal{I}}^{\star}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}\right\|_2 \\
&= \left\|\vec{\mathbf{1}}\left(\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}\right\|_2 \\
&\leqslant \|\vec{\mathbf{1}}\|_2 \cdot \left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2 \cdot \left\|\mathbf{g}^{[s]}\right\|_2 \\
&\overset{£}{\leqslant} \sqrt{K} \cdot \left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2 \cdot \left\|g^{[s]}\right\|_2 \\
&\overset{\$}{\leqslant} 2\sqrt{K} \cdot \underbrace{\left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2}_{\mathrm{err}(\mathbf{G}_{(\mathcal{I})})} \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2
\end{aligned}$$

where £ follows from the facts that $\|\mathbf{g}^{[s]}\|_2 \leqslant \|g^{[s]}\|_2$ and $\|\vec{\mathbf{1}}\|_2 = \sqrt{K}$, and \$ from

(3.3) and sub-multiplicativity of matrix norms. □

In Theorem 3.3.3, we show the accuracy of the approximate gradient of iterative GC approaches based on sketching techniques that satisfy the ℓ_2-s.e. property $\|\mathbf{I}_d - \mathbf{U}^\top\mathbf{S}^\top\mathbf{S}\mathbf{U}\|_2 \leqslant \epsilon$ from (3.8) (w.h.p.). This then holds true for our approach through expansion networks, by Theorem 3.3.1 and Remark 3.3.2.

Theorem 3.3.3. *Assume that the induced sketching matrix* $\mathbf{S}$ *from a GC scheme satisfies* $\|\mathbf{I}_d - \mathbf{U}^\top\mathbf{S}^\top\mathbf{S}\mathbf{U}\|_2 \leqslant \epsilon$ *(w.h.p.). Then, the updated approximate gradient estimate* $\hat{g}^{[s]}$ *at any iteration, satisfies (w.h.p.):*

$$\left\|g^{[s]} - \hat{g}^{[s]}\right\|_2 \leqslant 2\epsilon \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2. \tag{3.28}$$

Specifically, it satisfies (3.27) *with* $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) = \epsilon/\sqrt{K}$.

Proof. Now, consider $\hat{g}^{[s]}$ the approximated gradient of our scheme for linear regression with gradient (3.3). It follows that

$$\begin{aligned}
\left\|g^{[s]} - \hat{g}^{[s]}\right\|_2 &= \|2\mathbf{A}^\top(\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}) - 2\mathbf{A}^\top(\mathbf{S}^\top\mathbf{S})(\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b})\|_2 \\
&= 2\|\mathbf{A}^\top(\mathbf{I}_N - \mathbf{S}^\top\mathbf{S})(\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b})\|_2 \\
&\leqslant 2\|\mathbf{A}\|_2 \cdot \|\mathbf{I}_N - \mathbf{S}^\top\mathbf{S}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2 \\
&= 2\|\mathbf{A}\|_2 \cdot \|\mathbf{U}^\top(\mathbf{I}_N - \mathbf{S}^\top\mathbf{S})\mathbf{U}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2 \\
&= 2\|\mathbf{A}\|_2 \cdot \|\mathbf{I}_d - \mathbf{U}^\top\mathbf{S}^\top\mathbf{S}\mathbf{U}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2 \\
&\overset{\flat}{\leqslant} 2\epsilon \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2
\end{aligned}$$

where in $\flat$ we make use of the assumption that $\mathbf{S}$ satisfies (3.8). Our approximate GC approach therefore (w.h.p.) satisfies (3.27), with $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) = \epsilon/\sqrt{K}$. □

3.4 Experiments

In this section, we corroborate our theoretical results, and show their benefits on fabricated datasets. The minimum benefit of Algorithm 5 occurs when $\Pi_{\{K\}}$ is close to uniform. For this reason, and the fact that our expansion approach depends on the implicit distribution through $r_{\{K\}}$, we construct dataset matrices whose resulting sampling distributions and block leverage scores are non-uniform.

For the first experiment, we considered $\mathbf{A} \in \mathbb{R}^{2000\times 40}$ following a t-distribution, and standard Gaussian noise was added to an arbitrary vector from $\mathrm{im}(\mathbf{A})$ to define

b. We considered $K = 100$ blocks, thus $\tau = 20$. The effective dimension $N = 2000$ was reduced to $r = 1000$, *i.e.* $q = 20$. We compared the iterative approach with exact block leverage scores (*i.e.* $\beta = 1$), against analogous approaches using the block-SRHT and Gaussian sketches, and uncoded regular SD.

In Figure III.4 we ran 600 iterations on six different instances for each approach, and varied ξ for each experiment by logarithmic factors of $\xi^{\times} = 2/\sigma_{\max}(\mathbf{A})^2$. The average log residual errors $\log_{10}\left(\|\mathbf{x}_{ls}^{\star} - \hat{\mathbf{x}}\|_2/\sqrt{N}\right)$ are depicted in Figure III.4, and reported in Table III.1. In Figure III.5 we observe the convergence of the different approaches, in the case where $\xi \approx 0.42$. In this case, our method (block-lvg) outperforms the Gaussian sketching approach and regular SD. The fact that the performance of the block-SRHT is similar to our proposed algorithm, reflects the result of Proposition 2.3.1.

Figure III.4: residual error for varying ξ_s

Average log **residual error** $\log_{10}\left(\|\mathbf{x}_{ls}^{\star} - \hat{\mathbf{x}}\|_2/\sqrt{N}\right)$				
$\log_{10}(\xi)$	0.0004	0.0042	0.0421	0.4207
Regular SD	0.0566	0.0517	0.0440	0.0078
Gaussian	0.0590	0.0538	0.0416	-0.0114
block-SRHT	0.0603	0.0550	0.0431	-0.0110
block-lvg	0.0556	0.0502	0.0380	-0.0178

Table III.1: Average log residual errors, for six instances of SD with fixed steps, when performing Gaussian sketching with updated sketches, iterative block-SRHT and iterative block leverage score sketching, and uncoded SD.

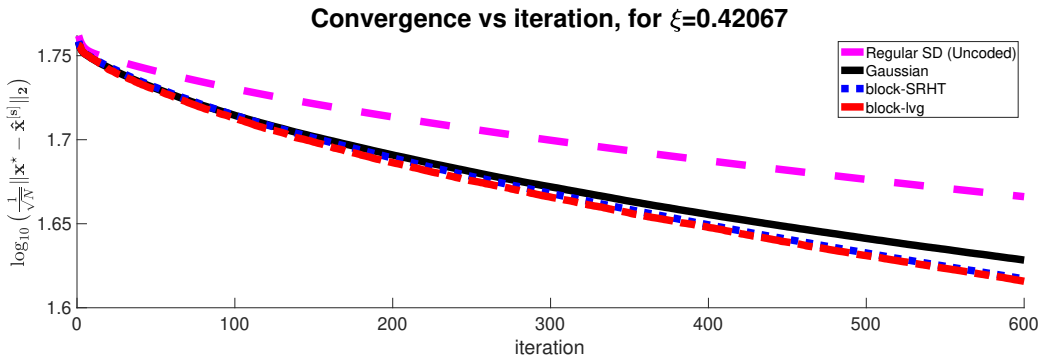


Figure III.5: log residual error convergence

We also considered the same experiment with $\mathbf{A}$ drawn from a t-distribution, with and optimal step-size $\xi_s^\star = \langle \mathbf{A}g^{[s]}, \mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\rangle / \|\mathbf{A}g^{[s]}\|_2^2$ at each iteration. From Figure III.6, we observe that our iterative sketching approach outperforms Gaussian sketching with updated sketches; and iterative sketching is superior to non-iterative. Furthermore, we validate Lemma 3.3.2 and Theorem 3.3.3, as our iterative sketching approach and SSD have similar convergence. Furthermore, it was observed that in some case cases when our iterative sketching method would outperform regular SD (and SSD). We also compared our method to iterative and non-iterative approaches according to the block leverage score sampling, block-SRHT, and Rademacher sketching methods, in which our corresponding approach again produced more accurate final approximations.

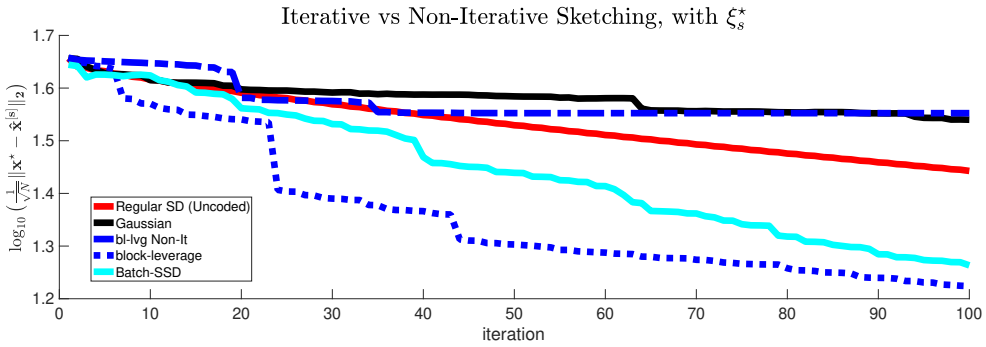


Figure III.6: log convergence with $\xi_s^\star$

3.5 Conclusion and Future Work

In this chapter, we showed how one can exploit results from RandNLA to distributed CC, in the context of GC. By taking enough samples, or equivalently; wait-

ing long enough, the approximation errors can be made arbitrarily small. In terms of CC, the advantages are that *encodings* correspond to a scalar multiplication, and *no* decoding step is required. By utilizing these techniques, we are also advantageous over other CC approximation schemes [25, 52, 53, 58, 59, 112, 120, 144, 155, 239, 253]; by incorporating information from our dataset into our scheme.

Our methods were validated numerically through various experiments, presented in Section 3.4. Further experiments were performed on various distributions for $\mathbf{A}$, in which similar results were obtained. We also considered the empirical distribution from real-server completion times taken from 500 AWS-servers [19], and emulated the proposed CC scheme. In this experiment, we obtained the expected results in terms of ℓ_2-s.e., misestimation factors $\beta_{\Pi,\bar{\Pi}}$, and metrics $\Delta_{\Pi,\bar{\Pi}}$, $d_{\Pi,\bar{\Pi}}$.

Even though we focused on leverage score sampling for linear regression, other sampling algorithms and problems could benefit by designing analogous replication schemes. One such problem is the *column subset selection problem*, which can be used to compute partial SVD, QR decompositions, as well as low-rank approximations [201]. As for the sampling technique we studied, one can judiciously define a sampling distribution to approximate solutions to such problems [34], which are known to be NP-hard under the Unique Games Conjecture assumption [66].

Furthermore, existing block sampling algorithms can also benefit from the proposed expansion networks, *e.g.* CR-multiplication [53] and CUR decomposition [221]. For instance, a coded matrix multiplication algorithm of minimum variance can been designed, where the sampling distribution proposed in [53] is used to determine the replication numbers of the expansion network. In terms of matrix decompositions, the block leverage score algorithm of [221] can be used to distributively determine an additive ϵ-error decomposition of $\mathbf{A}$, in the CC setting. Another future direction is generalizing existing tensor product and factorization algorithms to block sampling, according to both approximate and exact sampling distributions, in order to make them practical for distributed environments.

CHAPTER IV

Iterative Sketching for Secure Coded Regression

4.1 Introduction

Random projections[1] are a classical way of performing dimensionality reduction, and are widely used in algorithmic and learning contexts [93, 96, 284, 294]. Distributed computations in the presence of *stragglers* have gained a lot of attention in the information theory community. Coding-theoretic approaches have been adopted for this [43, 48, 51, 53, 55, 97, 166, 178, 179, 183, 185, 222, 224, 237, 245, 250, 251, 302], and fall under the framework of *coded computing* (CC). Data security is also an increasingly important issue in CC [181]. In this work, we propose methods to securely speed up linear regression by simultaneously leveraging random projections and sketching, and distributed computations. Our results are presented in terms of the system model proposed in [178], though they extend to any centralized distributed model, *i.e.* distributed systems with a central server which updates and communicates the parameters to the computational worker nodes. Furthermore, the desiderata of our approaches are to:

(I) produce straggler-resilient accurate approximations,

(II) secure the information,

(III) carry out the computations efficiently and distributively.

We focus on sketching for *steepest descent* (SD) in the context of solving overdetermined linear systems. Part of our theoretical results are given for the sketch-and-solve paradigm [255], which can be utilized by a single server; who can also store a compressed version of the data. The application through CC results in iterative sketching

[1] By '*projections*' we refer to random matrices, not idempotent matrices.

methods for distributively solving overdetermined linear systems. We propose applying a random orthonormal projection to the linear system before distributing the data, and then performing SD distributively on the transformed system through *approximate gradient coding*. Under the straggler scenario and the assumptions we make, this results in a *mini-batch stochastic steepest descent* (SSD) procedure of the original system. A special case of such a projection is the *Subsampled Randomized Hadamard Transform* (SRHT) [33, 96, 282], which utilizes Kronecker products of the Hadamard transform; and relates to the *fast Johnson-Lindenstrauss transform* [5, 153].

The benefit of applying an orthonormal matrix transformation is that we rotate and/or reflect the data's orthonormal basis, which *cannot* be reversed without knowledge of the transformation. This is leveraged to give security guarantees, while simultaneously ensuring that we recover well-approximated gradients, and an approximate solution of the linear system. Such sketching matrices are also referred to as *randomized orthonormal systems* [20]. We also discuss how one can use recursive Kronecker products of an orthonormal matrix of dimension greater than 2 in place of the Hadamard transform, to obtain the a more efficiency encoding and encryption than through a random and unstructured orthonormal matrix.

In the CC paradigm, the workers are assumed to be *homogeneous* with the same expected response time. In the proposed methods, we stop receiving computations once a fixed fraction of the workers respond; producing a different induced sketch at each iteration. A predominant task which has been studied in the CC framework is the gradient computation of differentiable and additively separable objective functions [25, 42, 47, 52, 58, 59, 62, 134, 144, 155, 223, 239, 279, 289, 292, 298]. These schemes are collectively called *gradient coding* (GC). We note that iterative sketching has proven to be a powerful tool for second-order methods [173, 231], though it has not been explored in first-order methods. Since we indirectly consider a different underlying modified problem at each iteration, the methods we propose are *approximate* GC schemes (GCSs). Related approaches have been proposed in [25, 52, 58, 59, 62, 155, 144, 239, 289, 292]. Two important benefits of our approaches are that we do not require a decoding step, nor an encoding step by the workers; at each iteration, avoiding prevalent bottlenecks.

An advantage of using an updated sketch at each iteration, is that we do not have a bias towards the samples that would be selected/returned when the sketching takes place, which occurs in the sketch-and-solve approach. Specifically, we do not solve a modified problem which only accounts for a reduced dimension; determined at the beginning of the iterative process. Instead, we consider a different reduced system at

each iteration. This is also justified numerically.

Another benefit of our approach, is that random projections secure the information from potential eavesdroppers, honest but curious; and colluding workers. We show information-theoretic security for the case where a random orthonormal projection is utilized in our sketching algorithm. Furthermore, the security of the SRHT, which is a crucial aspect, has not been extensively studied. Unfortunately, the SRHT is inherently insecure, which we show. We propose a modified projection which guarantees computational security of the SRHT.

There are related works to what we study. The work of [20] focuses on parameter averaging for variance reduction, but only mentions a security guarantee for the Gaussian sketch, derived in [306]. Another line of work [159, 160], focuses on introducing redundancy through equiangular tight frames (ETFs), partitioning the system into smaller linear systems, and then averaging the solutions of a fraction of them. A drawback of using ETFs, is that most of them are over $\mathbb{C}$. The authors of [263] study privacy of random projections, though make the assumption that the projections meet the 'ε-MI-DP constraint'. Recently, the authors of [192] considered CC privacy guarantees through the lens of differential privacy, with a focus on matrix multiplication. Lastly, a secure GCS is studied in [300], though it does not utilize sketching. We also clarify that even though we guarantee cryptographic security, our methods may still be vulnerable to various privacy attacks, *e.g.* membership inference attacks [114] and model inversion attacks [262]. This is another interesting line of work, though is not a focus of our approach.

The chapter is organized as follows. In 4.2 we review the framework and background for coded linear regression, the notions of security we will be working with, the ℓ_2-subspace embedding property, and list the main properties we seek to satisfy through our constructions; in order to meet the aforementioned desiderata. In 4.3 we present the proposed iterative sketching algorithm, and in 4.4 the special case where the projection is the randomized Hadamard transform; which we refer to as the "*block-SRHT*". The subspace embedding results for the general algorithm and the block-SRHT are presented in the respective sections. We consider the case where the central server may adaptively change the step-size of its SD procedure in 4.5, which can be viewed as an *adaptive GCS*. In 4.6 we present the security guarantees of our algorithm and the modified version of the block-SRHT. Finally, we present numerical experiments in 4.7; and concluding remarks in 4.8.

4.2 Coded Linear Regression

4.2.1 Least Squares Approximation and Steepest Descent

In linear least squares approximation [96], it is desired to approximate the solution

$$\mathbf{x}_{ls}^{\star} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ L_{ls}(\mathbf{A},\mathbf{b};\mathbf{x}) \coloneqq \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 \right\} \tag{4.1}$$

where $\mathbf{A}\in\mathbb{R}^{N\times d}$ and $\mathbf{b}\in\mathbb{R}^N$. This corresponds to the regression coefficients $\mathbf{x}$ of the model $\mathbf{b}=\mathbf{A}\mathbf{x}+\vec{\varepsilon}$, which is determined by the dataset $\mathcal{D}=\{(\mathbf{a}_i,b_i)\}_{i=1}^N \subsetneq \mathbb{R}^d\times\mathbb{R}$ of N samples, where $(\mathbf{a}_i,b_i)$ represent the features and label of the i^{th} sample, *i.e.* $\mathbf{A}=\begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_N\end{bmatrix}^T$ and $\mathbf{b}=\begin{bmatrix}b_1 & \cdots & b_N\end{bmatrix}^T$.

To simplify our presentation, we first define some notational conventions. Row vectors of a matrix $\mathbf{M}$ are denoted by $\mathbf{M}_{(i)}$, and column vectors by $\mathbf{M}^{(j)}$. Our embedding results are presented in terms of an arbitrary partition $\mathbb{N}_N=\bigsqcup_{\iota=1}^K \mathcal{K}_\iota$, for $\mathbb{N}_N\coloneqq\{1,\cdots,N\}$ the index set of $\mathbf{M}$'s rows. Each $\mathcal{K}_\iota$ corresponds to a *block* of $\mathbf{M}$. The notation $\mathbf{M}_{(\mathcal{K}_\iota)}$ denotes the submatrix of $\mathbf{M}$ comprised of the rows indexed by $\mathcal{K}_\iota$. That is: $\mathbf{M}_{(\mathcal{K}_\iota)}=\mathbf{I}_{(\mathcal{K}_\iota)}\mathbf{M}$, for $\mathbf{I}_{(\mathcal{K}_\iota)}$ the corresponding submatrix of $\mathbf{I}_N$ of size $|\mathcal{K}_\iota|\times N$. We call $\mathbf{M}_{(\mathcal{K}_\iota)}$ the 'ι^{th} *block of* $\mathbf{M}$'. We abbreviate $(1-\epsilon)\cdot\|\vec{b}\|\leqslant\|\vec{a}\|\leqslant(1+\epsilon)\cdot\|\vec{b}\|$, to $\|\vec{a}\|\leqslant_\epsilon\|\vec{b}\|$. Lastly, '$\leftarrow$' denotes a numerical assignment of a varying quantity, and '$\xleftarrow{U}$' a realization of a random variable through uniform sampling.

By $\mathbf{\Pi}$ we denote the random orthonormal matrix we apply to the data matrix $\mathbf{A}$; which is drawn uniformly at random from a finite subset $\tilde{O}_{\mathbf{A}}$ of the set orthonormal matrices $O_N(\mathbb{R})$, *i.e.* $\mathbf{\Pi}\xleftarrow{U}\tilde{O}_{\mathbf{A}}\subsetneq O_N(\mathbb{R})$. By $\hat{\mathbf{\Pi}}$ and $\tilde{\mathbf{\Pi}}$ we denote the special cases where $\mathbf{\Pi}$ is a orthonormal matrix used for the *block-SRHT* and *garbled block-SRHT* respectively.

We address the overdetermined case where $N\gg d$. Existing exact methods find a solution vector $\mathbf{x}_{ls}^{\star}$ in $\mathcal{O}(Nd^2)$ time, where $\mathbf{x}_{ls}^{\star}=\mathbf{A}^{\dagger}\mathbf{b}$. A common way to approximate $\mathbf{x}_{ls}^{\star}$ is through SD, which iteratively updates the gradient

$$g_{ls}^{[t]} \coloneqq \nabla_{\mathbf{x}} L_{ls}(\mathbf{A},\mathbf{b};\mathbf{x}^{[t]}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{x}^{[t]}-\mathbf{b}) \tag{4.2}$$

followed by updating the parameter vector

$$\mathbf{x}^{[t+1]} \leftarrow \mathbf{x}^{[t]} - \xi_t\cdot g_{ls}^{[t]}\ . \tag{4.3}$$

In our setting, the step-size $\xi_t>0$ is determined by the central server. The script $[t]$ indexes the iteration $t=0,1,2,\ldots$ which we drop when clear from the context. In 4.5,

we derive the optimal step-size $\xi_t^\star$ for (4.2) and the modified problems we consider, given the updated gradients and parameters of iteration t.

4.2.2 The Straggler Problem and Gradient Coding

Gradient coding is deployed in centralized computation networks, *i.e.* a central server communicates $\mathbf{x}^{[t]}$ to m workers; who perform computations and then communicate back their results. The central server distributes the dataset $\mathcal{D}$ among the m workers, to facilitate the solution of optimization problems with additively separable and differentiable objective functions. For linear regression (4.1), the data is partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^T & \cdots & \mathbf{A}_K^T \end{bmatrix}^T \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1^T & \cdots & \mathbf{b}_K^T \end{bmatrix}^T \tag{4.4}$$

where $\mathbf{A}_i \in \mathbb{R}^{\tau \times d}$ and $\mathbf{b}_i \in \mathbb{R}^\tau$ for all i, and $\tau = N/K$. For ease of exposition, we assume that $K|N$. Then we have $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}) = \sum_{i=1}^K L_{ls}(\mathbf{A}_i, \mathbf{b}_i; \mathbf{x})$. A regularizer $\mu R(\mathbf{x})$ can also be added to $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x})$ if desired.

In GC [279], the workers encode their computed *partial gradients*

$$g_i \coloneqq \nabla_{\mathbf{x}} L_{ls}(\mathbf{A}_i, \mathbf{b}_i; \mathbf{x})$$

which are then communicated to the central server. Once a certain fraction of encodings is received, the central server applies a decoding step to recover the gradient $g = \nabla_{\mathbf{x}} L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}) = \sum_{i=1}^K g_i$. This can be computationally prohibitive, and takes place at every iteration. To the best of our knowledge, the lowest decoding complexity is $\mathcal{O}\left((s+1) \cdot \lceil \frac{m}{s+1} \rceil\right)$; where s is the number of stragglers [47].

In our approach we trade time; by not requiring encoding nor decoding steps at each iteration, with accuracy of approximating $\mathbf{x}_{ls}^\star$. Unlike conventional GCSs, in this chapter the workers carry out the computation on the encoded data. The resulting gradient, is that of the modified least squares problem

$$\hat{\mathbf{x}}_{ls} = \arg\min_{\mathbf{x} \in \mathbb{R}^d} \left\{ L_{\mathbf{S}^{[t]}}(\mathbf{A}, \mathbf{b}; \mathbf{x}) \coloneqq \|\mathbf{S}^{[t]}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2 \right\} \tag{4.5}$$

for $\mathbf{S}^{[t]} \in \mathbb{R}^{r \times N}$ a sketching matrix, with $r \ll N$ and $r > d$. This is the core idea behind our approximation, where we incorporate iterative sketching with orthonormal matrices and random sampling; and generalizations of the SRHT for $\mathbf{S}^{[t]}$, for our approximate GCSs. The sketching approach we take is to first apply a random pro-

jection, which also provides security against the workers and eavesdroppers, and then sample computations carried out on the blocks of the transformed data uniformly at random; which corresponds to the responses of the homogeneous non-stragglers.

For q the total number of responsive workers, we can mitigate up to $s = m - q$ stragglers. Specifically, the number of responsive workers $m - s$ in the CC model, corresponds to the number of sampling trials q of our sketching algorithm, *i.e.* $q = m - s$. At iteration t, a SD update of the modified least squares problem (4.5) is obtained distributively. Furthermore, we assume that the data is partitioned into as many blocks as there are workers, *i.e.* $K = m$. The stragglers are assumed to be uniformly random and may differ at each iteration. Thus, there is a different sketching matrix $\mathbf{S}^{[t]}$ at each epoch.

In conventional GCSs the objective is to construct an encoding matrix $\mathbf{G} \in \mathbb{R}^{m \times K}$ (can have $m \neq K$) and decoding vectors $\mathbf{a}_{\mathcal{I}} \in \mathbb{R}^{1 \times q}$, such that $\mathbf{a}_{\mathcal{I}} \mathbf{G}_{(\mathcal{I})} = \vec{\mathbf{1}}$ for any set of non-straggling workers $\mathcal{I}$. Furthermore, it is assumed that multiple replications of each encoded block is shared among the workers, such that $m \gtrsim q \geqslant K$.[2] From the fact that $\mathbf{a}_{\mathcal{I}} \mathbf{G}_{(\mathcal{I})} = \vec{\mathbf{1}}$ for any $\mathcal{I}$, in *approximate* GC [59], the optimal decoding vector for a set $\mathcal{I}$ of size $q = m - s$ is determined by

$$\mathbf{a}_{\mathcal{I}}^{\star} = \arg \min_{\mathbf{a} \in \mathbb{R}^{1 \times q}} \left\{ \|\mathbf{a}\mathbf{G}_{(\mathcal{I})} - \vec{\mathbf{1}}\|_2^2 \right\} \qquad \Longrightarrow \qquad \mathbf{a}_{\mathcal{I}}^{\star} = \vec{\mathbf{1}} \mathbf{G}_{(\mathcal{I})}^{\dagger} \ ,$$

for $\mathbf{G}_{(\mathcal{I})}^{\dagger}$ the pseudoinverse of $\mathbf{G}_{(\mathcal{I})}$. The error in the approximated gradient $\grave{g}^{[t]}$ of an optimal approximate linear regression GCS $(\mathbf{G}, \mathbf{a}_{\mathcal{I}}^{\star})$, is then

$$\left\| g^{[t]} - \grave{g}^{[t]} \right\|_2 \leqslant 2\sqrt{K} \cdot \mathrm{err}(\mathbf{G}_{(\mathcal{I})}) \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\|_2 \ , \tag{4.6}$$

for $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) \coloneqq \left\| \mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger} \mathbf{G}_{(\mathcal{I})} \right\|_2$.

4.2.3 Secure Coded Computing Schemes

Modern cryptography is split into two main categories, *information-theoretic security* and *computational security*. The former is also referred to as *Shannon secrecy*, while the latter is also referred to as *asymptotic security*. In this subsection, we give the definitions which will allow us to characterize the security level of our GCSs.

[2] As we mention in 4.3.1, this can be done in order to mimic sampling *with replacement* through the CC network. The reason we require $q \geqslant K$, is to define $\mathbf{a}_{\mathcal{I}}^{\star} = \vec{\mathbf{1}} \mathbf{G}_{(\mathcal{I})}^{\dagger}$. This idea has been extensively studied in [56].

Definition 4.2.1. *A **secure CC** scheme, is the pair of encoding and decoding algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *of the CC scheme, such that* $\mathsf{Enc}(\mathbf{A})$ *also guarantees a level of security of* $\mathbf{A}$*, and* Dec *recovers the hidden information; i.e.* $\mathsf{Dec}(\mathsf{Enc}(\mathbf{A})) = \mathbf{A}$.

In our work, Enc corresponds to a linear transformation through a randomly selected orthonormal matrix $\mathbf{\Pi}$. The orthogonal group in dimension N, is denoted by $O_N(\mathbb{R})$. By encryption, we refer to this linear transformation, which is utilized in our GCS. Furthermore, we do not require a decryption step by the central server, as it computes an unbiased estimate of the gradient at the end of each iteration. Also, since $\mathbf{\Pi}^T = \mathbf{\Pi}^{-1}$, it follows that $\mathbf{\Pi}^T$ meets the requirement of Dec, so in the following definition we refer to the encoding-decoding pair by only referencing Enc. Furthermore, Enc depends on a secret key k which is randomly generated. In our case, this is simply $\mathbf{\Pi}$.

Definition 4.2.2 (Ch.2 [161])**.** *An encryption scheme* Enc *with message, ciphertext and key spaces* $\mathcal{M}$*,* $\mathcal{C}$ *and* $\mathcal{K}$ *respectively is **Shannon secret** w.r.t. a probability distribution* D *over* $\mathcal{M}$*, if for all* $\bar{m} \in \mathcal{M}$ *and all* $\bar{c} \in \mathcal{C}$*:*

$$\Pr_{\substack{m\sim D \\ k\xleftarrow{U}\mathcal{K}}}[m = \bar{m} \mid \mathsf{Enc}_k(m) = \bar{c}] = \Pr_{m\sim D}[m = \bar{m}] \ . \tag{4.7}$$

*An equivalent condition is **perfect secrecy**, which states that for all* $m_0, m_1 \in \mathcal{M}$*:*

$$\Pr_{k\xleftarrow{U}\mathcal{K}}[\mathsf{Enc}_k(m_0) = \bar{c}] = \Pr_{k\xleftarrow{U}\mathcal{K}}[\mathsf{Enc}_k(m_1) = \bar{c}] \ . \tag{4.8}$$

Definition 4.2.3 (Ch.3 [161])**.** *An encryption scheme is **computationally secure** if any probabilistic polynomial-time adversary succeeds in breaking it, with at most negligible probability. By negligible, we mean it is asymptotically smaller than any inverse polynomial function.*

4.2.4 The ℓ_2-subspace embedding Property

For the analysis of the sketching matrices $\mathbf{S}_{\mathbf{\Pi}}$ we propose in Algorithm 7, we consider any orthonormal basis $\mathbf{U} \in \mathbb{R}^{N\times d}$ of the column-space of $\mathbf{A}$, *i.e.* $\mathrm{im}(\mathbf{A}) = \mathrm{im}(\mathbf{U})$. The subscript of $\mathbf{S}_{\mathbf{\Pi}}$, indicates the dependence of the sketching matrix on $\mathbf{\Pi}$.

Recall that the ℓ_2*-subspace embedding* (ℓ_2-s.e.) property [294, 106] states that any $\mathbf{y} \in \mathrm{im}(\mathbf{U})$ satisfies:

$$\|\mathbf{S}_{\mathbf{\Pi}}\mathbf{y}\|_2 \leqslant_\epsilon \|\mathbf{y}\|_2 \iff \|\mathbf{I}_d - (\mathbf{S}_{\mathbf{\Pi}}\mathbf{U})^T(\mathbf{S}_{\mathbf{\Pi}}\mathbf{U})\|_2 \leqslant \epsilon \tag{4.9}$$

for $\epsilon > 0$. In turn, this characterizes the approximation's error of the solution $\hat{\mathbf{x}}_{ls}$ of (4.5) for $\mathbf{S} \leftarrow \mathbf{S}_{\mathbf{\Pi}}$, as

$$\|\mathbf{A}\hat{\mathbf{x}}_{ls} - \mathbf{b}\|_2 \leqslant \frac{1+\epsilon}{1-\epsilon}\|\mathbf{A}\mathbf{x}^{\star}_{ls} - \mathbf{b}\|_2 \leqslant (1 + \mathcal{O}(\epsilon))\|\mathbf{A}\mathbf{x}^{\star}_{ls} - \mathbf{b}\|_2$$

with high probability, and $\|\mathbf{A}(\mathbf{x}^{\star}_{ls} - \hat{\mathbf{x}}_{ls})\|_2 \leqslant \epsilon\|(\mathbf{I}_N - \mathbf{U}\mathbf{U}^T)\mathbf{b}\|_2$.

4.2.5 Properties of our Approach

A key property in the construction of our sketching matrices, is to sample *blocks* (*i.e.* submatrices) of a transformation of the data matrix, which permits us to then perform the computations in parallel. The additional properties we seek to satisfy with our GCSs through block sampling are the following:

(a) the underlying sketching matrix satisfies the ℓ_2-s.e. property,

(b) the block leverage scores are flattened through the random projection $\mathbf{\Pi}$,

(c) the projection is over $\mathbb{R}$,

(d) the central server computes an unbiased gradient estimate at each iteration,

(e) do not require encoding/decoding at each iteration,

(f) guarantee security of the information from the workers and potential eavesdroppers,

(g) $\mathbf{\Pi}$ can be applied efficiently, *i.e.* in $\mathcal{O}(Nd \log N)$ operations.

The seven properties listed above, are grouped together with respect to the desiderata mentioned in 4.1. Specifically, desideratum (I) encompasses properties (a), (b), (c), (d), desideratum (II) corresponds to (f), and (III) encompasses (b), (c), (e), (g).

Property (a) is motivated by the sketch-and-solve approach, though through the iterative process, in practice we benefit by having fresh sketches. Leverage scores define the key structural non-uniformity that must be dealt with in developing fast randomized matrix algorithms; and are formally defined in 4.3.2. If property (b) is met, we can then sample uniformly at random in order to guarantee (a). We require $\mathbf{\Pi}$ to be over $\mathbb{R}$, as if it were over $\mathbb{C}$, the communication cost from the central server to the workers; and the necessary storage space at the workers would double. Additionally, performing computations over $\mathbb{C}$ would result in further round-off errors and numerical instability. Properties (d) and (e) are met by requiring $\mathbf{\Pi}$ to be an

orthonormal matrix. By allowing the projection to be random; we can secure the data, *i.e.* satisfy (f). Furthermore, the action of applying an orthonormal projection for our encryption; is reversed through the computation of the partial gradients, hence no decryption step is required.

By considering a larger ensemble of orthonormal projections to select from, we can give stronger security guarantees. Specifically, by not restricting the structure of $\mathbf{\Pi}$, we can guarantee Shannon secrecy, though this prevents us from satisfying (g). On the other hand, if we let $\mathbf{\Pi}$ be structured, we can satisfy (g) at the cost of only guaranteeing computational security.

We point out that even though Gaussian and random Rademacher sketches satisfy satisfy (a), (b), (c) and (f), they do not satisfy (d), (e) nor (g) in our CC setting. Experimentally, we observe that our proposed sketching matrices outperform the Gaussian and random Rademacher sketches, primarily due to the fact that (d) is satisfied. Furthermore, for $\mathbf{\Pi} \in O_N(\mathbb{R})$, our distributive procedure results in a SSD approach.

4.3 Block Subsampled Orthonormal Sketches

Sampling blocks for sketching least squares has not been explored as extensively as sampling rows, though there has been interest in using "block-iterative methods" for solving systems of linear equations [101, 131, 215, 240]. Our interest in sampling blocks, is to invoke results and techniques from *randomized numerical linear algebra* (RandNLA) to CC. Specifically, we apply the transformation before partitioning the data and sharing it between the workers, who will compute the respective partial gradients. Then, the slowest s workers will be disregarded. The proposed sketching matrices are summarised in Algorithm 7.

To construct the sketch $\hat{\mathbf{A}}$, we first transform the orthonormal basis $\mathbf{U}$ by applying $\mathbf{\Pi}$ to $\mathbf{A}$. Then, we subsample q many blocks from $\mathbf{\Pi A}$, to reduce the dimension. Finally, we normalize by $\sqrt{N/r}$ to reduce the variance of the estimator $\hat{\mathbf{A}}$. Analogous steps are carried out on $\mathbf{\Pi b}$, to construct $\hat{\mathbf{b}}$.

4.3.1 Distributed Steepest Descent and Iterative Sketching

We now discuss the workers' computational tasks of our proposed GCS, when SD is carried out distributively. The encoding corresponds to $\tilde{\mathbf{A}} = \mathbf{G} \cdot \mathbf{A}$ and $\tilde{\mathbf{b}} = \mathbf{G} \cdot \mathbf{b}$ for $\mathbf{G} \coloneqq \sqrt{N/r} \cdot \mathbf{\Pi}$, which are then partitioned into K encoded block pairs $(\tilde{\mathbf{A}}_i, \tilde{\mathbf{b}}_i)$; similar to (4.4), and are sent to distinct workers. Specifically, $\tilde{\mathbf{A}}_i = \mathbf{I}_{(\mathcal{K}_i)}(\mathbf{GA})$ and

Algorithm 7: Subsampled Orthonormal Sketches

Input: $\mathbf{A} \in \mathbb{R}^{N\times d}$, $\mathbf{b} \in \mathbb{R}^N$, $\mathbf{x}^{[0]} \in \mathbb{R}^d$, $\tau = \frac{N}{K}$, $q = \frac{r}{\tau} > \frac{d}{\tau}$
Output: approximate solution $\hat{\mathbf{x}} \in \mathbb{R}^d$ to (4.1)
Randomly Select: $\mathbf{\Pi} \in O_N(\mathbb{R})$, an orthonormal matrix
for $t = 0, 1, 2, \ldots$ **do**
  Initialize: $\mathbf{\Omega} = \mathbf{0}_{q\times K}$
  Select: step-size $\xi_t > 0$
  for $i = 1$ *to* q **do**
    uniformly sample with replacement j_i from $\mathbb{N}_K$
    $\mathbf{\Omega}_{i,j_i} = \sqrt{N/r} = \sqrt{K/q}$
  end
  $\widetilde{\mathbf{\Omega}}_{[t]} \leftarrow \mathbf{\Omega} \otimes \mathbf{I}_\tau$ ▷ $\mathbf{S}_{\mathbf{\Pi}}^{[t]} = \widetilde{\mathbf{\Omega}}_{[t]} \cdot \mathbf{\Pi}$
  $\hat{\mathbf{A}}_{[t]} \leftarrow \widetilde{\mathbf{\Omega}}_{[t]} \cdot (\mathbf{\Pi}\mathbf{A}) = \mathbf{S}_{\mathbf{\Pi}}^{[t]} \cdot \mathbf{A}$
  $\hat{\mathbf{b}}_{[t]} \leftarrow \widetilde{\mathbf{\Omega}}_{[t]} \cdot (\mathbf{\Pi}\mathbf{b}) = \mathbf{S}_{\mathbf{\Pi}}^{[t]} \cdot \mathbf{b}$
  Update: $\hat{\mathbf{x}}^{[t+1]} \leftarrow \hat{\mathbf{x}}^{[t]} - \xi_t \cdot \nabla_{\mathbf{x}} L_{ls}\big(\hat{\mathbf{A}}_{[t]}, \hat{\mathbf{b}}_{[t]}; \hat{\mathbf{x}}^{[t]}\big)$
end

$\tilde{\mathbf{b}}_i = \mathbf{I}_{(\mathcal{K}_i)}(\mathbf{G}\mathbf{b})$. This differs from most GCSs, in that the encoding is usually done locally by the workers on the computed results; at each iteration.

If each worker respectively computes $\nabla_{\mathbf{x}} L_{ls}(\tilde{\mathbf{A}}_i, \tilde{\mathbf{b}}_i; \mathbf{x}^{[t]}) = 2\tilde{\mathbf{A}}_i^T(\tilde{\mathbf{A}}_i^T\mathbf{x}^{[t]} - \tilde{\mathbf{b}}_i)$ at iteration t, and the index set of the first q responsive workers is $\mathcal{S}^{[t]}$, the aggregated gradient

$$\hat{g}^{[t]} = 2\sum_{j\in\mathcal{S}^{[t]}} \tilde{\mathbf{A}}_j^T\left(\tilde{\mathbf{A}}_j\mathbf{x}^{[t]} - \tilde{\mathbf{b}}_j\right) \tag{4.10}$$

is equal to the gradient of $L_{\mathbf{S}}$ for $\mathbf{S} \leftarrow \mathbf{S}_{\mathbf{\Pi}}^{[t]}$ the induced sketching matrix at that iteration, *i.e.* $\hat{g}^{[t]} = \nabla_{\mathbf{x}} L_{\mathbf{S}_{\mathbf{\Pi}}^{[t]}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$. The sampling matrix $\widetilde{\mathbf{\Omega}}_{[t]}$ and index set $\mathcal{S}^{[t]}$, correspond to the q responsive workers. We illustrate our procedure in Figure IV.1.

In Algorithm 7, Theorems 4.3.2 and 4.4.1, we assume sampling with replacement. In what we just described, we used one replica of each block, thus $K = m$. To compensate for this, more than one replica of each block could be distributed. This is not a major concern with uniform sampling, as the probability that the i^{th} block would be sampled more than once is $(q-1)/K^2$, which is negligible for large K.

Lemma 4.3.1. *At any iteration t, with no replications of the blocks across the network, the resulting sketching matrix $\mathbf{S}_{[t]}$ satisfies $\mathbb{E}\left[\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\right] = \mathbb{E}\left[\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]}\right] = \mathbf{I}_N$.*

It is worth noting that by Lemma 4.3.1, $\mathbf{S}_{[t]}$ in expectation satisfies the ℓ_2-s.e.

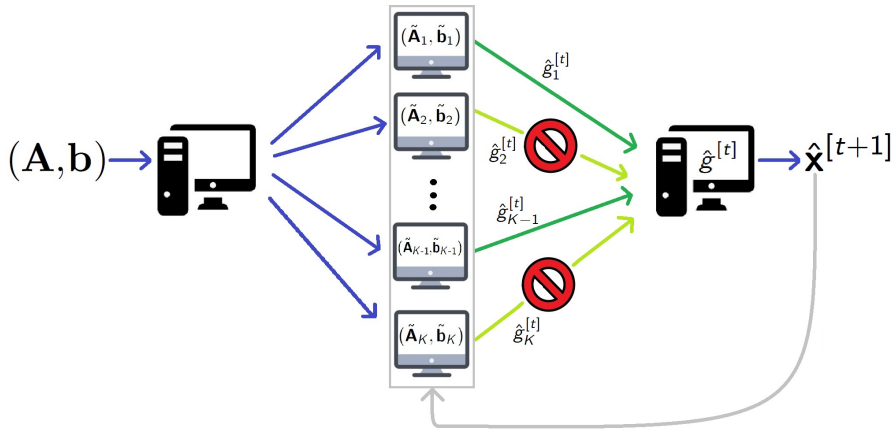


Figure IV.1: Illustration of our iterative sketching based GCS, at epoch $t+1$.

identity (4.9) with $\epsilon = 0$, as

$$\mathbb{E}\left[\mathbf{U}^T\left(\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\right)\mathbf{U}\right] = \mathbf{U}^T\mathbb{E}\left[\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\right]\mathbf{U} = \mathbf{U}^T\mathbf{U} = \mathbf{I}_d \ .$$

Theorem 4.3.1. *The proposed GCS results in a mini-batch stochastic steepest descent procedure for*

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^d}\left\{L_{\mathbf{G}}(\mathbf{A},\mathbf{b};\mathbf{x}) := L_{ls}(\mathbf{GA},\mathbf{Gb};\mathbf{x})\right\} . \tag{4.11}$$

Moreover $\mathbb{E}\left[\hat{g}^{[t]}\right] = \frac{q}{K}\cdot g_{ls}^{[t]}$.

Lemma 4.3.2. *The optimal solution of the modified least squares problem* $L_{\mathbf{G}}$*, is equal to the optimal solution* $\mathbf{x}_{ls}^\star$ *of* (4.1).

To prove Theorem 4.3.1, note that $\widetilde{\mathbf{\Omega}}_{[t]}$ corresponds to a uniform random selection of q out of K batches for each t; as in SSD, while in our procedure we consider the partial gradients of the q fastest responses. When computing $\nabla_{\mathbf{x}} L_{\mathbf{G}}$, the factor $\mathbf{\Pi}$ is annihilated; and the scaling factor $\sqrt{K/q}$ is squared.

Since $\mathbb{E}\left[\hat{g}^{[t]}\right] = \frac{q}{K}g_{ls}^{[t]}$, the estimate $\hat{g}^{[t]}$ is unbiased after an appropriate rescaling; which could be incorporated in the step-size ξ_t. By Theorem 4.3.1 and Lemma 4.3.2, it follows that with a diminishing step-size, our updates $\hat{\mathbf{x}}^{[t]}$ converge to $\mathbf{x}_{ls}^\star$ in expectation; at a rate of $\mathcal{O}(1/\sqrt{t} + r/t)$ [38, 79].

Corollary 4.3.1. *Consider the problems* (4.1) *and* (4.11)*, which are respectively solved through SD and our iterative sketching based GCS. Assume that the two approaches have the same starting point* $\mathbf{x}^{[0]}$ *and index set* $\mathcal{S}^{[t]}$ *at each* t*; and* $\hat{\xi}_t = \frac{K}{q}\xi_t$ *the step-sizes used for our scheme. Then, in expectation, our approach through Al-*

gorithm 7 has the same update at each step t as SD at the corresponding update, i.e $\mathbb{E}\left[\hat{\mathbf{x}}^{[t]}\right] = \mathbf{x}^{[t]}$.

By Lemma 4.3.2 and Corollary 4.3.1, the updated parameter estimates $\hat{\mathbf{x}}^{[0]}, \hat{\mathbf{x}}^{[1]}, \hat{\mathbf{x}}^{[2]}, \ldots$ of Algorithm 7 approach the optimal solution $\mathbf{x}_{ls}^{\star}$ of (4.1), by solving the modified regression problem (4.11) through SSD. It is also worth noting that the contraction rate of our GC approach, in expectation is equal to that of regular SD. This can be shown through an analogous derivation of [56, Theorem 6].

In the next subsection, we present our main ℓ_2-s.e. result.

4.3.2 Subspace Embedding of Algorithm 7

To give an embedding guarantee for Algorithm 7, we first show that the block leverage scores of $\mathbf{\Pi A}$ are "flattened", *i.e.* they are all approximately equal. This is precisely what allows us to sample blocks for the construction of $\mathbf{S_\Pi}$; and in the distributed approach the computations, *uniformly* at random. Recall that the *leverage scores* of $\tilde{\mathbf{U}} := \mathbf{\Pi U}$ are $\ell_i := \|\tilde{\mathbf{U}}_{(i)}\|_2^2$ for $i \in \mathbb{N}_N$, and the *block leverage scores* [52, 221] are defined as $\tilde{\ell}_\iota := \|\tilde{\mathbf{U}}_{(\mathcal{K}_\iota)}\|_F^2 = \sum_{j\in\mathcal{K}_\iota} \ell_j$ for all $\iota \in \mathbb{N}_K$. A lot of work has been done regarding ℓ_2-s.e. by leverage score sampling [91, 93, 95, 96, 294] as an importance sampling technique. By generalizing these to sampling blocks, one can show analogous results (*e.g.* [52, 56]).

Lemma 4.3.3 suggests that the *normalized* block leverage scores $\grave{\ell}_i = \frac{\tilde{\ell}_i}{K}$ of $\tilde{\mathbf{U}}$ are approximately uniform for all ι with high probability. This is the key step to proving that each $\mathbf{S}_\mathbf{\Pi}^{[t]}$ of Algorithm 7, satisfy (4.9). We illustrate the flattening of the scores for the various random projections considered in this chapter, in Figure IV.2.

Lemma 4.3.3. *For all $\iota \in \mathbb{N}_K$ and $\mathcal{K}_\iota \subsetneq \mathbb{N}_N$ of size $\tau = N/K$*

$$\Pr\left[\grave{\ell}_\iota <_{N\rho} 1/K\right] = \Pr\left[\left|\grave{\ell}_\iota - 1/K\right| < \tau\rho\right] > 1 - \delta,$$

for $\rho \geqslant \sqrt{\log(2\tau/\delta)/2}$.

Theorem 4.3.2. *Fix $\epsilon > 0$ such that $\epsilon \ll 1/N$. By Lemma 4.3.3, we can then assume that $\grave{\ell}_\iota = 1/K$ for all $\iota \in \mathbb{N}_K$. Then, $\mathbf{S_\Pi}$ of Algorithm 7 is a ℓ_2-s.e. sketching matrix of $\mathbf{A}$, according to (4.9). Specifically, for $\delta > 0$ and $q = \Theta\left(\frac{d}{\tau}\log\left(2d/\delta\right)/\epsilon^2\right)$:*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^T\mathbf{S}_\mathbf{\Pi}^T\mathbf{S}_\mathbf{\Pi}\mathbf{U}\|_2 \leqslant \epsilon\right] \geqslant 1 - \delta.$$

To prove Lemma 4.3.3 we use Hoeffding's inequality to show that the individual leverage scores are flattened, and then group them together by applying the binomial

approximation. This is then directly applied to a generalized version of the leverage score sketching matrix which samples blocks instead of individual rows [56, Theorem 1], to prove Theorem 4.3.2.

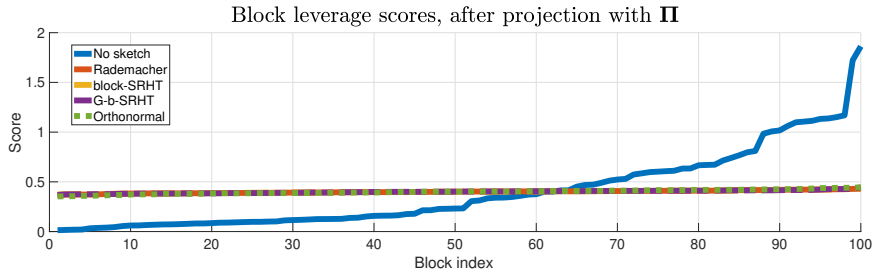


Figure IV.2: Flattening of block-scores, for $\mathbf{A}$ following a t-distribution. We abbreviate the garbled block-SRHT to 'G-b-SRHT'.

We note that there is no benefit in considering an overlap across the block batches which are sent to the workers (*e.g.* if a worker receives $\left[\tilde{\mathbf{A}}_{i-1}^T \ \tilde{\mathbf{A}}_i^T\right]^T$ and another receives $\left[\tilde{\mathbf{A}}_i^T \ \tilde{\mathbf{A}}_{i+1}^T\right]^T$), in terms of sampling. The reason is that since the computations are received uniformly at random, there is still the same chance that $\hat{g}_i$ and $\hat{g}_j$ would be considered, for any $i \neq j$.

Before moving onto the block-SRHT, we show how our scheme compares to other approximate GCSs in terms of the approximation error (4.6), when we consider multiple replications of each encoded block being shared among the workers. The result of Proposition 4.3.1 also applies to other sketching approaches, which satisfy (4.9).

Proposition 4.3.1. *By Theorem 4.3.2,* $\mathbf{S}_{\mathbf{\Pi}}$ *satisfies* (4.9) *(w.h.p.). Hence, the approximate gradients* $\hat{g}^{[t]}$ *of Algorithm 7 satisfy* (4.6) *(w.h.p.), with* $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) = \epsilon/\sqrt{K}$.

4.4 The Block-SRHT

In this section, we focus on a special case of $\mathbf{\Pi}$ which can be utilized in Algorithm 7; the *randomized Hadamard transform*. By utilizing this transform we satisfy property (g), and also avoid the extra computational cost which is needed to generate a random orthonormal matrix [150].

The SRHT is comprised of three matrices: $\mathbf{\Omega} \in \mathbb{R}^{r \times N}$ a uniform sampling and rescaling matrix of r rows, $\hat{\mathbf{H}}_N \in \{\pm 1/\sqrt{N}\}^{N \times N}$ the normalized Hadamard matrix for $N = 2^n$, and $\mathbf{D} \in \{0, \pm 1\}^{N \times N}$ with i.i.d. diagonal Rademacher random entries; *i.e.* it is a signature matrix. The main intuition of the projection is that it expresses the original signal or feature-row in the Walsh-Hadamard basis. Furthermore, $\hat{\mathbf{H}}_N$ can be

applied efficiently due to its structure. As in the case where we transformed the left orthonormal basis and column-space of $\mathbf{A}$ by multiplying its columns with a random orthonormal matrix $\mathbf{\Pi}$, in the new basis $\hat{\mathbf{H}}_N\mathbf{DU}$; the block leverage scores are close to uniform. Hence, we perform uniform sampling through $\widetilde{\mathbf{\Omega}}$ on the blocks of $\hat{\mathbf{H}}_N\mathbf{DA}$ to reduce the effective dimension N, whilst the information of $\mathbf{A}$ is maintained.

To exploit the SRHT in distributed GC for linear regression, we generalize it to subsampling blocks instead of rows; of the transformed data matrix, as in Algorithm 7. We give a ℓ_2-s.e. guarantee for the block-wise sampling version or SRHT, which characterizes the approximation of our proposed GCS for linear regression.

We refer to this special case as the "block-SRHT", for which $\hat{\mathbf{\Pi}}$ is taken from the subset $\hat{H}_N$ of $O_N(\mathbb{R})$

$$\hat{H}_N \coloneqq \left\{ \hat{\mathbf{H}}_N\mathbf{D} : \mathbf{D} = \mathrm{diag}(\pm 1) \in \{0, \pm 1\}^{N\times N} \right\}, \tag{4.12}$$

where $\mathbf{D}$ is a random signature matrix with equiprobable entries of +1 and -1, and $\hat{\mathbf{H}}_N$ for $N = 2^n$ is defined by

$$\mathbf{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \hat{\mathbf{H}}_N = \frac{1}{\sqrt{N}} \cdot \mathbf{H}_2^{\otimes \log_2(N)} \ .$$

Equivalently, $\hat{\mathbf{H}}_N$ can be defined entry-wise through the n-bit binary representation of i,j as

$$\hat{\mathbf{H}}_{ij} = (-1)^{\langle i,j\rangle_2}/\sqrt{N} \ \text{ for } \ \langle i,j\rangle_2 = \left(\sum_{l=0}^{n-1} i_l \cdot j_l \right) \bmod 2 \ .$$

The SRHT introduced in [96] corresponds to the case where we select $\tau = 1$, *i.e.* $K = N$. Henceforth, we drop the subscript N.

The main differences between the SRHT and the proposed block-SRHT $\mathbf{S}_{\tilde{\mathbf{\Pi}}}$ for $\tilde{\mathbf{\Pi}} \overset{U}{\leftarrow} \hat{H}_N$, is the sampling matrix $\widetilde{\mathbf{\Omega}}$; and that $q = r/\tau$ sampling trials take place instead of r. The limiting computational step of applying $\mathbf{S}_{\hat{\mathbf{\Pi}}}$ in (4.5) is the multiplication by $\hat{\mathbf{H}}$. The recursive structure of $\hat{\mathbf{H}}$ permits us to compute $\mathbf{S}_{\hat{\mathbf{\Pi}}}\mathbf{A}$ in $\mathcal{O}(Nd\log N)$ time, through Fourier methods [220].

4.4.1 Subspace Embedding of the Block-SRHT

To show that $\mathbf{S}_{\hat{\mathbf{\Pi}}}$ with $\hat{\mathbf{\Pi}} \overset{U}{\leftarrow} \hat{H}_N$ satisfies (4.9), we first present a key result, analogous to that of Lemma 4.3.3. Considering the orthonormal basis $\hat{\mathbf{V}} \coloneqq \hat{\mathbf{H}}\mathbf{DU}$ of the transformed data $\hat{\mathbf{H}}\mathbf{DA}$ with individual leverage scores $\{\ell_i\}_{i=1}^N$, Lemma 4.4.1 suggests

that the resulting block leverage scores $\tilde{\ell}_\iota = \|\hat{\mathbf{V}}_{(\mathcal{K}_\iota)}\|_F^2 = \sum_{j\in\mathcal{K}_\iota} \ell_j$ are approximately uniform for all $\iota \in \mathbb{N}_K$. Note that the diagonal entries of $\mathbf{D}$ is the only place in which randomness takes place other than the sampling. This then allows us to prove our ℓ_2-s.e. result regarding the block-SRHT, Theorem 4.4.1.

Lemma 4.4.1. *For all $\iota \in \mathbb{N}_K$ and $\mathcal{K}_\iota \subsetneq \mathbb{N}_N$ of size $\tau = N/K$*

$$\Pr\left[\tilde{\ell}_\iota \leqslant \eta d \cdot \log(Nd/\delta)/K\right] > 1 - \tau\delta/2, \tag{4.13}$$

for $0 < \eta \leqslant 2 + \log(16)/\log(Nd/\delta)$ a constant.

Theorem 4.4.1. *The block-SRHT $\mathbf{S}_{\hat{\mathbf{\Pi}}}$ is a ℓ_2-s.e. of $\mathbf{A}$. For $\delta > 0$ and $q = \Theta\big(\frac{d}{\tau}\log(Nd/\delta)\cdot\log(2d/\delta)/\epsilon^2\big)$:*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}\mathbf{U}\|_2 \leqslant \epsilon\right] \geqslant 1 - \delta\ .$$

Compared to Theorem 4.3.2, the above theorem has an additional logarithmic dependence on N. This is a consequence of applying the union bound, in order to show that the leverage scores of $\hat{\mathbf{H}}\mathbf{D}\mathbf{U}$ are flattened (Lemma 4.6.1). In the proof of Lemma 4.3.3, we instead applied Hoeffding's inequality, which removes such conditioning. Since Lemma 4.3.3 also holds for the block-SRHT, $\mathbf{S}_{\hat{\mathbf{\Pi}}}$ also satisfies the ℓ_2-s.e. guarantee stated in Theorem 4.3.2.

In Subsection 4.6.1 we alter the transformation $\hat{\mathbf{H}}\mathbf{D}$ by permuting its rows. While our ℓ_2-s.e. result remains intact, under mild but necessary assumptions, this transformation now also guarantees computational security.

4.4.2 Recursive Kronecker Products of Orthonormal Matrices

One could consider more general sets of matrices to sample from, while still benefiting from the recursive structure leveraged in Fourier methods. For a fixed 'base dimension' of $k \in \mathbb{Z}_{>2}$, let $\mathbf{\Pi}_k \in O_k(\mathbb{R})$, and define $\mathbf{\Pi} = \mathbf{\Pi}_k^{\otimes\lceil\log_k(N)\rceil}$. Carrying out the multiplication $\mathbf{\Pi}\mathbf{A}$ now takes $\mathcal{O}(Ndk^2\log_k N)$ time.

In the case where $k = 2$, up to a permutation of the rows and columns; we have $O_2(\mathbb{R}) = \{\mathbf{I}_2, \hat{\mathbf{H}}_2\}$, which is limiting compared to $O_k(\mathbb{R})$ for $k \geqslant 3$. This allows more flexibility, as more 'base matrices' $\mathbf{\Pi}_k$ can be considered, and the security can therefore be improved, as now we do not rely only on applying a random permutation to $\hat{\mathbf{\Pi}}$ (discussed in 4.6.1).

4.5 Optimal Step-Size and Adaptive GC

Recently, *adaptive gradient coding* (AGC) has been proposed in [42]. The objective is to adaptively design an exact GCS without prior information about the behavior of potential persistent stragglers, in order to gradually minimize the communication load. This though comes at the cost of further delays due to intermediate designs of GC encoding-decoding pairs, as well as performing the encoding and decoding steps. Furthermore, the assumptions made in [42] are more stringent compared to the ones we have made thus far.

In this section, we further speed up our process, by adaptively selecting a step-size which reduces the total number of iterations required for convergence to the solutions of problems (4.1), (4.5) and (4.11), when SD is carried out. The proposed choice $\xi_t^\star$ for the step-size, is based on the latest gradient update of (4.1) and (4.5). To determine $\xi_t^\star$, we solve

$$\xi_t^\star = \arg\min_{\xi\in\mathbb{R}_{\geqslant 0}} \left\{ \|\mathbf{A}\mathbf{x}^{[t+1]} - \mathbf{b}\|_2^2 \right\} = \arg\min_{\xi\in\mathbb{R}_{\geqslant 0}} \left\{ \|\mathbf{A}(\mathbf{x}^{[t]} - \xi \cdot g^{[t]})\mathbf{b}\|_2^2 \right\} \tag{4.14}$$

for each t. If $\xi_t = 0$, we have reached the global optimum.

Since (4.14) has a closed form solution, determining $\xi_t^\star$ at each iteration reduces to matrix-vector multiplications. In the distributive setting, this will be determined by the central server once sufficiently many workers have responded at iteration t, who will then update $\mathbf{x}^{[t+1]}$ according to (4.3).

Compared to AGC, this is a more practical model, as we do not design and deploy multiple codes across the network. The authors of [42] minimize the communication load of individual communication rounds. In contrast, we reduce the total number of iterations of the SD procedure, which leads to fewer communication rounds. Depending on the application and threshold parameters we pick for the two respective AGC methods, our proposed approach would most likely have a lower overall communication load. This of course would also depend on the selected step-size used in the AGC for [42], and termination criterion. Furthermore, we are also flexible in tolerating a different number of stragglers s at each iteration, which was a motivation for the design of AGC schemes.

Proposition 4.5.1. *Given the respective gradient $g^{[t]}$ and update $\mathbf{x}^{[t]}$ of the underlying objective function, the optimal step-size according to (4.14) for $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$,*

$L_{\mathbf{G}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ *and* $L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]}) \coloneqq \|\mathbf{\Pi}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2$*, is:*

$$\xi_t^\star = \langle \mathbf{A}g^{[t]}, \mathbf{A}\mathbf{x}^{[t]} - \mathbf{b} \rangle / \|\mathbf{A}g^{[t]}\|_2^2 \, . \tag{4.15}$$

In our distributive stochastic procedure, one could select an adaptive step-size ξ_{t+1} which minimizes $L_{\mathbf{S}_{\mathbf{\Pi}}^{[t]}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$; but the induced sketching matrix $\mathbf{S}_{\mathbf{\Pi}}^{[t]}$ would need to be explicitly determined once q workers have responded. This would result in further computations from the central server. Instead, we propose using the step-size (4.15), as it is optimal in expectation.

The bottleneck in using $\xi_t^\star$, is that it can only be updated once the $\tilde{g}^{[t]} \coloneqq \nabla_{\mathbf{x}} L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ has been determined, which causes a delay in updating $\hat{\mathbf{x}}^{[t+1]}$. Even so, we significantly reduce the number of iterations, which is evident through our experiments in Section 4.7. The overall computation of the entire network is therefore also reduced. Furthermore, $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{A}\mathbf{b}$ which appear in the expansion of (4.15) can be computed beforehand, so that $\mathbf{A}^T\mathbf{A}\mathbf{x}^{[t]}$ can be calculated by the central server while the workers are carrying out their tasks.

Corollary 4.5.1. *Assume that we know the parameter update* $\hat{\mathbf{x}}^{[t]}$*, and the gradient* $\tilde{g}^{[t]}$*. Over the possible index sets* $\mathcal{S}^{[t]}$ *at iteration* t*, the optimal step-size according to*

$$\arg\min_{\xi \in \mathbb{R}} \left\{ \mathbb{E}\left[\left\| \mathbf{S}_{\mathbf{\Pi}}^{[t]} \left(\mathbf{A}\hat{\mathbf{x}}^{[t+1]} - \mathbf{b} \right) \right\|_2^2 \right] \right\}$$

matches $\xi_t^\star$ *of* (4.15)*.*

4.6 Security of Orthonormal Sketches

In this section, we discuss the security of the proposed orthonormal-based sketching matrices and the block-SRHT. The idea behind securing the resulting sketches is that there is a large ensemble of orthonormal matrices $\mathbf{\Pi}$ to select from, making it near-impossible for adversaries to discover the inverse transformation.

To give information-theoretic security guarantees, we make some mild but necessary assumptions regarding Algorithm 7 and the data matrix $\mathbf{A}$. The message space $\mathcal{M}$ needs to be finite, which $\mathcal{M}$ in our case corresponds to the set of possible orthonormal bases of the column-space of $\mathbf{A}$. This is something we do not have control over, and it depends on the application and distribution from which we assume the data is gathered. Therefore, we assume that $\mathcal{M}$ is finite. For this reason, we consider a finite multiplicative subgroup $(\tilde{O}_{\mathbf{A}}, \cdot)$ of $O_N(\mathbb{R})$ (thus $\mathbf{I}_N \in \tilde{O}_{\mathbf{A}}$, and if $\mathbf{Q} \in \tilde{O}_{\mathbf{A}}$

then $\mathbf{Q}^T \in \tilde{O}_{\mathbf{A}}$), which contains all potential orthonormal bases of $\mathbf{A}$.[3] Recall that $O_N(\mathbb{R})$ is a regular submanifold of $\mathrm{GL}_N(\mathbb{R})$. Hence, we can define a distribution on any subset of $O_N(\mathbb{R})$.

We then let $\mathcal{M} = \tilde{O}_{\mathbf{A}}$, and assume $\mathbf{U}_{\mathbf{A}}$ the $N \times N$ orthonormal basis of $\mathbf{A}$ is drawn from $\mathcal{M}$ w.r.t. D. For simplicity, we consider D to be the uniform distribution. A simple method of generating a random matrix that follows the uniform distribution on the Stiefel manifold $V_n(\mathbb{R}^n)$ can be found in [64, Theorem 2.2.1]. Alternatively, one could generate a random Gaussian matrix and then perform Gram–Schmidt in order to orthonormalize it. Furthermore, an inherent limitation of Shannon secrecy is that $|\mathcal{K}| \geqslant |\mathcal{M}|$.

Theorem 4.6.1. *In Algorithm 7, sample* $\mathbf{\Pi}$ *uniformly at random from* $\tilde{O}_{\mathbf{A}}$. *The application of* $\mathbf{\Pi}$ *to* $\mathbf{A}$ *before partitioning the data, provides Shannon secrecy to* $\mathbf{A}$ *w.r.t. D uniform, for* $\mathcal{K}, \mathcal{M}, \mathcal{C}$ *all equal to* $\tilde{O}_{\mathbf{A}}$.

4.6.1 Securing the SRHT

Unfortunately, the guarantee of Theorem 4.6.1 does not apply to the block-SRHT, as in this case it is restrictive to assume that $\mathbf{U}_{\mathbf{A}} \in \hat{H}_N$. A simple computation on a specific example also shows that this sketching approach does not provide Shannon secrecy.[4] For instance, if $\mathbf{U}_0 = \mathbf{I}_2$, $\mathbf{U}_1 = \hat{\mathbf{H}}_2$ and the observed transformed basis $\bar{\mathbf{C}}$ has two zero entries, then

$$\Pr_{\mathbf{\Pi} \xleftarrow{U} H_2} \left[\mathbf{\Pi} \cdot \mathbf{U}_1 = \bar{\mathbf{C}}\right] > \Pr_{\mathbf{\Pi} \xleftarrow{U} H_2} \left[\mathbf{\Pi} \cdot \mathbf{U}_0 = \bar{\mathbf{C}}\right] = 0.$$

Furthermore, since $\hat{\mathbf{H}}$ is a known orthonormal matrix, it is a trivial task to invert this projection and reveal $\mathbf{DA}$. This shows that the inherent security of the SRHT is relatively weak. Proposition 4.6.1 is proven by constructing a counterexample.

Proposition 4.6.1. *The SRHT does not provide Shannon secrecy.*

To secure the SRHT and the block-SRHT, we randomly permute the rows of $\hat{\mathbf{H}}$, before applying it to $\mathbf{A}$. That is, for $\mathbf{P} \in S_N$ where $S_N \subsetneq \{0,1\}^{N \times N}$ is the permutation group on $N \times N$ matrices, we let $\tilde{\mathbf{H}} \coloneqq \mathbf{P}\hat{\mathbf{H}} \in \{\pm 1/\sqrt{N}\}^{N \times N}$, and the new sketching matrix is

$$\mathbf{S}_{\tilde{\mathbf{\Pi}}} = \widetilde{\mathbf{\Omega}} \cdot (\mathbf{P} \cdot \hat{\mathbf{H}}) \cdot \mathbf{D} = \widetilde{\mathbf{\Omega}} \cdot \tilde{\mathbf{H}} \cdot \mathbf{D} = \widetilde{\mathbf{\Omega}} \cdot \tilde{\mathbf{\Pi}}, \tag{4.16}$$

[3] In Appendix 3.4.2, we give an analogy between our approach and the OTP.
[4] Please check Appendix 3.4.1 for the details.

for which our flattening result (Corollary 4.6.1) still holds. We "garble" $\hat{\mathbf{H}}$ so that the projection applied to $\mathbf{A}$ now inherently has more randomness, and allows us to draw from a larger ensemble. Specifically, for a fixed N, the block-SRHT has 2^N options for $\hat{\mathbf{\Pi}} = \hat{\mathbf{H}}\mathbf{D}$, while for $\tilde{\mathbf{\Pi}} = \tilde{\mathbf{H}}\mathbf{D}$ there are $2^N N! = \mathcal{O}\big((2N/e)^N\sqrt{N}\big)$ options for $\tilde{\mathbf{\Pi}}$. Moreover, for

$$\tilde{H}_N \coloneqq \Big\{\mathbf{P}\mathbf{\Pi} : \mathbf{P} \in S_N \text{ and } \mathbf{\Pi} \in \hat{H}_N\Big\} \tag{4.17}$$

the set of all possible *garbled Hadamard transforms*, it follows that $(\tilde{H}_N, \cdot)$ is a finite multiplicative subgroup of $O_N(\mathbb{R})$. Hence, we can also define a distribution on $\tilde{H}_N$. We also get the benefits of permuting $\hat{\mathbf{H}}$'s columns without explicitly applying a second permutation, through $\mathbf{D}$.

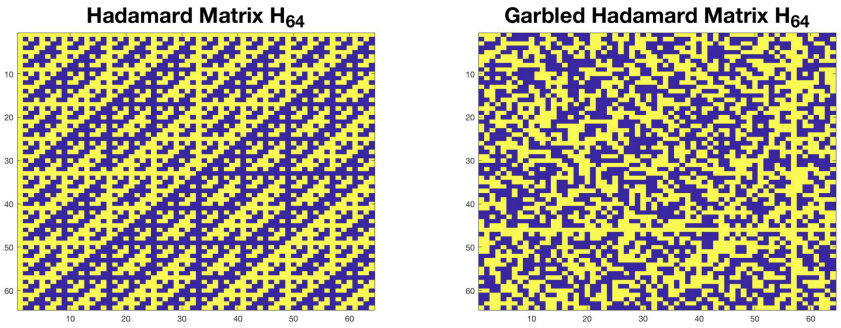


Figure IV.3: Example of how $\mathbf{P}$ and $\mathbf{D}$ modify the projection matrix $\hat{\mathbf{H}}_{64}$.

By the following Corollary, we deduce that Theorem 4.4.1 also holds for the "*garbled block-SRHT*" (an analogous result is used to prove Lemma 4.4.1). Thus, we can apply any $\tilde{\mathbf{\Pi}} \overset{U}{\leftarrow} \tilde{H}_N$ in Algorithm 7, and get a valid sketch.

Corollary 4.6.1. *For $\mathbf{y} \in \mathbb{R}^N$ a fixed (orthonormal) column vector of $\mathbf{U}$, and $\mathbf{D} \in \{0, \pm 1\}^{N\times N}$ with random equi-probable diagonal entries of ± 1, we have:*

$$\Pr\left[\|\tilde{\mathbf{H}}\mathbf{D}\cdot\mathbf{y}\|_\infty > C\sqrt{\log(Nd/\delta)/N}\right] \leqslant \frac{\delta}{2d} \tag{4.18}$$

for $0 < C \leqslant \sqrt{2 + \log(16)/\log(Nd/\delta)}$ a constant.

Moreover, Corollary 4.6.1 also holds true for random projections $\mathbf{R}$ whose entries are rescaled Rademacher random variables, *i.e.* $\mathbf{R}_{ij} = \pm 1/\sqrt{N}$ with equal probability. The advantage of this is that we have a larger set of projections

$$\tilde{R}_N \coloneqq \Big\{\mathbf{R} \in \{\pm 1/\sqrt{N}\}^{N\times N} : \Pr[\mathbf{R}_{ij} = +1/\sqrt{N}] = 1/2\Big\}$$

to draw from. This makes it even harder for an adversary to determine which projection was applied. Specifically $|\tilde{R}_N| = 2^{N^2}$, which is significantly larger than $|\tilde{H}_N|$. Two drawbacks of applying a random Rademacher projection $\mathbf{R}$ is that it is much slower than its Hadamard-based counterpart, and the resulting gradients $\hat{g}^{[t]}$ are not unbiased.

Next, we provide a computationally secure guarantee for the garbled block-SRHT $\mathbf{S}_{\tilde{\mathbf{\Pi}}} \leftarrow \widetilde{\mathbf{\Omega}}\tilde{\mathbf{\Pi}}$. The guarantee of Theorem 4.6.2 against computationally bounded adversaries, relies heavily on the assumption that *strong pseudorandom permutations* (s-PRPs) and *one-way functions* (OWFs) exist. Through a long line of work, it was shown that s-PRPs exist if and only if OWFs exist. Even though OWFs are minimal cryptographic objects, it is not known whether such functions exist [161]. Proving their existence is non-trivial, as this would then imply that $\mathsf{P} \neq \mathsf{NP}$. In practice however, this is not unreasonable to assume. The proof of Theorem 4.6.2 entails a reduction to inverting the s-PRP $\mathbf{P}$. In practice, *block ciphers* are used for s-PRPs.

Theorem 4.6.2. *Assume that* $\mathbf{P}$ *is a s-PRP. Then,* $\mathbf{S}_{\tilde{\mathbf{\Pi}}}\mathbf{A}$ *is computationally secure against polynomial-bounded adversaries, for* $\mathbf{S}_{\tilde{\mathbf{\Pi}}} \leftarrow \widetilde{\mathbf{\Omega}}\tilde{\mathbf{\Pi}}$ *the garbled block-SRHT.*

As discussed in 4.4, the Hadamard matrix satisfies the desired properties (b), (c), (d), (g), while any other form of a discrete Fourier transform would violate (c). By applying $\mathbf{P}$ to $\hat{\mathbf{H}}$, the matrix $\mathbf{P}\hat{\mathbf{H}}$ still satisfies the aforementioned properties, while also incorporating security; *i.e.* property (f). It would be interesting to see if other structured matrices exist which also satisfy (b)-(g). Similar to what we saw with the block-SRHT, if (b) is met; then we can achieve (a) through uniform sampling.

4.6.2 Exact Gradient Recovery

In the case where the *exact* gradient is desired, one can use the proposed orthonormal projections to encrypt the information from the workers, while requiring that the computations from *all* the workers are received. From Theorems 4.6.1 and 4.6.2, we know that under certain assumptions we can secure $\mathbf{A}$.

Since the projections are orthonormal, it would follow that $\hat{g}^{[t]} = g_{ls}^{[t]}$. Thus, as long as *all* workers respond, the aggregated gradient is equal to the exact gradient. One can utilize this idea to encrypt other distributive computations, *e.g.* matrix multiplication or inversion and logistic regression, which are discussed in Appendix 3.5. This resembles a homomorphic encryption scheme, but is by no means fully-homomorphic.

4.7 Experiments

We compared our proposed distributed GCSs to analogous approaches where the projection $\mathbf{\Pi}$ is a Gaussian sketch or a Rademacher random matrix. Our approach was found to outperform both these sketching methods in terms of convergence and approximation error, as the resulting gradients through these alternative approaches are not unbiased. In all experiments, the same initialization $\mathbf{x}^{[0]}$ was selected for each sketching methods.

Our approach was also compared to uncoded (regular) SD. Random matrices $\mathbf{A} \in \mathbb{R}^{2000\times 40}$ with non-uniform block leverage scores were generated for the experiments. Standard Gaussian noise was added to an arbitrary vector from $\text{im}(\mathbf{A})$, to define $\mathbf{b}$. We considered $K = 100$ blocks, thus $\tau = 20$. The effective dimension N was reduced to $r = 1000$.

For the experiments in Figure IV.4 we ran 600 iterations on six different instances for each one, and varied ξ for each experiment by logarithmic factors of the step-size $\xi^{\times} = 2/\sigma_{\max}(\mathbf{A})^2$. The average log residual errors $\log_{10}\left(\|\mathbf{x}_{ls}^{\star} - \hat{\mathbf{x}}\|_2/\sqrt{N}\right)$ are depicted in Figure IV.4. Step-size $\xi^{\times}$ was considered, as it guarantees descent at each iteration, though it is too conservative.

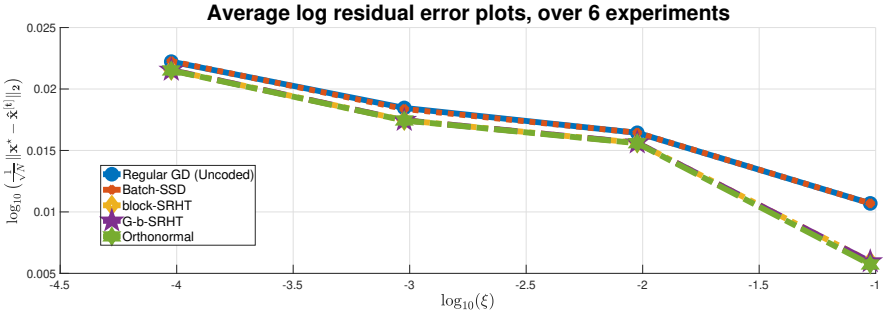


Figure IV.4: log residual error, for $\mathbf{A}$ following a t-distribution.

In contrast to the Gaussian sketch, orthonormal matrices $\mathbf{\Pi}$ also act as preconditioners. One example is the experiment depicted in Figure IV.5, in which the only modification we made from the previous experiments, was our initial choice of $\mathbf{x}^{[0]}$, which was scaled by $1/N$.

Next, we consider the case where ξ_t was updated according to (4.15). As above, our sketching approaches outperformed the case where a Gaussian sketch was applied. From Figure IV.6, our orthonormal sketching approach performs just as well as regular SD for the first 30 iterations, though it slows down afterwards, and is slightly worse than regular SD by the time 50 iterations have been completed. By the discussion

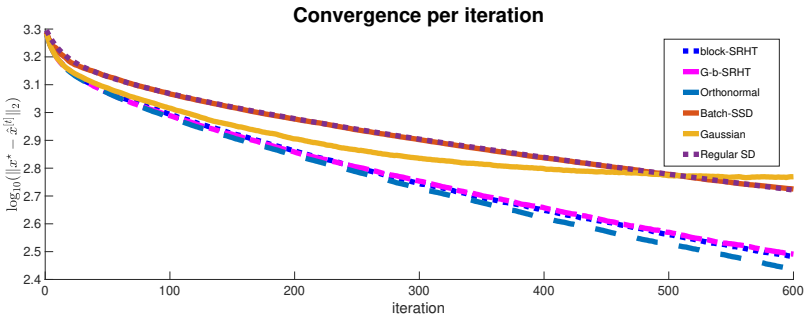


Figure IV.5: Example where $\mathbf{\Pi}$ also acts as a preconditioner.

in 4.6.2, we can achieve the performance of regular SD if we wait until all workers respond; and consider no stragglers, while our security guarantees still hold. This is true also for the block-SRHT and garbled block-SRHT, but not for the Gaussian sketch.

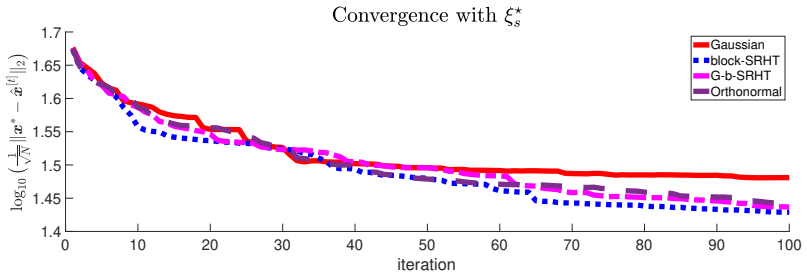


Figure IV.6: Adaptive step-size update, for $\mathbf{A}$ following a t-distribution.

Lastly, we give an example where it is clear that iterative sketching leads to better convergence than the sketch-and-solve approach. In the experiment depicted in Figure IV.7, we considered three sketching approaches: the iterative block-SRHT and garbled block-SRHT, and the non-iterative garbled block-SRHT. The step-size was adaptive at each iteration, as was done in the experiment of Figure IV.6.

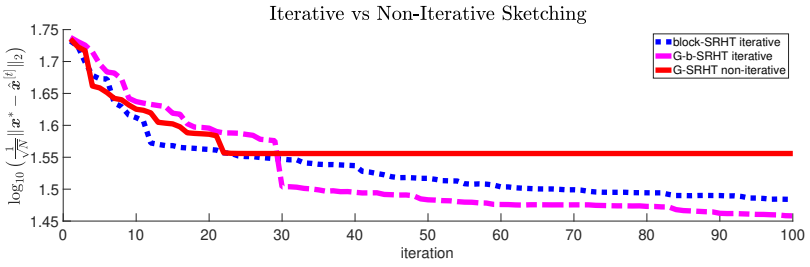


Figure IV.7: Convergence at each step, for the iterative block-SRHT and garbled block-SRHT, and the non-iterative garbled block-SRHT.

We carried out similar experiments when considering other dense and sparse matrices $\mathbf{A}$, with non-uniform block leverage scores. Similar results regarding our approaches were observed, as the ones provided above.

4.8 Concluding Remarks and Future Work

In this work, we proposed approximately solving a linear system by distributively leveraging iterative sketching and performing first-order SD simultaneously. In doing so, we benefit from both approximate GC and RandNLA. A difference to other RandNLA works is that our sketching matrices sample *blocks* uniformly at random, after applying a random orthonormal projection. An additional benefit is that by considering a large ensemble of orthonormal matrices to pick from, under necessary assumptions, we guarantee information-theoretic security while performing the distributed computations. This approach also enables us to not require encoding and decoding steps at every iteration. We also studied the special case where the projection is the randomized Hadamard transform, and discussed its security limitation. To overcome this, we proposed a modified "garbled block-SRHT", which guarantees computational security.

We note that applying orthonormal random matrices also secures coded matrix multiplication. There is a benefit when applying a garbled Hadamard transform in this scenario, as the complexity of multiplication resulting from the sketching is less than that of regular multiplication. Also, if such a random projection is used before performing CR-multiplication distributively [43, 53, 250], the approximate result will be the same. Moreover, our dimensionality reduction algorithm can be utilized by a single server, to store low-rank approximations of very large data matrices.

Partial stragglers, have also been of interest in the GC literature. These are stragglers who are able to send back a portion of their requested tasks. Our work is directly applicable, as we can consider smaller blocks, with multiple ones allocated to each worker.

There are several interesting directions for future work. We observed experimentally in Figure IV.5 that $\mathbf{\Pi}$ and $\hat{\mathbf{H}}_N$ may act as preconditioners for SSD. This mere observation requires further investigation. Another direction is to see if the proposed ideas could be applied to federated learning scenarios, in which security and privacy are major concerns. Some of the projections we considered, rely heavily on the recursive structure of $\hat{\mathbf{H}}$ in order to satisfy (g). One thing we overlooked, is whether other efficient multiplication algorithms (*e.g.* Strassen's [276]) could be exploited, in order

to construct suitable projections. It would be interesting to see if other structured or sparse matrices exist which also satisfy our desired properties (a)-(g).

There has been a lot of work regarding second-order algorithms with iterative sketching, *e.g.* [173, 231]. Utilizing iterative Hessian sketching or sketched Newton's method in CC has been explored in a tangential work [127], though the security aspect of these algorithms has not been extensively studied. A drawback here is that the local computations at the workers would be much larger, though we expect the number of iterations to be significantly reduced; for the same termination criterion to be met, compared to first-order methods. Deeper exploration of the theoretical guarantees of iterative sketched first-order methods, along with a comparison to their second-order counterparts, as well as studying their effect in logistic regression and other applications, are also of potential interest.

CHAPTER V

Securely Aggregated Coded Matrix Inversion

5.1 Introduction and Related Work

Inverting a matrix is one of the most important operations in numerous applications, such as, signal processing, machine learning, and scientific computing [126, 140]. A common way of inverting a matrix is to perform Gaussian elimination, which requires $\mathcal{O}(N^3)$ operations for square matrices of order N. In high-dimensional applications, this can be cumbersome. Over the past few years the machine learning (ML) community has made much progress on *federated learning* (FL), focusing on iterative methods.

The objective of FL is to leverage computation, communication and storage resources to perform distributed computations for ML models, where the data of each federated worker is never shared with the coordinator of the network; that aggregates local computations in order to update the model parameters. In FL applications it is important that the data is kept private and secure.

Distributed computations in the presence of *stragglers* (workers who fail to compute their task or have longer response time than others) must account for the effect of non-responsive workers. Coding-theoretic approaches have been adopted for this purpose [178, 302], and fall under the framework of *coded computing* (CC). Other techniques have also been utilized; to develop *approximate* CC schemes, *e.g.* equiangular tight frames [159] and sketching [49]. Data security is also an increasingly important issue in CC [181]. Despite the fact that multiplication algorithms imply inversion algorithms and vice versa, in the context of CC; matrix inversion has not been studied as extensively as *coded matrix multiplication* (CMM) [303]. The main reason for this is the fact that the latter is non-linear and non-parallelizable as an operator. We point out that distributed inversion algorithms do exist, though these make assumptions on the matrix, are specific for distributed and parallel computing

platforms, and require a matrix factorization; or heavy and multiple communication instances between the workers and the coordinator.

In [54] a CC method[1] was proposed based on *gradient coding* (GC) [134], which approximates the inverse of a matrix $\mathbf{A}$. In order to overcome the obstacle of non-linearity, the columns of $\mathbf{A}^{-1}$ are approximated. When assuming infinite floating-point precision, this CCM introduces no numerical nor approximation errors. Note that GC and not CMM was utilized, as the latter does not require the encoding to be done locally by the workers.

Though the two areas of FL and CC seem to be closely related, on the surface they appear incompatible. For instance, in CC one often assumes there is a master server that distributes the data and may perform the encoding (encoding by the master server is done in CMM, but not in GC), while in FL the central coordinator never has access to the distributed local training data; which are located at different client nodes or workers.

There are a few recent works that leverage CC in order to devise secure FL methods for distributed regression and iterative optimization [81, 132, 169, 233, 256, 295]. In this work, we combine optimization and CC, using erasure coding to protect against stragglers as in CC and locally approximating the inverse without revealing the data to the coordinator, to design a CCM which inverts a matrix from data aggregated through clients; and guarantees security against eavesdroppers. Our approach, is based on the *coded matrix inversion method* (CMIM) we develop, which utilizes *balanced Reed-Solomon* (BRS) codes [135, 136]. This results in an efficient decoding in terms of the threshold number of responsive workers needed to perform an error free computation. We show that the general class of *maximum distance separable* (MDS) generator matrices could be used to generate a suitable erasure code (Theorem 5.5.1). The focus is on BRS codes, which have the following advantages:

(i) minimum redundancy per task across the network,

(ii) they optimize communication from workers to the master,

(iii) we can efficiently decode the resulting method.

As noted in [81, 256], most CCMs are not applicable in FL. In our case, the obstacle is that all clients need to know each others data in order to invert the aggregated matrix, which we elaborate on in 5.5.1. For this reason, we relax the

[1] We abbreviate 'coded computing method/methods' to CCM/CCMs.

privacy restriction of FL and allow the clients to recover the aggregated matrix $\mathbf{A}$, which is necessary and unavoidable for matrix inversion.

Our CMIM can also be used to compute the Moore–Penrose pseudoinverse $\mathbf{Y}^\dagger$ of a data matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ for $M \gg N$, which is more general than inverting a square matrix. By using the fact that $\mathbf{Y}^\dagger = (\mathbf{Y}^\top \mathbf{Y})^{-1} \mathbf{Y}^\top$, the bottleneck is computing the inverse of $\mathbf{A} = \mathbf{Y}^\top \mathbf{Y}$. In addition, two more matrix multiplications need to take place distributively: computing $\mathbf{A}$ before the inversion; and $\widehat{\mathbf{A}^{-1}} \mathbf{Y}^\top$ after the inverse has been approximated. The matrix products can be computed distributively using various CCMs, *e.g.* we can use a modification of the coded FL approaches of [256] and a CMM from [47]; both of which are based on GC. For the remainder of the chapter, we focus on the generic problem of inverting a square matrix $\mathbf{A}$.

The proposed FL approach applies to general linear regression problems. Compared to traditional FL iterative approaches [167], the difference is that for $\mathbf{Y}\theta = \boldsymbol{p}$; with $\boldsymbol{p}$ the label vector and θ the model parameters, the pseudoinverse-regularized regression solution is $\hat{\theta} = \widehat{\mathbf{Y}^\dagger} \boldsymbol{p}$. Unlike conventional FL methods, this regularized regression can be computed non-iteratively. The non-iterative nature of the proposed approach is advantageous in settings such as Kalman filtering, where the matrix inverse must be updated in real time as measurements come in, as well as when dealing with time-series; and regularized regression with varying regularized coefficients.

This chapter is organized as follows. In 5.2 we recall basic facts on matrix inversion, least squares approximation and finite fields. In 5.3 we review BRS codes, and prove two key lemmas regarding their generator matrices. In 5.4 we present the matrix inverse approximation algorithm we utilize in our CCM. The main contribution is presented in 5.5. Our federated approach is split into four phases, which we group in pairs of two. First, we discuss information sharing from the coordinator to the workers (we consider all the clients' servers as the network's workers), and then information sharing between the workers. Second, we show how our inversion algorithm can be incorporated in linear CCMs, and describe how this fits into the relaxed FL setting we are considering. Concluding remarks and future work are presented in 5.6.

5.1.1 Overview of the Coded Matrix Inversion Method

In CC the computational network is centralized, and is comprised of a master server who communicates with n workers. The idea behind our approximation is that the workers use a least squares solver to approximate multiple columns of $\mathbf{A}^{-1}$, resulting in a set of local approximations to submatrices of $\widehat{\mathbf{A}^{-1}}$, which we refer to as *blocks*. We present approximation guarantees and simulation results for *steepest*

descent (SD) and *conjugate gradient* (CG) iterative optimization methods. By locally approximating the columns in this way, the workers can linearly encode the blocks of $\widehat{\mathbf{A}^{-1}}$. The clients have a block of data $\{\mathbf{A}_\iota\}_{\iota=1}^{k}$, which constitute the data matrix $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \cdots & \mathbf{A}_k\end{bmatrix}$. To simplify our presentation, we assume that each local data block is of the same size; *i.e.* $\mathbf{A}_\iota \in \mathbb{R}^{N\times T}$ for $T = N/k$, and that client i has n_i servers. Therefore, the total number of workers is $n = \sum_{j=1}^{k} n_j$. We assume the blocks are of the same size, so that the encodings carried out by the clients are consistent. In 5.5, we show that this assumption is not necessary. Moreover, for the CCM, it is not required that the number of blocks equal the number of clients. For a given natural number γ, assume that γ divides T; denoted $\gamma \mid T$ (each local data block $\mathbf{A}_\iota$ is further divided into γ sub-blocks). In the case where $k \nmid N$ or $\gamma \nmid T$, we can pad the blocks of $\widehat{\mathbf{A}^{-1}}$ so that these assumptions are met.

A limitation of our proposed CMIM, is the fact that each worker needs to have full knowledge of $\mathbf{A}$, in order to estimate columns of $\mathbf{A}^{-1}$ through a least squares solver. The sensitivity of Gaussian elimination and matrix inversion also require that all clients have knowledge of each others' data [54]. This limitation is shared by other coded federated learning methods, *e.g.* CodedPaddedFL [256], and further justifies our requirement that task allocations need to be carefully distributed across the workers, especially in the context of FL. In contrast to CC and GC; where a master server has access to all the data, in FL the data is inherently distributed across devices, thus GC cannot be applied directly. We also assume that the coordinator does not intercept the communication between the clients, otherwise she could recover the local data. Also, we trust that the coordinator will not invert $\widehat{\mathbf{A}^{-1}}$, to approximate $\mathbf{A}$ — this would be computationally difficult, for N large.

Before broadcasting the data amongst themselves, the clients encode their block $\mathbf{A}_i$, which guarantees security from outside eavesdroppers. When the clients receive the encoded data, they can decrypt and recover $\mathbf{A}$. Then, their servers act as the workers of the proposed CMIM and carry out their assigned computations, and directly communicate their computations back to the coordinator. Once the *recovery threshold* (the minimum number of responses needed to recover the computation) is met, the approximation $\widehat{\mathbf{A}^{-1}}$ is recoverable.

5.1.2 Coded Federated Learning

There are few works that leverage CC to devise secure FL schemes. Most of these works have focused on distributed regression and iterative methods, which is the primary application for FL [81, 169, 233, 256, 295]. Below, we describe and compare

these approaches to our work.

The authors of [81] proposed *coded federated learning*, in which they utilize a CMM scheme. Their security relies on the use of random linear codes, to define the parity data. Computations are carried out locally on the systematic data, and only the parity data is sent to the coordinator. The main drawback compared to our scheme is that each worker has to generate a random encoding matrix and apply a matrix multiplication for the encodings, while we use the same BRS generator matrix across the network, based on GC, to linearly encode the local computations. The drawback in our case, is that the workers need to securely share their data with each other. This is an artifact of the operation (inversion) we are approximating, and is inevitable in the general case where $\mathbf{A}$ has no structure. Under the relaxed FL setting we are considering, where the data is gathered or generated locally and is not i.i.d., we cannot make any assumptions on the structure of $\mathbf{A}$.

In [256], two methods were proposed. CodedPaddedFL combines one-time-padding with GC to carry out the FL task. Some of its disadvantages are that a *one-time-pad* (OTP) needs to be generated by each worker, and that the OTPs are shared with the coordinator, which means that if she gets hold of the encrypted data, she can decrypt it, compromising security. Furthermore, there is a heavy communication load and the coordinator needs to store all the pads in order to recover the computed gradients. In the proposed CMIM, the coordinator generates a set of interpolation points, and shares them with the clients. If the coordinator can intercept the communication between the workers, she can decrypt the encrypted data blocks. The second method proposed in [256], CodedSecAgg, relies on *Shamir's secret sharing* (SSS); which is based on polynomial interpolation over finite fields. In contrast, our CMIM relies on GC and Lagrange interpolation.

Lastly, we discuss the method proposed in [295], which is based on the McEliece cryptosystem, and moderate-density parity-check codes. This scheme considers a communication delay model which defines stragglers as the workers who respond slower than the fastest worker, and time out after a predetermined amount of time Δ. As the iterative SD process carries on, such workers are continuously disregarded. Due to this, there is a data sharing step at each iteration, at which the new stragglers communicate encrypted versions of their data to the active workers. Our scheme is non-iterative, and has a fixed recovery threshold. Unlike some of the works previously mentioned, which guarantee information-theoretic security, the McEliece based systems and our approach have *computational* privacy guarantees.

5.1.3 Lagrange Interpolation and Polynomial CCMs

While there is extensive literature on matrix-vector and matrix-matrix multiplication, and computing the gradient in the presence of stragglers, there is limited work on computing or approximating the inverse of a matrix [296]. The non-linearity of matrix inversion prohibits linear or polynomial encoding of the data before the computations are to be performed. Consequently, most CCMs cannot be directly utilized. Gradient coding is the appropriate CC set up to consider [279], precisely because the encoding takes place once the computation has been completed, in contrast to most CMM methods where the encoding is done by the master, before the data is distributed. This helps improve the recovery threshold, which is a primary objective of the CMM problem.

Here, we give a brief overview of the GC scheme on which our CMIM is based. We also review "Lagrange Coded Computing" (LCC), which has relations to our approach. Then, we give a summary of our proposed CMIM. All these rely on Lagrange interpolation over finite fields. We then mention related CMM schemes based on Lagrange or polynomial interpolation.

Gradient codes are a class of codes designed to mitigate the effect of stragglers in data centers, by recovering the gradient of differentiable and additively separable objective functions in distributed first-order methods [279]. The proposed CMIM utilizes BRS generator matrices constructed for GC [134]. The main difference from our work is that in GC the objective is to construct an encoding matrix $\mathbf{G} \in \mathbb{C}^{n\times k}$ and decoding vectors $\mathbf{a}_{\mathcal{I}} \in \mathbb{C}^k$, such that $\mathbf{a}_{\mathcal{I}}^\top \mathbf{G}_{\mathcal{I}} = \vec{\mathbf{1}}$ for any set of non-straggling workers indexed by $\mathcal{I}$. To do so, the decomposition of the BRS generator matrices $\mathbf{G}_{\mathcal{I}} = \mathbf{H}_{\mathcal{I}}\mathbf{P}$ is exploited, where $\mathbf{H}_{\mathcal{I}}$ is a Vandermonde matrix; and the first row of $\mathbf{P}$ is equal to $\vec{\mathbf{1}}$. Subsequently $\mathbf{a}_{\mathcal{I}}^\top$ is extracted as the first row of $\mathbf{H}_{\mathcal{I}}^{-1}$.

In the proposed CMIM framework, the objective is to design an *encoding-decoding pair* $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$ for which $\tilde{\mathbf{D}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}} = \mathbf{I}_N$, for all $\mathcal{I} \subsetneq \mathbb{N}_n$ of size k. The essential reason for requiring this condition, as opposed to that of GC, is that the empirical gradient of a given dataset is the sum of each individual gradients, while in our scenario if the columns of $\widehat{\mathbf{A}^{-1}}$ are summed; they cannot then be recovered.

The state-of-the art CC framework is LCC, which is used to compute arbitrary multivariate polynomials of a given dataset [302]; and has since been considered in various settings [107, 266, 267, 278, 307]. This approach is based on Lagrange interpolation, and it achieves the optimal trade-off between resiliency, security, and privacy. The problem we are considering is not a multivariate polynomial in terms of $\mathbf{A}$. To securely communicate $\mathbf{A}$ to the workers, we encode it through Lagrange

interpolation. Though similar ideas appear in LCC, the purpose and application of the interpolation is different. Furthermore, LCC is a *point-based* approach and requires additional interpolation and linear combination steps after the decoding takes place, while ours is a *coefficient-based* CCM [164, 165].

Recall that the workers in the CMIM must compute blocks of $\widehat{\mathbf{A}^{-1}}$. Once they complete their computations, they encode them by computing a linear combination with coefficients determined by a sparsest-balanced MDS generator matrix. Referring to the advantages claimed for CMIM in Section 5.1, working with MDS generator matrices allows us to meet points (i) and (ii), while BRS generator matrices also helps us satisfy (iii). Once the recovery threshold is met, the coordinator can recover the approximation $\widehat{\mathbf{A}^{-1}}$. The structure of sparsest-balanced generator matrices is also leveraged to optimally allocate tasks to the workers, while linear encoding is what allows minimal communication load from the workers to the master. Security against eavesdroppers is guaranteed by encoding the local data through a modified Lagrange interpolation polynomial, before it is shared by the clients. This CMIM also extends to approximating $\mathbf{A}^{\dagger}$ [54].

Some of the earliest interpolation based CMM schemes are the Polynomial [303] and MatDot codes [98], both of which are point-based. The construction of 'Polynomial Codes' has since been generalized to 'Entangled Polynomial Codes' [304], which define similar polynomials to ours (5.12), though their use differs. We use (5.12) to encrypt the data block of each client; and our decryption is an evaluation at a finite field point. For Entangled Polynomial Codes two such polynomials are defined; one for each input matrix, and their product determines another degree $R-2$ polynomial which is evaluated by each worker at a different point, before proceeding to the decoding step.

In MatDot codes [98] two polynomials are defined, one corresponding to each input, where instead of the Lagrange polynomial in (5.12); a monic monomial is multiplied by the partitions of the respective block submatrices. Then, analogous steps to those of Entangled Polynomial Codes take place, in order to recover the matrix product.

5.2 Preliminary Background

The set of $N \times N$ non-singular matrices is denoted by $\mathrm{GL}_N(\mathbb{R})$. Recall that $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$ has a unique inverse $\mathbf{A}^{-1}$, such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_N$. The simplest way of computing $\mathbf{A}^{-1}$ is by performing Gaussian elimination on $\left[\mathbf{A}|\mathbf{I}_N\right]$, which gives $\left[\mathbf{I}_N\middle|\mathbf{A}^{-1}\right]$ in $\mathcal{O}(N^3)$ operations. In Algorithm 8, we approximate $\mathbf{A}^{-1}$

column-by-column. We denote the i^{th} row and column of $\mathbf{A}$ respectively by $\mathbf{A}_{(i)}$ and $\mathbf{A}^{(i)}$. The *condition number* of $\mathbf{A}$ is $\kappa_2 = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2$. The largest, smallest and i^{th} singular values of $\mathbf{A}$ are denoted by $\sigma_{\max}(\mathbf{A})$, $\sigma_{\min}(\mathbf{A})$ and $\sigma_i(\mathbf{A})$ respectively. For $\mathcal{I}$ an index subset of the rows of a matrix $\mathbf{M}$, the matrix consisting only of the rows indexed by $\mathcal{I}$, is denoted by $\mathbf{M}_{\mathcal{I}}$. We denote the set of integers between 1 and ν by $\mathbb{N}_\nu$. The support of a vector $\mathbf{v}$ is denoted by $\text{supp}(\mathbf{v})$, and the number of nonzero elements of $\mathbf{A}$ by $\text{nnzr}(\mathbf{A})$.

In the proposed algorithm we approximate N instances of the least squares minimization problem

$$\theta^\star_{ls} = \arg\min_{\theta\in\mathbb{R}^M} \left\{\|\mathbf{A}\theta - \mathbf{y}\|_2^2\right\} \tag{5.1}$$

for $\mathbf{A} \in \mathbb{R}^{N\times M}$ and $\mathbf{y} \in \mathbb{R}^N$. In many applications $N \gg M$, where the rows represent the feature vectors of a dataset. This has the closed-form solution $\theta^\star_{ls} = \mathbf{A}^\dagger\mathbf{y}$.

Computing $\mathbf{A}^\dagger$ to solve (5.1) is intractable for large M, as it requires computing the inverse of $\mathbf{A}^\top\mathbf{A}$. Instead, we use gradient methods to get *approximate* solutions, *e.g.* SD or CG, which require less operations, and can be done distributively. One could use second-order methods; *e.g.* Newton–Raphson, Gauss-Newton, Quasi-Newton, BFGS, or Krylov subspace methods instead. This would be worthwhile future work.

When considering a minimization problem with a convex differentiable objective function $\psi\colon \Theta \to \mathbb{R}$ over an open convex set $\Theta \subseteq \mathbb{R}^M$, as in (5.1), the SD procedure selects an initial $\theta^{[0]} \in \Theta$, and then updates θ according to:

$$\theta^{[t+1]} = \theta^{[t]} - \xi_t \cdot \nabla_\theta\psi(\theta^{[t]})$$

for $t = 0, 1, 2, \ldots$ until a termination criterion is met, for ξ_t the step-size. The CG method is the most used and prominent iterative procedure for numerically solving systems of positive-definite equations.

Our proposed coding scheme is defined over the multiplicative cyclic group $(\mathbb{F}_q^\times, \cdot)$, for $\mathbb{F}_q$ the finite field of q elements and $\mathbb{F}_q^\times = \mathbb{F}_q\backslash\{0_{\mathbb{F}_q}\}$ its set of units. For implementation purposes, we identify $\mathbb{F}_q^\times$ with its realization in $\mathbb{C}$ as a subgroup of the circle group, since we assume our data is over $\mathbb{R}$. All operations can therefore be carried out over $\mathbb{C}$. Specifically, we identify β as an arbitrary primitive generator of $(\mathbb{F}_q^\times, \cdot)$. One such case is to identify $\beta \mapsto e^{2\pi i/(q-1)}$. Thus, for all $j \in \mathbb{N}_{q-1}$; we identify $\beta^j \mapsto e^{2\pi ij/(q-1)}$.

5.3 Balanced Reed-Solomon Codes

A Reed-Solomon code $\mathsf{RS}_q[n,k]$ over $\mathbb{F}_q$ for $q > n > k$, is the encoding of polynomials of degree at most $k-1$, for k the message length and n the code length [243]. It represents our message over the *defining set of points* $\mathcal{A} = \{\alpha_j\}_{j=1}^n \subset \mathbb{F}_q$

$$\mathsf{RS}_q[n,k] = \Big\{ \big[f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_n)\big] \;\Big|\; f(X) \in \mathbb{F}_q[X] \text{ of degree } \leqslant k-1 \Big\}$$

where $\alpha_j = \alpha^j$, for α a primitive root of $\mathbb{F}_q$. Hence, each α_i is distinct. A natural interpretation of $\mathsf{RS}_q[n,k]$ is through its encoding map. Each message $(m_0, ..., m_{k-1}) \in \mathbb{F}_q^k$ is interpreted as $f(\mathsf{x}) = \sum_{i=0}^{k-1} m_i \mathsf{x}^i \in \mathbb{F}_q[\mathsf{x}]$, and f is evaluated at each point of $\mathcal{A}$. From this, $\mathsf{RS}_q[n,k]$ can be defined through the generator matrix

$$\mathbf{G} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \dots & \alpha_n^{k-1} \end{pmatrix} \in \mathbb{F}_q^{n \times k},$$

thus, RS codes are linear codes over $\mathbb{F}_q$. Furthermore, they attain the Singleton bound, *i.e.* $d = n - k + 1$, where d is the code's distance, which implies that they are MDS.

Balanced Reed-Solomon codes [135, 136] are a family of linear MDS error-correcting codes with generator matrices $\mathbf{G} \in \mathbb{F}_q^{n \times k}$ that are:

- **sparsest**: each *column* has the least possible number of nonzero entries
- **balanced**: each *row* contains the same number of nonzero entries

for the given code parameters k and n. The design of these generators are suitable for our purposes, as:

1. we have balanced loads across homogeneous workers,
2. sparse generator matrices reduce the computation tasks across the network,
3. the MDS property permits an efficient decoding step,
4. linear codes produce a compressed representation of the encoded blocks.

5.3.1 Balanced Reed-Solomon Codes for CC

In the proposed CMIM, we leverage BRS generator matrices to approximate $\mathbf{A}^{-1}$ distributively. For simplicity, we will consider the case where $d = s + 1 = \frac{nw}{k}$ is a positive integer[2], for n the number of workers and s the number of stragglers. Furthermore, d is the distance of the code and $\|\mathbf{G}^{(j)}\|_0 = d$ for all $j \in \mathbb{N}_k$; $\|\mathbf{G}_{(i)}\|_0 = w$ for all $i \in \mathbb{N}_n$, and $d > w$ since $n > k$. For decoding purposes, we require that at least $k = n - s$ workers respond. Consequently, $d = s + 1$ implies that $n - d = k - 1$. For simplicity, we also assume $d \geqslant n/2$. In our setting, each column of $\mathbf{G}$ corresponds to a computation task of computing a block of $\widehat{\mathbf{A}^{-1}}$; which we will denote by $\hat{\mathcal{A}}_i$, and each row corresponds to a worker.

Our choice of such a generator matrix $\mathbf{G} \in \mathbb{F}_q^{n\times k}$, solves

$$\begin{aligned} \arg \min_{\mathbf{G}\in\mathbb{F}_q^{n\times k}} \quad & \big\{ \operatorname{nnzr}(\mathbf{G}) \big\} \\ \text{s.t.} \quad & \|\mathbf{G}_{(i)}\|_0 \geqslant w, \ \forall i \in \mathbb{N}_n \\ & \|\mathbf{G}^{(j)}\|_0 \geqslant d, \ \forall j \in \mathbb{N}_k \\ & \operatorname{rank}(\mathbf{G}_{\mathcal{I}}) = k, \ \forall \mathcal{I} : |\mathcal{I}| = k \end{aligned} \tag{5.2}$$

which determines an optimal task allocation among the workers of the proposed CMIM. The first and second constraints are analogous to the bound of [279, Theorem 1], which is met with equality in "perfectly balanced GC schemes". This theorem states that if all rows of $\mathbf{G}$ have the same number of nonzeros, then $\|\mathbf{G}_{(i)}\|_0 \geqslant k(s+1)/n$, for all i. By construction, the generator matrix $\mathbf{G}$ we propose, meets the first and second constraints with equality, for all $i \in \mathbb{N}_n$ and $j \in \mathbb{N}_k$.

Under the above assumptions, the entries of the generator matrix of a $\mathsf{BRS}_q[n, k]$ code meet the following:

- each column is sparsest, with exactly d nonzero entries
- each row is balanced, with $w = \frac{dk}{n}$ nonzero entries

where d equals to the number of workers who are tasked to compute each block, and w is the number of blocks that are computed by each worker.

Each column $\mathbf{G}^{(j)}$ corresponds to a polynomial $p_j(\mathsf{x})$, whose entries are the evaluation of the polynomial we define in (5.3) at each of the points of the defining set $\mathcal{A}$, *i.e.* $\mathbf{G}_{ij} = p_j(\alpha_i)$ for $(i, j) \in \mathbb{N}_n \times \mathbb{N}_k$. To construct the polynomials $\{p_j(\mathsf{x})\}_{j=1}^k$, for which

[2]The case where $\frac{nw}{k} \in \mathbb{Q}_+ \backslash \mathbb{Z}_+$ is analyzed in [134], and also applies to our approach. We restrict our discussion to the case where $\frac{nw}{k} \in \mathbb{Z}_+$.

$\deg(p_j) \leqslant \operatorname{nnzr}(\mathbf{G}^{(j)}) = n - d = k - 1$, we first need to determine a sparsest and balanced *mask matrix* $\mathbf{M} \in \{0,1\}^{n\times k}$, which is ρ-sparse for $\rho = \frac{d}{n}$; *i.e.* $\operatorname{nnzr}(\mathbf{G}) = \rho nk$. We use the construction from [134], though it is fairly easy to construct more general such matrices, by using the Gale-Ryser Theorem [77, 168]. Even though this was not pointed out in [134], their construction of $\mathbf{M}$ (Algorithm 12) does not always produce a mask matrix of the given parameters when we select $d < n/2$. This is why in our work we require $d \geqslant n/2$. Furthermore, deterministic constructions resemble generator matrices of cyclic codes.

For our purposes we use $\mathcal{B} = \{\beta_j\}_{j=1}^{n}$ as our defining set of points, where each point corresponds to the worker with the same index. The objective now is to devise the polynomials $p_j(\mathsf{x})$, for which $p_j(\beta_i) = 0$ if and only if $\mathbf{M}_{ij} = 0$. Therefore:

$$\text{(I)}\ \ \mathbf{M}_{ij} = 0 \quad \Longrightarrow \quad (\mathsf{x} - \beta_i) \mid p_j(\mathsf{x})$$

$$\text{(II)}\ \ \mathbf{M}_{ij} \neq 0 \quad \Longrightarrow \quad p_j(\beta_i) \in \mathbb{F}_q^{\times}$$

for all pairs (i, j).

The construction of $\mathsf{BRS}_q[n, k]$ from [135] is based on what the authors called *scaled polynomials*. Below, we summarize the polynomial construction based on Lagrange interpolation [134]. We then prove a simple but important result that allows us to efficiently perform the decoding step.

The univariate polynomials corresponding to each column $\mathbf{G}^{(j)}$, are defined as:

$$p_j(\mathsf{x}) \coloneqq \prod_{i:\mathbf{M}_{ij}=0} \left(\frac{\mathsf{x} - \beta_i}{\beta_j - \beta_i}\right) = \sum_{\iota=1}^{k} p_{j,\iota} \cdot \mathsf{x}^{\iota-1} \in \mathbb{F}_q[\mathsf{x}] \tag{5.3}$$

which satisfy (I) and (II). By the BCH bound [207, Chapter 9], it follows that $\deg(p_j) \geqslant n - d = k - 1$ for all $j \in \mathbb{N}_k$. Since each $p_j(\mathsf{x})$ is the product of $n - d$ monomials, we conclude that the bound on the degree is satisfied and met with equality, hence $p_{j,\iota} \in \mathbb{F}_q^{\times}$ for all coefficients.

By construction, $\mathbf{G}$ is decomposable into a Vandermonde matrix $\mathbf{H} \in \mathcal{B}^{n\times k}$ and a matrix comprised of the polynomial coefficients $\mathbf{P} \in (\mathbb{F}_q^{\times})^{k\times k}$ [134]. Specifically, $\mathbf{G} = \mathbf{H}\mathbf{P}$ where $\mathbf{H}_{ij} = \beta_i^{j-1} = \beta^{i(j-1)}$ and $\mathbf{P}_{ij} = p_{j,i}$ are the coefficients from (5.3). This can be interpreted as $\mathbf{P}^{(j)}$ defining the polynomial $p_j(\mathsf{x})$, and $\mathbf{H}_{(i)}$ is comprised of the first k positive powers of β_i in ascending order, therefore

$$p_j(\beta_i) = \sum_{\iota=1}^{k} p_{j,\iota} \cdot \beta_i^{\iota-1} = \langle \mathbf{H}_{(i)}, \mathbf{P}^{(j)} \rangle.$$

The following lemmas will help us respectively establish in our CC setting the efficiency of our decoding step and the optimality of the allocated tasks to the workers. For Lemma 5.3.1, recall that efficient matrix multiplication algorithms have complexity $\mathcal{O}(N^\omega)$, for $\omega < 2.373$ the *matrix multiplication exponent* [11].

Lemma 5.3.1. *The restriction $\mathbf{G}_\mathcal{I} \in \mathbb{F}_q^{k\times k}$ of $\mathbf{G}$ to any of its k rows indexed by $\mathcal{I} \subsetneq \mathbb{N}_n$, is an invertible matrix. Moreover, its inverse can be computed online in $\mathcal{O}(k^2 + k^\omega)$ operations.*

Proof. The matrices $\mathbf{H}$ and $\mathbf{P}$ are of size $n \times k$ and $k \times k$ respectively. The restricted matrix $\mathbf{G}_\mathcal{I}$ is then equal to $\mathbf{H}_\mathcal{I}\mathbf{P}$, where $\mathbf{H}_\mathcal{I} \in \mathbb{F}_q^{k\times k}$ is a square Vandermonde matrix, which is invertible in $\mathcal{O}(k^2)$ time [26]. Specifically

$$\mathbf{H}_\mathcal{I} = \begin{pmatrix} 1 & \beta_{\mathcal{I}_1} & \beta_{\mathcal{I}_1}^2 & \dots & \beta_{\mathcal{I}_1}^{k-1} \\ 1 & \beta_{\mathcal{I}_2} & \beta_{\mathcal{I}_2}^2 & \dots & \beta_{\mathcal{I}_2}^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \beta_{\mathcal{I}_k} & \beta_{\mathcal{I}_k}^2 & \dots & \beta_{\mathcal{I}_k}^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k\times k}.$$

It follows that

$$\det(\mathbf{H}_\mathcal{I}) = \prod_{\{i<j\}\subseteq\mathcal{I}} (\beta_j - \beta_i)$$

which is nonzero, since β is primitive. Therefore, $\mathbf{H}_\mathcal{I}$ is invertible. By [135, Lemma 1] and the BCH bound, we conclude that $\mathbf{P}$ is also invertible. Hence, $\mathbf{G}_\mathcal{I}$ is invertible for any set $\mathcal{I}$.

Note that the inverse of $\mathbf{P}$ can be computed a priori by the master before we deploy our CCM. Therefore, computing $\mathbf{G}_\mathcal{I}^{-1}$ online with knowledge of $\mathbf{P}^{-1}$, requires an inversion of $\mathbf{H}_\mathcal{I}$ which takes $\mathcal{O}(k^2)$ operations; and then multiplying it by $\mathbf{P}^{-1}$. Thus, it requires $\mathcal{O}(k^2 + k^\omega)$ operations in total. □

Lemma 5.3.2. *The generator matrix $\mathbf{G} \in \mathbb{F}_q^{n\times k}$ of a $\mathsf{BRS}_q[n, k]$ MDS code defined by the polynomials $p_j(\mathsf{x})$ of (5.3), solves the optimization problem (5.2).*

Proof. The first two constraints are satisfied by the construction of $\mathbf{G}$, which meets the sparsest and balanced constraints with equality; for the given parameters. The last constraint is implied by the MDS theorem, which states that every set of k rows of $\mathbf{G}$ is linearly independent.

The sparsity constraints of (5.2) imply that $\mathrm{nnzr}(\mathbf{G}) \geqslant \max\{nw, kd\}$, and for our parameters we have $nw = kd$. Both the first and second constraints are met with

equality for the chosen $\mathbf{G}$. Moreover

$$\begin{aligned}
\operatorname{nnzr}(\mathbf{G}) &= \sum_{j \in \mathbb{N}_k} \operatorname{nnzr}(\mathbf{G}^{(j)}) \\
&= \sum_{j \in \mathbb{N}_k} \#\big\{p_j(\beta_i) \neq 0 : \beta_i \in \mathcal{B}\big\} \\
&= \sum_{j \in \mathbb{N}_k} n - \big\{i : \mathbf{M}_{ij} = 0\big\} \\
&= \sum_{j \in \mathbb{N}_k} n - (n - d) \\
&= kd
\end{aligned}$$

and the proof is complete. □

We conclude this section by recalling how the decomposition $\mathbf{G} = \mathbf{HP}$ is utilized for GC [134]. Each column of $\mathbf{G}$ corresponds to a partition of the data whose partial gradient is to be computed. The polynomials are judiciously constructed in this scheme, such that the constant term of each polynomial is 1, thus $\mathbf{P}_{(1)} = \vec{\mathbf{1}}$. By this, the decoding vector $\mathbf{a}_{\mathcal{I}}^{\top}$ is the first row of $\mathbf{H}_{\mathcal{I}}^{-1}$, for which $\mathbf{a}_{\mathcal{I}}^{\top}\mathbf{H}_{\mathcal{I}} = \mathbf{e}_1^{\top}$. A direct consequence of this is that $\mathbf{a}_{\mathcal{I}}^{\top}\mathbf{G}_{\mathcal{I}} = \mathbf{e}_1^{\top}\mathbf{P} = \mathbf{P}_{(1)} = \vec{\mathbf{1}}$, which is the objective for constructing a GC scheme.

5.4 Inverse Approximation Algorithm

Our goal is to estimate $\mathbf{A}^{-1} = \big[\mathbf{b}_1 \ \cdots \ \mathbf{b}_N\big]$, for $\mathbf{A}$ a square matrix of order N. A key property to note is

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}\big[\mathbf{b}_1 \ \cdots \ \mathbf{b}_N\big] = \big[\mathbf{A}\mathbf{b}_1 \ \cdots \ \mathbf{A}\mathbf{b}_N\big] = \mathbf{I}_N$$

which implies that $\mathbf{A}\mathbf{b}_i = \mathbf{e}_i$ for all $i \in \mathbb{N}_N$, where $\mathbf{e}_i$ are the standard basis column vectors. Assume for now that we use any black-box least squares solver to estimate

$$\hat{\mathbf{b}}_i \approx \arg\min_{\mathbf{b} \in \mathbb{R}^N} \Big\{ g_i(\mathbf{b}) \coloneqq \|\mathbf{A}\mathbf{b} - \mathbf{e}_i\|_2^2 \Big\} \tag{5.4}$$

which we call N times, to recover $\widehat{\mathbf{A}^{-1}} \coloneqq \big[\hat{\mathbf{b}}_1 \ \cdots \ \hat{\mathbf{b}}_N\big]$. This approach may be viewed as approximating

$$\widehat{\mathbf{A}^{-1}} \approx \arg \min_{\mathbf{B} \in \mathbb{R}^{N \times N}} \left\{ \|\mathbf{A}\mathbf{B} - \mathbf{I}_N\|_F^2 \right\}.$$

Alternatively, one could estimate the rows of $\mathbf{A}^{-1}$. Algorithm 8 shows how this can be performed by a single server.

Algorithm 8: Estimating $\mathbf{A}^{-1}$

Input: $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$
for *i=1 to N* **do**
  approximate $\hat{\mathbf{b}}_i \approx \arg\min_{\mathbf{b}\in\mathbb{R}^N} \{\|\mathbf{A}\mathbf{b} - \mathbf{e}_i\|_2^2\}$
end
return $\widehat{\mathbf{A}^{-1}} \leftarrow \left[\hat{\mathbf{b}}_1 \ \cdots \ \hat{\mathbf{b}}_N\right]$

In the case where SD is used to approximate $\hat{\mathbf{b}}_i$ from (5.4), the overall operation count is $\mathcal{O}(\mathcal{T}_i N^2)$; for $\mathcal{T}_i$ the total number of descent iterations used. An upper bound on the number of iterations can be determined by the underlying termination criterion, *e.g.* the criterion $g_i(\hat{\mathbf{b}}^{[t]}) - g_i(\mathbf{b}_{ls}^{\star}) \leqslant \epsilon$ is guaranteed to be satisfied after $\mathcal{T} = \mathcal{O}(\log(1/\epsilon))$ iterations [36]. The overall error of $\widehat{\mathbf{A}^{-1}}$ may be quantified as

- $\mathrm{err}_{\ell_2}(\widehat{\mathbf{A}^{-1}}) \coloneqq \|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_2$
- $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) \coloneqq \|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F$
- $\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}}) \coloneqq \frac{\|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F}{\|\mathbf{A}^{-1}\|_F} = \frac{\left(\sum_{i=1}^{N} \|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2\right)^{1/2}}{\|\mathbf{A}^{-1}\|_F}$.

To approximate $\mathbf{A}^{-1}$ distributively, each of the n workers are asked to estimate τ-many $\hat{\mathbf{b}}_i$'s in parallel. When using SD, the worst-case runtime by the workers is $\mathcal{O}(\tau\mathcal{T}_{\max}N^2)$, for $\mathcal{T}_{\max}$ the maximum number of iterations of SD among the workers. If CG is used, each worker needs no more than a total of $N\tau$ CG steps to exactly compute its task, *i.e.* $\mathcal{O}(\tau N\kappa_2)$ operations; as each instance of (5.4) is expected to converge in $\mathcal{O}(\kappa_2)$ iterations, which is the worst case runtime [261, 281].

In order to bound $\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}}) = \frac{\|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F}{\|\mathbf{A}^{-1}\|_F}$, we first upper bound the numerator and then lower bound the denominator. Since $\|\mathbf{A}^{-1} - \widehat{\mathbf{A}^{-1}}\|_F^2 = \sum_{i=1}^{N} \|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$, bounding the numerator reduces to bounding $\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$ for all $i \in \mathbb{N}_N$. This is straightforward

$$\begin{aligned}
\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2 &\overset{\diamondsuit}{\leqslant} 2\left(\|\mathbf{A}^{-1}\mathbf{e}_i\|_2^2 + \|\hat{\mathbf{b}}_i\|_2^2\right) \\
&\overset{\$}{\leqslant} 2\left(\|\mathbf{A}^{-1}\|_2^2 \cdot \|\mathbf{e}_i\|_2^2 + \|\hat{\mathbf{b}}_i\|_2^2\right) \\
&= 2\left(1/\sigma_{\min}(\mathbf{A})^2 + \|\hat{\mathbf{b}}_i\|_2^2\right) \qquad (5.5)
\end{aligned}$$

where in $\diamondsuit$ we use the fact that $\|\mathbf{u}-\mathbf{v}\|_2^2 \leqslant 2(\|\mathbf{u}\|_2^2+\|\mathbf{v}\|_2^2)$, and in \$ the submultiplicativity of the ℓ_2-norm is invoked. For the denominator, by the definition of the Frobenius norm

$$\|\mathbf{A}^{-1}\|_F^2 = \sum_{i=1}^{N} \frac{1}{\sigma_i(\mathbf{A})^2} \geqslant \frac{N}{\sigma_{\max}(\mathbf{A})^2} \ . \tag{5.6}$$

By combining (5.5) and (5.6) we get

$$\begin{aligned}\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) &\leqslant \sqrt{2}\left(\frac{N/\sigma_{\min}(\mathbf{A})^2+\sum_{i=1}^{N}\|\hat{\mathbf{b}}_i\|_2^2}{N/\sigma_{\max}(\mathbf{A})^2}\right)^{1/2} \\ &= \sqrt{2}\left(\kappa_2^2+\frac{\sigma_{\max}(\mathbf{A})^2}{N}\cdot\sum_{i=1}^{N}\|\hat{\mathbf{b}}_i\|_2^2\right)^{1/2} .\end{aligned}$$

This is an additive error bound in terms of the problem's condition number, which also shows a dependency on the estimates $\{\hat{\mathbf{b}}_i\}_{i=1}^N$. Propositions 5.4.1 and 5.4.2 give error bounds when using SD and CG as the subroutine of Algorithm 8 respectively.

Proposition 5.4.1. *For $\mathbf{A}\in\mathrm{GL}_N(\mathbb{R})$, we have $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) \leqslant \frac{\epsilon\sqrt{N/2}}{\sigma_{min}(\mathbf{A})^2}$ and $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) \leqslant \frac{\epsilon\sqrt{N/2}}{\sigma_{\min}(\mathbf{A})}$, when using SD to solve (5.4) with termination criteria $\|\nabla g_i(\mathbf{b}^{[t]})\|_2 \leqslant \epsilon$ for each i.*

Proof. Recall that for a strongly-convex function with strong-convexity parameter μ, we have the following optimization gap [36, Section 9.1.2]

$$g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^{\star}) \leqslant \frac{1}{2\mu}\cdot\|\nabla g_i(\mathbf{b})\|_2^2 \ . \tag{5.7}$$

For $\mathbf{A}\in\mathrm{GL}_N(\mathbb{R})$ in (5.4), the constant is $\mu=\sigma_{\min}(\mathbf{A})^2$. By fixing $\epsilon=\sqrt{2\sigma_{\min}(\mathbf{A})^2\eta}$, we have $\eta=\frac{1}{2}\cdot\left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2$. Thus, by (5.7) and our termination criterion:

$$\|\nabla g_i(\mathbf{b})\|_2 \leqslant \sqrt{2\sigma_{\min}(\mathbf{A})^2\eta} \quad\Longrightarrow\quad g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^{\star}) \leqslant \eta \ ,$$

so when solving (5.4) we get

$$g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^{\star}) = g_i(\mathbf{b}) - 0 = \|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2 \ ,$$

hence

$$\|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2 \leqslant \frac{1}{2}\cdot\left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \tag{5.8}$$

for all $i \in \mathbb{N}_N$. We want an upper bound for each summand $\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$ of the numerator of $\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}})^2$:

$$\begin{aligned}
\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2 &= \|\mathbf{A}^{-1}(\mathbf{e}_i - \mathbf{A}\hat{\mathbf{b}}_i)\|_2^2 \\
&\leqslant \|\mathbf{A}^{-1}\|_2^2 \cdot \|\mathbf{e}_i - \mathbf{A}\hat{\mathbf{b}}_i\|_2^2 \\
&\overset{\sharp}{\leqslant} \|\mathbf{A}^{-1}\|_2^2 \cdot \frac{1}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \qquad (5.9) \\
&= \frac{\epsilon^2}{2\sigma_{\min}(\mathbf{A})^4} \qquad (5.10)
\end{aligned}$$

where $\sharp$ follows from (5.8), thus $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}})^2 \leqslant \frac{N\epsilon^2}{2\sigma_{\min}(\mathbf{A})^4}$. Substituting (5.9) into the definition of $\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}})$ gives us

$$\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}})^2 \leqslant \frac{\|\mathbf{A}^{-1}\|_2^2}{\|\mathbf{A}^{-1}\|_F^2} \cdot \frac{N}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \overset{\ddagger}{\leqslant} \frac{N\epsilon^2/2}{\sigma_{\min}(\mathbf{A})^2}$$

where $\ddagger$ follows from the fact that $\|\mathbf{A}^{-1}\|_2^2 \leqslant \|\mathbf{A}^{-1}\|_F^2$. □

In the experiments of Subsection 5.4.1, we verify that Proposition 5.4.1 holds for Gaussian random matrices. The dependence on $1/\sigma_{\min}(\mathbf{A})$ is an artifact of using gradient methods to solve the underlying problems (5.4), since the error will be multiplied by $\|\mathbf{A}^{-1}\|_2^2$. In theory, this can be annihilated if one runs the algorithm on $p\mathbf{A}$ for $p \approx 1/\sigma_{\min}(\mathbf{A})$, followed by multiplication of the final result by p. This is a way of preconditioning SD. In practice, the scalar p should not be selected to be much larger than $1/\sigma_{\min}(\mathbf{A})$, as it could result in $\widehat{\mathbf{A}^{-1}} \approx \mathbf{0}_{N\times N}$.

Proposition 5.4.2. *Assume Algorithm 8 uses CG to solve (5.4). Then, in no more than $\mathcal{O}\left(N\sqrt{\kappa_2}\ln(1/\epsilon)\right)$ iterations, we have $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) \leqslant N\epsilon$. Moreover, if $\mathbf{A}^\top\mathbf{A}$ has $\tilde{N}$ distinct eigenvalues, it converges in at most $\tilde{N}N$ steps.*

Proof. By [261, Section 10] and [38, Section 2], we know that for each subroutine (5.4) of Algorithm 13, CG requires at most $\mathcal{O}(\sqrt{\kappa_2}\ln(1/\epsilon))$ iterations in order to attain an ϵ-optimal point, for each $\hat{\mathbf{b}}_i$. Hence, considering all approximate columns $\{\hat{\mathbf{b}}_i\}_{i=1}^N$, we conclude that the total error in terms of the Frobenius norm of $\widehat{\mathbf{A}^{-1}}$, is at most $N\epsilon$.

Recall that in order to solve (5.1) with the CG method in the case where $\mathbf{A}$ is neither symmetric, positive-definite, nor square, we apply the CG iteration to the normal equations

$$\mathbf{A}^\top\mathbf{A}\mathbf{y} = \mathbf{A}^\top\theta \ .$$

This follows by setting the derivative of (5.1) to zero. In our scenario, we are assuming that $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$, hence $\mathbf{A}^\top \mathbf{A}$ is full-rank and symmetric, thus CG in its simplest form can be used to solve the minimization problems of Algorithm 8. By [281, Theorem 38.4], it follows that each instance of (5.4) converges in at most $\tilde{N}$ steps. □

Even though Proposition 5.4.2 guarantees convergence in at most $\tilde{N}N$ steps, it does not assume finite floating-point precision. Therefore, this does not hold in practical settings. Our experiments though show that after significantly less steps, we achieve approximations of negligible error, which is sufficient for ML and FL applications.

5.4.1 Numerical Experiments

The accuracy of the proposed algorithm was tested on randomly generated matrices, using both SD and CG for the subroutine optimization problems. The depicted results are averages of 20 runs, with termination criteria $\|\nabla g_i(\mathbf{b}^{[t]})\|_2 \leqslant \epsilon$ for SD and $\|\mathbf{b}_i^{[t]} - \mathbf{b}_i^{[t-1]}\|_2 \leqslant \epsilon$ for CG, for the given ϵ accuracy parameters. We considered $\mathbf{A} \in \mathbb{R}^{100\times 100}$. The error subscripts represent $\mathscr{A} = \{\ell_2, F, \mathrm{r}F\}$, $\mathscr{N} = \{\ell_2, F\}$. We note that significantly fewer iterations took place when CG was used for the same ϵ, though this depends heavily on the choice of the step-size. The errors observed in the case of CG, are due to floating-point arithmetic. Therefore, as expected; there is a trade-off between accuracy and speed when using SD vs. CG.

Average $\widehat{\mathbf{A}^{-1}}$ errors, for $\mathbf{A} \sim 50 \cdot \mathcal{N}(0,1)$ — SD					
ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
$\mathrm{err}_{\mathscr{A}}$	$\mathcal{O}(10^{-2})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-7})$	$\mathcal{O}(10^{-9})$	$\mathcal{O}(10^{-12})$

Table V.1: Numerical experiments for $\widehat{\mathbf{A}^{-1}}$.

Average $\widehat{\mathbf{A}^{-1}}$ errors, for $\mathbf{A} \sim 50 \cdot \mathcal{N}(0,1)$ — CG					
ϵ	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$\mathrm{err}_{\mathscr{N}}$	$\mathcal{O}(10^{-3})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-8})$	$\mathcal{O}(10^{-11})$	$\mathcal{O}(10^{-12})$
$\mathrm{err}_{\mathrm{r}F}$	$\mathcal{O}(10^{-3})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-7})$	$\mathcal{O}(10^{-10})$	$\mathcal{O}(10^{-12})$

Table V.2: Numerical experiments for $\widehat{\mathbf{A}^{\dagger}}$.

We utilized Algorithm 8 in Newton's method, for classifying images of four and nine from MNIST, by solving a regularized logistic regression minimization problem. For Algorithm 8, we used CG with a fixed number of iteration per column estimation. It is clear from Figure V.1 that we require no more than 18 iterations per column estimate, for $N = 785$, to attain the optimal classification rate. With more than 18 CG iterations, the same classification rate was obtained.

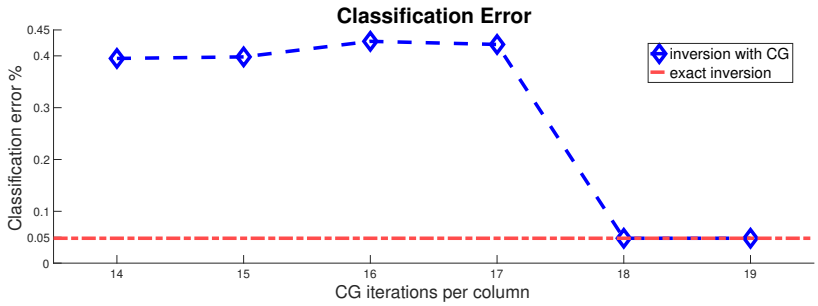


Figure V.1: MNIST classification error, where Algorithm 8 is used in Newton's method. In red, we depict the error when exact inversion was used.

5.5 Secure Coded Matrix Inversion

In this section, we describe the proposed CMIM (also presented in [54]) which makes Algorithm 8 resilient to stragglers, and show how it can be applied to the relaxed FL scenario described in the introduction. The CMIM workflow is depicted in Figure V.2.

Our scheme can be broken up in to four phases: (a) the coordinator shares elements $\beta, \mathcal{H}$ of a finite field with all the clients, (b) the clients each generate a *pseudorandom permutation* (PRP) σ_ι, encrypt their corresponding data block $\mathbf{A}_\iota$ through a matrix polynomial $f_\iota(\mathsf{x})$, and broadcast $\{f_\iota(\mathsf{x}), \sigma_\iota\}$ to the other clients, (c) the clients recover $\mathbf{A}$, compute and encode their assigned task $\mathbf{W}_\iota$, which is communicated to the coordinator, (d) the coordinator decodes once sufficiently many workers respond. It is also possible that $\beta, \mathcal{H}$ are determined collectively by the clients, or by a single client, which makes the data sharing secure against a curious and dishonest coordinator.

In our proposed FL approach, we assume there is a trustworthy coordinator who shares certain parameters to each of the k clients which constitute the network; *e.g.* hospitals in a health care network, each of which are comprised of multiple servers. What we present works for the case where the clients have local datasets of different sizes, $\{N_i\}_{i=1}^k$. This would result in the encoding functions $f_\iota(\mathsf{x})$ having different

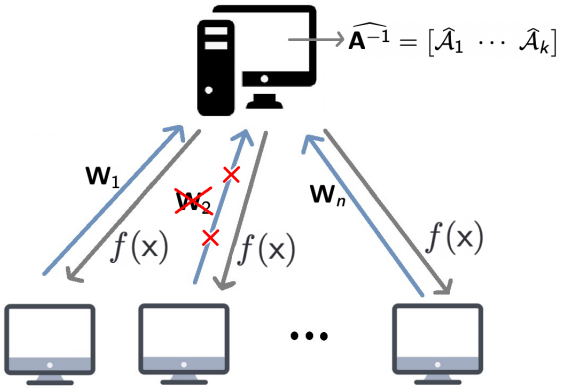


Figure V.2: Algorithmic workflow of the CMIM, as proposed in [54]. The master shares $f(\mathsf{x})$, an encoding analogous to (5.12), along with $\beta, \{\eta_j^{-1}\}_{j=1}^k$. The workers then recover $\mathbf{A}$, compute their assigned tasks, and encode them according to $\mathbf{G}$. Once k encodings $\mathbf{W}_\iota$ are sent back, $\widehat{\mathbf{A}^{-1}}$ can be recovered.

degrees, or their matrix coefficients being of a different size. In our setting we assume the workers are *homogeneous*, *i.e.* they have the same computational power. Therefore, equal computational loads are assigned to each of them. In order to keep the notation and size of the communication loads consistent, we assume w.l.o.g. that $\mathbf{A}_\iota \in \mathbb{R}^{N\times T}$ for all $\iota \in \mathbb{N}_k$. If this is not the case, before $f_\iota(\mathsf{x})$ are determined, the clients could perform a data exchange phase (*e.g.* [295]), so that $N_i = N_j$ for all $i \neq j$. By this, it follows that the number of blocks does not have to be equal to the number of clients. The example we describe, is simply a motivation. A flowchart of our approach is presented in Figure V.3.

Moreover, in the case where $M > N$; for $M = \sum_{i=1}^k N_i$, we can select a subset of features and/or samples, so that the resulting data matrix we consider is square. This can be interpreted as using the surrogate $\tilde{\mathbf{A}} = \mathbf{SA}$, where $\mathbf{S} \in \mathbb{R}^{N\times M}$ is an appropriate (sparse) *sketching* matrix for matrix inversion [125], which the workers agree on.

First, in 5.5.1 we argue why all of $\mathbf{A}$ needs to be known by each of the workers, in order to recover entries or columns of its inverse. Then, in 5.5.2 we focus on phases (a) and (b), where we utilize Lagrange interpolation to securely share $\mathbf{A}$ among the workers. We discuss the computation tasks the workers are requested to compute, which are blocks of $\widehat{\mathbf{A}^{-1}}$; and collectively correspond to the subroutine problems of Algorithm 8. In 5.5.3 we focus on (c) and (d), where we show how the workers encode their computations, and describe the coordinator's decoding step. Optimality of BRS

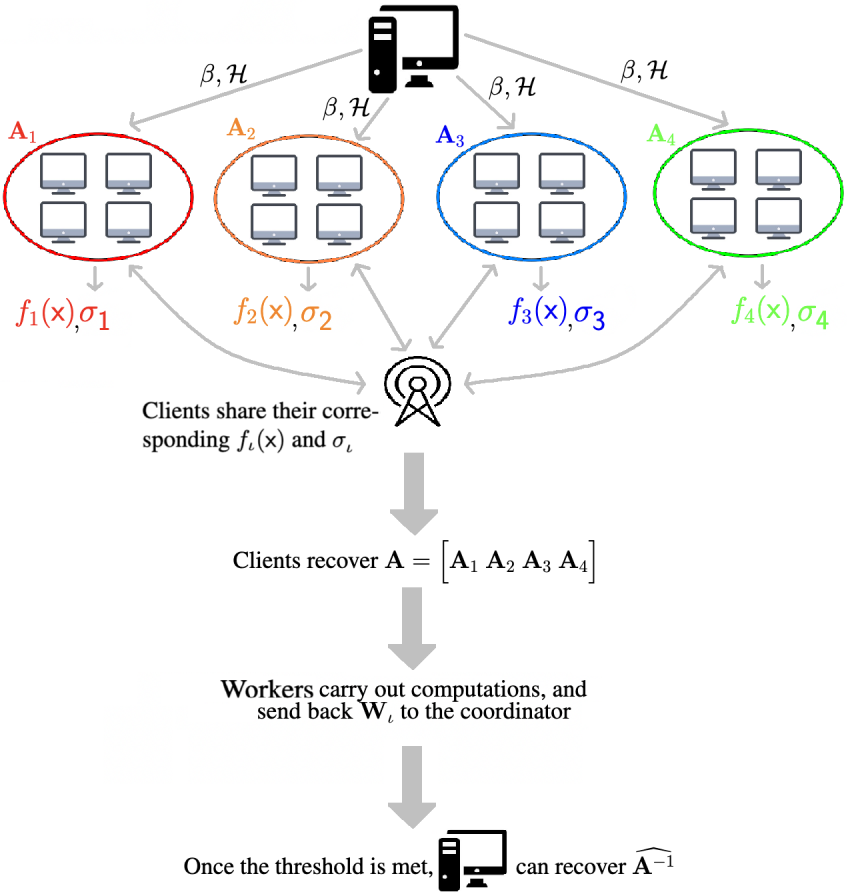


Figure V.3: Flowchart of our proposal, where $k = n_i = 4$ for all $i \in \mathbb{N}_4$.

generator matrices in terms of the encoded communication loads is established in 5.5.4.

When assuming no floating-point errors, our approach introduces no numerical nor approximation errors. The errors are a consequence of using iterative solvers to estimate (5.4), which we utilize to linearly separate the computations. Therefore, if the workers can recover the optimal solutions to the underlying minimization problems, our scheme would be *exact*.

5.5.1 Knowledge of A is necessary

A bottleneck when computing the inverse of a matrix; or estimating its columns, is that the entire matrix needs to be known. A single change in the matrix's entries may result in a non-singular matrix, which conveys how sensitive Gaussian elimination is. Such problems are extensively studied in conditioning and stability of numerical analysis [281], and in perturbation theory. This is not a focus of our work.

In the case where only one column is not known, one can determine the subspace in which the missing column lies, but without the knowledge of at least one entry of that column, it would be impossible to recover that column. Even with such an approach or a matrix completion algorithm, the entire $\mathbf{A}$ is determined before we proceed to inverting $\mathbf{A}$; or performing linear regression to approximate $\mathbf{Ab} = \mathbf{e}_i$ as in (5.4).

A similar issue, relating to our FL set up, is the case where one of the blocks is different. This could lead to drastic miscalculations. In the following example, we consider $n = k = 2$ and $N = 4$, where the second worker sends two different blocks, which are indicated by a different color and font:

$$\boldsymbol{A}_1 = \begin{pmatrix} 6 & 2 & 2 & -5 \\ 0 & -1 & 2 & 0 \\ -5 & 6 & -1 & -3 \\ 5 & -3 & -4 & 3 \end{pmatrix} \qquad \boldsymbol{A}_2 = \begin{pmatrix} 6 & 2 & -1 & -3 \\ 0 & -1 & 5 & 6 \\ -5 & 6 & 3 & -2 \\ 5 & -3 & 1 & 6 \end{pmatrix} .$$

It follows that $\|\boldsymbol{A}_1^{-1}\|_F \approx 90.45$, $\|\boldsymbol{A}_2^{-1}\|_F \approx 1$, and $\|\boldsymbol{A}_1^{-1} - \boldsymbol{A}_2^{-1}\|_0 = 16$; *i.e.* no entries of $\boldsymbol{A}_1^{-1}$ and $\boldsymbol{A}_2^{-1}$ are equal.

Furthermore, by the data processing inequality [71, Corollary pg.35], the above imply that no less than N^2 information symbols can be known by each worker, while hoping to approximate a column of $\mathbf{A}^{-1}$. Hence, all clients need full knowledge of each others information, and cannot communicate less than NT symbols to each other. This is a consequence of the fact that a dense vector is not recoverable from underdetermined linear measurements. They can however send an encoded version of their respective block $\mathbf{A}_\iota \in \mathbb{R}^{N\times T}$ to the other clients consisting of NT symbols, determined by a modified Lagrange polynomial, which guarantees security against eavesdroppers.

Similar cryptographic protocols date back to the SSS algorithm [27, 258], which is also based on RS codes (for details on recent developments and variants of SSS, refer to [63, 268]). This idea has been extensively exploited in LCC [302], yet differs from our approach.

5.5.2 Phases (a), (b) — Data Encryption and Sharing

Let $k, \gamma \in \mathbb{Z}_+$ be factors of N and T respectively, so that $T = \frac{N}{k}$ and $\Gamma = \frac{T}{\gamma}$.[3] The coordinator constructs a set of distinct *interpolation points* $\mathcal{B} = \{\beta_j\}_{j=1}^{n} \subsetneq \mathbb{F}_q^{\times}$, for $q > n \geqslant \gamma$.[4] To construct this set, it suffices to sample $\beta \in \mathbb{F}_q^{\times}$; any one of the $\phi(q-1)$ primitive roots of $\mathbb{F}_q$ (ϕ is Euler's totient function), which is a generator of the multiplicative group $(\mathbb{F}_q^{\times}, \cdot)$, and define each point as $\beta_j = \beta^j$. Then, a random multiset $\mathcal{H} = \left\{\eta_j \in \mathbb{F}_q^{\times} \mid \forall j \in \mathbb{N}_\gamma\right\}$ of size γ is generated, *i.e.* repetitions in $\mathcal{H}$ are allowed, which will be used to remove the structure of the Lagrange coefficients, as the adversaries could exploit their structure to reveal β.

The element β and set $\mathcal{H}$, are broadcasted securely to all the workers through a public-key cryptosystem, *e.g.* RSA [247] or McEliece [206]. Matrices $\mathbf{A}_\iota$ are partitioned into γ blocks

$$\mathbf{A}_\iota = \left[\mathbf{A}_\iota^1 \ \cdots \ \mathbf{A}_\iota^\gamma\right] \quad \text{where } \mathbf{A}_\iota^i \in \mathbb{R}^{N\times\Gamma}, \ \forall i \in \mathbb{N}_\gamma, \tag{5.11}$$

and each client generates a PRP $\sigma_\iota \in \ S_\gamma$. The blocks $\{\mathbf{A}_\iota\}_{\iota=1}^{k}$ are encrypted locally through the univariate polynomials

$$f_\iota(\mathsf{x}) = \sum_{j=1}^{\gamma} \mathbf{A}_\iota^j \cdot \eta_{\sigma_\iota(j)} \left(\prod_{l\neq j} \frac{\mathsf{x} - \beta_l}{\beta_j - \beta_l}\right) \tag{5.12}$$

for which $f_\iota(\beta_j) = \eta_{\sigma_\iota(j)}\mathbf{A}_\iota^j$.

The clients securely broadcast $\{f_\iota(\mathsf{x}), \sigma_\iota\}$ to each other, and their servers can then recover all $\mathbf{A}_\iota$'s as follows:

$$\mathbf{A}_\iota = \left[\eta_{\sigma_\iota(1)}^{-1} f_\iota(\beta_1) \ \cdots \ \eta_{\sigma_\iota(\gamma)}^{-1} f_\iota(\beta_\gamma)\right] \in \mathbb{R}^{N\times T}. \tag{5.13}$$

The coefficients of $f_\iota(\mathsf{x})$ are comprised of $N\Gamma$ symbols, thus, each polynomial consists of a total of NT symbols, which is the minimum number of symbols needed to be communicated. The PRP σ_ι is generated locally by the clients, to ensure that each $f_\iota(\mathsf{x})$ differs by more than just the matrix partitions.

We assume Kerckhoffs' principle, which states that everyone has knowledge of the system, including the messages $f_\iota(\mathsf{x})$. For the proposed CMIM, as long as $\{\beta, \mathcal{H}\}$

[3] If $\gamma \nmid T$, append $\mathbf{0}_{T\times 1}$ to the end of the first $\tilde{\gamma} = T(\mathrm{mod}\gamma)$ blocks which are each comprised of $\tilde{\Gamma} = \lfloor \frac{T}{\gamma} \rfloor$ columns of $\mathbf{A}_\iota$, while the remaining $\gamma - \tilde{\gamma}$ blocks are comprised of $\tilde{\Gamma} + 1$ columns. Now, each block is of size $T \times (\tilde{\Gamma} + 1)$.

[4] For the encodings of the $\mathbf{A}_\iota$'s, γ points suffice, and we only need to require $q > \gamma$. We select $\mathcal{B}$ of cardinality n and require $q > n \geqslant \gamma$, in order to reuse $\mathcal{B}$ in our CCM.

and σ_ι are securely communicated, even if $f_\iota(\mathsf{x})$ is revealed, the block $\mathbf{A}_\iota$ is secure against polynomial-bounded adversaries (this is the security level assumed by the cryptosystems used for the communication).

Proposition 5.5.1. *The encryptions of $\mathbf{A}_\iota$ through $f_\iota(\mathsf{x})$, are as secure against eavesdroppers as the public-key cryptosystems which are used when broadcasting $\{\beta, \mathcal{H}\}$ and σ_ι. To recover $\mathbf{A}_\iota$, an adversary needs to intercept both communications, and break both cryptosystems.*

Proof. We prove this by contradiction. Assume that an adversary was able to reverse the encoding $f_\iota(\mathsf{x})$ of $\mathbf{A}_\iota$. This implies that he was able to reveal β and $\sigma_\iota(\mathcal{H}) \coloneqq \{\eta_{\sigma_\iota(j)}\}_{j=1}^{\gamma}$. The only way to reveal these elements, is if he was able to both intercept and decipher the public-key cryptosystem used by the coordinator, which contradicts the security of the cryptosystem.

In order to invert the multiplications of $\sigma_\iota(\mathcal{H})$ for each of the evaluations of $f_\iota(\mathsf{x})$, both $\mathcal{H}$ and σ_ι need to be known. To do so, the adversary needs to intercept both the communication between the coordinator and the clients, and the communication between the clients, as well as breaking both the cryptosystems used to securely carry out these communications. □

5.5.3 Phases (c), (d) — Computations, Encoding and Decoding

At this stage, the workers have knowledge of everything they need in order to recover $\mathbf{A}$, before they carry out their computation tasks. By (5.13), the recovery is straightforward.

For Algorithm 8, any CCM in which the workers compute an encoding of partitions of the resulting computation $\mathbf{E} = \begin{bmatrix} E_1 & \cdots & E_k \end{bmatrix}$ could be utilized. It is crucial that the encoding takes place on the computed tasks $\{E_i\}_{i=1}^{k}$ in the scheme, and *not* the assigned data or partitions of the matrices that are being computed over (such CMM leverage the linearity of matrix multiplication), otherwise the algorithm could potentially not return the correct approximation. This also means that utilizing such encryption approaches (*e.g.* [302]) for guaranteeing security against the workers, is not an option. We face these restrictions due to the fact that matrix inversion is a non-linear operator.

The computation tasks E_i correspond to a partitioning $\widehat{\mathbf{A}^{-1}} = \begin{bmatrix} \hat{\mathcal{A}}_1 & \cdots & \hat{\mathcal{A}}_k \end{bmatrix}$, of our approximation from Algorithm 8. We propose a linear encoding of the computed blocks $\{\hat{\mathcal{A}}_i\}_{i=1}^{k}$ based on generators satisfying (5.2). Along with the proposed decoding step, we have a MDS-based CCM for matrix inversion.

We consider the same parameters as in 5.5.2, in order to reuse $\mathcal{B}$ in the proposed CMIM. Each $\hat{\mathcal{A}}_i$ is comprised of T distinct but consecutive approximations of (5.4), *i.e.*

$$\hat{\mathcal{A}}_i = \left[\hat{\mathbf{b}}_{(i-1)T+1} \ \cdots \ \hat{\mathbf{b}}_{iT}\right] \in \mathbb{R}^{N\times T} \quad \forall i \in \mathbb{N}_k,$$

which could also be approximated by iteratively solving

$$\hat{\mathcal{A}}_i \approx \arg\min_{\mathbf{B}\in\mathbb{R}^{N\times T}} \left\{ \left\| \mathbf{AB} - \left[\mathbf{e}_{(i-1)T+1} \ \cdots \ \mathbf{e}_{iT}\right] \right\|_F^2 \right\}.$$

Without loss of generality, we assume that the workers use the same algorithms and parameters for estimating the columns $\{\hat{\mathbf{b}}_i\}_{i=1}^N$. Therefore, workers allocated the same tasks are expected to get equal approximations in the same amount of time.

For our CCM, we leverage BRS generator matrices for both the encoding and decoding steps. We adapt the GC framework, so we need an analogous condition to $\mathbf{a}_{\mathcal{I}}^\top \mathbf{G}_{\mathcal{I}} = \vec{\mathbf{1}}$ for the CMIM; in order to invoke Algorithm 8. The condition we require is $\tilde{\mathbf{D}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}} = \mathbf{I}_N$, for an encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.

From our discussion on BRS codes in 5.3.1, we set $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$ and $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$ for any given set of k responsive workers indexed by $\mathcal{I}$. The index set of blocks requested from the ι^{th} worker to compute is $\mathcal{J}_\iota \coloneqq \mathrm{supp}(\mathbf{G}_{(\iota)})$, and has cardinality w. The workers' encoding steps correspond to

$$\tilde{\mathbf{G}} \cdot (\widehat{\mathbf{A}^{-1}})^\top = (\mathbf{I}_T \otimes \mathbf{G}) \cdot \begin{bmatrix} \hat{\mathcal{A}}_1^\top \\ \vdots \\ \hat{\mathcal{A}}_k^\top \end{bmatrix} = \begin{pmatrix} \sum_{j\in\mathcal{J}_1} p_j(\beta_1) \cdot \hat{\mathcal{A}}_j^\top \\ \vdots \\ \sum_{j\in\mathcal{J}_n} p_j(\beta_n) \cdot \hat{\mathcal{A}}_j^\top \end{pmatrix} \tag{5.14}$$

which are carried out locally, once they have computed their assigned tasks. We denote the encoding of the ι^{th} worker by $\mathbf{W}_\iota \in \mathbb{C}^{T\times N}$, *i.e.* $\mathbf{W}_\iota = \sum_{j\in\mathcal{J}_\iota} p_j(\beta_\iota) \cdot \hat{\mathcal{A}}_j^\top$, which is sent to the coordinator. The received encoded computations by any distinct k workers indexed by $\mathcal{I}$, constitute $\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$.

Lemma 5.3.1 implies that as long as k workers respond, the approximation $\widehat{\mathbf{A}^{-1}}$ is recoverable. Moreover, the decoding step reduces to a matrix multiplication of $k \times k$ matrices. Applying $\mathbf{H}_{\mathcal{I}}^{-1}$ to a square matrix can be done in $\mathcal{O}(k^2 \log k)$, through the IFFT algorithm. The prevailing computation in our decoding, is applying $\mathbf{P}^{-1}$. The

decoding step is

$$\begin{aligned}\tilde{\mathbf{D}}_{\mathcal{I}} \cdot \left(\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^{\top}\right) &= \left(\mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}\right) \cdot \left(\mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}\right) \cdot (\widehat{\mathbf{A}^{-1}})^{\top} \\ &= (\mathbf{I}_T \cdot \mathbf{I}_T) \otimes \left(\mathbf{G}_{\mathcal{I}}^{-1} \cdot \mathbf{G}_{\mathcal{I}}\right) \cdot (\widehat{\mathbf{A}^{-1}})^{\top} \\ &= \mathbf{I}_T \otimes \mathbf{I}_k \cdot (\widehat{\mathbf{A}^{-1}})^{\top} \\ &= (\widehat{\mathbf{A}^{-1}})^{\top}\end{aligned}$$

and our scheme is valid.

The above CCM therefore has a linear encoding done locally by the workers (5.14), is MDS since $s = d - 1$, and its decoding step reduces to computing and applying $\mathbf{G}_{\mathcal{I}}^{-1}$ (Lemma 5.3.1). The security of the encodings rely on the secrecy of $\mathcal{B}$, which were sent from the coordinator to the workers. For an additional security layer, the interpolation points of $\mathcal{B}$ could instead be defined as $\beta_j = \beta^{\pi(j)}$, for $\pi \in S_n$ a PRP. In this case, π^{-1} would also need to be securely broadcasted.

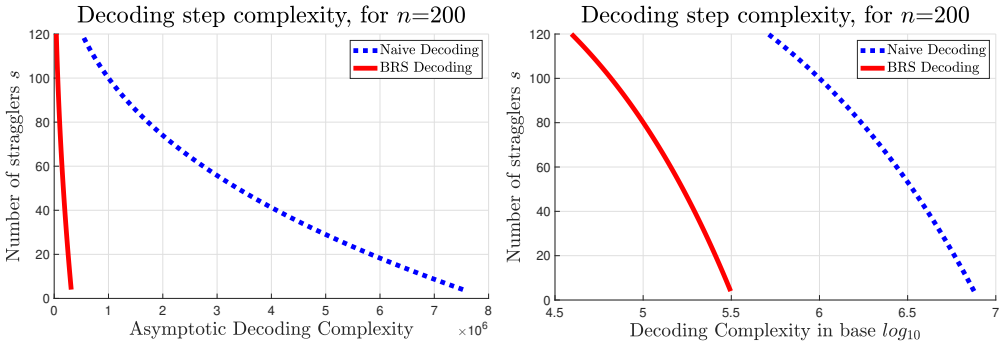


Figure V.4: Comparison of decoding complexity, when naive matrix inversion is used (so $\mathcal{O}(k^3)$) compared to the decoding step implied by Lemma 5.3.1, for $n = 200$ and varying s. We also provide a logarithmic scale comparison.

Remark 5.5.1. *With the above framework, any sparsest-balanced generator MDS matrix [77] would suffice, as long as it satisfies the MDS theorem [191]. By Lemma 5.3.1, if we set $k = \Omega(\sqrt{N})$ (as in [303]), the decoding step could then be done in $\mathcal{O}\left(N^{\omega/2}\right) = o\left(N^{1.187}\right)$ time, which is close to linear in terms of N.*

Theorem 5.5.1. *Let $\mathbf{G} \in \mathbb{F}^{n \times k}$ be a generator matrix of any MDS code over $\mathbb{F}$, for which $\|\mathbf{G}^{(j)}\|_0 = n - k + 1$ and $\|\mathbf{G}_{(i)}\|_0 = w$ for all $(i, j) \in \mathbb{N}_n \times \mathbb{N}_k$. By utilizing Algorithm 8, we can devise a linear MDS coded matrix inversion scheme; through the encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.*

Proof. The encoding coefficients applied locally by each of the n workers correspond to a row of $\mathbf{G}$. The encodings of all the workers then correspond to $\tilde{\mathbf{G}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$, for $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$, as in (5.14). Consider any set of responsive workers $\mathcal{I}$ of size k, whose encodings constitute $\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$. By the MDS theorem, $\mathbf{G}_{\mathcal{I}}$ is invertible. Hence, the decoding step reduces to inverting $\mathbf{G}_{\mathcal{I}}$; *i.e.* $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$, and is performed online. □

Constructions based on cyclic MDS codes, which have been used to devise GC schemes [239], can also be considered. These encoding matrices are not sparsest-balanced, which makes them suitable when considering *heterogeneous* workers.

Proposition 5.5.2. *Any cyclic $[n, k]$ MDS code $\mathcal{C}$ over $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ can be used to devise a coded matrix inversion encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.*

Proof. Consider a cyclic $[n, n-s]$ MDS code $\mathcal{C}$ over $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$. Recall that from our assumptions, we have $s = n - k$. By [239, Lemma 8], there exists a codeword $\mathbf{g}_1 \in \mathcal{C}$ of support $d = s + 1$, *i.e.* $\|\mathbf{g}_1\|_0 = d$. Since $\mathcal{C}$ is cyclic, it follows that the cyclic shifts of $\mathbf{g}_1$ also lie in $\mathcal{C}$. Denote the $n-1$ consecutive cyclic shifts of $\mathbf{g}_1$ by $\{\mathbf{g}_i\}_{i=2}^n \subsetneq \mathcal{C} \subsetneq \mathbb{F}^{1\times n}$, which are all distinct. Define the cyclic matrix

$$\bar{\mathbf{G}} \coloneqq \begin{pmatrix} | & | & & | \\ \mathbf{g}_1^\top & \mathbf{g}_2^\top & \dots & \mathbf{g}_n^\top \\ | & | & & | \end{pmatrix} \in \mathbb{F}^{n\times n}.$$

Since $\|\mathbf{g}_i\|_0 = d$ and $\mathbf{g}_i$ is a cyclic shift of $\mathbf{g}_{i-1}$ for all $i > 1$, it follows that $\|\bar{\mathbf{G}}_{(i)}\|_0 = \|\bar{\mathbf{G}}_{(j)}\|_0 = d$ for all $i, j \in \mathbb{N}_n$, *i.e.* $\bar{\mathbf{G}}$ is sparsest and balanced. If we erase *any* $s = n - k$ columns of $\bar{\mathbf{G}}$, we get $\mathbf{G} \in \mathbb{F}^{n\times k}$. By erasing arbitrary columns of $\bar{\mathbf{G}}$, the resulting $\mathbf{G}$ is *not* balanced, *i.e.* we have $\|\mathbf{G}_{(i)}\|_0 \neq \|\mathbf{G}_{(j)}\|_0$ for some pairs $i, j \in \mathbb{N}_n$. Similar to our construction based on BRS generator matrices, we define the encoding matrix to be $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$. The local encodings are then analogous to (5.14).

Consider an arbitrary set of k non-straggling workers $\mathcal{I} \subsetneq \mathbb{N}_n$, and the corresponding matrix $\mathbf{G}_{\mathcal{I}} \in \mathbb{F}^{k\times k}$. By [239, Lemma 12, B4.], $\mathbf{G}_{\mathcal{I}}$ is invertible. The decoding matrix is then $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$, and the condition $\tilde{\mathbf{D}}_{\mathcal{I}}\tilde{\mathbf{G}} = \mathbf{I}_N$ is met. □

5.5.4 Optimality of MDS BRS Codes

Under the assumption that $k = n - s$, by utilizing the $\mathsf{BRS}_q[n, k]$ generator matrices, we achieved the minimum possible communication load from the workers to

the coordinator. From our discussion in 5.5.1, we cannot hope to receive an encoding of less than N^2/k symbols; when we require that k workers respond with the same amount of information symbols in order to recover $\widehat{\mathbf{A}^{-1}} \in \mathbb{R}^{N\times N}$, unless we make further assumptions on the structure of $\mathbf{A}$ and $\mathbf{A}^{-1}$. Each encoding $\mathbf{W}_\iota$ consists of $NT = N^2/k$ symbols, so we have achieved the lower bound on the minimum amount of information needed to be sent to the coordinator. Moreover, $\mathbf{W}_\iota \in \mathbb{C}^{T\times N}$ for any sparsest-balance generator MDS matrix. This also holds true for other generator matrices which can be used in Theorem 5.5.1, as the encodings are linear (*e.g.* Proposition 5.5.2).

We also require the workers to estimate the least possible number of columns for the given recovery threshold k. For our choice of parameters, the bound of [279, Theorem 1] is met with equality. That is, for all $i \in \mathbb{N}_n$:

$$\|\mathbf{G}_{(i)}\|_0 = w = \frac{k}{n} \cdot d = \frac{k}{n} \cdot (n - k + 1) \ ,$$

which means that for homogeneous workers, we cannot get a sparser generator matrix. This, along with the requirement that $\mathbf{G}_{\mathcal{I}}$ should be invertible for all possible $\mathcal{I}$, are what we considered in (5.2).

5.5.5 Time and Space Complexity

Next, we discuss the complexity of our method. Communication loads and storage are measured in symbols over $\mathbb{R}$. For simplicity, we assume that the n workers are homogeneous and the local data blocks $\{\mathbf{A}_\iota\}_{\iota=1}^{k}$ are of size $N \times T$, for $T = N/k = \Gamma\gamma$. To further simplify our expressions and for fair comparisons to other polynomial codes, we set $k = n = \Omega(\sqrt{N})$ (as in [303]).

Let $\nu = |\mathsf{Enc}(\beta)|$ denote the number of symbols required for the encoding of β through a public-key cryptosystem used to securely broadcast a symbol. Further note that $|\mathsf{Enc}(\mathcal{H})| = \gamma\nu$ and $|\mathsf{Enc}(\sigma_\iota)| \leqslant \gamma\nu/2$, when the same cryptosystem is used to broadcast $\mathcal{H}$ and σ_ι respectively. Hence, phase (a) requires a communication load of $\mathcal{O}(\nu\gamma)$ symbols per client.

There are γ modified Lagrange polynomials in (5.12); each of which require $\mathcal{O}(n-1)$ operations to compute. Multiplying each polynomial with one of the γ sub-blocks $\{\mathbf{A}_\iota^j\}_{j=1}^{\gamma}$, requires $\gamma \cdot \mathcal{O}(N\Gamma) = \mathcal{O}(NT) = \mathcal{O}(N^{3/2})$ additional operations in total. Finally, summing over the encoded blocks requires $\mathcal{O}(N\Gamma(\gamma-1)) = \mathcal{O}(N^{3/2})$ operations. Hence, the encoding through the polynomials $f_\iota(\mathsf{x})$ has complexity $\mathcal{O}\big(2N^{3/2} + \gamma(n-1)\big) = \mathcal{O}\big(\sqrt{N}(N+\gamma)\big)$ for each data block, which is done locally

by the clients. The PRP used could be any block cipher, *e.g.* the Feistel cipher; which has time complexity $\mathcal{O}(n)$. Therefore, phase (b) has complexity $\mathcal{O}\big(\sqrt{N}(N+\gamma+1)\big)$ and the clients communicate $2NT+n=\Omega\big(\sqrt{N}(N+1)\big)$ symbols to each other, which accounts for a total communication load of $(k-1)\cdot\Omega\big(\sqrt{N}(N+1)\big)=\Omega(N^2)$ symbols per client.

At phase (c), the ι^{th} client first recovers $\mathbf{A}$ by evaluating $\eta^{-1}_{\sigma_\iota(j)}f_\iota(\beta_j)$; for each $j\in\mathbb{N}_\gamma$, which requires a total of $\mathcal{O}(\gamma\cdot N\Gamma)=\mathcal{O}\big(N^{3/2}\big)$ operations. The complexity of the computation tasks depends on the underlying optimization algorithm used by the workers; and the desired level of accuracy. By Proposition 5.4.2, under the given assumptions, when using CG we converge after $\tilde{N}$ iterations per column estimate, and each iteration has complexity $\mathcal{O}(N)$. Therefore, the complexity of the workers' tasks are $\mathcal{O}(\tilde{N}NT)=\mathcal{O}\big(\tilde{N}N^{3/2}\big)$. All in all, the computation tasks at phase (c) have total complexity $\mathcal{O}\big((\tilde{N}+1)N^{3/2}\big)=\mathcal{O}\big(\tilde{N}N^{3/2}\big)$ per worker. Since $\mathbf{W}_\iota\in\mathbb{C}^{T\times N}$ for each ι, the communication load is $2NT=2N^2/k=\Omega\big(N^{3/2}\big)$ symbols. Furthermore, the baseline to computing the $\mathbf{A}^{-1}$ is Gaussian elimination; which has complexity $\mathcal{O}(N^3)$ when carried out on one server, while our approach through CG has complexity $k\cdot\mathcal{O}\big(\tilde{N}N^{3/2}\big)=\mathcal{O}\big(\tilde{N}N^2\big)$.

By Lemma 5.3.1 and Remark 5.5.1; the decoding takes $\mathcal{O}(N^{\omega/2})=o(N^{1.187})$ time, which amounts to the complexity of phase (d). Since our recovery threshold is k, phase (d) no more than $k\cdot\Omega(NT)=\Omega(N^2)$ symbols need to be received and stored by the coordinator, who finally recovers a matrix of size $N\times N$.

We summarize the communication loads (C.L.) and time complexity (T.C.) of each of the four phases in Table V.3. The time complexity for phase (a) depends on the encryption method that is used to securely communicate $\beta,\mathcal{H}$; which we do not study, and there is no communication taking place in phase (d). Phases (a) and (d) correspond to the coordinator, (b) to each client, and (c) to each worker.

Communication Loads & Time Complexities				
Phase	(a) *share* $\beta,\mathcal{H}$	(b) *encrypt* $\mathbf{A}_\iota$	(c) *CC job*	(d) *decode*
C.L.	$\mathcal{O}(\nu\gamma)$	$\Omega(N^2)$	$\Omega\big(N^{3/2}\big)$	✗
T.C.	✗	$\mathcal{O}\big(\sqrt{N}(N+\gamma)\big)$	$\mathcal{O}\big(\tilde{N}N^{3/2}\big)$	$o\big(N^{1.187}\big)$

Table V.3: Communication loads and time complexities of our proposed matrix inversion scheme.

A bottleneck of our approach is the workers' storage requirement. As was discussed in 5.5.1, the workers need to recover $\mathbf{A}$, so they need to store a total of N^2 symbols.

The central server receives a total of k completed tasks $\{\mathbf{W}_i\}_{i\in\mathcal{I}}$, which constitute to a total of $2N^2$ symbols. Further examining this drawback would be worthwhile future work.

5.5.6 Comparison to Exact Matrix Inversion

We conclude this section with a discussion on the conditions under which our CMIM will have advantages over standard matrix inversion approaches. First of all, the main bottleneck of our approach is the fact that each worker has a storage requirement of N^2 symbols. When N is relatively small, matrix inversion can be performed by a single server; though the time complexity is still high, in which case our distributed approach is beneficial. In this scenario, the storage constraint is not an issue. For N very large, our approach is still advantageous in terms of time complexity, as a single server would need to perform the entire computation on its own; while also storing the entire matrix. In this case, the storage requirement of our approach is disadvantageous, since we require total storage of kN^2 symbols across the network, while matrix inversion only requires N^2. This is the cost we pay for performing our method distributively.

The second point of comparison is approximation accuracy. The accuracy of standard finite precision matrix inversion is controlled by the number of bits of precision used by the multiplier. On the other hand, by design, our proposed algorithm introduces an additional approximation error due to its reliance on successive approximation iterations. As we showed numerically though in Figure V.1, after a few iterations of Algorithm 8 with CG; we can achieve the same error rate as when exact matrix inversion is used, with a lower complexity. Furthermore, in building risk minimizing ML models, approximate solutions using iterative approximations are often faster and sufficient for achieving desired performance benchmarks. Additionally, the approximation accuracy of our proposed matrix inversion method is controllable by adjusting the number of iterations carried out locally by the workers.

Lastly, we discuss when Algorithm 8 might have advantages over exact matrix inversion in terms of computational complexity and waiting time. For simplicity, we assume that exact computation of $\mathbf{A}^{-1}$ requires $\mathcal{O}(N^{2.373})$ operations. When utilizing our algorithm with CG on a single server, we require $\mathcal{O}(\tilde{N}N^2)$ operations to guarantee convergence. Thus, in this case; Algorithm 8 is beneficial when $\tilde{N} < N^{0.373}$, where $\tilde{N}$ is the number of distinct eigenvalues of $\mathbf{A}^\top\mathbf{A}$. When employing a distributed implementation, in terms of the waiting time through phase (c); our approach is beneficial when $\tilde{N} < N^{0.873}$.

5.6 Conclusion and Future Work

In this chapter, we addressed the problem of approximate computation of the inverse of a matrix distributively in a relaxed FL setting, under the possible presence of straggling workers. We provided approximation error bounds for our approach, as well as security and recovery guarantees. We also provided numerical experiments that validated our proposed approach.

There are several interesting future directions. One avenue to consider is incorporating fully homomorphic encryption in our phases (b),(c),(d), to obtain a FL scheme; and prevent the requirement of clients need to recover each others' information. An important issue is the numerical stability of the BRS approach, so exploring other suitable generator matrices could be beneficial; *e.g.* circulant permutation and rotation matrices [236]. It is also worth investigating if we can reduce the communication rounds when computing the pseudoinverse through our approach. This depends on the CMM which is being utilized, though using different ones for each of the two multiplications may also be beneficial. An interesting approach to also consider, is if divide and conquer algorithms could be leveraged in CC to recover exact or approximate solutions to $\mathbf{A}^{-1}$.

In terms of coding-theory, it would be interesting to see if it is possible to reduce the complexity of our decoding step. Specifically, could well-known RS decoding algorithms such as the Berlekamp-Welch algorithm be exploited? Another direction, is leveraging approximate CCMs. The work of [148] considers the GC problem for *approximate* and *exact* recovery through Lagrange interpolation, for heterogeneous workers in the presence of stragglers and adversaries. A potential scheme for matrix inversion could also be developed through the methods of [148]. In terms of our approximation algorithms, an avenue worth exploring is that of incorporating approximate and/or sparse Gaussian elimination [171, 172] into our distributed CCM.

Tribute to Alex Vardy: The chapter was submitted for publication to a 2023 special issue of the IEEE Journal on Selected Areas in Information Theory, which was dedicated to the memory of Alexander Vardy. As such, we mention how this work relates to some of Alex Vardy's work. Even though Alex had not worked on CC, his contributions to RS codes are immense. A focus of this chapter was to reduce the decoding complexity of the proposed BRS-based CCM, while in [130] it was shown by Guruswami and Vardy that maximum-likelihood decoding of RS codes is NP-hard. Another highly innovative work of Alex's is [228], in which the 'Parvaresh-Vardy

codes' were introduced; and the associated list-decoding algorithm was shown to yield an improvement over the Guruswami–Sudan algorithm. This was subsequently improved by Guruswami and Rudra [129], whose techniques were exploited in [266] to introduce list-decoding in CC. An expository monograph on the developments of list-decoding can be found in [44], which I wrote as part of Mahdi's wonderful course on 'Coding Theory for Theoretical Computer Science'.

CHAPTER VI

Approximate Matrix Multiplication by Joint Leverage Score Sampling

6.1 Introduction

Matrix multiplication is one of the key underlying operations used in applications and algorithms in domains such as computer science, data science, numerical analysis, machine learning, network analysis and scientific computing. Frequently, this operation occurs thousands of times, making it a bottleneck and an impediment in large scale computations.

Considering square matrices of order N, naive matrix multiplication requires $O(N^3)$ operations. More elegant algorithms exist, which require $O(N^\omega)$ operations, for $\omega < 2.81$ [70, 176, 276]. The smallest known matrix multiplication exponent $\omega < 2.373$, was recently achieved in [11]. A drawback of these faster algorithms though, is that the hidden constants in their asymptotic analysis are significantly large.

In this chapter, we propose a randomized algorithm which judiciously subsamples pairs of rows from $\mathbf{A}$ and $\mathbf{B}$, to approximate

$$\overbrace{\begin{pmatrix} \\ \mathbf{A}^\top \\ \\ \end{pmatrix}}^{L\times N} \cdot \overbrace{\begin{pmatrix} \\ \mathbf{B} \\ \\ \\ \end{pmatrix}}^{N\times M} \approx \overbrace{\begin{pmatrix} \\ \hat{\mathbf{A}}^\top \\ \\ \end{pmatrix}}^{L\times r} \cdot \overbrace{\begin{pmatrix} \\ \hat{\mathbf{B}} \\ \\ \end{pmatrix}}^{r\times M}$$

by applying a carefully chosen *sketching matrix* $\mathbf{S} \in \mathbb{R}^{r \times N}$ with $r \ll N$, on both matrices. For our proposed $\mathbf{S}$, we define a sampling distribution based on the left leverage scores of both $\mathbf{A}$ and $\mathbf{B}$, which we call '*joint leverage score distribution*'. These scores reveal influential pairs of rows across the two matrices. The proposed algorithm is similar to the $CR-$MM approximate matrix multiplication (AMM) algorithm [46, 53, 88, 89, 90, 201, 217, 294], and is motivated by the leverage score sampling algorithm for ℓ_2-subspace embedding [52, 96, 195]. Both there algorithms have been extensively studied in the literature of randomized numerical linear algebra (RandNLA) [201, 294].

The benefit of sampling according to leverage scores, is that we can derive a spectral bound on a characterization of AMM, analogous to that of a ℓ_2-subspace embedding. Such spectral guarantees have not been derived for the general case of $CR-$MM, whose error is quantified in terms of the Frobenius norm. A spectral bound for the $CR-$MM is only known for the case where $\mathbf{A} = \mathbf{B}$ [96, 201]. It is worth noting that leverage scores have also been used for graph sparsification algorithms [92, 273].

A seemingly unrelated object, which has many applications in data science, is the of a graph Laplacian. Graphs are often used to model phenomena in which there is a flow around a network with high and low pressure regions; *e.g.* electrical circuits, and are prevalent in many domains of modern computer science and engineering. Due to the large size of the networks considered, a sparser version is preferred to store and process. This gives rise to the notion of a *graph spectral sparsifier* [46, 273, 275], which requires that the Laplacian quadratic form of the sparsifier approximate that of the original. In [46], it was shown how $CR-$MM can be utilized to produce minimum variance unbiased *additive* and *multiplicative* spectral sparsifiers. Furthermore, our AMM algorithm can be viewed as a generalization of sampling according to effective resistances, the state-of-the approach to spectral sparsification through edge sampling [273].

Our work on AMM is closely related to that of [69]. Specifically, they give a similar spectral AMM characterization to ours, both of which imply the same spectral error bound (Lemma 6.2.1). The work of Cohen et al. focuses on oblivious subspace embeddings, and what can be implied by such embeddings if certain moment bounds and other conditions are met. They do not give an explicit AMM algorithm, and have a black box reliance on the subspace embedding primitive in their proofs. Similar results and deductions have also been derived in [255]. As in [69], any AMM sketching matrix which satisfies our condition, is a ℓ_2-subspace embedding sketching matrix. This is achieved by simply applying it to the case where $\mathbf{B} = \mathbf{A}$, in which case we

move from $\mathbf{A}^\top\mathbf{A}$ to the least squares objective function $\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$. For the algorithm we propose; which is not data-oblivious, the resulting embedding is the leverage score ℓ_2-subspace embedding sketch [96]. It is also worth noting that tangentially related work is that of [138], which considers spectral guarantees of the CUR decomposition.

The chapter is organized as follows. In Section 6.2 we first recall the notion of leverage scores, and the $CR-$MM algorithm. Next, we describe our sampling approach to AMM; which we call *joint leverage score sampling*. Then, we present our AMM algorithm. Our spectral characterization of AMM is derived in Subsection 6.2.3. In Subsection 6.2.4 we present the theory behind our approach, and our spectral guarantees. In Section 6.3 we discuss other extensions of our approach, which suggest worthwhile future work. We conclude this chapter with Section 6.4.

6.2 Joint Leverage Score Multiplication

6.2.1 Preliminaries

We first recall the $CR-$MM approximation algorithm. Consider the two matrices $\mathbf{A} \in \mathbb{R}^{N\times L}$ and $\mathbf{B} \in \mathbb{R}^{N\times M}$, for which we want to approximate the product $\mathbf{A}^\top\mathbf{B}$. It is known that the product may be approximated by subsampling with replacement (w.r.) row pairs from $\mathbf{A}$ and $\mathbf{B}$, where the sampling probabilities are proportional to their Euclidean norms. That is, we sample with replacement r pairs $(\mathbf{A}_{(i)}, \mathbf{B}_{(i)})$[1] for $i \in \mathbb{N}_N \coloneqq \{1, \cdots, N\}$ and $r \ll N$, with probability

$$p_i = \|\mathbf{A}_{(i)}\|_2 \cdot \|\mathbf{B}_{(i)}\|_2 / \phi$$

for $\phi \coloneqq \sum_{j=1}^{N} \|\mathbf{A}_{(j)}\|_2 \cdot \|\mathbf{B}_{(j)}\|$, and sum a rescaling of the samples' outer-products:

$$\mathbf{A}^\top\mathbf{B} \approx \frac{1}{r} \cdot \left(\sum_{j\in\mathcal{I}} \frac{1}{p_j} \mathbf{A}_{(j)}^\top \mathbf{B}_{(j)} \right) = \sum_{j\in\mathcal{I}} \frac{\mathbf{A}_{(j)}^\top}{\sqrt{rp_j}} \cdot \frac{\mathbf{B}_{(j)}}{\sqrt{rp_j}} =: Y \tag{6.1}$$

where $\mathcal{I}$ is the multiset consisting of the indices (possibly repeated) of the sampled pairs, *i.e.* $|\mathcal{I}| = r$. Denote the corresponding sketches of $\mathbf{A}^\top$ and $\mathbf{B}$ in $CR-$MM respectively by $C \in \mathbb{R}^{L\times r}$ and $R \in \mathbb{R}^{r\times M}$ respectively. The approximation (6.1) satisfies $\|\mathbf{A}^\top\mathbf{B} - CR\|_F = O(\|\mathbf{A}\|_F\|\mathbf{B}\|_F/\sqrt{r})$. Further details on this algorithm can be found in [53, 88, 89, 90, 201, 217, 294]. We summarize what is known about the $CR-$MM algorithm in Theorem 6.2.1.

[1] We denote the i^{th} row of matrix $\mathbf{M}$ by $\mathbf{M}_{(i)}$.

Theorem 6.2.1 ([201] Section 3.2)**.** *The estimator $Y = CR$ from (6.1) is unbiased, while the sampling probabilities $\{p_i\}_{i=1}^N$ minimize the variance, i.e.*

$$\{p_i\}_{i=1}^N = \underset{\sum_{i=1}^N p_i = 1}{\arg\min} \left\{ \mathrm{Var}(Y) = \mathbb{E}\left[\|\mathbf{A}^\top \mathbf{B} - CR\|_F^2 \right] \right\} \tag{6.2}$$

and it is an ϵ-multiplicative error approximation of the matrix product, with high probability. Specifically, for $\delta \geqslant 0$ and $r \geqslant \frac{1}{\delta^2 \epsilon^2}$ the number of sampling trials which take place:

$$\Pr\left[\|\mathbf{A}^\top \mathbf{B} - CR\|_F \leqslant \epsilon \cdot \|\mathbf{A}\|_F \|\mathbf{B}\|_F \right] \geqslant 1 - \delta \tag{6.3}$$

for any $\epsilon > 0$.

The special case where $\mathbf{A} = \mathbf{B}$ in the $CR-$MM algorithm has also been studied, as it appears in numerous applications. This restriction allows us to use statements from random matrix theory [219], to get stronger spectral norm bounds. One such bound is Theorem 6.2.2 [96, Theorem 4], [201, Theorem 8].

Theorem 6.2.2 ([201] Theorem 8)**.** *Let $\mathbf{A} \in \mathbb{R}^{N \times L}$ with $\sigma_{\max}(\mathbf{A}) = \|\mathbf{A}\|_2 \leqslant 1$, and approximate the product $Y \approx \mathbf{A}^\top \mathbf{A}$ using $CR-$MM. Let $\epsilon \in (0, 1)$ be an accuracy parameter, and assume that $\|\mathbf{A}\|_F^2 \geqslant 1/24$. If*

$$r \geqslant \frac{96 \|\mathbf{A}\|_F^2}{\epsilon^2} \ln\left(\frac{96 \|\mathbf{A}\|_F^2}{\epsilon^2 \sqrt{\delta}} \right) \geqslant \frac{4}{\epsilon^2} \ln\left(\frac{4}{\epsilon^2 \sqrt{\delta}} \right) \tag{6.4}$$

for $r \leqslant N$, then

$$\Pr\left[\|\mathbf{A}^\top \mathbf{A} - Y\|_2 \leqslant \epsilon \right] \geqslant 1 - \delta \ . \tag{6.5}$$

The left *leverage scores* $\{\ell_i\}_{i=1}^N$ of $\mathbf{M} \in \mathbb{R}^{N \times d}$ are defined as the diagonal entries of the projection $P_{\mathbf{M}} = \mathbf{M}\mathbf{M}^\dagger$, *i.e.* $\ell_i = (P_{\mathbf{M}})_{ii} = \|\mathbf{M}_{(i)}\|_{\mathbf{M}^\dagger}^2$. Equivalently, considering the reduced left singular vectors matrix $U \in \mathbb{R}^{N \times d}$ of $\mathbf{M}$, we have $\ell_i = \|U_{(i)}\|_2^2$. A sampling distribution is defined over the rows of $\mathbf{M}$ by normalizing the scores $\{\ell_i\}_{i=1}^N$, where each row of $\mathbf{M}$ has respective probability of being sampled: $p_i = \ell_i / \left(\sum_{j=1}^N \ell_j \right) = \ell_i / \|U\|_F^2 = \ell_i / d$. It is a known fact that

$$\sum_{i=1}^N \ell_i = \mathrm{tr}(P_{\mathbf{M}}) = \mathrm{tr}(UU^\top) = \mathrm{rank}(\mathbf{M}) \leqslant d \ . \tag{6.6}$$

We note that methods for approximating the leverage scores are available [91, 142, 201, 249, 273], which do not require directly computing the left orthonormal basis of each matrix. Furthermore, any orthonormal basis of the column-space of $\mathbf{M}$ would

suffice to deduce the leverage scores. We will be working with the left singular vectors matrix, to simplify certain expressions.

Lastly, we recall the following noncommutative Bernstein inequality, which we use in order to prove the spectral characterization of our AMM algorithm.

Theorem 6.2.3 ([197, 241]). *Let $\mathbf{X}_1, \cdots, \mathbf{X}_r$ be independent copies of a zero-mean random matrix $\mathbf{X} \in \mathbb{R}^{d_1 \times d_2}$, with $\sigma^2 \geqslant \max\left\{\left\|\mathbb{E}\left[\mathbf{X}^\top \mathbf{X}\right]\right\|_2, \left\|\mathbb{E}\left[\mathbf{X}\mathbf{X}^\top\right]\right\|_2\right\}$ and $\|\mathbf{X}\|_2 \leqslant \gamma$. Then, $\forall \epsilon > 0$:*

$$\Pr\left[\left\|\frac{1}{r}\sum_{l=1}^{r}\mathbf{X}_l\right\|_2 > \epsilon\right] \leqslant (d_1 + d_2) \cdot \exp\left(\frac{-r\epsilon^2/2}{\sigma^2 + \gamma\epsilon/3}\right).$$

6.2.2 Joint Leverage Score Sampling

We define the *joint leverage score* of the pair $(\mathbf{A}_{(i)}, \mathbf{B}_{(j)})$ as the geometric mean of corresponding leverage scores, *i.e.*

$$\mathscr{J}_{ij} = \sqrt{\bar{\ell}_i \cdot \tilde{\ell}_j}$$

where $\bar{\ell}_i$ and $\tilde{\ell}_i$ are respectively the i^{th} and j^{th} leverage score of $\mathbf{A}$ and $\mathbf{B}$. By $U_{\mathbf{A}} \in \mathbb{R}^{N \times L}$ and $U_{\mathbf{B}} \in \mathbb{R}^{N \times M}$ we denote the reduced left singular vectors matrices of $\mathbf{A}$ and $\mathbf{B}$, with respective rows $\{\bar{u}_i\}_{i=1}^N$ and $\{\tilde{u}_j\}_{j=1}^N$. By the definition of leverage scores, it follows that

$$\bar{\ell}_i = \bar{u}_i \bar{u}_i^\top \quad \text{and} \quad \tilde{\ell}_i = \tilde{u}_i \tilde{u}_i^\top \quad \text{for all } i \in \mathbb{N}_N.$$

For $P_{\mathbf{A}} = U_{\mathbf{A}} U_{\mathbf{A}}^\top$ and $P_{\mathbf{B}} = U_{\mathbf{B}} U_{\mathbf{B}}^\top$ the respective projection matrices of $\mathbf{A}$ and $\mathbf{B}$, let $\vec{d}_{\mathbf{A}} = \sqrt{\mathrm{diag}(P_{\mathbf{A}})}$ and $\vec{d}_{\mathbf{B}} = \sqrt{\mathrm{diag}(P_{\mathbf{B}})}$ denote the vectors of the square roots of the leverage scores of $\mathbf{A}$ and $\mathbf{B}$, hence $\|\vec{d}_{\mathbf{A}}\|_2 = \left(\sum_{l=1}^N \bar{\ell}_l\right)^{1/2}$. Let $\mu \coloneqq \sqrt{LM}$. By the Cauchy–Schwarz inequality and (6.6):

$$\tau \coloneqq \sum_{l=1}^{N} \mathscr{J}_{ll} = \langle \vec{d}_{\mathbf{A}}, \vec{d}_{\mathbf{B}} \rangle \leqslant \|\vec{d}_{\mathbf{A}}\|_2 \cdot \|\vec{d}_{\mathbf{B}}\|_2 \leqslant \sqrt{LM} \quad \Longrightarrow \quad \tau \leqslant \mu\,. \tag{6.7}$$

In this work we sample pairs of rows of the same index, and do not consider pairs $(\mathbf{A}_{(i)}, \mathbf{B}_{(j)})$ for $i \neq j$. Hence, we only consider the joint leverage scores $\{\mathscr{J}_{ii}\}_{i=1}^N$, which we normalize to determine the sampling probabilities

$$\pi_i \coloneqq \mathscr{J}_{ii}/\tau\,. \tag{6.8}$$

Following a similar approach to $CR-$MM and the leverage score sampling algorithms, we construct a sketching matrix $\mathbf{\Pi} \in \mathbb{R}^{r \times N}$ to reduce the effective dimension from N to r, and approximate the product:

$$\mathbf{A}^\top \mathbf{B} \approx (\mathbf{\Pi A})^\top (\mathbf{\Pi B}) =: \hat{\mathbf{A}}^\top \hat{\mathbf{B}} \ .$$

Once the sketches $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ are known, computing $\hat{\mathbf{A}}^\top \hat{\mathbf{B}}$ requires $O(LMr)$ operations. The pseudocode of our approach is presented in Algorithm 9.

Note that any approximate leverage score algorithm [91, 142, 201, 249, 273] could also be used to speed up the computation of $\{\pi_i\}_{i=1}^N$, as determining them exactly is a cumbersome task. The guarantees we give will assume that the distribution used in Algorithm 9 is approximate.

Algorithm 9: Joint Leverage Score Sampling AMM

Input: $\mathbf{A} \in \mathbb{R}^{N \times L}$, $\mathbf{B} \in \mathbb{R}^{N \times M}$, $r < N$
Output: Approximate product $\hat{\mathbf{A}}^\top \hat{\mathbf{B}} \approx \mathbf{A}^\top \mathbf{B}$
Initialize: $\hat{\mathbf{\Pi}} = \mathbf{0}_{r \times N}$
Determine: approximate distribution $\{\hat{\pi}_l\}_{l=1}^N$ to (6.8)
for $j = 1$ *to* r **do**
  sample w.r. $i(j)$ from $\mathbb{N}_N$, according to $\{\hat{\pi}_l\}_{l=1}^N$
  $\hat{\mathbf{\Pi}}_{j,i(j)} = 1/\sqrt{r \hat{\pi}_{i(j)}}$
end
return $\hat{\mathbf{A}}^\top \hat{\mathbf{B}} = (\mathbf{\Pi A})^\top (\mathbf{\Pi B})$

Using probabilities exactly equal to $\{\pi_i\}_{i=1}^N$ is an overkill. Similar results hold if we instead use approximate probabilities $\{\hat{\pi}_i\}_{i=1}^N$ that are "close" in the following sense: $\hat{\pi}_i \geqslant \beta \pi_i$ for all i, where $\beta \in (0, 1]$ is a misestimation factor. This factor quantifies how far $\{\hat{\pi}_i\}_{i=1}^N$ is from $\{\pi_i\}_{i=1}^N$. If we sample according to the approximate distribution, the same embedding result will follow, if we modify the sampling complexity by a factor of $1/\beta$, *i.e.* we oversample. The key point here is that it is essential not to underestimate high joint leverage score row pairs too much. It is however acceptable if we overestimate and thus oversample some low score pairs, as long it is not done excessively [201]. Throughout this chapter, we will denote the resulting sketching matrices of Algorithm 9 by $\mathbf{\Pi}$ and $\hat{\mathbf{\Pi}}$, when sampling takes place according to $\{\pi_i\}_{i=1}^N$ and $\{\hat{\pi}_i\}_{i=1}^N$ respectively.

6.2.3 Spectral Characterization for AMM

Recall that a sketching matrix $\mathbf{S} \in \mathbb{R}^{r\times N}$ is a ℓ_2-*subspace embedding* of $\mathbf{M} \in \mathbb{R}^{N\times d}$ for $N \gg d$ and $N > r > d$, if it satisfies

$$\|\mathbf{I}_d - (\mathbf{S}U)^\top(\mathbf{S}U)\|_2 \leqslant \epsilon \tag{6.9}$$

for $\epsilon > 0$, with high probability [294]. Next, we derive an analogous characterization for AMM, in order to quantify the error of our approximation.

Considering the reduced SVDs of $\mathbf{A}$ and $\mathbf{B}$, an AMM sketching matrix $\mathbf{S} \in \mathbb{R}^{r\times N}$ ought to satisfy

$$\mathbf{A}^\top \cdot \mathbf{B} = V_\mathbf{A}\Sigma_\mathbf{A}U_\mathbf{A}^\top \cdot U_\mathbf{B}\Sigma_\mathbf{B}V_\mathbf{B}^\top \approx V_\mathbf{A}\Sigma_\mathbf{A}U_\mathbf{A}^\top\mathbf{S}^\top \cdot \mathbf{S}U_\mathbf{B}\Sigma_\mathbf{B}V_\mathbf{B}^\top = (\mathbf{SA})^\top \cdot (\mathbf{SB}) \ .$$

Hence, a sufficient condition which implies $U_\mathbf{A}^\top U_\mathbf{B} \approx (\mathbf{S}U_\mathbf{A})^\top(\mathbf{S}U_\mathbf{B})$, is

$$\|U_\mathbf{A}^\top U_\mathbf{B} - (\mathbf{S}U_\mathbf{A})^\top(\mathbf{S}U_\mathbf{B})\|_2 \leqslant \epsilon \tag{6.10}$$

for $\epsilon > 0$ a small constant. We denote $D_\mathbf{S} = \big(U_\mathbf{A}^\top U_\mathbf{B} - (\mathbf{S}U_\mathbf{A})^\top(\mathbf{S}U_\mathbf{B})\big)$, and

$$\Delta_{\mathbf{A},\mathbf{B}} = \mathbf{A}^\top\mathbf{B} - \hat{\mathbf{A}}^\top\hat{\mathbf{B}} = V_\mathbf{B}\Sigma_\mathbf{A} \cdot D_\mathbf{S} \cdot \Sigma_\mathbf{B}V_\mathbf{B} \ . \tag{6.11}$$

Lemma 6.2.1. *If* $\mathbf{S}$ *satisfies* (6.10)*, then* $\mathrm{err}_{\mathbf{A},\mathbf{B}}^{\ell_2,\mathbf{S}} \coloneqq \|\Delta_{\mathbf{A},\mathbf{B}}\|_2 \leqslant \epsilon \cdot \sigma_1(\mathbf{A})\sigma_1(\mathbf{B})$.

Proof. This is a direct application of the Cauchy–Schwarz inequality to (6.11). □

Remark 6.2.1. *For* $\mathbf{A} = \mathbf{B}$*, the sampling probabilities* $\{\pi_i\}_{i=1}^N$ *are equal to the normalized leverage scores of* $\mathbf{A}$*, and condition* (6.10) *results in the* ℓ_2*-subspace embedding property* (6.9)*. Furthermore,* $\mathbf{\Pi}$ *would be the same sketching matrix as that of the leverage score* ℓ_2*-subspace embedding sampling algorithm. Additionally, for* $\mathbf{A}$*,* $\mathbf{B}$ *both orthonormal matrices, the output of Algorithm 9 is the same as that of the* $CR-$MM *algorithm.*

Proposition 6.2.1. *Any algorithm that constructs a sketching matrix* $\mathbf{S}$ *which satisfies* (6.10)*, can be used to construct a* ℓ_2*-subspace embedding sketch of* $\mathbf{A}$*.*

Proof. Assume we have an algorithm that when given $\mathbf{A}$ and $\mathbf{B}$, produces $\mathbf{S}$ which satisfies (6.10). If it is only given $\mathbf{A}$ as an input, *i.e.* let $\mathbf{B} = \mathbf{A}$, it then constructs $\mathbf{S}$ for the case where $U_\mathbf{B} = U_\mathbf{A}$. Thus $U_\mathbf{A}^\top U_\mathbf{B} = \mathbf{I}_d$, and the resulting $\mathbf{S}$ satisfies (6.9). □

6.2.4 Approximation Guarantee

In this subsection, we present our theoretical result of Algorithm 9. Unlike the guarantees of the $CR-$MM algorithm for $\mathbf{A} \neq \mathbf{B}$, whose error is quantified in terms of the Frobenius norm; the guarantee we give is in terms of the spectral norm. Specifically, we show that (6.10) is achieved with a high probability.

To prove our main theorem, we first need to determine the expectation; and bound the spectral norm and variance of appropriately defined matrix random variables. Once these quantities are determined, we can directly apply Theorem 6.2.3.

For $i(j) \in \mathbb{N}_N$, which denotes the index of the sampled pair $\left(\mathbf{A}_{(i(j))}, \mathbf{B}_{(i(j))}\right)$ at the j^{th} sampling trial of Algorithm 9, we define

$$\mathbf{X}_{i(j)} \coloneqq \left(U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - \frac{\bar{u}_{i(j)}^{\top} \tilde{u}_{i(j)}}{\hat{\pi}_{i(j)}} \right) \in \mathbb{R}^{L \times M} \ . \tag{6.12}$$

The matrices $\left\{\mathbf{X}_{i(j)}\right\}_{j=1}^{r}$ are independent copies of the matrix random variable $\mathbf{X}$ we are considering, since we are sampling from with replacement. Note that the realizations $\mathbf{X}_{\iota}$ of $\mathbf{X}$ correspond to $D_{\hat{\mathbf{\Pi}}}$ of (6.10); *i.e.* $D_{\mathbf{S}}$ for $\mathbf{S} \leftarrow \hat{\mathbf{\Pi}}$, with $r = 1$.

Lemma 6.2.2. *For $\mathbf{X}_{i(j)}$ as defined in (6.12), we have $\|\mathbf{X}_{i(j)}\|_2 \leqslant 1 + \sqrt{L \cdot M}$ and $\mathbb{E}\left[\mathbf{X}_{i(j)}\right] = \mathbf{0}_{M \times L}$.*

Proof. A straightforward computation gives us

$$\mathbb{E}\left[\mathbf{X}_{i(j)}\right] = U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - \sum_{l=1}^{n} \hat{\pi}_l \cdot \left(\frac{\bar{u}_l^{\top} \tilde{u}_l}{\hat{\pi}_l} \right) = U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - \sum_{l=1}^{n} \bar{u}_l^{\top} \tilde{u}_l = U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - U_{\mathbf{A}}^{\top} U_{\mathbf{B}}$$

thus $\mathbb{E}\left[\mathbf{X}_{i(j)}\right] = \mathbf{0}_{L \times M}$. For the second part of the lemma, we have

$$\|\mathbf{X}_{i(j)}\|_2 \leqslant \|U_{\mathbf{A}}^{\top} U_{\mathbf{B}}\|_2 + \frac{1}{\hat{\pi}_{i(j)}} \cdot \|\bar{u}_{i(j)}^{\top} \tilde{u}_{i(j)}\|_2$$

where we observe that

$$\begin{aligned}
\|\bar{u}_{i(j)}^{\top} \tilde{u}_{i(j)}\|_2 &\leqslant \|\bar{u}_{i(j)}^{\top} \tilde{u}_{i(j)}\|_F \\
&= \left(\operatorname{tr}\left(\tilde{u}_{i(j)}^{\top} \bar{u}_{i(j)} \bar{u}_{i(j)}^{\top} \tilde{u}_{i(j)}\right)\right)^{1/2} \\
&= \left(\operatorname{tr}\left(\bar{u}_{i(j)} \bar{u}_{i(j)}^{\top} \cdot \tilde{u}_{i(j)} \tilde{u}_{i(j)}^{\top}\right)\right)^{1/2} \\
&= \sqrt{\bar{\ell}_{i(j)} \cdot \tilde{\ell}_{i(j)}} \\
&= \mathscr{J}_{i(j)i(j)}
\end{aligned}$$

and

$$\|U_{\mathbf{A}}^{\top} U_{\mathbf{B}}\|_2 \leqslant \|U_{\mathbf{A}}\|_2 \cdot \|U_{\mathbf{B}}\|_2 \leqslant 1 \ .$$

Combining these two observations, we conclude that

$$\|\mathbf{X}_{i(j)}\|_2 \leqslant 1 + \frac{\mathscr{J}_{i(j)i(j)}}{\hat{\pi}_{i(j)}} = 1 + \frac{\tau}{\beta} \leqslant 1 + \frac{\sqrt{L \cdot M}}{\beta}$$

where in the last inequality we invoked (6.7). □

In order to upper bound the variance parameter of the random matrix variable defined in (6.12), we define the matrices $\tilde{\Phi} = \sum_{l=1}^{N} \sqrt{\bar{\ell}_l / \tilde{\ell}_l} \cdot \tilde{u}_l^{\top} \tilde{u}_l$ and $\bar{\Phi} = \sum_{l=1}^{N} \sqrt{\tilde{\ell}_l / \bar{\ell}_l} \cdot \bar{u}_l^{\top} \bar{u}_l$. Furthermore, to simplify our notation, let $\alpha_i = \sqrt{\bar{\ell}_l / \tilde{\ell}_l}$ for all $i \in \mathbb{N}_N$. We note that $\{\tilde{u}_i\}_{i=1}^{N}$ and $\{\bar{u}_i\}_{i=1}^{N}$ are respectively linearly dependent spanning sets of both the column and row spaces of $\tilde{\Phi}$ and $\bar{\Phi}$ respectively, when $N > M, L$. This prohibits us from explicitly computing the two spectral norms, though an upper bound suffices.

Lemma 6.2.3. *For $\mathbf{X}_{i(j)}$ as defined in (6.12), for $\iota \leftarrow i(j)$ we have $\left\|\mathbb{E}\left[\mathbf{X}_{\iota}^{\top} \mathbf{X}_{\iota}\right]\right\|_2 \leqslant \frac{\tau}{\beta} \cdot \lambda_1(\tilde{\Phi}) + 1$ and $\left\|\mathbb{E}\left[\mathbf{X}_{\iota} \mathbf{X}_{\iota}^{\top}\right]\right\|_2 \leqslant \frac{\tau}{\beta} \cdot \lambda_1(\bar{\Phi}) + 1$.*

Proof. By a straightforward computation

$$\begin{aligned}
\mathbf{X}_{\iota}^{\top} \mathbf{X}_{\iota} &= \left(U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - \frac{\bar{u}_{\iota}^{\top} \tilde{u}_{\iota}}{\hat{\pi}_{\iota}} \right)^{\top} \left(U_{\mathbf{A}}^{\top} U_{\mathbf{B}} - \frac{\bar{u}_{\iota}^{\top} \tilde{u}_{\iota}}{\hat{\pi}_{\iota}} \right) \\
&= U_{\mathbf{B}}^{\top} U_{\mathbf{A}} U_{\mathbf{A}}^{\top} U_{\mathbf{B}} + \frac{\tilde{u}_{\iota}^{\top} \bar{u}_{\iota} \bar{u}_{\iota}^{\top} \tilde{u}_{\iota}}{\hat{\pi}_{\iota}^2} - 2 \frac{\tilde{u}_{\iota}^{\top} \bar{u}_{\iota}}{\hat{\pi}_{\iota}} U_{\mathbf{A}}^{\top} U_{\mathbf{B}}
\end{aligned}$$

where in the second equality we used the fact that

$$U_{\mathbf{B}}^{\top} U_{\mathbf{A}} \frac{\bar{u}_{\iota}^{\top} \tilde{u}_{\iota}}{\hat{\pi}_{\iota}} = \frac{\tilde{u}_{\iota}^{\top} \bar{u}_{\iota}}{\hat{\pi}_{\iota}} \cdot U_{\mathbf{A}}^{\top} U_{\mathbf{B}} \ .$$

Let $\Upsilon = U_{\mathbf{B}}^\top U_{\mathbf{A}} U_{\mathbf{A}}^\top U_{\mathbf{B}}$, and $E_\iota = \mathbb{E}\big[\mathbf{X}_\iota^\top \mathbf{X}_\iota\big]$. It follows that

$$\begin{aligned}
E_\iota &= \Upsilon + \mathbb{E}\left[\frac{\tilde{u}_\iota^\top \bar{u}_\iota \bar{u}_\iota^\top \tilde{u}_\iota}{\hat{\pi}_\iota^2}\right] - 2 \cdot \mathbb{E}\left[\frac{\tilde{u}_\iota^\top \bar{u}_\iota}{\hat{\pi}_\iota}\right] \cdot U_{\mathbf{A}}^\top U_{\mathbf{B}} \\
&= \Upsilon + \mathbb{E}\left[\frac{\bar{\ell}_\iota}{\hat{\pi}_\iota^2} \cdot \tilde{u}_\iota^\top \tilde{u}_\iota\right] - 2 \cdot \left(\sum_{l=1}^N \hat{\pi}_l \cdot \frac{\tilde{u}_l^\top \bar{u}_l}{\hat{\pi}_l}\right) \cdot U_{\mathbf{A}}^\top U_{\mathbf{B}} \\
&= \Upsilon + \left(\sum_{l=1}^N \hat{\pi}_l \cdot \frac{\bar{\ell}_l}{\hat{\pi}_l^2} \cdot \tilde{u}_l^\top \tilde{u}_l\right) - 2 \cdot U_{\mathbf{B}}^\top U_{\mathbf{A}} \cdot U_{\mathbf{A}}^\top U_{\mathbf{B}} \\
&= \Upsilon + \left(\sum_{l=1}^N \frac{\bar{\ell}_l}{\hat{\pi}_l} \cdot \tilde{u}_l^\top \tilde{u}_l\right) - 2\Upsilon \\
&\leqslant \left(\sum_{l=1}^N \frac{\tau}{\beta} \cdot \sqrt{\bar{\ell}_l / \tilde{\ell}_l} \cdot \tilde{u}_l^\top \tilde{u}_l\right) - \Upsilon \\
&= \frac{\tau}{\beta} \cdot \tilde{\Phi} - \Upsilon \ .
\end{aligned}$$

By the triangle inequality, and the fact that $\|U_{\mathbf{A}}\|_2 = \|U_{\mathbf{B}}\|_2 = 1$, we conclude that

$$\begin{aligned}
\|E_\iota\|_2 &= \left\| \frac{\tau}{\beta} \cdot \tilde{\Phi} - \Upsilon \right\|_2 \\
&\leqslant \frac{\tau}{\beta} \cdot \|\tilde{\Phi}\|_2 + \|\Upsilon\|_2 \\
&\leqslant \frac{\tau}{\beta} \cdot \|\tilde{\Phi}\|_2 + 1 \\
&= \frac{\tau}{\beta} \cdot \lambda_1(\tilde{\Phi}) + 1 \ .
\end{aligned}$$

An analogous calculation yields that $\big\|\mathbb{E}\big[\mathbf{X}_\iota \mathbf{X}_\iota^\top\big]\big\|_2 \leqslant \frac{\tau}{\beta} \cdot \lambda_1(\bar{\Phi}) + 1$. □

Lemma 6.2.4. *Considering any $\mathbf{A} \in \mathbb{R}^{N \times L}$ and $\mathbf{B} \in \mathbb{R}^{N \times M}$, we have:*

- $\lambda_1(\tilde{\Phi}) \leqslant \max_{l \in \mathbb{N}_N}\{\alpha_l\} \cdot \max_{i \in \mathbb{N}_N}\{\tilde{\ell}_i\}$, *and*
- $\lambda_1(\bar{\Phi}) \leqslant \max_{l \in \mathbb{N}_N}\{1/\alpha_l\} \cdot \max_{i \in \mathbb{N}_N}\{\bar{\ell}_i\}$.

Proof. Recall that $\lambda_1(\mathbf{M}) = \max_{\mathbf{x} \in \mathbb{S}^{n-1}}\big\{\mathbf{x}^\top \mathbf{M} \mathbf{x}\big\}$, for $\mathbf{M} \in \mathbb{R}^{n \times n}$ and $\mathbb{S}^{n-1}$ the $n-1$

dimensional ℓ_2-sphere, i.e $\mathbb{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}$. It follows that

$$\begin{aligned}
\lambda_1(\tilde{\Phi}) &= \max_{\mathbf{x}\in\mathbb{S}^{M-1}} \left\{ \mathbf{x}^\top \left(\sum_{i=1}^N \alpha_i \cdot \tilde{u}_i^\top \tilde{u}_i \right) \mathbf{x} \right\} \\
&= \max_{\mathbf{x}\in\mathbb{S}^{M-1}} \left\{ \begin{bmatrix} \mathbf{x}^\top & \vec{0} \end{bmatrix} \left(\sum_{i=1}^N \alpha_i \cdot (U_{\mathbf{B}})_{(i)}^\top (U_{\mathbf{B}})_{(i)} \right) \begin{bmatrix} \mathbf{x} \\ \vec{0} \end{bmatrix} \right\} \\
&\leqslant \max_{\mathbf{y}\in\mathbb{S}^{N-1}} \left\{ \mathbf{y}^\top \left(\sum_{i=1}^N \alpha_i \cdot (U_{\mathbf{B}})_{(i)}^\top (U_{\mathbf{B}})_{(i)} \right) \mathbf{y} \right\} \\
&\leqslant \max_{l\in\mathbb{N}_N}\{\alpha_i\} \cdot \max_{\mathbf{y}=\begin{bmatrix}\mathbf{y}_1^\top & \mathbf{y}_2^\top\end{bmatrix}^\top \in\mathbb{S}^{N-1}} \left\{ \begin{bmatrix} \mathbf{y}_1^\top & \mathbf{y}_2^\top \end{bmatrix} \left(\sum_{i=1}^N (U_{\mathbf{B}})_{(i)}^\top (U_{\mathbf{B}})_{(i)} \right) \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \right\} \\
&\leqslant \max_{l\in\mathbb{N}_N}\{\alpha_i\} \cdot \max_{\mathbf{y}=\begin{bmatrix}\mathbf{y}_1^\top & \mathbf{y}_2^\top\end{bmatrix}^\top \in\mathbb{S}^{N-1}} \left\{ \mathbf{y}_1^\top \left(\sum_{i=1}^N \tilde{u}_i^\top \tilde{u}_i \right) \mathbf{y}_1 \right\} \\
&\leqslant \max_{l\in\mathbb{N}_N}\{\alpha_i\} \cdot \max_{\mathbf{y}=\begin{bmatrix}\mathbf{y}_1^\top & \mathbf{y}_2^\top\end{bmatrix}^\top \in\mathbb{S}^{N-1}} \left\{ \mathbf{y}_1^\top \left(\sum_{i=1}^N \tilde{u}_i^\top \tilde{u}_i \right) \mathbf{y}_1 \right\} \\
&= \max_{l\in\mathbb{N}_N}\{\alpha_i\} \cdot \max_{\mathbf{y}=\begin{bmatrix}\mathbf{y}_1^\top & \mathbf{y}_2^\top\end{bmatrix}^\top \in\mathbb{S}^{N-1}} \left\{ \langle \mathbf{y}_1, \tilde{u}_i \rangle^2 \right\} \\
&= \max_{l\in\mathbb{N}_N}\{\alpha_i\} \cdot \max_{i\in\mathbb{N}_N}\{\tilde{\ell}_i\} \ ,
\end{aligned}$$

which proves the first claim. The bound on $\lambda_1(\bar{\Phi})$ is be proven similarly. □

We now have everything we need in order to prove our main result regarding Algorithm 9. By combining Lemmas 6.2.3 and 6.2.4, we get that

$$\begin{aligned}
\sigma^2 &= \max\left\{ \|\mathbb{E}[\mathbf{X}^\top \mathbf{X}]\|_2, \|\mathbb{E}[\mathbf{X}\mathbf{X}^\top]\|_2 \right\} \\
&\leqslant \frac{\tau}{\beta} \cdot \max\left\{ \max_{l\in\mathbb{N}_N}\{\alpha_l\} \cdot \max_{i\in\mathbb{N}_N}\{\tilde{\ell}_i\}, \max_{l\in\mathbb{N}_N}\{1/\alpha_l\} \cdot \max_{i\in\mathbb{N}_N}\{\bar{\ell}_i\} \right\} + 1 \ .
\end{aligned}$$

To simplify our statement, we assume that

$$\max\left\{ \max_{l\in\mathbb{N}_N}\{\alpha_l\} \cdot \max_{i\in\mathbb{N}_N}\{\tilde{\ell}_i\}, \max_{l\in\mathbb{N}_N}\{1/\alpha_l\} \cdot \max_{i\in\mathbb{N}_N}\{\bar{\ell}_i\} \right\} \leqslant \frac{\mu}{\tau} \ , \tag{6.13}$$

which implies that $\sigma^2 \leqslant \mu/\beta + 1$.

Theorem 6.2.4. *Let $L > M$, $\delta > 0$ and assume that (6.13) holds. Then, the sketching matrix $\hat{\mathbf{\Pi}}$ of Algorithm 9 with samplin according to $\{\hat{\pi}_i\}_{i=1}^N$; meets condition (6.10) with probability at least $(1 - \delta \cdot \sqrt{L/M})$, for $r = \Theta(\mu \log(2\mu/\delta)/(\beta\epsilon^2))$.*

Proof. Under the assumption that (6.13) holds for the pair of matrices $\mathbf{A}$ and $\mathbf{B}$, we know that $\sigma^2 \leqslant \mu/\beta + 1$; for all random sampling trials of Algorithm 9. The corresponding argument $D_{\hat{\mathbf{\Pi}}}$ of (6.10) for $r \geqslant 1$, is equal to $\frac{1}{r}\sum_{l=1}^{r} \mathbf{X}_l$ for r realizations of (6.12). By substituting σ and γ according to Lemma 6.2.2 in the exponent of the bound in Theorem 6.2.3, we get

$$\begin{aligned}
\frac{-r\epsilon^2/2}{\mu/\beta + 1 + (\mu+1)\epsilon/3} &\leqslant \frac{-r\epsilon^2/2}{\mu/\beta + \beta + (\mu+1)\epsilon/3} \\
&= \frac{-r\beta\epsilon^2/2}{(\mu+1)(1+\epsilon/3)} \\
&= \frac{-\Theta\left(\mu \log(2\mu/\delta)/(\beta\epsilon^2)\right) \frac{\beta\epsilon^2}{2(1+\epsilon/3)}}{\mu+1} \\
&= -\Theta(\log(2\mu/\delta)) \ ,
\end{aligned}$$

thus

$$\begin{aligned}
\Pr\left[\|D_{\hat{\mathbf{\Pi}}}\|_2 > \epsilon\right] &\leqslant (L+M)\cdot e^{-\Theta(\log(2\mu/\delta))} \\
&= 2e^{\log L - \Theta(\log(2\mu/\delta))} \\
&= 2e^{\Theta\left(\log\left(\frac{2L\delta}{\mu}\right)\right)} \\
&= e^{\Theta\left(\log\left(\delta\cdot\sqrt{L/M}\right)\right)} = \delta\cdot\sqrt{L/M} \ .
\end{aligned}$$

Taking the complementary event completes the proof. □

As was briefly discussed in Subsection 6.2.2, the only difference in sampling according to approximate joint leverage scores $\{\hat{\pi}_i\}_{i=1}^N$ and exact scores $\{\pi_i\}_{i=1}^N$, is that with the former we need to oversample by a factor of $1/\beta$. We see this explicitly through their guarantees in terms of the error defined in Lemma 6.2.1, provided in Proposition 6.2.2 and Corollary 6.2.1 respectively.

Proposition 6.2.2. *For $\tilde{\varepsilon}, \delta > 0$ and $r = \Theta\big(\mu\log(2\mu/\delta)\cdot\big(\sigma_1(\mathbf{A})\sigma_1(\mathbf{B})/(\beta\tilde{\varepsilon})\big)^2\big)$, the sketching matrix $\hat{\mathbf{\Pi}}$ of Algorithm 9 with sampling according to $\{\hat{\pi}_i\}_{i=1}^N$, satisfies*

$$\Pr\left[\mathrm{err}_{\mathbf{A},\mathbf{B}}^{\ell_2,\hat{\mathbf{\Pi}}} \leqslant \tilde{\varepsilon}\right] \geqslant 1 - \delta\cdot\sqrt{L/M}.$$

Proof. Let $\tilde{\varepsilon} = \epsilon\cdot\sigma_1(\mathbf{A})\sigma_1(\mathbf{B})$. By Lemma 6.2.1 we know that $\mathrm{err}_{\mathbf{A},\mathbf{B}}^{\ell_2,\hat{\mathbf{\Pi}}} \leqslant \epsilon\cdot\sigma_1(\mathbf{A})\sigma_1(\mathbf{B})$, so by our choice of $\tilde{\varepsilon}$ we have $\Pr\left[\mathrm{err}_{\mathbf{A},\mathbf{B}}^{\ell_2,\hat{\mathbf{\Pi}}} \leqslant \tilde{\varepsilon}\right] = \Pr\left[\|\Delta_{\mathbf{A},\mathbf{B}}\| \leqslant \epsilon\right]$. By Theorem 6.2.4, it follows that $\Pr\left[\|\Delta_{\mathbf{A},\mathbf{B}}\| \leqslant \epsilon\right] \leqslant 1 - \delta\cdot\sqrt{\frac{L}{M}}$, for $r = \Theta(\mu\log(2\mu/\delta)/\epsilon^2) = \Theta\big(\mu\log(2\mu/\delta)\cdot(\sigma_1(\mathbf{A})\sigma_1(\mathbf{B})/\tilde{\varepsilon})^2\big)$. □

Corollary 6.2.1. *Let $\tilde{\varepsilon}, \delta > 0$ and $r = \Theta\big(\mu \log(2\mu/\delta) \cdot \big(\sigma_1(\mathbf{A})\sigma_1(\mathbf{B})/\tilde{\varepsilon}\big)^2\big)$, and consider sampling according to the joint leverage score distribution $\{\pi_i\}_{i=1}^N$, for which Algorithm 9 produces the sketching matrix $\mathbf{\Pi}$. We then have*

$$\Pr\left[\mathrm{err}_{\mathbf{A},\mathbf{B}}^{\ell_2,\mathbf{\Pi}} \leqslant \tilde{\varepsilon}\right] \geqslant 1 - \delta \cdot \sqrt{L/M}.$$

Proof. If we sample according to the exact distribution $\{\pi_i\}_{i=1}^N$, it follows that $\beta = 1$. By Proposition 6.2.2, the statement is immediate. □

A drawback of the above results is that the largest singular value of both matrices appear in the sampling complexities. Note though that if we made an analogous assumption to Theorem 6.2.2; that $\|\mathbf{A}\|_2 \leqslant 1$, this would not be a concern. Since we are dealing with the spectral norm though, it seems inevitable that $\sigma_1(\mathbf{A})$ and $\sigma_1(\mathbf{B})$ would not appear.

To make a fair comparison with the $CR-$MM guarantee of Theorem 6.2.2, let us assume that $\|\mathbf{A}\|_2, \|\mathbf{B}\|_2 \leqslant 1$, and that the sampling complexities for the two algorithms are the same. For Algorithm 9 we would then have $r = \Theta\big(\mu \log(2\mu/\delta')/\epsilon^2\big)$ for $\delta' > 0$, $\mu = 96\|\mathbf{A}\|_F\|\mathbf{B}\|_F$, and we would get the same probability of error when the accuracy parameters are selected such that $\delta' = \Theta\big(\epsilon^2\sqrt{\delta}\big)$. Moreover, what we presented holds true for the product of any pair of distinct matrices $\mathbf{A}$ and $\mathbf{B}$, while Theorem 6.2.2 only considers the Gram matrix of $\mathbf{A}$.

6.3 Implications to other AMM Algorithms

In this section, we discuss how our approach through joint leverage scores can be used to derive spectral guarantees for other similar AMM algorithms, and briefly discuss how it can be utilized for spectral sparsification of 'intersection graphs'. In Subsection 6.3.1 we derive a guarantee for $CR-$MM, and in Subsection 6.3.2 we briefly discuss how our approach can benefit by first applying a random projection and then performing uniform random sampling on the transformed matrices' row pairs. In Subsection 6.3.3, we bridge a connection to graph sparsification.

6.3.1 $CR-$MM Through Approximate Joint Leverage Scores

A pitfall of our algorithm, is that it requires carrying out an singular value or QR decomposition for each of the matrices $\mathbf{A}$ and $\mathbf{B}$, in order to calculate the joint leverage scores. We therefore need $O(N^2(L+M))$ operations to compute $\{\pi_i\}_{i=1}^N$. Instead, more efficient leverage score approximation algorithms could be used [91,

142, 249, 273], that do not require directly computing a left orthonormal basis of each matrix. This is part of the reason Theorem 6.2.4 and Proposition 6.2.2 are present in terms of an approximate distribution $\{\hat{\pi}_i\}_{i=1}^N$.

The sampling procedure of the $CR-$MM algorithm on the other hand, can be done efficiently with $O(1)$ additional storage space, by the pass-efficient SELECT algorithm [201, page 13]. In Proposition 6.3.1, we quantify the AMM spectral bound $CR-$MM, by using the distribution of $\{p_i\}_{i=1}^N$ of $CR-$MM as a surrogate to $\{\pi_i\}_{i=1}^N$. In many practical cases, we empirically observed that the two distributions were relatively close to each other, *i.e.* $p_i \geqslant \tilde{\beta}\pi_i$ for a small $\tilde{\beta} \in (0, 1]$.

Lemma 6.3.1. *For all $i \in \mathbb{N}_N$, we have $\pi_i \cdot \left(\frac{\tau}{\phi} \cdot \sigma_{\min}(\mathbf{A}) \cdot \sigma_{\min}(\mathbf{B})\right) \leqslant p_i$.*

Proof. Recall that $p_i = \|\mathbf{A}_{(i)}\|_2 \cdot \|\mathbf{B}_{(i)}\|_2/\phi$; for $\phi \coloneqq \sum_{j=1}^N \|\mathbf{A}_{(j)}\|_2 \cdot \|\mathbf{B}_{(j)}\|$. Consider the reduced QR factorization of $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q} \in \mathbb{R}^{N\times L}$ matches $U_{\mathbf{A}}$, and $\mathbf{R}$ is an upper triangular matrix. It follows that $\mathbf{Q} = \mathbf{AR}^{-1}$ for $\mathbf{R}^{-1}$ also an upper triangular matrix, and

$$\sqrt{\ell_i} = \|\mathbf{Q}_{(i)}\|_2 = \|\mathbf{A}_{(i)} \cdot \mathbf{R}^{-1}\|_2 \leqslant \|\mathbf{A}_{(i)}\|_2 \cdot \|\mathbf{R}^{-1}\|_2 = \|\mathbf{A}_{(i)}\|_2 \cdot \frac{1}{\sigma_{\min}(\mathbf{A})} \ .$$

Similarly, one can show that $\sqrt{\ell_i} \leqslant \|\mathbf{B}_{(i)}\|_2 \cdot \frac{1}{\sigma_{\min}(\mathbf{B})}$. By combining the two bounds, we get

$$\mathscr{J}_{ii} \cdot \sigma_{\min}(\mathbf{A}) \cdot \sigma_{\min}(\mathbf{B}) \leqslant \|\mathbf{A}_{(i)}\|_2 \cdot \|\mathbf{B}_{(i)}\|_2 = p_i \cdot \phi \quad \Longrightarrow \quad \pi_i \cdot \left(\frac{\tau}{\phi} \cdot \sigma_{\min}(\mathbf{A}) \cdot \sigma_{\min}(\mathbf{B})\right) \leqslant p_i$$

which completes the proof. □

Proposition 6.3.1. *Let $L > M$ and $\delta, \epsilon > 0$. For $\beta = \left((\tau/\phi) \cdot \sigma_{\min}(\mathbf{A}) \cdot \sigma_{\min}(\mathbf{B})\right)$ and $r = \Theta(\mu \log(2\mu/\delta)/(\beta\epsilon^2))$, $CR-$MM satisfies*

$$\Pr\left[\|U_{\mathbf{A}}^\top U_{\mathbf{B}} - (\hat{\mathbf{\Pi}} U_{\mathbf{A}})^\top (\hat{\mathbf{\Pi}} U_{\mathbf{B}})\|_2 \leqslant \epsilon\right] \geqslant 1 - \sqrt{L/M} \cdot \delta \ .$$

Proof. This follows directly by taking $\tilde{\beta} = \left((\tau/\phi) \cdot \sigma_{\min}(\mathbf{A}) \cdot \sigma_{\min}(\mathbf{B})\right)$; based on the conclusion of Lemma 6.3.1, and using $\{p_i\}_{i=1}^N$ as an approximate distribution to $\{\pi_i\}_{i=1}^N$ in Theorem 6.2.4. Furthermore, the constant $\tilde{\beta}$ lies in $(0, 1]$, as all its terms are positive; and $\{pi_i\}_{i=1}^N$, $\{\pi_i\}_{i=1}^N$ are both valid sampling distributions. □

6.3.2 Data-Oblivious AMM through Joint Leverage Scores

Recall that the '*Subsampled Randomized Hadamard Transform*' (SRHT) [96] is comprised of three matrices: $\mathbf{\Omega} \in \mathbb{R}^{r\times N}$ a uniform sampling and rescaling matrix of r rows, $\hat{\mathbf{H}}_N \in \{\pm 1/\sqrt{N}\}^{N\times N}$ the normalized Hadamard matrix for $N = 2^n$, and $\mathbf{D} \in \{0, \pm 1\}^{N\times N}$ with i.i.d. diagonal Rademacher random entries. The structured sketching matrix $\hat{\mathbf{S}} = \mathbf{\Omega}\hat{\mathbf{H}}_N\mathbf{D} \in \mathbb{R}^{r\times N}$ of a matrix $\mathbf{A} \in \mathbb{R}^{N\times L}$, flattens the leverage scores of $\mathbf{A}$, *i.e.* the normalized leverage scores of $\widetilde{\mathbf{A}} \coloneqq \hat{\mathbf{H}}_N\mathbf{D}\mathbf{A}$ are approximately equal [5, 96]. Hence, once we apply the randomized transform $\hat{\mathbf{H}}_N\mathbf{D}$ to $\mathbf{A}$, we can sample rows of $\widetilde{\mathbf{A}}$ uniformly at random with replacement (which is done through $\mathbf{\Omega}$) without directly utilizing any information from $\mathbf{A}$; to get an *oblivious* ℓ_2-subspace embedding of $\hat{\mathbf{A}} \coloneqq \hat{\mathbf{S}}\mathbf{A}$.

Now, consider two distinct matrices $\mathbf{A} \in \mathbb{R}^{N\times L}$ and $\mathbf{B} \in \mathbb{R}^{N\times M}$. By applying to each of them the above randomized transform, we get $\widetilde{\mathbf{A}} = \hat{\mathbf{H}}_N\mathbf{D}\mathbf{A}$ and $\widetilde{\mathbf{B}} = \hat{\mathbf{H}}_N\mathbf{D}\mathbf{B}$ both of which have *approximately* uniform leverage scores, hence, the joint leverage scores of the pair $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{B}}$ is also approximately uniform. From this, the conclusion of Theorem 6.2.4 directly applies to the approximate product

$$\mathbf{A}^\top \cdot \mathbf{B} \approx \left(\hat{\mathbf{H}}_N\mathbf{D} \cdot \mathbf{A}\right)^\top \cdot \left(\mathbf{\Omega}^\top\mathbf{\Omega}\right) \cdot \left(\hat{\mathbf{H}}_N\mathbf{D} \cdot \hat{\mathbf{B}}\right) = \hat{\mathbf{A}}^\top \cdot \hat{\mathbf{B}} \,.$$

Furthermore, random Gaussian and orthonormal matrices could also be considered in place of the transform $\hat{\mathbf{H}}_N\mathbf{D}$, as these also flatten the leverage scores of the matrices; and additionally have security guarantees [49, 306]. The main drawback of these approaches, is that they are slower than the SRHT; which can compute the sketch $\hat{\mathbf{A}}$ in $O(NL \log N)$ time through Fourier methods.

6.3.3 Graph Spectral Sparsifiers

The state-of-the-art approach to spectral sparsification of a weighted graph $G = (V, E, w)$ is to sample edges according to *effective resistances*, which has connections to leverage scores of the graph's boundary matrix $\boldsymbol{B} \in \mathbb{R}_{\geqslant 0}^{|E|\times|V|}$ [92, 271, 273]. This approach leads to a nearly-linear time algorithm that produces high-quality sparsifiers, by approximating G's *Laplacian* $\boldsymbol{L}$ which is defined as the outer-product of its *boundary matrix*; *i.e.* $\boldsymbol{L} = \boldsymbol{B}^\top\boldsymbol{B}$. That is, we get a sparser graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$ with Laplacian $\tilde{\boldsymbol{L}} \approx \boldsymbol{L}$.

The benefit of using the bound of Theorem 6.2.4, is that it can be applied to directly approximate the *intersection* of two different graphs on V; say $G_1 = (V, E_1, w^1)$ and $G_2 = (V, E_2, w^2)$, which has found applications in areas such as image segmen-

tation. The intersection of these two graphs is defined as $H = \left(V, E_H, w^H\right)$, where $E_H = E_1 \cap E_2$ and $w_e^H = \sqrt{w_e^1 \cdot w_e^2}$ for each $e \in E_H$ (w.l.o.g., we assume that the indices of the edges are the same in G_1 and G_2).

We can use AMM through joint leverage scores to approximate $\boldsymbol{L}_{1,2} = \boldsymbol{B}_1^T \boldsymbol{B}_2$, for $\boldsymbol{B}_1, \boldsymbol{B}_2$ the boundary matrices of the two graphs. The techniques of [273] on sampling according to effective resistances do not apply, though through what was proposed in this chapter; one can extend their work to consider '*joint effective resistances*' defined across the two graphs.

The orientation of an edge $e = (u, v)$, is represented in the *incidence vector* $\chi_e = \mathbf{e}_u - \mathbf{e}_v$; for $\mathbf{e}_i \in \mathbb{R}^V$ the standard basis vectors. Let $\tilde{x}_e = \sqrt{\boldsymbol{L}^\dagger}\chi_e$, for each $e \in E$. Then, the effective resistances of e is defined as $r_e = \|\tilde{x}_e\|_2^2$. Denote the effective resistances of G_ι and the corresponding individual leverage scores of $\boldsymbol{B}_\iota$ by $\left\{r_e^\iota\right\}_{e \in E_\iota}$ and $\{\ell_e^\iota\}_{e \in E_\iota}$ respectively, for each $\iota \in \{1, 2\}$. If now we consider the aforementioned distinct matrices G_1 and G_2, we can define their '*joint effective resistances*' as

$$r_{e'}^H = \frac{\sqrt{\ell_{e'}^1 \cdot \ell_{e'}^2}}{\sqrt{w_{e'}^1 \cdot w_{e'}^2}} = \sqrt{r_{e'}^1 \cdot r_{e'}^2}$$

for each of $e' \in E_H$. Further investigation of this would be worthwhile future work.

6.4 Concluding Remarks and Future Work

The main contributions of this chapter are the following. First, we connected ideas from two distinct RandNLA algorithms to (i) define a new generalized notion of leverage scores; which when $\mathbf{A} = \mathbf{B}$ results in the leverage scores of $\mathbf{A}$, and (ii) developed an AMM algorithm for which a spectral guarantee was provided. Our algorithm produces a low-rank approximation, which is more efficient to work with in terms of storage and computation, in comparison to the exact matrix product. To do so, we proposed a condition for AMM which is similar to the ℓ_2-subspace embedding property. We also gave a connection to graph sparsifiers, and presented a generalized notion of effective resistances, which is worth investigating further. Another area which could be benefit from joint leverage scores, is that of tensor products and decompositions; where an analogous notion could potentially be defined. It would be interesting to see what other optimization problems and applications can benefit from utilizing sampling according to the joint leverage score distribution.

CHAPTER VII

Conclusion and Future Work

This dissertation primarily focused on randomized sketching algorithms for Euclidean subspace embeddings through block sampling, and coded computing; for both exact and approximate computations. We also considered encryption through random projections; for iterative sketching, encryption through Lagrange interpolation in the coded computing paradigm, and spectral sparsification of graph Laplacians.

Though the two approaches of coded computing and sketching used to accelerating computations utilize vastly different techniques, in Chapters III and IV we gave approaches to how one can utilize sketching techniques for distributed steepest descent through approximate gradient coding. These ideas could also be utilized in distributed coded matrix multiplication. In Chapter II we focused on binary gradient coding and coded matrix multiplication, where we generalized fractional binary repetition codes. Chapter V dealt with the problem of coded matrix inversion; which has not been studied as extensively as other coded computations. Finally, in Chapter VI a new approximate matrix-matrix multiplication algorithm was proposed, which guarantees a spectral approximation of the exact product. In Appendix E we bridge a connection between approximate matrix-matrix multiplication and spectral sparsification of graph Laplacians. Next, we summarize worthwhile future work pertaining to each chapter.

- **Chapter II — Generalized Fractional Repetition Codes for Binary Coded Computations**:

  In this chapter, we introduced a binary GCS for distributed optimization. The main advantages of our code design is that (i) it provides numerically stable computations of the gradient, (ii) it removes the limiting assumption $(s+1) \mid n$, and (iii) it has an improved and more tractable decoding stage compared to that of the first GCS proposed in [279]. We provided an analysis

of the proposed design, and showed that it is optimal in terms of the minimizing function d_s defined in Definition 2.3.1. Both homogeneous and heterogeneous workers were considered. It is worth noting that more recent work also considers this direction [269], though their focus is on improving the recovery threshold. We then presented two CMM approaches; as extensions of our binary GCS.

There are several interesting directions for future work. We have seen that the proposed schemes accommodate various matrix operations. It would be interesting to see what other operations they can accommodate, in order to devise exact and approximate straggler resilient coded computing schemes. Another direction is to incorporate privacy and security into our schemes. A third direction, is to further explore the connections between coded computations and codes for distributed storage systems. Specifically, it would be worthwhile to explore the connections between the proposed GCS, the GCS of [155], and the distributed storage systems of [100], which we briefly described in Subsection 2.6.4.

- **Chapter III — Gradient Coding through Iterative Block Leverage Score Sampling**:

  In this chapter, we showed how one can exploit results from RandNLA to distributed CC, in the context of GC. By taking enough samples, or equivalently; waiting long enough, the approximation errors can be made arbitrarily small. In terms of CC, the advantages are that *encodings* correspond to a scalar multiplication, and *no* decoding step is required. By utilizing these techniques, we are also advantageous over other CC approximation schemes [25, 52, 53, 58, 59, 112, 120, 144, 155, 239, 253]; by incorporating information from our dataset into our scheme.

  Our methods were validated numerically through various experiments, presented in Section 3.4. Further experiments were performed on various distributions for $\mathbf{A}$, in which similar results were obtained. We also considered the empirical distribution from real-server completion times taken from 500 AWS-servers [19], and emulated the proposed CC scheme. In this experiment, we obtained the expected results in terms of ℓ_2-s.e., misestimation factors $\beta_{\Pi,\bar{\Pi}}$, and metrics $\Delta_{\Pi,\bar{\Pi}}$, $d_{\Pi,\bar{\Pi}}$.

  Even though we focused on leverage score sampling for linear regression, other sampling algorithms and problems could benefit by designing analogous

replication schemes. One such problem is the *column subset selection problem*, which can be used to compute partial SVD, QR decompositions, as well as low-rank approximations [201]. As for the sampling technique we studied, one can judiciously define a sampling distribution to approximate solutions to such problems [34], which are known to be NP-hard under the Unique Games Conjecture assumption [66].

Furthermore, existing block sampling algorithms can also benefit from the proposed expansion networks, *e.g.* CR-multiplication [53] and CUR decomposition [221]. For instance, a coded matrix multiplication algorithm of minimum variance can been designed, where the sampling distribution proposed in [53] is used to determine the replication numbers of the expansion network. In terms of matrix decompositions, the block leverage score algorithm of [221] can be used to distributively determine an additive ϵ-error decomposition of $\mathbf{A}$, in the CC setting. Another future direction is generalizing existing tensor product and factorization algorithms to block sampling, according to both approximate and exact sampling distributions, in order to make them practical for distributed environments.

- **Chapter IV — Iterative Sketching for Secure Coded Regression**:

  In this chapter, we proposed approximately solving a linear system by distributively leveraging iterative sketching and performing first-order SD simultaneously. In doing so, we benefit from both approximate GC and RandNLA. A difference to other RandNLA works is that our sketching matrices sample *blocks* uniformly at random, after applying a random orthonormal projection. An additional benefit is that by considering a large ensemble of orthonormal matrices to pick from, under necessary assumptions, we guarantee information-theoretic security while performing the distributed computations. This approach also enables us to not require encoding and decoding steps at every iteration. We also studied the special case where the projection is the randomized Hadamard transform, and discussed its security limitation. To overcome this, we proposed a modified "garbled block-SRHT", which guarantees computational security.

  We note that applying orthonormal random matrices also secures coded matrix multiplication. There is a benefit when applying a garbled Hadamard transform in this scenario, as the complexity of multiplication resulting from the sketching is less than that of regular multiplication. Also, if such a random projection is used before performing CR-multiplication distributively [43, 53, 250],

the approximate result will be the same. Moreover, our dimensionality reduction algorithm can be utilized by a single server, to store low-rank approximations of very large data matrices.

Partial stragglers, have also been of interest in the GC literature. These are stragglers who are able to send back a portion of their requested tasks. Our work is directly applicable, as we can consider smaller blocks, with multiple ones allocated to each worker.

There are several interesting directions for future work. We observed experimentally in Figure IV.5 that $\mathbf{\Pi}$ and $\hat{\mathbf{H}}_N$ may act as preconditioners for SSD. This mere observation requires further investigation. Another direction is to see if the proposed ideas could be applied to federated learning scenarios, in which security and privacy are major concerns. Some of the projections we considered, rely heavily on the recursive structure of $\hat{\mathbf{H}}$ in order to satisfy (g). One thing we overlooked, is whether other efficient multiplication algorithms (*e.g.* Strassen's [276]) could be exploited, in order to construct suitable projections. It would be interesting to see if other structured or sparse matrices exist which also satisfy our desired properties (a)-(g).

There has been a lot of work regarding second-order algorithms with iterative sketching, *e.g.* [173, 231]. Utilizing iterative Hessian sketching or sketched Newton's method in CC has been explored in a tangential work [127], though the security aspect of these algorithms has not been extensively studied. A drawback here is that the local computations at the workers would be much larger, though we expect the number of iterations to be significantly reduced; for the same termination criterion to be met, compared to first-order methods. Deeper exploration of the theoretical guarantees of iterative sketched first-order methods, along with a comparison to their second-order counterparts, as well as studying their effect in logistic regression and other applications, are also of potential interest.

- **Chapter V — Securely Aggregated Coded Matrix Inversion**:

  In this chapter, we addressed the problem of approximate computation of the inverse of a matrix distributively in a relaxed FL setting, under the possible presence of straggling workers. We provided approximation error bounds for our approach, as well as security and recovery guarantees. We also provided numerical experiments that validated our proposed approach.

There are several interesting future directions. One avenue to consider is incorporating fully homomorphic encryption in our phases (b),(c),(d), to obtain a FL scheme; and prevent the requirement of clients need to recover each others' information. An important issue is the numerical stability of the BRS approach, so exploring other suitable generator matrices could be beneficial; *e.g.* circulant permutation and rotation matrices [236]. It is also worth investigating if we can reduce the communication rounds when computing the pseudoinverse through our approach. This depends on the CMM which is being utilized, though using different ones for each of the two multiplications may also be beneficial. An interesting approach to also consider, is if divide and conquer algorithms could be leveraged in CC to recover exact or approximate solutions to $\mathbf{A}^{-1}$.

In terms of coding-theory, it would be interesting to see if it is possible to reduce the complexity of our decoding step. Specifically, could well-known RS decoding algorithms such as the Berlekamp-Welch algorithm be exploited? Another direction, is leveraging approximate CCMs. The work of [148] considers the GC problem for *approximate* and *exact* recovery through Lagrange interpolation, for heterogeneous workers in the presence of stragglers and adversaries. A potential scheme for matrix inversion could also be developed through the methods of [148]. In terms of our approximation algorithms, an avenue worth exploring is that of incorporating approximate and/or sparse Gaussian elimination [171, 172] into our distributed CCM.

- **Chapter VI — Approximate Matrix Multiplication by Joint Leverage Score Sampling**:

  The main contributions of this chapter are the following. First, we connected ideas from two distinct RandNLA algorithms to (i) define a new generalized notion of leverage scores; which when $\mathbf{A} = \mathbf{B}$ results in the leverage scores of $\mathbf{A}$, and (ii) developed an AMM algorithm for which a spectral guarantee was provided. Our algorithm produces a low-rank approximation, which is more efficient to work with in terms of storage and computation, in comparison to the exact matrix product. To do so, we proposed a condition for AMM which is similar to the ℓ_2-subspace embedding property. We also gave a connection to graph sparsifiers, and presented a generalized notion of effective resistances, which is worth investigating further. Another area which could be benefit from joint leverage scores, is that of tensor products and decompositions; where an analogous notion could potentially be defined. It would be interesting to see

what other optimization problems and applications can benefit from utilizing sampling according to the joint leverage score distribution.

- **Appendix E — Graph Sparsification by Approximate Matrix Multiplication**:

  In this appendix, we proposed a graph sparsifier that approximates Laplacian through the use of $CR-$MM; a sampling with replacement technique, adapted from RandNLA. Applications of the proposed method to spectral clustering through *block* sampling [53, 217] would be worthwhile future work. Specifically, cliques of a given graph may be determined by approximating their Laplacians. The proposed computationally efficient spectral approximation may permit the identification of highly connected vertices without the need to traverse through the entire graph.

APPENDICES

Appendix A

Appendix to Chapter II

1.1 Pseudocode of Encoding Matrices $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$ and $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$

In this appendix we provide the pseudocode of the encoding matrices $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$ and $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$ described in 2.4.2 and 2.4.3, which are combined to give the encoding matrix $\mathbf{B}$ of our GCS.

Algorithm 10: Encoding $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$ — $\mathfrak{C}_1 = \{[i]_{s+1}\}_{i=0}^{r-1}$

Input: number of workers n and stragglers s, where $s, n \in \mathbb{Z}_+$
Output: encoding matrix $\tilde{\mathbf{B}}_{\mathfrak{C}_1} \in \{0,1\}^{n\times n}$ ▷ assume $n = k$
$\tilde{\mathbf{B}}_{\mathfrak{C}_1} \leftarrow \mathbf{0}_{n\times n}$, and use the division algorithm to get the parameters:
$n = \ell \cdot (s+1) + r \qquad r = t \cdot \ell + q \qquad n = \lambda \cdot (\ell+1) + \tilde{r}$
for $i = 0$ *to* $r-1$ **do**
  if $\ell + r > s$ **then**
    for $j = 1$ *to* $\ell + r - s$ **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_1}\Big[(j-1)(s+1)+i, (j-1)(s+1)+1 : j(s+1)\Big] = \mathbf{1}_{1\times(s+1)}$
    end
    for $j = \ell + r - s + 1$ *to* $\ell + 1$ **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_1}\Big[(j-1)(s+1)+i, (j-1)s+(\ell+r-s)+1 : (j-1)s+\ell+r\Big] = \mathbf{1}_{1\times s}$
    end
  end
  else if $\ell + r \le s$ **then**
    for $j = 1$ *to* $\tilde{r}$ **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_1}\Big[(j-1)(s+1)+i, (j-1)(\lambda+1)+1 : j(\lambda+1)\Big] = \mathbf{1}_{1\times(\lambda+1)}$
    end
    for $j = \tilde{r} + 1$ *to* $\ell + 1$ **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_1}\Big[(j-1)(s+1)+i, (j-1)\lambda+\tilde{r}+1 : (j-1)\lambda+\tilde{r}+\lambda\Big] = \mathbf{1}_{1\times\lambda}$
    end
  end
end
return $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$

Algorithm 11: Encoding $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$ — $\mathfrak{C}_2 = \{[i]_{s+1}\}_{i=r}^{s}$

Input: number of workers n and stragglers s, where $s, n \in \mathbb{Z}_+$
Output: encoding matrix $\tilde{\mathbf{B}}_{\mathfrak{C}_2} \in \{0,1\}^{n\times n}$ ▷ assume $n = k$
$\tilde{\mathbf{B}}_{\mathfrak{C}_2} \leftarrow \mathbf{0}_{n\times n}$, and use the division algorithm to get the parameters:
$n = \ell \cdot (s+1) + r \qquad r = t \cdot \ell + q \qquad n = \lambda \cdot (\ell + 1) + \tilde{r}$
for $i = r$ *to* s **do**
  if $q \equiv 0$ **then**
    for $j = 1$ *to* ℓ **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_2}\Big[(j-1)(s+1)+i, (j-1)(s+t+1)+1 : j(s+t+1)\Big] = \mathbf{1}_{1\times(s+t+1)}$
    end
  end
  else if $q > 0$ **then**
    for $j = 1$ *to* q **do**
      $\tilde{\mathbf{B}}_{\mathfrak{C}_2}\Big[(j-1)(s+1)+i, (j-1)(s+t+2)+1 : j(s+t+1)\Big] = \mathbf{1}_{1\times(s+t+2)}$
    end
    for $j = q+1$ *to* ℓ **do**
      $\mathbf{B}\Big[(j-1)(s+1)+i, (j-1)(s+t+1)+q+1 : j(s+t+1)+q\Big] = \mathbf{1}_{1\times(s+t+1)}$
    end
  end
end
return $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$

1.2 Special Case of CMM-2

In this appendix, we present a special of CMM-2; where $k_1 = 1$ and $k = k_2$, as this may have certain applications where one cannot partition both matrices, *e.g.* [53]. Additionally, it is simpler to understand this case, and then view CMM-2 as applying this special case of the code k_1 times.

In contrast to the partitioning (2.28) of CMM-1; and the general case of CMM-2 (2.34), in this approach we partition only one of the two matrices, say B; along its columns. Each worker computes the product of a submatrix of B with the matrix A, and then the central server augments the received computations accordingly. The key property that we utilize is the following:

$$AB = A \cdot \Big[\bar{B}_1 \ \cdots \ \bar{B}_k\Big] = \Big[A\bar{B}_1 \ \cdots \ A\bar{B}_k\Big] = \Big[\tilde{C}_1 \ \cdots \ \tilde{C}_k\Big] \tag{1.1}$$

where $\tilde{C}_j \coloneqq A\bar{B}_j$, for all $j = 1, \cdots, k$. For convenience we assume that $(s+1) \mid M$, and similar to our GCS that $n = k$. To further simplify our construction and description, under the assumption that $(s+1) \mid M$, we can assume that the equipotent partitions of $\{\bar{B}_i\}_{i=1}^{k}$ which are distributed to the workers are of size $\mathrm{T} = M/k \in \mathbb{Z}_+$, which implies that $(s+1) \mid k$. All in all, each worker receives only one $\bar{B}_j \in \mathbb{R}^{N\times T}$; where

T is a multiple of $(s+1)$, and $\tilde{C}_j \in \mathbb{R}^{L\times T}$. We also note that a similar multiplication can take place if we instead partition A along its rows — this corresponds to the case where $k_2 = 1$ and $k = k_1$. For completeness, we present the corresponding encoding through matrix $\tilde{\mathbf{B}} \in \{0,1\}^{k\mathrm{T}(s+1)\times k\mathrm{T}}$

$$\overbrace{\left(\mathbf{I}_k \otimes \mathbf{1}_{(s+1)\times 1} \otimes \mathbf{I}_{\mathrm{T}}\right)}^{\tilde{\mathbf{B}}\in\{0,1\}^{k\mathrm{T}(s+1)\times k\mathrm{T}}} \cdot \begin{bmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_k^T \end{bmatrix} = \begin{bmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_1^T \\ \vdots \\ \tilde{C}_k^T \\ \vdots \\ \tilde{C}_k^T \end{bmatrix}, \tag{1.2}$$

where the transpose of each submatrix $\tilde{C}_j$ appears $s+1$ times along the rows of the encoding $\tilde{\mathbf{B}}C^T \in \mathbb{R}^{(s+1)M\times L}$, each corresponding to one of the $s+1$ potential workers that are asked to compute $\tilde{C}_j$. Lastly, in this special case of CMM-2, no rearrangement is needed once the decoding step has been applied.

1.3 Numerical Example of the Proposed Encodings and Decodings

In this appendix, we give examples of our encoding and decoding algorithms, to convey the main ideas and help visualize the task allocations which take place. For our GCS of Section 2.3, consider the case where $n = k = 11$ and $s = 3$.

By (2.8), (2.9), (2.10) we then have $\ell = 2, r = 3, t = 1, q = 1$; thus $\ell > r - s$, and the task allocation for $\mathfrak{C}_1$ is described by $\mathbf{B}_{\mathfrak{C}_1} \in \{0,1\}^{(\ell+1)\cdot r\times n}$:

$$\mathbf{B}_{\mathfrak{C}_1} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & & & & & \\ \mathit{1} & \mathit{1} & \mathit{1} & \mathit{1} & & & & & & & \\ \mathtt{1} & \mathtt{1} & \mathtt{1} & \mathtt{1} & & & & & & & \\ & & & & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ & & & & \mathit{1} & \mathit{1} & \mathit{1} & \mathit{1} & & & \\ & & & & \mathtt{1} & \mathtt{1} & \mathtt{1} & \mathtt{1} & & & \\ & & & & & & & & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & & & & & & & & \mathit{1} & \mathit{1} & \mathit{1} \\ & & & & & & & & \mathtt{1} & \mathtt{1} & \mathtt{1} \end{bmatrix},$$

where each congruence class is represented by a different color and font for clarity. The zero entries are omitted. The indicated dimensions are for the case where $r > 0$, *i.e.* the remainder block is not empty. The encoding corresponding to the congruence classes 0 to $r-1$ constructed by Algorithm 10, is obtained from $\mathbf{B}_{\mathfrak{C}_1}$ by properly appending zero vectors. Specifically, $\mathbf{B}_{\mathfrak{C}_1}$ is the restriction of $\tilde{\mathbf{B}}_{\mathfrak{C}_1}$ to the rows with nonzero entries.

For the remaining congruence classes, r to s, since $q = 1$; we have:

$$\mathbf{B}_{\mathfrak{C}_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & & & & & \\ & & & & & & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

where $\mathbf{B}_{\mathfrak{C}_2} \in \{0,1\}^{\ell\cdot(s+1-r)\times n}$ is the restriction of $\tilde{\mathbf{B}}_{\mathfrak{C}_2}$ constructed by Algorithm 11 to the rows with nonzero entries.

The final step is to appropriately merge the two matrices together, so that the congruence classes are in ascending order. Considering Algorithms 10 and 11, this corresponds to $\mathbf{B} = \tilde{\mathbf{B}}_{\mathfrak{C}_1} + \tilde{\mathbf{B}}_{\mathfrak{C}_2}$. We therefore get the following encoding matrix:

$$\mathbf{B} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & & & & & \\ \mathit{1} & \mathit{1} & \mathit{1} & \mathit{1} & & & & & & & \\ \mathtt{1} & \mathtt{1} & \mathtt{1} & \mathtt{1} & & & & & & & \\ 1 & 1 & 1 & 1 & 1 & 1 & & & & & \\ & & & & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ & & & & \mathit{1} & \mathit{1} & \mathit{1} & \mathit{1} & & & \\ & & & & \mathtt{1} & \mathtt{1} & \mathtt{1} & \mathtt{1} & & & \\ & & & & & & 1 & 1 & 1 & 1 & 1 \\ & & & & & & & & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & & & & & & & & \mathit{1} & \mathit{1} & \mathit{1} \\ & & & & & & & & \mathtt{1} & \mathtt{1} & \mathtt{1} \end{bmatrix} \in \{0,1\}^{n\times n}.$$

As discussed in Section 2.4.5, one can apply permutations to the rows and columns of $\mathbf{B}$ and obtain a solution to (IP). Furthermore, one can apply a different permutation on the columns for each congruence class, and still obtain a valid GCS. An example of how the allocations can be modified for each congruence class is given below, where the superposition of any set of row vectors of the same color result in $\mathbf{1}_{1\times 11}$, and all colors appear exactly once in each column:

$$\bar{\mathbf{B}} = \begin{bmatrix} 1 & 1 & 1 & 1 & & & & & & & \\ & 1 & & 1 & & 1 & & 1 & & & \\ & & 1 & & 1 & & 1 & & 1 & & \\ 1 & 1 & & 1 & & & 1 & & 1 & & 1 \\ & & & & 1 & 1 & 1 & 1 & & & \\ & & 1 & & 1 & & 1 & & 1 & & \\ 1 & 1 & & & & & & & & 1 & 1 \\ & & 1 & & 1 & 1 & & 1 & & 1 & \\ & & & & & & & & 1 & 1 & 1 \\ 1 & & & & & & & & & 1 & 1 \\ & & & 1 & & 1 & & 1 & & & \end{bmatrix}.$$

1.3.1 Example of Algorithm 2

We now demonstrate the decoding procedure of Algorithm 2. Considering our earlier example, we have the corresponding encoding matrices $\mathbf{B}_{\mathfrak{C}_1}$ and $\mathbf{B}_{\mathfrak{C}_2}$:

$$\mathbf{B}_{\mathfrak{C}_1} = \begin{bmatrix} 1 & 1 & 1 & 1 & & & & & & & \\ 1 & 1 & 1 & 1 & & & & & & & \\ 1 & 1 & 1 & 1 & & & & & & & \\ & & & & 1 & 1 & 1 & 1 & & & \\ & & & & 1 & 1 & 1 & 1 & & & \\ & & & & 1 & 1 & 1 & 1 & & & \\ & & & & & & & & 1 & 1 & 1 \\ & & & & & & & & 1 & 1 & 1 \\ & & & & & & & & 1 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{B}_{\mathfrak{C}_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & & & & & \\ & & & & & & \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} \end{bmatrix}.$$

Here, each set of computations is represented by a different color and font, and we require that at least one worker of each of color from one of the two encoding matrices has responded.

In Algorithm 2 we add only the first received computation of each represented color and font to C. In contrast to Algorithm 1 where we required all workers of a single color to respond in order to perform the decoding step, now we require at least one worker from each block, from either $\mathbf{B}_{\mathfrak{C}_1}$ or $\mathbf{B}_{\mathfrak{C}_2}$ to respond. We do this separately

for the subroutines corresponding to $\mathbf{B}_{\mathfrak{C}_1}$ and $\mathbf{B}_{\mathfrak{C}_2}$, and terminate whenever at least one worker from each block, from one of the two groups $\mathfrak{C}_1$ or $\mathfrak{C}_2$ has responded. This is equivalent to decoding a repetition erasure code. Lastly, recall that the number of workers per block in $\mathbf{B}_{\mathfrak{C}_2}$ is $s+1-r$, which in our toy example happens to be one.

1.3.2 Example of CMM-2, with $k_1 = 1$

An example of CMM-2 with $k_1 = 1$ is provided to help visualize the encoding task assignments, as well as the decoding. Let $n = k_2 = 8$, $s = 1$ and M be arbitrary, with $\mathrm{T} = M/k_2$ and $\mathrm{S} = L/1$. Let $\mathbf{0}_{\mathrm{T}}$ denote the $\mathrm{T} \times \mathrm{T}$ zero matrix. For $I = 1$ and $[I]_{s+1} = \{1, 3, 5, 7\}$, the encoding-decoding pair is

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{I}_{\mathrm{T}} & & & \\ \mathbb{I}_{\mathbb{T}} & & & \\ & \mathbf{I}_{\mathrm{T}} & & \\ & \mathbb{I}_{\mathbb{T}} & & \\ & & \mathbf{I}_{\mathrm{T}} & \\ & & \mathbb{I}_{\mathbb{T}} & \\ & & & \mathbf{I}_{\mathrm{T}} \\ & & & \mathbb{I}_{\mathbb{T}} \end{bmatrix}, \quad \tilde{\mathbf{a}}_{[I]_{s+1}} = \begin{bmatrix} \mathbf{0}_{\mathrm{T}} & & & \\ \mathbb{I}_{\mathbb{T}} & & & \\ & \mathbf{0}_{\mathrm{T}} & & \\ & \mathbb{I}_{\mathbb{T}} & & \\ & & \mathbf{0}_{\mathrm{T}} & \\ & & \mathbb{I}_{\mathbb{T}} & \\ & & & \mathbf{0}_{\mathrm{T}} \\ & & & \mathbb{I}_{\mathbb{T}} \end{bmatrix}$$

where both matrices are of the same size. From this example, it is also clear that $\tilde{\mathbf{a}}_{[I]}$ is in fact the restriction of $\tilde{\mathbf{B}}$ to the workers corresponding to the congruence class I.

1.4 Proofs of Section 2.4

In this appendix we present proofs of Theorems 2.4.2 and 2.4.3.

Proof. [Theorem 2.4.2] To simplify the presentation of the proof, we restrict it to the case where $n = k$. As was previously mentioned, to meet this assumption we can incorporate instances of the data point $(\mathbf{0}_{p\times 1}, 0)$ until N' points total are considered; such that $n|N'$, and let $k = n$.

Firstly, in the simplest case where $(s+1)|n$; we have $\|\mathbf{B}_{(i)}\|_0 = \frac{k}{n}(s+1) = \frac{k}{\ell} = s+1$ for all $i \in \mathbb{N}_{0,n}$. This results in a block diagonal matrix $\mathbf{B} = \mathbf{1}_{(s+1)\times(s+1)} \otimes \mathbf{I}_\ell \in \{0,1\}^{n\times k}$, which is a permutation of the FRC scheme presented in [279], for which $d_s(\mathbf{B}) = 0$; and $|\operatorname{supp}(\mathbf{B})| = \ell \cdot (s+1)^2 = k \cdot (s+1)$. The minimum value of the function d_s is therefore attained, while meeting all inequality constraints with an

upper bound of 0. By Proposition 2.3.1 and the fact that $\mathbf{B}$ is block diagonal with $\mathbf{B}_{ij} = 1$ if $\mathbf{B}_{ij} \neq 0$, the fourth constraint is also met.

For the remainder of the proof, we will consider the case where $(s+1) \nmid n$. We first give an equivalent optimization problem (IP-B) to (IP) for the case where we are only considering *binary* GC schemes, and show that our construction through Algorithms 10 and 11 meets the constraints of (IP-B). We then argue by contradiction, to show that our construction is a solution to (IP-B).

By imposing the constraints that $\mathbf{B} \in \{0,1\}^{n\times k}$ and $\mathbf{a}_{\mathcal{I}} \in \{0,1\}^n$ for each $\mathcal{I} \in \mathcal{I}_f^n$, the fourth constraint of (IP) can be replaced by (2.7) for all $i \in \mathbb{N}_{0,s}$, and by the fact that

$$\begin{aligned}
\operatorname{nnzr}(\mathbf{B}) &= \sum_{\iota=1}^{s+1} \sum_{j\in\mathcal{K}_\iota} \|\mathbf{B}_{(j)}\|_0 && (1.3)\\
&= \sum_{\iota=1}^{s+1} \|\mathbf{1}_{1\times k}\|_0 && (1.4)\\
&= \sum_{\iota=1}^{s+1} k && (1.5)\\
&= k\cdot(s+1) && (1.6)
\end{aligned}$$

we can drop the first constraint of (IP). We therefore have the following binary equivalent integer program (IP-B) of (IP):

$$\begin{aligned}
\text{(IP-B)} \qquad \arg\min_{\mathbf{B}\in\{0,1\}^{n\times k}} \quad & \big\{d_s(\mathbf{B})\big\}\\
\text{s.t.} \quad & \bigsqcup_{i=0}^{s} \mathcal{K}_i = \mathbb{N}_{0,n-1} \;:\; \big||\mathcal{K}_j| - |\mathcal{K}_l|\big| \le 1,\ \forall j,l \in \mathbb{N}_{0,s}\\
& \big|\|\mathbf{B}_{(j)}\|_0 - \|\mathbf{B}_{(l)}\|_0\big| \le 1,\ \forall j,l \in \mathcal{K}_i,\ \forall i \in \mathbb{N}_{0,s}\\
& \sum_{j\in\mathcal{K}_i} \mathbf{B}_{(j)} = \mathbf{1}_{1\times k},\ \forall i \in \mathbb{N}_{0,s}
\end{aligned}.$$

For our construction through Algorithms 10 and 11, the partitioning is in terms of congruence classes is $\{[i]_{s+1}\}_{i=0}^{s} = \mathfrak{C}_1 \bigsqcup \mathfrak{C}_2 = \mathbb{N}_{0,n-1}$, for which $\big||[j]_{s+1}| - |[l]_{s+1}|\big| = 1$ if $[j]_{s+1}$ and $[l]_{s+1}$ are in distinct $\mathfrak{C}_\iota$'s, and $\big||[j]_{s+1}| - |[l]_{s+1}|\big| = 0$ if they are in the same. Hence, the first constraint of (IP-B) is met.

Considering our partitioning of $\mathbb{N}_{0,n-1}$, the second constraint of (IP-B) can be split into the cases:

(a) $\big|\|\mathbf{B}_{(j)}\|_0 - \|\mathbf{B}_{(l)}\|_0\big| \le 1$, for all $j,l \in \mathfrak{C}_1$

(b) $\big|\|\mathbf{B}_{(j)}\|_0 - \|\mathbf{B}_{(l)}\|_0\big| \leq 1$, for all $j, l \in \mathfrak{C}_2$

where (a) and (b) correspond to Algorithms 10 and 11 respectively. By construction, it is clear that both (a) and (b) are met. The final constraint of (IP-B) for the partitioning through congruence classes, can be reformulated as in (2.15), which is also met by construction; for each $c \in \mathbb{N}_{0,s}$.

Now, for a contradiction, assume that there is a $\mathbf{B}' \in \{0,1\}^{n\times k}$ that satisfies the three constraints of (IP-B), for which $d_s(\mathbf{B}') < d_s(\mathbf{B})$. Denote the respective summands by $d'_i \coloneqq \big|\|\mathbf{B}'_{(i)}\|_0 - (s+1)\big|$ and $d_i \coloneqq \big|\|\mathbf{B}_{(i)}\|_0 - (s+1)\big|$; for each $i \in \mathbb{N}_{0,n-1}$. That is, $d(\mathbf{B}') = \sum_{j=0}^{n-1} d'_j$ and $d(\mathbf{B}) = \sum_{j=0}^{n-1} d_j$. Since $d(\mathbf{B}') < d_s(\mathbf{B})$, it follows that there is an $i \in \mathbb{N}_{0,n-1}$ for which $d'_i < d_i$; *i.e.* $d_i - d'_i = \delta$ for a positive integer δ.

By the first constraint of (IP-B) and (2.8), it follows that the rows of $\mathbf{B}'$ are partitioned into r groups $\{\mathcal{K}_\iota\}_{\iota=0}^{r-1}$ of size $\ell+1$; and $(s+1-r)$ groups $\{\mathcal{K}_\iota\}_{\iota=r}^{s}$ of size ℓ. Without loss of generality, we assume that

- $\mathcal{K}_\iota = \{\iota + z\cdot(s+1) : z \in \mathbb{N}_{0,\ell}\}$, for each $\iota \in \mathbb{N}_{0,r-1}$
- $\mathcal{K}_\iota = \{\iota + z\cdot(s+1) : z \in \mathbb{N}_{0,\ell-1}\}$, for each $\iota \in \{r, r+1, \ldots, s\}$

i.e. $\{\mathcal{K}_\iota\}_{\iota=0}^{s} = \mathfrak{C}_1 \bigsqcup \mathfrak{C}_2$; where $\mathfrak{C}_1 \equiv \bigsqcup_{\iota=0}^{r-1} \mathcal{K}_\iota$ and $\mathfrak{C}_2 \equiv \bigsqcup_{\iota=r}^{s} \mathcal{K}_\iota$ — this assumption can met by a simple permutation on the rows of $\mathbf{B}'$; which does not affect $d_s(\mathbf{B}')$ nor any of the constraints on (IP-B).

We first consider the case where $d'_i < d_i$; for some $i \in \mathfrak{C}_1$. Recall that Algorithm 10 can be reduced to only include the **else if** statement, as when $\ell > s - r$ we have $\lambda = s$ and $\tilde{r} = \ell + r - s > 0$. Thus, the two **if** loops are equivalent for $\ell > s - r$. It therefore suffices to only consider the **else if** statement of the algorithm. We know that $d_i = \big|\|\mathbf{B}_{(i)}\|_0 - (s+1)\big| \in \big\{|\lambda-(s+1)|, |\lambda+1-(s+1)|\big\}$. When $\lambda = \|\mathbf{B}_{(i)}\|_0 \leq s$; it follows that $\|\mathbf{B}'_{(i)}\|_0 = \|\mathbf{B}_{(i)}\| - \delta$, and in order to meet (2.7); there is at least one $j \in [i]_{s+1}$ for which $\|\mathbf{B}_{(j)}\|_0 = \lambda + 1$ and $\|\mathbf{B}'_{(j)}\|_0 \geq \|\mathbf{B}_{(j)}\|_0 + 1$. Therefore

$$\big|\|\mathbf{B}'_{(j)}\|_0 - \|\mathbf{B}'_{(i)}\|_0\big| = \|\mathbf{B}'_{(j)}\|_0 - \|\mathbf{B}'_{(i)}\|_0 \geq \lambda + 2 - \big(\|\mathbf{B}_{(i)}\|_0 - \delta\big) = \delta + 2 > 1 \quad (1.7)$$

which violates the second constraint of (IP-B). By a symmetric argument, one shows a similar contradiction for when $\|\mathbf{B}_{(i)}\|_0 = \lambda + 1 \geq s + 1$.

Next, we consider the case where $d'_i < d_i$ for $i \in \mathfrak{C}_2$. When $q = 0$, we have $\|\mathbf{B}_{(j)}\|_0 = s+t+1$ and $d_j = |s+t+1-(s+1)| = t$ for all $j \in [i]_{s+1}$, thus $d_s(\mathbf{B}) = \ell\cdot t$. Note that we cannot have $d'_i = d_i - \delta$ for $\delta \geq t+1$, as this would imply that

$$\big|\|\mathbf{B}'_{(i)}\| - (s+1)\big| = \big|\|\mathbf{B}_{(i)}\|_0 - (s+1)\big| - \delta = \big|s+t+1-(s+1)\big| - \delta \leq t - (t+1) = -1 \quad (1.8)$$

a contradiction. We therefore restrict this difference to $\delta \in \{1, 2, \ldots, t\}$, for which it follows that $\|\mathbf{B}'_{(i)}\|_0 = \|\mathbf{B}_{(i)}\| - \delta$. In order to meet (2.7), there is at least one $j \in [i]_{s+1}$ for which $\|\mathbf{B}'_{(j)}\|_0 \geq \|\mathbf{B}_{(j)}\|_0 + 1$, thus

$$\big|\|\mathbf{B}'_{(j)}\|_0 - \|\mathbf{B}'_{(i)}\|_0\big| = \|\mathbf{B}'_{(j)}\|_0 - \|\mathbf{B}'_{(i)}\|_0 \geq \|\mathbf{B}_{(j)}\|_0 + 1 - \big(\|\mathbf{B}_{(i)}\|_0 - \delta\big) = \delta + 1 > 1 \quad (1.9)$$

which violates the second constraint of (IP-B).

Lastly, we consider the case where $i \in \mathfrak{C}_2$, and $q > 0$. In the case where $\|\mathbf{B}_{(i)}\|_0 = s + t + 1$, the argument is the same as above. In the case where $\|\mathbf{B}_{(i)}\|_0 = s + t + 2$ and $\delta = 1$, by the third constraint of (IP-B) it follows that there is at least one $j \in [i]_{s+1}$ for which $\|\mathbf{B}'_{(j)}\|_0 > \|\mathbf{B}_{(j)}\| + 1$, implying that $d(\mathbf{B}') \geq d(\mathbf{B})$, contradicting the assumption that $d(\mathbf{B}') < d(\mathbf{B})$. Hence, for the case where $\|\mathbf{B}_{(i)}\|_0 = s + t + 2$, the only differences δ we need to consider are $\delta \in \{2, 3, \ldots, t + 1\}$. This though, reduces to the same argument as above, which led to the contradiction in (1.9).

We therefore conclude that any $\mathbf{B}'\{0, 1\}^{n \times k}$ for which $d_s(\mathbf{B}') < d_s(\mathbf{B})$, violates at least one constraint of (IP-B). Therefore, our construction of $\mathbf{B}$ through Algorithms 10 and 11 is a solution to (IP-B), the binary version of (IP). □

Proof. [Theorem 2.4.3] Assume condition 1) holds. By our construction of $\mathbf{a}_{\mathcal{I}}$, we consider each congruence class separately. The superposition of the rows corresponding to a complete residue system $[i]_{s+1}$, is equal to the sum of these rows over $\mathbb{R}$. We denote this superposition for the i^{th} congruence class by $\bar{\mathbf{b}}_{[i]}$, *i.e.*

$$\bar{\mathbf{b}}_{[i]} \coloneqq \left(\sum_{\iota \in [i]_{s+1}} \bar{\mathbf{B}}_{(\iota)} \right), \quad (1.10)$$

for which $\mathbf{a}_{\mathcal{I}}^T \bar{\mathbf{B}} = \bar{\mathbf{b}}_{[i]}$ if $\{\iota : \iota \in [i]_{s+1}\} \subsetneq \mathcal{I}$. Since the vectors are binary, the superposition results in $\mathbf{1}_{1 \times k}$ only when 1 appears in each position in a single row of this congruence class. This is precisely condition 2); for $\bar{\mathbf{B}}$ satisfying 1).

Now, assume condition 2) holds. For binary rows $\{\bar{\mathbf{B}}_{(j)}\}_{j=1}^n$, condition 1) ensures that the cardinality of each of these vectors is equal to that of the corresponding row $\mathbf{B}_{(j)}$, *i.e.* the same number of partitions are allocated to the j^{th} worker through both $\mathbf{B}$ and $\bar{\mathbf{B}}$. Therefore, for $\bar{\mathbf{B}}$ satisfying 1), we get

$$\bar{\mathbf{b}}_{[i]} = \left(\sum_{\iota \in [i]_{s+1}} \mathbf{B}_{(\iota)} \right) = k \ . \quad (1.11)$$

for all $i \in \{0, \cdots, s\}$. Under the assumption that 2) is satisfied, we have $(\bar{\mathbf{b}}_{[i]})_l \in \{0,1\}$ for all $l \in \mathbb{N}_k$, thus $\bar{\mathbf{b}}_{[i]} = \mathbf{1}_{1\times k}$. Specifically, we saw that for $\bar{\mathbf{B}} \in \{0,1\}^{n\times k}$ satisfying condition 2), we have $\bar{\mathbf{b}}_{[i]} \in \{0,1\}^{1\times k}$ for all $i \in \mathbb{N}_{0,s}$. When condition 1) is also satisfied, we then have $\bar{\mathbf{b}}_{[i]} = \mathbf{a}_{\mathcal{I}}^T \bar{\mathbf{B}} = \mathbf{1}_{1\times k}$ for i such that $\{\iota : \iota \in [i]_{s+1}\} \subsetneq \mathcal{I}$. We conclude that if 1) and 2) are simultaneously satisfied, the first statement holds.

Condition 2) guarantees that $\|\bar{\mathbf{B}}^{(i)}\|_0 \leq s+1$ for all i. Since we are applying a permutation on each set of rows corresponding to a complete residue system separately, we get that $\|\bar{\mathbf{B}}^{(i)}\|_0 \geq \|\mathbf{B}^{(i)}\|_0$, and by our construction of $\mathbf{B}$, we are guaranteed that $\|\mathbf{B}^{(i)}\|_0 = s+1$ for all i. By antisymmetry, it is clear that $\|\bar{\mathbf{B}}^{(i)}\|_0 = s+1$ for all i. This completes the proof. □

1.5 Numerical Experiment — Coded vs. Uncoded

In the following experiment, we justify the benefit of encoding computations in distributed platforms. We considered the fastest 250 AWS (Amazon Web Services) server completion times from [18] to model the delays of our experiment. Similar experiments have been considered in other works, *e.g.* [279, 134, 178], though these consider artificially delayed stragglers, with significantly smaller n and s.

We considered $A \in \mathbb{R}^{L\times N}$ and $B \in \mathbb{R}^{N\times M}$ for $L = M = N = 10^4$, which we partition across the respective dimension N to accommodate our scheme CMM-1 with $n = 250$ workers; that will tolerate s stragglers. Specifically, deployed CMM-1 with blocks of size $\tau = \frac{N}{k} = 20$ for $k = 500$, and in the coded setting, each worker was assigned $s+1$ different blocks, where s varied for different experiments. Once the computation times were calculated, we added the delay times from the AWS server completion times [18] mentioned above. We ordered the delay times in ascending order, and note that there was significant difference between workers responses 184 and 185 (3.325 seconds), and workers 238 and 239 (7.1463 seconds).

In Table A.1, we report the respective response time of the slowest worker which was needed in order to recover the matrix product (*i.e.* waiting time for the $(n-s+1)$ fastest worker; which corresponds to the recovery threshold of CMM-1), and the slowest worker of the distributed computation when no coding was applied. It is worth mentioning that our approach could have a greater speed up when the decoding step of Algorithm 2 were to be used instead. In this experiment, we report the worst case scenario of our approach. Since these trials were carried out on the same personal laptop, we expect that the times reported in the "Uncoded" row should be the same. They differ slightly, as the matrix product corresponding to each column, was for

different random matrices A and B. Our approach was beneficial in the case where CMM-1 was deployed with $s \in \{6, 8, 10, 12, 14, 16\}$, which is a consequence of the delay times of the slower servers.

Emulated AWS Recovery Times for CMM-1								
s	2	4	6	8	10	12	14	16
CMM-1	21.0316	21.1006	**20.8975**	**20.7138**	**20.7548**	**20.1759**	**12.9986**	**12.9136**
Uncoded	**21.0137**	**21.0190**	21.0189	21.0288	21.0201	21.0213	21.0181	21.0290

Table A.1: Emulated AWS response times, for CMM-1 and uncoded distributed matrix multiplication. We report the waiting times of the slowest responsive worker we need in order to perform CMM-1; *i.e.* the time of the fastest $n - s + 1$ worker, and the slowest of the 250^{th} workers in the uncoded scenario. In bold, we indicate which of the two respective times was faster. The times reported are in seconds.

1.6 Application of CMM to Distributed Gradient Descent for Frobenius norm Minimization

In this appendix we first review gradient descent, and then focus on gradient descent for Frobenius norm minimization problems, as defined in (1.15); for the objective function L_F defined in (1.13). We briefly describe the following motivating problems which try to solve (1.15) or similar optimization problems: *nonnegative matrix factorization* (NMF), k-SVD, low rank matrix approximation, sparse coding and the best k-rank approximation; which relates to principal component analysis. This is not an exhaustive list of where the objective function L_F has been utilized, since many more applications do exist.

Recall that in gradient descent, we consider a minimization problem with a convex differentiable objective function $L \colon \mathcal{C} \to \mathbb{R}$ over an open constrained set $\mathcal{C} \subseteq \mathbb{R}^p$. Then, given an initial $\theta^{[0]} \in \mathcal{C}$; the following update is performed at iteration $t + 1$:

$$\theta^{[t+1]} \leftarrow \theta^{[t]} - \xi_t \cdot \nabla_\theta L(\mathcal{D}; \theta^{[t]}), \qquad \text{for } t = 0, 1, 2, \ldots \tag{1.12}$$

until a specified termination criterion is met. The parameter ξ_t is the step-size, which may be adaptive or fixed. Note that GC is only concerned with computing the gradient at each step and hence, selecting an appropriate step-size was not discussed in this chapter.

In the literature regarding gradient descent for coded computing thus far, only the case where the gradient of the objective function (2.1) is a vector has been considered or discussed. In order to tie together Sections 2.2 and 2.3 with Section 2.5, we discuss the case where the gradient is a matrix, *e.g.*

$$L_F(\mathbf{X}, \mathbf{Y}; \Theta) \coloneqq \|\mathbf{X}\Theta - \mathbf{Y}\|_F^2 = \sum_{i=1}^{m} \overbrace{\|\mathbf{X}\Theta^{(i)} - \mathbf{Y}^{(i)}\|_2^2}^{L_{ols}\left(\mathbf{X}, \mathbf{Y}^{(i)}; \Theta^{(i)}\right)}, \tag{1.13}$$

for $\mathbf{X} \in \mathbb{R}^{N \times p}$, $\Theta \in \mathbb{R}^{p \times m}$ and $\mathbf{Y} \in \mathbb{R}^{N \times m}$. The gradient is

$$\nabla_\Theta L_F(\mathbf{X}, \mathbf{Y}; \Theta) = 2\mathbf{X}^T(\mathbf{X}\Theta - \mathbf{Y}), \tag{1.14}$$

which is computed in order to approximate the solution to

$$\Theta^\star = \underset{\Theta \in \mathbb{R}^{p \times m}}{\arg\min} \left\{ L_F(\mathbf{X}, \mathbf{Y}; \Theta) \right\} \tag{1.15}$$

via gradient descent. Similar to the ordinary least squares objective function L_{ols}, (1.15) has the closed-form solution:

$$\Theta^\star = \mathbf{X}^\dagger \mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \tag{1.16}$$

which is intractable for large N. In practice, it is often preferred to approximate $\Theta^\star$.

A motivating application is if we have measurements $\{\mathbf{Y}^{(i)}\}_{i=1}^m$ from m different sensors in different locations or from different sources, for the same corresponding $\mathbf{X}$, and we want to interpolate the corresponding optimal models $\{\theta_i\}_{i=1}^m$ for each sensor or source.

More generally, the goal of most nonlinear regression problems is to solve the problem

$$\min_{\kappa \in \mathcal{H}} \left\{ \sum_{i=1}^{N} \left(\kappa(\mathbf{x}_i) - y_i \right)^2 \right\}, \tag{1.17}$$

where κ comes from a hypothesis class $\mathcal{H}$ that fits the training data, for which one can use the kernel trick to solve efficiently. What we present can be applied also to regression problems of this type, as well as kernel regression problems [257].

Throughout the gradient descent process, the second summand $2\mathbf{X}^T\mathbf{Y}$ is constant. Hence, at every iteration we only need to compute the matrix product $\hat{\mathbf{X}}\Theta$, where $\hat{\mathbf{X}} = 2\mathbf{X}^T\mathbf{X}$ is also a constant matrix. Then, depending on which matrix multiplication scheme we decide to use, the workers will receive the entire matrix $\hat{\mathbf{X}}$ or a submatrix

of it at the beginning of the distributed computation process, and a submatrix of Θ's update

$$\Theta^{[t+1]} \leftarrow \Theta^{[t]} - \xi_t \cdot \nabla_\Theta L_F(\mathbf{X}, \mathbf{Y}; \Theta^{[t]}) \tag{1.18}$$

at each iteration. In an iterative process, it is preferred to reduce the total communication cost as much as possible. Hence, we prefer to communicate only part of Θ when possible.

Solving for the loss function L_Θ may also be viewed as solving multiple linear regression problems simultaneously. This is due to its decomposition into a summation of m separate least squares objective functions, with the same data matrix $\mathbf{X}$. For $\Theta^{(i)} = \theta_i$, it follows that

$$\Theta^\star = \begin{pmatrix} | & | & & | \\ \theta_1^\star & \theta_2^\star & \dots & \theta_m^\star \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{p \times m}, \tag{1.19}$$

for $\theta_i^\star$ being the minimal solution to $L_{ols}(\mathbf{X}, \mathbf{Y}^{(i)}; \Theta^{(i)})$, for each $i \in \mathbb{N}_m$. To guarantee convergence for all θ_i, we can fix $\xi_t = 2/\sigma_{\max}(\mathbf{X})^2$ for all iterations.

We point out that the above problem could indeed be solved by using regular GC, as we have

$$\|\mathbf{A}\|_F^2 = \left\| \left[\left(\mathbf{A}^{(1)}\right)^T \ \cdots \ \left(\mathbf{A}^{(m)}\right)^T \right]^T \right\|_2^2, \tag{1.20}$$

for any real-valued matrix $\mathbf{A}$ comprised of m columns. We also note that the least squares regression problem in the presence of stragglers, was studied in [182].

1.6.1 Nonnegative Matrix Factorization

The NMF problem deals with decomposing a matrix $A \in \mathbb{R}_{\geq 0}^{L \times M}$ with nonnegative entries into two matrices $U \in \mathbb{R}_{\geq 0}^{L \times N}$ and $V \in \mathbb{R}_{\geq 0}^{N \times M}$, by attempting to solve

$$\min_{\substack{U \in \mathbb{R}_{\geq 0}^{L \times N} \\ V \in \mathbb{R}_{\geq 0}^{N \times M}}} \left\{ \|A - UV\|_F^2 \right\} \tag{1.21}$$

for U, V with the appropriate dimensions. In [177] a multiplicative update algorithm is proposed:

$$V \leftarrow V \cdot \frac{U^T A}{U^T U V} \qquad \text{and} \qquad U \leftarrow U \cdot \frac{A V^T}{U V V^T}, \tag{1.22}$$

where the division is done element-wise. Multiple multiplications are required for these updates, and the matrices can be quite large, *e.g.* when dealing with recom-

mender systems. Multiple distributive multiplications are required at each iteration, which makes this process a lot more cumbersome. Therefore, speeding up this process is even more crucial. Further details on this algorithm and how to incorporate gradient methods to solve NMF can be found in [105, 189, 190].

1.6.2 Low-Rank Approximation

Consider the problem of finding a low-rank approximation of a matrix $A \in \mathbb{R}^{L\times M}$. That is, for an approximation of rank k or less we want to find $B = U_B V_B$ for $U_B \in \mathbb{R}^{L\times k}$ and $V_B \in \mathbb{R}^{k\times M}$, which can be done by solving the problem

$$\min_{\substack{B\in\mathbb{R}^{L\times M}\\ \operatorname{rank}(B)\leq k}} \left\{\|A-B\|_F^2\right\} = \min_{\substack{U_B\in\mathbb{R}^{L\times k}\\ V_B\in\mathbb{R}^{k\times M}}} \left\{\|A-U_BV_B\|_F^2\right\}, \tag{1.23}$$

where it is easier to work with $B = U_B V_B$ in the case where both L and M are large, *e.g.* in terms of storage and computations.

The objective function of (1.23) is bi-convex. A common alternating minimization approach fixes one of the matrices and optimizes the other, and then alternates.

It is well known that the best k-rank approximation for many norms, including the Euclidean and Frobenius norms, can be computed through the truncated singular value decomposition (SVD). By the Eckart–Young theorem [99]

$$A_k = U\Sigma_k V^T = \sum_{i=1}^{k} \sigma_i U^{(i)}(V^{(i)})^T \tag{1.24}$$

solves (1.23). The SVD takes $O(LM\cdot\min\{L,M\})$ time itself to compute, which is cumbersome. To avoid computing the SVD, we can resort to approximating A_k by solving

$$\min_{\substack{U\in\mathbb{R}^{L\times k}\\ U^TU=\mathbf{I}_k}} \left\{\|A-UU^TA\|_F^2\right\} = \min_{\substack{U\in\mathbb{R}^{L\times k}\\ U^TU=\mathbf{I}_k}} \left\{-\operatorname{tr}(U^TAA^TU)\right\} \tag{1.25}$$

through gradient descent, where the gradient with respect to U is $-AA^TU$. Hence, at each iteration we distributively compute $AA^TU^{[t]}$ [156, 257]. For $\hat{U}$ our final solution to the above minimization problem, our k-rank approximation of A will be $\hat{U}\hat{U}^TA$, which is an approximation of A_k. For $U^\star$ being the exact solution, we have $A_k = U^\star(U^\star)^TA$.

Other problems in which CMM could be utilized in distributed gradient methods are the weighted low-rank matrix approximation [17] and the k-SVD algorithm [2]. These involve similar objective functions, whose gradients have a matrix form

which require at least one matrix-matrix multiplication. Thus, the process would be accelerated if these were to be computed distributively.

Appendix B

Appendix to Chapter III

2.1 Proofs of Subsection 3.3.2

In this appendix, we provide the proof of Theorem 3.3.1 and Corollary 3.3.1. We first recall the following matrix Chernoff bound [294, Fact 1], and prove Proposition 2.1.1.

Theorem 2.1.1 (Matrix Chernoff Bound, [294, Fact 1]). *Let $\mathbf{X}_1, \cdots, \mathbf{X}_q$ be independent copies of a symmetric random matrix $\mathbf{X} \in \mathbb{R}^{d\times d}$, with $\mathbb{E}[\mathbf{X}] = 0, \|\mathbf{X}\|_2 \leqslant \gamma$, $\|\mathbb{E}[\mathbf{X}^\top \mathbf{X}]\|_2 \leqslant \sigma^2$. Let $\mathbf{Z} = \frac{1}{q}\sum_{i=1}^q \mathbf{X}_i$. Then $\forall \epsilon > 0$:*

$$\Pr\left[\|\mathbf{Z}\|_2 > \epsilon\right] \leqslant 2d \cdot \exp\left(-\frac{q\epsilon^2}{\sigma^2 + \gamma\epsilon/3}\right).$$

Proposition 2.1.1. *The sketching matrix $\widetilde{\mathbf{S}}$ of Algorithm 5 with $\tilde{\Pi}_i \geqslant \beta \Pi_i$ for all i and $\beta \in (0, 1]$, guarantees*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^\top \widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}} \mathbf{U}\|_2 > \epsilon\right] \leqslant 2d \cdot e^{-q\epsilon^2 \Theta(\beta/d)} \tag{2.1}$$

for any $\epsilon > 0$, and $q = r/\tau > d/\tau$.

Proof. [Proposition 2.1.1] In order to use Theorem 2.1.1, we first need to define an appropriate symmetric random matrix $\mathbf{X}$, whose realizations $\mathbf{X}_{\{q\}}$ correspond to the sampling procedure of Algorithm 5, and $\widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}} = \frac{1}{q}\sum_{i=1}^q \mathbf{X}_i$. The realization of the matrix random variable are

$$\mathbf{X}_i = \mathbf{I}_d - \left(\frac{\mathbf{U}_{(\mathcal{K}_\iota^i)}^\top \mathbf{U}_{(\mathcal{K}_\iota^i)}}{\tilde{\Pi}_\iota}\right) = \mathbf{I}_d - \left(\frac{\sum_{l\in\mathcal{K}_\iota^i} \mathbf{U}_{(l)}^\top \mathbf{U}_{(l)}}{\tilde{\Pi}_\iota}\right)$$

where $\mathcal{K}_\iota^i$ indicates the ι^{th} block of $\mathbf{A} \in \mathbb{R}^{N\times d}$, which was sampled at trial i. This holds for all $\iota \in \mathbb{N}_K$. The expectation of the symmetric random matrix $\mathbf{X}$ is

$$\begin{aligned}
\mathbb{E}[\mathbf{X}] &\overset{\ddagger}{=} \mathbf{I}_d - \left(\sum_{j=1}^{K} \tilde{\Pi}_j \cdot \frac{\mathbf{U}_{(\mathcal{K}_j)}^\top \mathbf{U}_{(\mathcal{K}_j)}}{\tilde{\Pi}_j}\right) \\
&= \mathbf{I}_d - \sum_{j=1}^{K} \mathbf{U}_{(\mathcal{K}_j)}^\top \mathbf{U}_{(\mathcal{K}_j)} \\
&\overset{\sharp}{=} \mathbf{I}_d - \sum_{l=1}^{N} \mathbf{U}_{(l)}^\top \mathbf{U}_{(l)} \\
&= \mathbf{I}_d - \mathbf{I}_d \\
&= \mathbf{0}_{d\times d}
\end{aligned}$$

where $\ddagger$ follows from the fact that each realization $\mathbf{X}_i$ corresponding to each $\{\mathcal{K}_j^i\}_{j=1}^K$ of the random matrix is sampled with probability $\tilde{\Pi}_j$, and in $\sharp$ we simplify the expression in terms of rank-1 outer-product matrices. Furthermore, for $\{\tilde{\ell}_l\}_{l=1}^N$ the corresponding *approximate* leverage scores of $\mathbf{A}$

$$\|\mathbf{X}_i\|_2 \overset{\natural}{\leqslant} \|\mathbf{I}_d\|_2 + \frac{\|\mathbf{U}_{(\mathcal{K}_\iota^i)}^\top \mathbf{U}_{(\mathcal{K}_\iota^i)}\|_2}{\tilde{\Pi}_\iota} \overset{\diamond}{\leqslant} 1 + \frac{\sum_{l\in\mathcal{K}_\iota^i} \ell_l}{\left(\sum_{l\in\mathcal{K}_\iota^i} \tilde{\ell}_l\right)/d} = 1 + \frac{d\cdot\Pi_\iota}{\tilde{\Pi}_\iota} = 1 + \frac{d}{\beta}$$

where in $\natural$ we use the triangle inequality on the definition of $\mathbf{X}_i$, and in $\diamond$ we use it on the sum of outer-products (the numerator of second summand).

We now upper bound the variance of the copies of $\mathbf{X}$:

$$
\begin{aligned}
\left\|\mathbb{E}\left[\mathbf{X}_i^\top \mathbf{X}_i\right]\right\|_2 &= \left\|\mathbb{E}\left[\left(\mathbf{I}_d - \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}/\tilde{\Pi}_\iota\right)\right)^\top \left(\mathbf{I}_d - \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}/\tilde{\Pi}_i\right)\right)\right]\right\|_2 \\
&= \left\|\mathbf{I}_d - 2\cdot\mathbb{E}\left[\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}/\tilde{\Pi}_\iota\right]\right. + \left.\mathbb{E}\left[\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2/\tilde{\Pi}_\iota^2\right]\right\|_2 \\
&= \left\|2\Big(\mathbf{I}_d - \overbrace{\mathbb{E}\left[\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}/\tilde{\Pi}_\iota\right]}^{=\mathbf{I}_d}\Big) - \mathbf{I}_d + \mathbb{E}\left[\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2/\tilde{\Pi}_\iota^2\right]\right\|_2 \\
&= \left\|\mathbb{E}\left[\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2/\tilde{\Pi}_\iota^2\right] - \mathbf{I}_d\right\|_2 \\
&= \left\|\left(\sum_{\iota=1}^K \Pi_\iota \cdot \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2/\tilde{\Pi}_\iota^2\right) - \mathbf{I}_d\right\|_2 \\
&\overset{\sharp}{\leqslant} \left\|\left(\sum_{\iota=1}^K \frac{1}{\beta}\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2/\tilde{\Pi}_\iota\right) - \mathbf{I}_d\right\|_2 \\
&= \left\|\left(\sum_{\iota=1}^K \frac{d}{\beta}\cdot\frac{\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2}{\|\mathbf{U}_{(\mathcal{K}_\iota)}\|_F^2}\right) - \mathbf{I}_d\right\|_2 \\
&\leqslant \left\|\left(\sum_{\iota=1}^K \frac{d}{\beta}\cdot\frac{\left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2}{\|\mathbf{U}_{(\mathcal{K}_\iota)}\|_2^2}\right) - \mathbf{I}_d\right\|_2 \\
&\overset{\flat}{=} \left\|\frac{d}{\beta}\cdot\left(\sum_{\iota=1}^K \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)^2\right) - \mathbf{I}_d\right\|_2 \\
&\overset{\natural}{\leqslant} \left\|\frac{d}{\beta}\left(\sum_{\iota=1}^K \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)\left(\mathbf{I}_d - \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}^\top \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}\right)\right) - \mathbf{I}_d\right\|_2 \\
&= \left\|\frac{d}{\beta}\overbrace{\left(\sum_{\iota=1}^K \mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)}^{=\mathbf{I}_d} - \frac{d}{\beta}\left(\sum_{\iota=1}^K \left(\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)}\right)\left(\mathbf{U}_{(\bar{\mathcal{K}}_\iota)}^\top \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}\right)\right) - \mathbf{I}_d\right\|_2 \\
&= \left\|(d/\beta - 1)\cdot\mathbf{I}_d - \frac{d}{\beta}\Big(\sum_{\iota=1}^K \mathbf{U}_{(\mathcal{K}_\iota)}^\top \overbrace{\left(\mathbf{U}_{(\mathcal{K}_\iota)}\mathbf{U}_{(\bar{\mathcal{K}}_\iota)}^\top\right)}^{\mathbf{0}_{d\times d}} \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}\Big)\right\|_2 \\
&= \|(d/\beta - 1)\cdot\mathbf{I}_d\|_2 \\
&= d/\beta - 1
\end{aligned}
$$

where in $\sharp$ we used the fact $\Pi_\iota/\tilde{\Pi}_\iota \leqslant 1/\beta$, in $\flat$ that $\|\mathbf{U}_{(\mathcal{K}_\iota)}\|_2^2 = 1$, and in $\natural$ that $\mathbf{U}_{(\mathcal{K}_\iota)}^\top \mathbf{U}_{(\mathcal{K}_\iota)} = \mathbf{I}_d - \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}^\top \mathbf{U}_{(\bar{\mathcal{K}}_\iota)}$ for each ι.

According to Theorem 2.1.1, we substitute $\gamma = 1 + d$ and $\sigma^2 = d/\beta - 1$ to get

$$\begin{aligned}\frac{1}{q}\sum_{i=1}^{q}\mathbf{X}_i &= \frac{1}{q}\sum_{i=1}^{q}\left(\mathbf{I}_d - \frac{\mathbf{U}_{(\mathcal{K}_{j(i)})}^{\top}\mathbf{U}_{(\mathcal{K}_{j(i)})}}{\tilde{\Pi}_{j(i)}}\right)\\ &= \mathbf{I}_d - \frac{1}{q}\left(\sum_{i=1}^{q}\frac{\mathbf{U}_{(\mathcal{K}_{j(i)})}^{\top}\mathbf{U}_{(\mathcal{K}_{j(i)})}}{\tilde{\Pi}_{j(i)}}\right)\\ &= \mathbf{I}_d - \mathbf{U}^{\top}\widetilde{\mathbf{S}}^{\top}\widetilde{\mathbf{S}}\mathbf{U}\end{aligned}$$

where the last equality follows from the definition of $\widetilde{\mathbf{S}}$. Putting everything together into Theorem 2.1.1, we get that

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^{\top}\widetilde{\mathbf{S}}^{\top}\widetilde{\mathbf{S}}\mathbf{U}\|_2 > \epsilon\right] \leqslant 2d \cdot e^{-q\epsilon^2\Theta(\beta/d)}.$$

□

Proof. [Theorem 3.3.1] By substituting $q = \Theta\left(\frac{d}{\tau}\log\left(2d/\delta\right)/(\beta\epsilon^2)\right)$ in (2.1) and taking the complementary event, we attain the desired statement. □

Before we prove Corollary 3.3.1, we introduce the notion of *block α-balanced*, which is a generalization of *α-balanced* from [231]. The sampling distribution $\Pi_{\{K\}}$ is said to be *block α-balanced*, if

$$\max_{i\in\mathbb{N}_K}\{\Pi_i\} \leqslant \frac{\alpha}{N/\tau} = \frac{\alpha}{K} \tag{2.2}$$

for some constant α independent of K and q. Furthermore, in our context, if the individual leverage scores $\pi_{\{N\}}$ are α-balanced for α independent of N and r, then the block leverage scores $\Pi_{\{K\}}$ are block α-balanced, as

$$\max_{i\in\mathbb{N}_K}\{\Pi_i\} \leqslant \tau \cdot \max_{j\in\mathbb{N}_N}\{\pi_j\} \leqslant \tau \cdot \frac{\alpha}{N} = \frac{\alpha}{N/\tau} = \frac{\alpha}{K}\,. \tag{2.3}$$

Proof. [Corollary 3.3.1] From the proof of [231, Theorem 1], we simply need to show that

$$\left\|\mathbb{E}\left[\widetilde{\mathbf{S}}^{\top}\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^{\top}\right)^{-1}\widetilde{\mathbf{S}}\right]\right\|_2 \leqslant \eta \cdot \frac{r}{N}$$

for $\widetilde{\mathbf{S}}$ a single sketch produced in Algorithm 5, and an appropriate constant η independent of N and r. We assume that the individual leverage scores $\pi_{\{N\}}$ are α-balanced, where α is a constant independent of N and r. By (2.3), it follows that the block leverage scores $\Pi_{\{K\}}$ are block α-balanced; *i.e.* $\Pi_i \leqslant \Pi_K \leqslant \frac{\alpha}{K}$ for all $i \in \mathbb{N}_{K-1}$.

A direct computation shows that

$$\begin{aligned}\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^{\top}\right)^{-1} &= \Big((\mathbf{D}\cdot\boldsymbol{\Omega}\otimes\mathbf{I}_{\tau})\cdot(\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\otimes\mathbf{I}_{\tau})\Big)^{-1}\\ &= \Big(\big(\mathbf{D}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\big)\otimes\mathbf{I}_{\tau}\Big)^{-1}\\ &= \big(\mathbf{D}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\big)^{-1}\otimes\mathbf{I}_{\tau}\end{aligned}$$

and consequently

$$\begin{aligned}\widetilde{\mathbf{S}}^{\top}\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^{\top}\right)^{-1}\widetilde{\mathbf{S}} &= \big(\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\otimes\mathbf{I}_{\tau}\big)\cdot\Big(\big(\mathbf{D}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\big)^{-1}\otimes\mathbf{I}_{\tau}\Big)\cdot(\mathbf{D}\cdot\boldsymbol{\Omega}\otimes\mathbf{I}_{\tau})\\ &= \Big(\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\cdot\big(\mathbf{D}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\big)^{-1}\otimes\mathbf{I}_{\tau}\Big)\cdot(\mathbf{D}\cdot\boldsymbol{\Omega}\otimes\mathbf{I}_{\tau})\\ &= \underbrace{\Big(\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\cdot\big(\mathbf{D}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}^{\top}\cdot\mathbf{D}^{\top}\big)^{-1}\cdot\mathbf{D}\cdot\boldsymbol{\Omega}\Big)}_{\coloneqq Z\in\mathbb{R}_{\geqslant 0}^{K\times K}}\otimes\mathbf{I}_{\tau}\end{aligned}$$

where $Z=\sum_{\iota=1}^{q}Z_{\iota}$, for $\{Z_{\iota}\}_{\iota=1}^{q}$ rank-1 outer-product matrices of size $K\times K$ corresponding to each sampling trial, through $\boldsymbol{\Omega}$. For each sampling trial, we know that the i^{th} block is sampled with probability Π_i. Furthermore, if the i^{th} block is sampled at iteration ι, it follows that

$$\begin{aligned}Z_{\iota} &= \mathbf{e}_i\cdot\sqrt{\frac{1}{q\Pi_i}}\cdot\left(\mathbf{e}_i^{\top}\cdot\left(\sqrt{\frac{1}{q\Pi_i}}\right)^{2}\cdot\mathbf{e}_i\right)^{-1}\cdot\sqrt{\frac{1}{q\Pi_i}}\cdot\mathbf{e}_i^{\top}\\ &= \mathbf{e}_i\cdot\sqrt{\frac{1}{q\Pi_i}}\cdot\left(\sqrt{q\Pi_i}\right)^{2}\cdot\sqrt{\frac{1}{q\Pi_i}}\cdot\mathbf{e}_i^{\top}\\ &= \mathbf{e}_i\cdot\mathbf{e}_i^{\top}\end{aligned}$$

hence

$$\mathbb{E}\left[\widetilde{\mathbf{S}}^{\top}\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^{\top}\right)^{-1}\widetilde{\mathbf{S}}\right]=\mathbb{E}_{\boldsymbol{\Omega}}\left[\sum_{j=1}^{K}\mathbf{e}_j\cdot\mathbf{e}_j^{\top}\right]\otimes\mathbf{I}_{\tau}=\left(\sum_{j=1}^{K}\mathbb{E}_{\boldsymbol{\Omega}}\left[\mathbf{e}_j\cdot\mathbf{e}_j^{\top}\right]\right)\otimes\mathbf{I}_{\tau}=\mathbf{Q}\otimes\mathbf{I}_{\tau}$$

where $\mathbf{Q}=\operatorname{diag}\big(\{h_j\}_{j=1}^{q}\big)$, for $h_i=1-(1-\Pi_i)^q$ the probability that the i^{th} block was sampled at least once. Since we assume that the blocks $\mathbf{A}_{\{K\}}$ are indexed in ascending order according their block leverage scores; *i.e.* $\Pi_i\leqslant\Pi_{i+1}$ for all $i\in\mathbb{N}_{K-1}$,

it follows that $h_i \leqslant h_K$ for all i; thus

$$\mathbb{E}\left[\widetilde{\mathbf{S}}^\top\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^\top\right)^{-1}\widetilde{\mathbf{S}}\right] = \operatorname{diag}\left(\{h_j\}_{j=1}^q\right)\otimes\mathbf{I}_\tau \preceq h_K\cdot\mathbf{I}_N = \left(1-(1-\Pi_K)^q\right)\cdot\mathbf{I}_N \preceq q\Pi_K\cdot\mathbf{I}_N\ .$$

Consequently, since $\Pi_{\{K\}}$ are block α-balanced; we have

$$\left\|\mathbb{E}\left[\widetilde{\mathbf{S}}^\top\left(\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^\top\right)^{-1}\widetilde{\mathbf{S}}\right]\right\|_2 \leqslant q\cdot\Pi_K \leqslant \alpha\cdot\frac{q}{K} = \alpha\cdot\frac{r}{N}.$$

This completes the proof, as we can take $\eta=\alpha$. □

2.2 Concrete Example of Induced Sketching

In this appendix, we give a simple demonstration of the induced sketching matrices, through our proposed GC approach. For simplicity, we will consider exact sampling, with $\Pi_{\{K\}} = \{3/20, 3/20, 4/20, 5/20, 5/20\}$ for $K=5$, an arbitrary large block size of τ, and $m=20$. The optimal replication numbers resulting from this distribution are $r^\star_{\{K\}} = \{3,3,4,5,5\}$, hence $m=R$. In order to obtain a reduced dimension r which is 60% of the original N, we carry out $q=3$ sampling trials at each iteration.

Let the resulting index multisets corresponding to the encoded pairs $(\tilde{A}_j, \tilde{b}_j)$ of the first four iterations; along with the resulting estimated gradients, be:

1. $\mathcal{S}^{[1]} = \{1,4,5\} \implies \hat{g}^{[1]} = \nabla_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}_{[1]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[1]}\right) = \hat{g}_1^{[1]} + \hat{g}_4^{[1]} + \hat{g}_5^{[1]}$
2. $\mathcal{S}^{[2]} = \{3,5,5\} \implies \hat{g}^{[2]} = \nabla_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}_{[2]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[2]}\right) = \hat{g}_3^{[2]} + \hat{g}_5^{[2]} + \hat{g}_5^{[2]}$
3. $\mathcal{S}^{[3]} = \{2,4,5\} \implies \hat{g}^{[3]} = \nabla_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}_{[3]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[3]}\right) = \hat{g}_2^{[3]} + \hat{g}_4^{[3]} + \hat{g}_5^{[3]}$
4. $\mathcal{S}^{[4]} = \{4,1,4\} \implies \hat{g}^{[4]} = \nabla_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}_{[4]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[4]}\right) = \hat{g}_4^{[4]} + \hat{g}_1^{[4]} + \hat{g}_4^{[4]}\ .$

Then, the corresponding *induced* block leverage score sketching matrices of Algorithm 5, are:

1. $\widetilde{\mathbf{S}}_{[1]} = \begin{pmatrix} 1/\sqrt{3\Pi_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sqrt{3\Pi_4} & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{3\Pi_5} \end{pmatrix}\otimes\mathbf{I}_\tau$
2. $\widetilde{\mathbf{S}}_{[2]} = \begin{pmatrix} 0 & 0 & 1/\sqrt{3\Pi_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{3\Pi_5} \\ 0 & 0 & 0 & 0 & 1/\sqrt{3\Pi_5} \end{pmatrix}\otimes\mathbf{I}_\tau$

3. $\widetilde{\mathbf{S}}_{[3]} = \begin{pmatrix} 0 & 1/\sqrt{3\Pi_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sqrt{3\Pi_4} & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{3\Pi_5} \end{pmatrix} \otimes \mathbf{I}_\tau$

4. $\widetilde{\mathbf{S}}_{[4]} = \begin{pmatrix} 0 & 0 & 0 & 1/\sqrt{3\Pi_4} & 0 \\ 1/\sqrt{3\Pi_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sqrt{3\Pi_4} & 0 \end{pmatrix} \otimes \mathbf{I}_\tau$

each of which are of size $(3\tau) \times (5\tau)$.

2.3 Comparison to the block-SRHT

An alternative view point is that the matrix $\tilde{\mathbf{U}}_{\text{exp}} \coloneqq \mathbf{\Psi} \cdot (\mathbf{\Sigma}\mathbf{V}^\top)^{-1}$ has uniform Frobenius block scores, which further justifies the proposed GC approach for homogeneous servers. This resembles the main idea behind the SRHT [5, 96] and different variants known as *block-SRHT* [49, 16], where a random projection is applied to the data matrix to "flatten" its leverage scores, *i.e.* make them close to uniform. These two techniques for flattening the corresponding block scores are vastly different, and the proximity of the corresponding block scores are measured and quantified differently. In contrast to the block-SRHT, the quality of the approximation in our case depends on the misestimation factor $\beta_{\tilde{\Pi}}$; and is not quantified probabilistically.

We note that since $\mathbf{\Psi}$ has repeated blocks from the expansion, the scores we consider in Lemma 2.3.1 are not the block leverage scores of $\mathbf{\Psi}$. The Frobenius block scores of $\tilde{\mathbf{U}}_{\text{exp}}$, are in fact the corresponding block leverage scores of $\tilde{\mathbf{A}}$, which are replicated in the expansion. Moreover, note that the closer $\beta_{\tilde{\Pi}}$ is to 1, the closer the sampling distribution according to the Frobenius block scores of $\tilde{\mathbf{U}}_{\text{exp}}$; which we denote by $\tilde{Q}_{\{R\}}$, is to being exactly uniform. We denote the uniform sampling distribution by $\mathcal{U}_{\{R\}}$.

Lemma 2.3.1. *When $\tilde{\Pi}_{\{K\}} = \Pi_{\{K\}}$, the sampling distribution $\tilde{Q}_{\{R\}}$ is uniform. When $\tilde{\Pi}_\iota \geqslant \beta_{\tilde{\Pi}}\Pi_\iota$ for $\beta_{\tilde{\Pi}} = \min_{i\in\mathbb{N}_K}\{\Pi_i/\tilde{\Pi}_i\} \in (0,1)$ and all $\iota \in \mathbb{N}_K$, the resulting distribution $\tilde{Q}_{\{R\}}$ is approximately uniform, and satisfies $d_{\mathcal{U},\tilde{Q}} \leqslant 1/(R\beta_{\tilde{\Pi}})$.*

Proof. Let $\mathbf{U} = \begin{bmatrix}\mathbf{U}_1^\top & \cdots & \mathbf{U}_K^\top\end{bmatrix}^\top$ denote the corresponding blocks of $\mathbf{U}$ according the partitioning of $\mathcal{D}$. Without loss of generality, assume that the data points within each partition are consecutive rows of $\mathbf{A}$, and $\mathbf{U}_\iota \in \mathbb{R}^{\tau\times d}$ for all $\iota \in \mathbb{N}_K$.

From (3.19) and (3.22), it follows that

$$\begin{aligned}\boldsymbol{\Psi} = \widetilde{\mathbf{E}} \cdot \tilde{\mathbf{A}} &= (\mathbf{E} \otimes \mathbf{I}_\tau) \cdot \big(\mathbf{G} \cdot \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top\big) \\ &= (\mathbf{E} \otimes \mathbf{I}_\tau) \cdot \Big[\mathbf{U}_1^\top/\sqrt{q\tilde{\Pi}_1} \;\cdots\; \mathbf{U}_K^\top/\sqrt{q\tilde{\Pi}_K}\Big]^\top \cdot \boldsymbol{\Sigma}\mathbf{V}^\top \\ &=: (\mathbf{E} \otimes \mathbf{I}_\tau) \cdot \Big[\tilde{\mathbf{U}}_1^\top \;\cdots\; \tilde{\mathbf{U}}_K^\top\Big]^\top \cdot \boldsymbol{\Sigma}\mathbf{V}^\top \\ &=: \overbrace{\Big[\underbrace{\tilde{\mathbf{U}}_1^\top \;\cdots\; \tilde{\mathbf{U}}_1^\top}_{r_1}\; \underbrace{\tilde{\mathbf{U}}_2^\top \;\cdots\; \tilde{\mathbf{U}}_2^\top}_{r_2} \;\cdots\; \underbrace{\tilde{\mathbf{U}}_K^\top \;\cdots\; \tilde{\mathbf{U}}_K^\top}_{r_K}\Big]^\top}^{\tilde{\mathbf{U}}_{\exp} \in \mathbb{R}^{R\tau \times d}} \cdot \boldsymbol{\Sigma}\mathbf{V}^\top.\end{aligned}$$

Note that $\tilde{\mathbf{U}}_{\exp}\boldsymbol{\Sigma}\mathbf{V}^\top$ is not the SVD of $\boldsymbol{\Psi}$. For the normalizing factor of $\frac{q}{Rd}$:

$$\tilde{Q}_\iota = \frac{q}{Rd}\cdot\|\tilde{\mathbf{U}}\|_F^2 = \frac{q}{Rd}\cdot\frac{\|\mathbf{U}_\iota\|_F^2}{q\tilde{\Pi}_\iota} = \frac{\Pi_\iota}{R\tilde{\Pi}_\iota} \leqslant \frac{1}{R\beta_{\tilde{\Pi}}} \quad \Longrightarrow \quad \sum_{i=1}^{R} \big|\tilde{Q}_i - 1/R\big| \overset{\triangle}{\leqslant} \frac{R}{R\beta_{\tilde{\Pi}}} = \frac{1}{\beta_{\tilde{\Pi}}}\,,$$

where $\triangle$ follows from the fact that $\big|\tilde{Q}_i - 1/R\big| \leqslant 1/\big(R\beta_{\tilde{\Pi}}\big)$ for each $i \in \mathbb{N}_R$. After normalizing by $1/R$ according to the distortion metric, we deduce that $d_{\mathcal{U},\tilde{Q}} \leqslant 1/\big(R\beta_{\tilde{\Pi}}\big)$.

In the case where $\tilde{\Pi}_{\{K\}} = \Pi_{\{K\}}$, we have

$$\tilde{Q}_\iota = \frac{q}{Rd}\cdot\|\tilde{\mathbf{U}}_\iota\|_F^2 = \frac{q}{Rd}\cdot\frac{\|\mathbf{U}_\iota\|_F^2}{q\Pi_\iota} = \frac{\Pi_\iota}{R\Pi_\iota} = \frac{1}{R}$$

for all $\iota \in \mathbb{N}_K$, thus $\tilde{Q}_{\{K\}} = \mathcal{U}_{\{K\}}$. □

Finally, in Proposition 2.3.1 we show when the block leverage score sampling sketch of Algorithm 5 and the block-SRHT of [49] have the same ℓ_2-s.e. guarantee. We first recall the corresponding result to Theorem 3.3.1, of the block-SRHT.

Theorem 2.3.1 ([49, Theorem 7])**.** *The block-SRHT $\mathbf{S}_{\hat{\mathbf{H}}}$ is a ℓ_2-s.e. of $\mathbf{A}$. For $\delta > 0$ and $q = \Theta\big(\frac{d}{\tau}\log(Nd/\delta)\cdot\log(2d/\delta)/\epsilon^2\big)$:*

$$\Pr\big[\|\mathbf{I}_d - \mathbf{U}^\top\mathbf{S}_{\hat{\mathbf{H}}}^\top\mathbf{S}_{\hat{\mathbf{H}}}\mathbf{U}\|_2 \leqslant \epsilon\big] \geqslant 1-\delta\,.$$

Proposition 2.3.1. *Let $\beta = 1$. For $\delta = e^{\Theta(1)}/(Nd)$, the sketches of Algorithm 5 and the block-SRHT of [49] achieve the same asymptotic ℓ_2-s.e. guarantee, for the same number of sampling trials q.*

Proof. For $\delta = e^{\Theta(1)}/(Nd)$, the two sketching methods have the same q; and both satisfy the ℓ_2-s.e. property with error probability $1-\delta$. □

2.4 Contraction Rate of Block Leverage Score Sampling

In this appendix we quantify the contraction rate of our method on the error term $\mathbf{x}^{[s]} - \mathbf{x}^\star$, which further characterizes the convergence of SD after applying our method. The contraction rate is compared to that of regular SD.

Recall that the contraction rate of an iterative process given by a function $h(x^{[s]})$ is the constant $\gamma \in (0,1)$ for which at each iteration we are guaranteed that $h(x^{[s+1]}) \leqslant \gamma \cdot h(x^{[s]})$, therefore $h(x^{[s]}) \leqslant \gamma_s \cdot h(x^{[0]})$. Let ξ be a fixed step-size, $\widetilde{\mathbf{S}}_{[s]}$ the induced sketching matrix of Algorithm 5 at iteration s, and define $\mathbf{B}_{SD} = \left(\mathbf{I}_d - 2\xi \cdot \mathbf{A}^\top \mathbf{A}\right)$ and $\mathbf{B}_s = \left(\mathbf{I}_d - 2\xi \cdot \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{A}\right)$. We note that the contraction rates could be further improved if one also incorporates an optimal step-size. It is also worth noting that when weighting from Appendix 2.5 is introduced, we have the same contraction rate and straggler ratio.

Lemma 2.4.1. *For $\widetilde{\mathbf{S}}$ the sketching matrix of Algorithm 5, we have $\mathbb{E}\left[\widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}}\right] = \mathbf{I}_N$.*

Proof. Similar to the proof of Proposition 2.1.1, we define a symmetric random matrix $\mathbf{Y}$, whose realizations correspond to the sampled and rescaled submatrices of Algorithm 5. The realizations are

$$\mathbf{Y}_i = \frac{\mathbf{I}_{(\mathcal{K}_\iota^i)}^\top \mathbf{I}_{(\mathcal{K}_\iota^i)}}{q\tilde{\Pi}_\iota} = \frac{\sum_{l \in \mathcal{K}_\iota^i} \mathbf{e}_l \mathbf{e}_l^\top}{q\tilde{\Pi}_\iota} \ .$$

After q sampling trials, we have $\widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}} = \sum_{i=1}^q \mathbf{Y}_i$. It follows that

$$\begin{aligned}
\mathbb{E}\left[\widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}}\right] &= \mathbb{E}\left[\sum_{i=i}^q \mathbf{Y}_i\right] \\
&= \sum_{i=1}^q \mathbb{E}\left[\mathbf{Y}_i\right] \\
&= q \cdot \left(\sum_{j=1}^K \tilde{\Pi}_j \cdot \frac{\mathbf{I}_{(\mathcal{K}_j)}^\top \mathbf{I}_{(\mathcal{K}_j)}}{q\tilde{\Pi}_j}\right) \\
&= \sum_{j=1}^K \mathbf{I}_{(\mathcal{K}_j)}^\top \mathbf{I}_{(\mathcal{K}_j)} \\
&= \sum_{l=1}^N \mathbf{e}_{(l)} \mathbf{e}_{(l)}^\top \\
&= \mathbf{I}_N \ .
\end{aligned}$$

□

Theorem 2.4.1. *The contraction rate of our GC approach through the expected sketch $\widetilde{\mathbf{S}}_{[s]}$ at each iteration, is equal to the contraction rate of regular SD. Specifically, for $e_s \coloneqq \mathbf{x}^{[s]} - \mathbf{x}^\star$ the error at iteration s and $\gamma_{SD} = \lambda_1(\mathbf{B}_{SD})$ the contraction rate of regular SD, we have $\left\|\mathbb{E}[e_{s+1}]\right\|_2 \leqslant \gamma_{SD} \cdot \|e_s\|_2$.*

Proof. For a fixed step-size 2ξ, our SD parameter update at iteration $s+1$ is

$$\mathbf{x}^{[s+1]} \leftarrow \mathbf{x}^{[s]} - 2\xi \cdot \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \left(\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\right) \ ,$$

where for regular SD we have $\widetilde{\mathbf{S}}_{[s]} \leftarrow \mathbf{I}_N$. At iteration $s+1$, the error e_s of the previous iteration is not random, hence $\mathbb{E}[e_s] = e_s$. By substituting the expression of $\mathbf{B}_s$, it follows that

$$\begin{aligned}
e_{s+1} &= \mathbf{x}^{[s+1]} - \mathbf{x}^\star \\
&= \left(\mathbf{x}^{[s]} - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \left(\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\right)\right) - \mathbf{x}^\star \\
&= \mathbf{x}^{[s]} - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{A}\mathbf{x}^{[s]} + 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b} - \mathbf{x}^\star \\
&= \mathbf{B}_s \mathbf{x}^{[s]} - \left(\mathbf{x}^\star - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b}\right) \\
&= \mathbf{B}_s \mathbf{x}^{[s]} - \left(\mathbf{x}^\star - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \left(\mathbf{A}\mathbf{x}^\star + \mathbf{b}^\perp\right)\right) \\
&= \mathbf{B}_s \left(\mathbf{x}^{[s]} - \mathbf{x}^\star\right) - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b}^\perp \\
&= \mathbf{B}_s e_s - 2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b}^\perp
\end{aligned} \tag{2.4}$$

and by Lemma 2.4.1

$$\mathbb{E}\left[2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b}^\perp\right] = 2\xi \mathbf{A}^\top \mathbb{E}\left[\widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]}\right] \mathbf{b}^\perp = 2\xi \mathbf{A}^\top \mathbf{b}^\perp = \mathbf{0}_{d\times 1} \tag{2.5}$$

as $\mathbf{b}^\perp$ lies in the kernel of $\mathbf{A}^\top$, and

$$\mathbb{E}\left[\mathbf{B}_s\right] = \mathbf{I}_d - 2\xi \mathbf{A}^\top \mathbb{E}\left[\widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]}\right] \mathbf{A} = \mathbf{I}_d - 2\xi \mathbf{A}^\top \mathbf{A} = \mathbf{B}_{SD} \ . \tag{2.6}$$

From (2.4), (2.5) and (2.6), it follows that

$$\mathbb{E}\left[e_{s+1}\right] = \mathbb{E}\left[\mathbf{B}_s e_s\right] - \mathbb{E}\left[2\xi \mathbf{A}^\top \widetilde{\mathbf{S}}_{[s]}^\top \widetilde{\mathbf{S}}_{[s]} \mathbf{b}^\perp\right] = \mathbb{E}\left[\mathbf{B}_s\right] \cdot e_s = \mathbf{B}_{SD} \cdot e_s \ . \tag{2.7}$$

This gives us the contraction rate of the expected sketch through Algorithm 5

$$\left\|\mathbb{E}[e_{s+1}]\right\|_2 \leqslant \lambda_1\big(\mathbb{E}[\mathbf{B}_s]\big) \cdot \|e_s\|_2 = \lambda_1(\mathbf{B}_{SD}) \cdot \|e_s\|_2 \quad \Longrightarrow \quad \gamma_{s+1} = \lambda_1(\mathbf{B}_{SD}) \ . \tag{2.8}$$

By replacing $\mathbf{B}_s$ with $\mathbf{B}_{SD}$ in (2.4) and $\widetilde{\mathbf{S}}_{[s]} \leftarrow \mathbf{I}_N$, we conclude that the contraction rate of SD is $\gamma_{SD} = \lambda_1(\mathbf{B}_{SD})$. □

We conclude this appendix by stating the expected ratio of dimensions r to N, and stragglers to servers.

Remark 2.4.1. *The expected ratio of the reduced dimension r to the original dimension N is $q(T)\tau/N$, and the expected straggler to servers ratio is $(m-q)/m$. Since we stop receiving computations at a time instance T; we expect that $q \leftarrow q(T)$ computations are received, hence there are $m-q$ stragglers. Thus, the straggler to servers ratio is $(m-q)/m$. The expected ratio between the two dimensions is immediate from the fact that $r = q\tau$ is the new reduced dimension, in the case where $q \leqslant K$.*

2.5 Weighted Block Leverage Score Sketch

So far, we have considered sampling w.r. according to the normalized block leverage scores, to reduce the effective dimension N of $\mathbf{A}$ and $\mathbf{b}$ to $r = q\tau$. In this appendix, we show that by *weighting* the sampled blocks according to the sampling which has taken place through $\mathbf{\Omega}$ for the construction of the sketching matrix $\widetilde{\mathbf{S}}$, we can further compress the data matrix $\mathbf{A}$; and get the same results when first and second order optimization methods are used to solve (3.10). The weighting we propose is more beneficial with non-uniform distributions, as we expect the sampling w.r. to capture the importance of the more influence blocks. The *weighted sketching matrix* $\widetilde{\mathbf{S}}_{\mathbf{w}}$ we propose, is a simple extension to $\widetilde{\mathbf{S}}$ of Algorithm 5. For simplicity, we do not consider iterative sketching, though similar arguments and guarantees can be derived.

The main idea is to not keep repetitions of blocks which were sampled multiple times, but rather weigh each block by the number of times it was sampled. By doing so, we retain the *weighted sketch* $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$ of size $\bar{q}\tau \times d$; for $\bar{q}$ the number of distinct blocks that were sampled.[1] Additionally, the gradient and Hessian of $L_{\mathbf{S}}\big(\widetilde{\mathbf{S}}_{\mathbf{w}}, \mathbf{A}, \mathbf{b}; \mathbf{x}\big)$ are respectively equal to those of $L_{\mathbf{S}}\big(\widetilde{\mathbf{S}}, \mathbf{A}, \mathbf{b}; \mathbf{x}\big)$; and are unbiased estimators of the gradient and Hessian of $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x})$.

To achieve the weighting algorithmically, we count how many times each of the distinct $\bar{q}$ blocks were sampled, and at the end of the sampling procedure we multiply the blocks by their corresponding "weight". We initialize a weight vector $\mathbf{w} = \mathbf{0}_{1\times K}$,

[1] In practice, for highly non-uniform $\Pi_{\{K\}}$ we expect $\bar{q}\tau \ll r = q\tau$. The sketch $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$ could therefore be stored in much less space than $\widetilde{\mathbf{S}}\mathbf{A}$, and the system of equations $\widetilde{\mathbf{S}}_{\mathbf{w}}(\mathbf{A}-\mathbf{b}) = \mathbf{0}_{\bar{q}\tau\times 1}$ could have significantly fewer equations than $\widetilde{\mathbf{S}}(\mathbf{A}-\mathbf{b}) = \mathbf{0}_{q\tau\times 1}$.

and in the sampling procedure whenever the i^{th} partition is drawn, we update its corresponding weight: $\mathbf{w}_i \leftarrow \mathbf{w}_i + 1$. It is clear that once q trials are been carried out, we have $\|\mathbf{w}\|_1 = q$ for $\mathbf{w} \in \mathbb{N}_0^{1\times K}$.

Let $\mathcal{S}$ denote the index multiset observed after the sampling procedure of Algorithm 5, and $\bar{\mathcal{S}}$ the set of indices comprising $\mathcal{S}$. That is, $\mathcal{S}$ has cardinality q and may have repetitions, while $\bar{\mathcal{S}} = \mathbb{N}_K \cap \mathcal{S}$ has cardinality $\bar{q} = |\bar{\mathcal{S}}| \leqslant q$ with no repetitions. We denote the ratio of the two sets by $\zeta \coloneqq q/\bar{q} = \|\mathbf{w}\|_1/\|\mathbf{w}\|_0 \geqslant 1$, which indicates how much further compression we get by utilizing the fact that blocks may be sampled multiple times. The corresponding weighted sketching matrix $\widetilde{\mathbf{S}}_{\mathbf{w}}$ of $\widetilde{\mathbf{S}}$ is then

$$\widetilde{\mathbf{S}}_{\mathbf{w}} = \overbrace{\operatorname{diag}\left(\left\{\sqrt{\mathbf{w}_j/\left(q\Pi_j\right)}\right\}_{j\in\bar{\mathcal{S}}}\right)}^{\bar{\mathbf{W}}_{1/2}\in\mathbb{R}^{\bar{q}\times\bar{q}}_{\geqslant 0}} \cdot \mathbf{I}_{(\bar{\mathcal{S}})} \otimes \mathbf{I}_\tau \in \mathbb{R}^{\bar{q}\tau\times N} \tag{2.9}$$

where $\mathbf{I}_{(\bar{\mathcal{S}})} \in \{0,1\}^{\bar{q}\times K}$ is the restriction of $\mathbf{I}_K$ to the rows indexed by $\bar{\mathcal{S}}$.

For simplicity, assume that the sampling matrices which are devised for $\widetilde{\mathbf{S}}$ and $\widetilde{\mathbf{S}}_{\mathbf{w}}$ follow the ordering of the sampled blocks in order of the samples, *i.e.* if $\left(\mathbf{\Omega}_{(i)}\right)_j = 1$ then $\left(\mathbf{\Omega}_{(i+1)}\right)_l = 1$ for $l \geqslant j$; and equivalently $\mathcal{S}_l \leqslant \mathcal{S}_{l+1}$ and $\bar{\mathcal{S}}_l < \bar{\mathcal{S}}_{l+1}$ for all valid l. We restrict the sampling matrix $\mathbf{\Omega} \in \{0,1\}^{q\times K}$ to its unique rows, by applying $\bar{\mathbf{\Omega}} \in \{0,1\}^{\bar{q}\times q}$:

$$\bar{\mathbf{\Omega}}_{ij} = \begin{cases} 1 & \text{for } i = 1 \text{ and } j = \mathcal{S}_1 \\ 1 & \text{if } \mathcal{S}_j > \mathcal{S}_{j-1} \text{ for } j \in \mathbb{N}_q\backslash\{1\} \\ 0 & \text{otherwise} \end{cases}$$

to the left of the sampling matrix $\mathbf{\Omega}$ of Algorithm 5, *i.e.* $\mathbf{\Omega}_{\mathbf{w}} \coloneqq \bar{\mathbf{\Omega}} \cdot \mathbf{\Omega} \in \{0,1\}^{\bar{q}\times K}$. This sampling matrix then satisfies

$$\left(\mathbf{\Omega}_{\mathbf{w}}\right)_{ij} = \begin{cases} 1 & \text{if } j = \bar{\mathcal{S}}_j \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } j \in \mathbb{N}_{\bar{q}} \ .$$

Let $\tilde{\mathbf{w}} = \mathbf{w}_{|\bar{\mathcal{S}}} \in \mathbb{Z}_+^{1\times\bar{q}}$ be the restriction of $\mathbf{w}$ to its nonzero elements; hence $\|\tilde{\mathbf{w}}\|_1 = \|\mathbf{w}\|_1 = |\mathcal{S}| = q$, and define the rescaling diagonal matrix $\tilde{\mathbf{W}}_{1/2} = \operatorname{diag}\left(\left\{\sqrt{\tilde{\mathbf{w}}_i}\right\}_{i=1}^{\bar{q}}\right)$. We then have the following relationship

$$\widetilde{\mathbf{S}}_{\mathbf{w}} = \overbrace{\left(\tilde{\mathbf{W}}_{1/2} \cdot \bar{\mathbf{\Omega}} \cdot \mathbf{D} \cdot \bar{\mathbf{\Omega}}^\top\right)}^{=\bar{\mathbf{W}}_{1/2}} \cdot \mathbf{\Omega}_{\mathbf{w}} \otimes \mathbf{I}_\tau = \left(\bar{\mathbf{W}}_{1/2} \otimes \mathbf{I}_\tau\right) \cdot \left(\mathbf{\Omega}_{\mathbf{w}} \otimes \mathbf{I}_\tau\right) \tag{2.10}$$

when $\bar{\mathcal{S}} = \mathbb{N}_K \cap \mathcal{S}$.

As previously noted, $\widetilde{\mathbf{S}}_{\mathbf{w}}$ has ζ times less rows than $\widetilde{\mathbf{S}}$. Hence, the required storage space for the sketch $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$ drops by a multiplicatively factor of ζ, and the required operations are reduced analogously; according to the computation. The weighted sketching matrix $\widetilde{\mathbf{S}}_{\mathbf{w}}$ has the following guarantees, which imply that the proposed weighting will not affect first or second order iterative methods which are used to approximate (3.10).

Proposition 2.5.1. *The resulting gradient and Hessian of the modified least squares problem (3.10) when sketching with $\widetilde{\mathbf{S}}$ of Algorithm 5, are respectively identical to the resulting gradient and Hessian when sketching with $\widetilde{\mathbf{S}}_{\mathbf{w}}$ presented in (2.9) and (2.10).*

Proof. From Algorithm 5, the assumption on the ordering of the elements in $\mathcal{S}$ and $\bar{\mathcal{S}}$, and the construction of $\widetilde{\mathbf{S}}$, we have

$$\begin{aligned}
\widetilde{\mathbf{S}}_{\mathbf{w}}^{\top} \cdot \widetilde{\mathbf{S}}_{\mathbf{w}} &= \left(\left(\boldsymbol{\Omega}_{\mathbf{w}}^{\top} \cdot \tilde{\mathbf{W}}_{1/2}^{\top} \right) \otimes \mathbf{I}_{\tau} \right) \cdot \left(\left(\tilde{\mathbf{W}}_{1/2} \cdot \boldsymbol{\Omega}_{\mathbf{w}} \right) \otimes \mathbf{I}_{\tau} \right) \\
&= \left(\left(\boldsymbol{\Omega}^{\top} \cdot \mathbf{D}^{\top} \right) \otimes \mathbf{I}_{\tau} \right) \cdot \left(\left(\mathbf{D} \cdot \boldsymbol{\Omega} \right) \otimes \mathbf{I}_{\tau} \right) \\
&= \widetilde{\mathbf{S}}^{\top} \cdot \widetilde{\mathbf{S}} \ .
\end{aligned}$$

Let $\mathcal{T} = \biguplus_{j \in \mathcal{S}} \mathcal{K}_j$ and $\bar{\mathcal{T}} = \bigsqcup_{j \in \bar{\mathcal{S}}} \mathcal{K}_j$, thus $\bar{\mathcal{T}}$ is contained in $\mathcal{T}$ when both are viewed as multisets. Considering the objective function $L_{\mathbf{S}}(\mathbf{S}, \mathbf{A}, \mathbf{b}; \mathbf{x})$ of (3.10), the equivalence of gradients is observed through the following computation

$$\begin{aligned}
\nabla_{\mathbf{x}} L_{\mathbf{S}}(\widetilde{\mathbf{S}}, \mathbf{A}, \mathbf{b}; \mathbf{x}) &= 2\mathbf{A}^{\top} \left(\widetilde{\mathbf{S}}^{\top} \widetilde{\mathbf{S}} \right) (\mathbf{A}\mathbf{x} - \mathbf{b}) \\
&= 2 \sum_{l \in \mathcal{T}} \mathbf{A}_{(l)}^{\top} \cdot \mathbf{D}_{ll}^2 \cdot \left(\mathbf{A}_{(l)} \mathbf{x} - \mathbf{b}_l \right) \\
&= 2 \sum_{j \in \bar{\mathcal{T}}} \tilde{\mathbf{w}}_j \cdot \mathbf{A}_{(j)}^{\top} \cdot \mathbf{D}_{jj}^2 \cdot \left(\mathbf{A}_{(j)} \mathbf{x} - \mathbf{b}_j \right) \\
&= 2 \sum_{j \in \bar{\mathcal{T}}} \mathbf{A}_{(j)}^{\top} \cdot \left(\tilde{\mathbf{W}}_{1/2} \right)_{jj}^2 \cdot \left(\mathbf{A}_{(j)} \mathbf{x} - \mathbf{b}_j \right) \\
&= 2\mathbf{A}^{\top} \left(\widetilde{\mathbf{S}}_{\mathbf{w}}^{\top} \widetilde{\mathbf{S}}_{\mathbf{w}} \right) (\mathbf{A}\mathbf{x} - \mathbf{b}) \\
&= \nabla_{\mathbf{x}} L_{\mathbf{S}}(\widetilde{\mathbf{S}}_{\mathbf{w}}, \mathbf{A}, \mathbf{b}; \mathbf{x}) \ .
\end{aligned}$$

Recall that the Hessian of the least squares objective function (3.2) is $\nabla_{\mathbf{x}}^2 L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}) = 2\mathbf{A}^{\top}\mathbf{A}$. Considering the modified objective function (3.10) and our sketching matri-

ces, it follows that

$$\begin{aligned}
\nabla^2_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}, \mathbf{A}, \mathbf{b}; \mathbf{x}\right) &= 2\mathbf{A}^\top \left(\widetilde{\mathbf{S}}^\top \widetilde{\mathbf{S}}\right) \mathbf{A} \\
&= 2 \sum_{l \in \mathcal{T}} \mathbf{A}_{(l)}^\top \cdot \mathbf{D}_{ll}^2 \cdot \mathbf{A}_{(l)} \\
&= 2 \sum_{j \in \bar{\mathcal{T}}} \tilde{\mathbf{w}}_j \cdot \mathbf{A}_{(j)}^\top \cdot \mathbf{D}_{jj}^2 \cdot \mathbf{A}_{(j)} \\
&= 2 \sum_{j \in \bar{\mathcal{T}}} \mathbf{A}_{(j)}^\top \cdot \left(\tilde{\mathbf{W}}_{1/2}\right)_{jj}^2 \cdot \mathbf{A}_{(j)} \\
&= 2\mathbf{A}^\top \left(\widetilde{\mathbf{S}}_{\mathbf{w}}^\top \widetilde{\mathbf{S}}_{\mathbf{w}}\right) \mathbf{A} \\
&= \nabla^2_{\mathbf{x}} L_{\mathbf{S}}\big(\widetilde{\mathbf{S}}_{\mathbf{w}}, \mathbf{A}, \mathbf{b}; \mathbf{x}\big)
\end{aligned}$$

which completes the proof. □

Corollary 2.5.1. *At each iteration, the gradient and Hessian of the weighted sketch system of equations* $\widetilde{\mathbf{S}}_{\mathbf{w}}(\mathbf{A} - \mathbf{b}) = \mathbf{0}_{\bar{q}\tau \times 1}$*, are unbiased estimators of the gradient and Hessian of the original system* $(\mathbf{A} - \mathbf{b}) = \mathbf{0}_{N \times 1}$.

Proof. Denote the gradient and Hessian of the weighted sketch at iteration s by $\hat{g}_{\mathbf{w}}^{[s]}$ and $\hat{H}_{\mathbf{w}}^{[s]}$ respectively. By Proposition 2.5.1 we know that $\hat{g}_{\mathbf{w}}^{[s]} = \hat{g}^{[s]}$, and by Theorem 3.3.2 that $\mathbb{E}\left[\hat{g}^{[s]}\right] = g^{[s]}$. Hence $\mathbb{E}\left[\hat{g}_{\mathbf{w}}^{[s]}\right] = g^{[s]}$.

Following the same notation as in the proof of Theorem 3.3.2, the Hessian $\hat{H}^{[s]} = \nabla^2_{\mathbf{x}} L_{\mathbf{S}}\left(\widetilde{\mathbf{S}}^{[s]}, \mathbf{A}, \mathbf{b}; \mathbf{x}^{[s]}\right)$ is

$$\hat{H}^{[s]} = 2 \sum_{i \in \mathcal{I}^{[s]}} \frac{1}{q\tilde{\Pi}_i} \mathbf{A}_i^\top \mathbf{A}_i$$

thus

$$\mathbb{E}\left[\hat{H}^{[s]}\right] = 2\mathbb{E}\left[\sum_{i \in \mathcal{I}^{[s]}} \frac{1}{q\tilde{\Pi}_i} \mathbf{A}_i^\top \mathbf{A}_i\right] = 2 \sum_{i \in \mathcal{I}^{[s]}} \sum_{j=1}^{K} \tilde{\Pi}_j \frac{1}{q\tilde{\Pi}_j} \mathbf{A}_j^\top \mathbf{A}_j = 2q \cdot \sum_{j=1}^{K} \frac{1}{q} \mathbf{A}_i^\top \mathbf{A}_i = 2\mathbf{A}^\top \mathbf{A}$$

which is precisely the Hessian of (3.2). By Proposition 2.5.1, it follows that $\mathbb{E}\left[\hat{H}_{\mathbf{w}}^{[s]}\right] = 2\mathbf{A}^\top \mathbf{A}$, which completes the proof. □

Geometrically, from the point of view of adding vectors, the partial gradients of the partitions sampled will be scaled accordingly to their weights. Therefore, the partial gradients $\hat{g}_i$ with higher weights have a greater influence in the direction of the resulting gradient $\hat{g}$. This was also the fundamental idea behind our sketching and GC techniques, as the partitions sampled multiple times are of greater importance.

Next, we quantify the expected dimension of the weighted sketch $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$. This shows the dependence on $\tilde{\Pi}_{\{K\}}$, and further justifies that we attain a higher compression factor ζ when the block leverage scores are non-uniform.

Theorem 2.5.1. *The expected reduced dimension of* $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$ *is* $\left(K - \sum_{i=1}^{K}(1-\tilde{\Pi}_i)^q\right)\cdot\tau$, *which is maximal when* $\tilde{\Pi}_{\{K\}}$ *is uniform.*

Proof. It suffices to determine the expected number of distinct blocks $\mathbf{A}_i$ which are sampled after q trials when carrying out Algorithm 5. The probability of not sampling $\mathbf{A}_i$ at a given trial is $\left(1-\tilde{\Pi}_i\right)$, hence not sampling $\mathbf{A}_i$ at any trial occurs with probability $\left(1-\tilde{\Pi}_i\right)^q$; since the trials are identical and independent. Thus, the expected number of distinct blocks being sampled is

$$\begin{aligned}
\mathbb{E}[\bar{q}] &= \sum_{i=1}^{K} 1 \cdot \Pr\left[\mathbf{A}_i \text{ was sampled at least once sampled}\right] \\
&= \sum_{i=1}^{K} \left(1 - \cdot \Pr\left[\mathbf{A}_i \text{ was } \textit{not} \text{ sampled at any trial}\right]\right) \\
&= \sum_{i=1}^{K} \left(1 - \left(1-\tilde{\Pi}_i\right)^q\right) \\
&= K - \sum_{i=1}^{K} \left(1-\tilde{\Pi}_i\right)^q \ .
\end{aligned}$$

Thus, the expected reduced dimension is $\tau \cdot \mathbb{E}[\bar{q}]$.

Let $Q\big(\tilde{\Pi}_{\{K\}}\big) \coloneqq \sum_{i=1}^{K}(1-\tilde{\Pi}_i)^q$, and introduce the Lagrange multiplier $\lambda > 0$ to the constraint $R\big(\tilde{\Pi}_{\{K\}}\big) = \left(\sum_{i=1}^{K}\tilde{\Pi}_i - 1\right)$, to get the Lagrange function

$$\mathscr{L}\big(\tilde{\Pi}_{\{K\}}, \lambda\big) \coloneqq Q\big(\tilde{\Pi}_{\{K\}}\big) + \lambda \cdot R\big(\tilde{\Pi}_{\{K\}}\big) = \sum_{i=1}^{K}\left(\lambda\cdot\tilde{\Pi}_i + \left(1-\tilde{\Pi}_i\right)^q\right) - \lambda \tag{2.11}$$

for which

$$\frac{\partial\mathscr{L}\big(\tilde{\Pi}_{\{K\}}, \lambda\big)}{\partial\tilde{\Pi}_i} = \lambda - q(1-\tilde{\Pi}_i)^{q-1} = 0 \tag{2.12}$$

$$\implies \quad \tilde{\Pi}_i = 1 - (\lambda/q)^{1/(q-1)} \quad \text{and} \quad \lambda = q(1-\tilde{\Pi}_i)^{q-1} \tag{2.13}$$

for all $i \in \mathbb{N}_K$, and

$$\frac{\partial\mathscr{L}\big(\tilde{\Pi}_{\{K\}}, \lambda\big)}{\partial\lambda} = \sum_{i=1}^{K}\tilde{\Pi}_i - 1 = 0 \ . \tag{2.14}$$

Note that the uniform distribution $\mathcal{U}_{\{K\}} = \left\{\tilde{\Pi}_i = 1/K\right\}_{i=1}^{K}$ is a solution to (2.14).

We will now verify that $\mathcal{U}_{\{K\}}$ satisfies (2.12). From (2.13); for $\tilde{\Pi}_{\{K\}} \leftarrow \mathcal{U}_{\{K\}}$, we have $\lambda = q(1-1/K)^{q-1} > 0$, which we substitute into (2.12):

$$\lambda - q(1-\tilde{\Pi}_i)^{q-1} = q(1-1/K)^{q-1} - q(1-1/K)^{q-1} = 0 \ . \tag{2.15}$$

Hence, $\mathcal{U}_{\{K\}}$ is the solution to both (2.12) and (2.14).

By the second derivative test; since $\partial^2 \mathscr{L}\big(\tilde{\Pi}_{\{K\}}\big)/\partial \tilde{\Pi}_i^2 = q(q-1)(1-\tilde{\Pi}_i)^{q-2}$ is positive for $\tilde{\Pi}_i = 1/K$, we conclude that $Q\big(\mathcal{U}_{\{K\}}\big) \leqslant Q\big(\tilde{\Pi}_{\{K\}}\big)$ for any $\tilde{\Pi}_{\{K\}} \neq \mathcal{U}_{\{K\}}$. This implies that $\mathbb{E}[\bar{q}]$ is maximal when $\tilde{\Pi}_{\{K\}} = \mathcal{U}_{\{K\}}$, and so is the expected reduced dimension of $\widetilde{\mathbf{S}}_{\mathbf{w}}\mathbf{A}$. □

We further note that $\mathbb{E}[\bar{q}]$ from the proof of Theorem 2.5.1, is trivially minimal in the degenerate case where $\tilde{\Pi}_\iota = 1$ for a single $\iota \in \mathbb{N}_K$, and $\tilde{\Pi}_j = 0$ for every $j \in \mathbb{N}_K \backslash \{\iota\}$. This occurs in the case where $\mathbf{A}_j = \mathbf{0}_{\tau \times d}$ for each j, and $\bar{q}$ is therefore exactly

$$K - \sum_{j \neq \iota} (1-\tilde{\Pi}_j)^q = K - \sum_{j \neq \iota} 1^q = K - (K-1) = 1 \ .$$

Appendix C

Appendix to Chapter IV

3.1 Proofs of Section 4.3

3.1.1 Subsection 4.3.1

Note that in Lemma 4.3.1:

$$\mathbb{E}\left[\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]}\right] = \mathbf{I}_N \quad \Longrightarrow \quad \mathbb{E}\left[\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\right] = \mathbf{I}_N \ ,$$

as

$$\mathbb{E}\left[\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\right] = \mathbf{\Pi}^T\mathbb{E}\left[\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]}\right]\mathbf{\Pi} = \mathbf{\Pi}^T\mathbf{\Pi} = \mathbf{I}_N \ .$$

We provide both derivations separately in order to convey the respective importance behind the use of the Lemma in subsequent arguments, even though the main idea is the same. Furthermore, the proof of Theorem 4.3.1 is very similar to that of Lemma 4.3.1.

Proof. [Lemma 4.3.1] The only difference in $\mathbf{S}_{\mathbf{\Pi}}^{[t]}$ at each iteration, is $\mathcal{S}^{[t]}$ and $\widetilde{\mathbf{\Omega}}_{[t]}$. This corresponds to a uniform random selection of q out of K batches of the data which determine the gradient at iteration t — all blocks are scaled by the same factor

$\sqrt{K/q}$ in $\widetilde{\mathbf{\Omega}}_{[t]}$. Let $\mathcal{Q}$ be the set of all subsets of $\mathbb{N}_K$ of size q. Then

$$\begin{aligned}
\mathbb{E}\big[\mathbf{S}_{[t]}^T\mathbf{S}_{[t]}\big] &= \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \frac{1}{\binom{K}{q}} \cdot \big(\mathbf{S}_{[t]} \cdot \mathbf{S}_{[t]}\big) \\
&= \frac{1}{\binom{K}{q}} \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \sum_{i\in\mathcal{S}^{[t]}} \Big(\sqrt{K/q}\Big)^2 \cdot \mathbf{\Pi}_{(\mathcal{K}_i)}^T\mathbf{\Pi}_{(\mathcal{K}_i)} \\
&= \frac{\binom{K-1}{q-1}}{\binom{K}{q}} \sum_{i=1}^{K} \frac{K}{q} \cdot \mathbf{\Pi}_{(\mathcal{K}_i)}^T\mathbf{\Pi}_{(\mathcal{K}_i)} \\
&= \frac{\binom{K-1}{q-1} \cdot \frac{K}{q}}{\binom{K}{q}} \sum_{i=1}^{K} \mathbf{\Pi}_{(\mathcal{K}_i)}^T\mathbf{\Pi}_{(\mathcal{K}_i)} \\
&= \mathbf{\Pi}^T\mathbf{\Pi} \\
&= \mathbf{I}_N
\end{aligned}$$

where $\binom{K-1}{q-1}$ is the number of sets in $\mathcal{Q}$ which include i, for each $i \in \mathbb{N}_K$. This completes the first part of the proof.

Note that the sampling and rescaling matrices $\widetilde{\mathbf{\Omega}}_{[t]}$ of Algorithm 7, may also be expressed as

$$\widetilde{\mathbf{\Omega}}_{[t]} = \sqrt{K/q} \cdot \sum_{\iota\in\mathcal{S}^{[t]}} \mathbf{I}_{(\mathcal{K}_\iota)} \ .$$

Further notice that $\widetilde{\mathbf{\Omega}}_{[t]}$'s corresponding sampling and rescaling matrix of size $N \times N$, which appears in the expansion the objective function (4.5), is

$$\begin{aligned}
\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]} &= \Big(\sqrt{K/q}\Big)^2 \cdot \sum_{\iota\in\mathcal{S}^{[t]}} \big(\mathbf{I}_{(\mathcal{K}_\iota)}\big)^T\mathbf{I}_{(\mathcal{K}_\iota)} \\
&= \frac{K}{q} \cdot \sum_{j\in \bigsqcup_{\iota\in\mathcal{S}^{[t]}} \mathcal{K}_\iota} \mathbf{e}_j\mathbf{e}_j^T \ .
\end{aligned}$$

Let $\mathcal{B}$ denote the set of all possible block sampling and rescaling matrices of size $r \times N$, which sample q out of K blocks. For $\mathbf{\Phi} \in \mathcal{B}$, by $\mathbf{I}_{(\mathcal{K}_\iota)} \subseteq \mathbf{\Phi}$ we denote the condition that $\mathbf{I}_{(\mathcal{K}_\iota)}$ is a submatrix of $\mathbf{\Phi}$. Note that for each $\iota \in \mathbb{N}_K$, there are $\binom{K-1}{q-1}$

matrices in $\mathcal{B}$ which have $\mathbf{I}_{(\mathcal{K}_\iota)}$ as a submatrix. For our set up, we then have

$$\begin{aligned}
\mathbb{E}\left[\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]}\right] &= \sum_{\mathbf{\Phi}\in\mathcal{B}} \frac{1}{\binom{K}{q}} \cdot \left(\mathbf{\Phi}^T\mathbf{\Phi}\right) \\
&= \frac{\left(\sqrt{K/q}\right)^2}{\binom{K}{q}} \cdot \sum_{\mathbf{\Phi}\in\mathcal{B}} \sum_{\mathbf{I}_{(\mathcal{K}_\iota)}\subseteq\mathbf{\Phi}} \left(\mathbf{I}_{(\mathcal{K}_\iota)}\right)^T\mathbf{I}_{(\mathcal{K}_\iota)} \\
&= \frac{\binom{K-1}{q-1}\cdot(K/q)}{\binom{K}{q}} \cdot \sum_{\iota\in\mathbb{N}_K} \left(\mathbf{I}_{(\mathcal{K}_\iota)}\right)^T\mathbf{I}_{(\mathcal{K}_\iota)} \\
&= \sum_{\iota\in\mathbb{N}_K} \left(\mathbf{I}_{(\mathcal{K}_\iota)}\right)^T\mathbf{I}_{(\mathcal{K}_\iota)} \\
&= \sum_{j\in\mathbb{N}_N} \mathbf{e}_j^T\mathbf{e}_j \\
&= \mathbf{I}_N
\end{aligned}$$

and the proof is complete.

□

Proof. [Theorem 4.3.1] The only difference in $\mathbf{S}_{\mathbf{\Pi}}^{[t]}$ at each iteration, is $\mathcal{S}^{[t]}$ and $\widetilde{\mathbf{\Omega}}_{[t]}$. This corresponds to a uniform random selection of q out of K batches of the data which determine the gradient at iteration t — all blocks are scaled by the same factor $\sqrt{K/q}$ in $\widetilde{\mathbf{\Omega}}_{[t]}$. By (4.10), the gradient update is equal to that of a batch stochastic steepest descent procedure.

We break up the proof of the second statement by first showing that $\mathbb{E}\left[\hat{g}^{[t]}\right] = \tilde{g}^{[t]}$; for $\tilde{g}$ the gradient in the basis $\mathbf{\Pi}\mathbf{U}$, and then showing that $\mathbb{E}\left[\tilde{g}^{[t]}\right] = \frac{q}{K}\cdot g_{ls}^{[t]}$.

Let $\mathcal{Q}$ be the set of all subsets of $\mathbb{N}_K$ of size q, $\hat{g}_{\mathcal{S}^{[t]}}$ the gradient determined by

the index set $\mathcal{S}^{[t]}$, and $\tilde{g}_i^{[t]}$ the respective partial gradients at iteration t. Then

$$\begin{aligned}
\mathbb{E}\left[\hat{g}^{[t]}\right] &= \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \frac{1}{\binom{K}{q}} \cdot \hat{g}_{\mathcal{S}^{[t]}} \\
&= \frac{1}{\binom{K}{q}} \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \sum_{i\in\mathcal{S}^{[t]}} \left(\sqrt{K/q}\right)^2 \cdot \tilde{g}_i^{[t]} \\
&= \frac{\binom{K-1}{q-1}}{\binom{K}{q}} \sum_{i=1}^{K} \frac{K}{q} \cdot \tilde{g}_i^{[t]} \\
&= \sum_{i=1}^{K} \tilde{g}_i^{[t]} \\
&= \tilde{g}^{[t]}
\end{aligned}$$

where $\binom{K-1}{q-1}$ is the number of sets in $\mathcal{Q}$ which include i, for each $i \in \mathbb{N}_K$.

We denote the resulting partial gradient on the sampled index set $\mathcal{S}^{[t]}$ of the gradient on (4.1) at iteration t; *i.e.* $g_{ls}^{[t]}$, by $g_{\mathcal{S}^{[t]}}$, and the individual partial gradients by $g_i^{[t]}$. Using the same notation as above, we get that

$$\begin{aligned}
\mathbb{E}\left[\tilde{g}^{[t]}\right] &= \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \frac{1}{\binom{K}{q}} \cdot g_{\mathcal{S}^{[t]}} \\
&= \frac{1}{\binom{K}{q}} \sum_{\mathcal{S}^{[t]}\in\mathcal{Q}} \sum_{i\in\mathcal{S}^{[t]}} g_i^{[t]} \\
&= \frac{\binom{K-1}{q-1}}{\binom{K}{q}} \sum_{i=1}^{K} g_i^{[t]} \\
&= \frac{q}{K} \cdot \sum_{i=1}^{K} \tilde{g}_i^{[t]} \\
&= \frac{q}{K} \cdot g^{[t]}
\end{aligned}$$

which completes the proof. □

Proof. [Lemma 4.3.2] Since $\mathbf{\Pi}$ is an orthonormal matrix, the solution of the least squares problem with the objective $L_{\mathbf{G}}(\mathbf{A}, \mathbf{b}; \mathbf{x})$ is equal to the optimal solution (4.1),

as

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{G}(\mathbf{A}\mathbf{x}-\mathbf{b})\|_2^2 \\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{\Pi}(\mathbf{A}\mathbf{x}-\mathbf{b})\|_2^2 \\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 \\ &= \mathbf{x}_{ls}^{\star}\ .\end{aligned}$$

□

Proof. [Corollary 4.3.1] We prove this by induction. From our assumptions we have a fixed starting point $\mathbf{x}^{[0]}$, for which $\hat{\mathbf{x}}^{[0]} = \mathbf{x}^{[0]}$. Our base case is therefore $\mathbb{E}[\hat{\mathbf{x}}^{[0]}] = \mathbb{E}[\mathbf{x}^{[0]}] = \mathbf{x}^{[0]}$. For the inductive hypothesis, we assume that $\mathbb{E}[\hat{\mathbf{x}}^{[\tau]}] = \mathbf{x}^{[\tau]}$ for $\tau \in \mathbb{N}$.

It then follows that at step $\tau + 1$ we have

$$\begin{aligned}\mathbb{E}\big[\hat{\mathbf{x}}^{[\tau+1]}\big] &= \mathbb{E}\big[\hat{\mathbf{x}}^{[\tau]} - \hat{\xi}_\tau \cdot \hat{g}^{[\tau]}\big] \\ &= \mathbb{E}\big[\hat{\mathbf{x}}^{[\tau]}\big] - \frac{K}{q}\cdot\xi_\tau\cdot\mathbb{E}\big[\hat{g}^{[\tau]}\big] \\ &= \mathbf{x}^{[\tau]} - \frac{q}{K}\cdot\left(\frac{K}{q}\cdot\xi_\tau\right)\cdot g_{ls}^{[\tau]} \\ &= \mathbf{x}^{[\tau]} - \xi_\tau\cdot g_{ls}^{[\tau]} \\ &= \mathbf{x}^{[\tau+1]}\end{aligned}$$

which completes the inductive step. □

3.1.2 Subsection 4.3.2

In this appendix, we provide the proofs of Lemma 4.3.3 and Theorem 4.3.2. First, we need to provide Lemmas 3.1.1 and 3.1.2, and Hoeffding's inequality; which we use to prove the latter Lemma. Throughout this subsection, by ℓ_i we denote the i^{th} leverage score of $\mathbf{\Pi A}$ for $\mathbf{\Pi}$ a random orthonormal matrix, *i.e.*

$$\ell_i = \|\tilde{\mathbf{U}}_{(i)}\|_2^2 = \|\mathbf{e}_i^T\tilde{\mathbf{U}}\|_2^2 = \mathbf{e}_i^T\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{e}_i \tag{3.1}$$

where $\tilde{\mathbf{U}} = \mathbf{\Pi U}$; for $\mathbf{U}$ the reduced left orthonormal matrix of $\mathbf{A}$. By $\mathbf{e}_i$ we denote the i^{th} standard basis vector of $\mathbb{R}^N$.

Lemma 3.1.1. *For each $i \in \mathbb{N}_N$, we have $\mathbb{E}[\ell_i] = \frac{d}{N}$.*

Proof. By (3.1), we have

$$\begin{aligned}
\mathbb{E}[\ell_i] &= \mathbb{E}\left[\mathrm{tr}(\mathbf{e}_i^T \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T \mathbf{e}_i)\right] \\
&= \mathbb{E}\left[\mathrm{tr}(\mathbf{e}_i \mathbf{e}_i^T \cdot \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T)\right] \\
&= \sum_{j=1}^{N} \frac{1}{N} \cdot \mathrm{tr}(\mathbf{e}_j \mathbf{e}_j^T \cdot \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T) \\
&= \frac{1}{N} \cdot \mathrm{tr}\left(\sum_{j=1}^{N} \mathbf{e}_j \mathbf{e}_j^T \cdot \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right) \\
&= \frac{1}{N} \cdot \mathrm{tr}\left(\mathbf{I}_N \cdot \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right) \\
&= \frac{1}{N} \cdot \mathrm{tr}\left(\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right) \\
&= \frac{d}{N} \ .
\end{aligned}$$

□

Let $\bar{\ell}_i$ denote the i^{th} normalized leverage score, *i.e.* $\bar{\ell}_i = \frac{\ell_i}{d}$. The ι^{th} normalized block leverage score of $\mathbf{A}$ is denoted by $\grave{\ell}_\iota$, *i.e.*

$$\grave{\ell}_\iota = \frac{1}{d} \cdot \|\mathbf{I}_{(\mathcal{K}_\iota)} \tilde{\mathbf{U}}\|_F^2 = \frac{1}{d} \cdot \Big(\sum_{j \in \mathcal{K}_\iota} \ell_j\Big) = \sum_{j \in \mathcal{K}_\iota} \bar{\ell}_j \ . \tag{3.2}$$

To prove Lemma 4.3.3, we first recall Hoeffding's inequality.

Theorem 3.1.1 (Hoeffding's Inequality, [201])**.** *Let $\{X_i\}_{i=1}^m$ be independent random variables such that $X_i \in [a_i, b_i]$ for all $i \in \mathbb{N}_m$, and let $X = \sum_{i=1}^m X_i$. Then*

$$\Pr\left[\left|X - \mathbb{E}[X]\right| \geqslant t\right] \leqslant 2 \cdot \exp\left\{\frac{-2t^2}{\sum_{j=1}^m (a_i - b_i)^2}\right\}.$$

Lemma 3.1.2. *The normalized leverage scores $\{\bar{\ell}_i\}_{i=1}^N$ of $\mathbf{\Pi A}$ satisfy*

$$\Pr\left[|\bar{\ell}_i - 1/N| < \rho\right] > 1 - 2 \cdot e^{-2\rho^2}$$

for any $\rho > 0$.

Proof. We know that $\ell_i \in [0, d]$ for each $i \in \mathbb{N}_N$, thus $\bar{\ell}_i \in [0, 1]$ for each i. By Lemma 3.1.1, it follows that

$$\mathbb{E}[\bar{\ell}_i] = \mathbb{E}[\ell_i/d] = \frac{1}{d} \cdot \mathbb{E}[\ell_i] = \frac{1}{N}.$$

Now, fix a constant $\rho > 0$. By applying Theorem 3.1.1 with $m = 1$, we get

$$\Pr\left[|\bar{\ell}_i - 1/N| \geqslant \rho\right] \leqslant 2 \cdot e^{-2\rho^2}$$

thus

$$\Pr\left[|\bar{\ell}_i - 1/N| < \rho\right] > 1 - 2 \cdot e^{-2\rho^2}.$$

□

Next, we complete the proof of the "flattening Lemma of block leverage scores" (Lemma 4.3.3).

Proof. [Lemma 4.3.3] To show that the two probability events of expression (4.13) are equal, note that:

1. $\grave{\ell}_\iota - \frac{1}{K} < \frac{N}{K}\rho \iff \grave{\ell}_\iota < (1 + N\rho)\frac{1}{K}$
2. $\frac{1}{K} - \grave{\ell}_\iota < \frac{N}{K}\rho \iff \grave{\ell}_\iota > (1 - N\rho)\frac{1}{K}$.

By combining the two inequalities, we conclude that

$$(1 - N\rho) \cdot \frac{1}{K} < \grave{\ell}_\iota < (1 + N\rho) \cdot \frac{1}{K} \iff \grave{\ell}_\iota <_{N\rho} 1/K. \tag{3.3}$$

By Lemma 3.1.2, it follows that

$$\begin{aligned}
\Pr\left[\left|\grave{\ell}_\iota - 1/K\right| < \tau\rho\right] &> \Pr\left[\bigwedge_{j \in \mathcal{K}_\iota} \left\{|\bar{\ell}_i - 1/N| < \rho\right\}\right] \\
&> \left(1 - 2 \cdot e^{-2\rho^2}\right)^\tau \\
&\overset{\bowtie}{\approx} 1 - 2\tau \cdot e^{-2\rho^2}
\end{aligned}$$

where in $\bowtie$ we applied the binomial approximation. By substituting $\rho \geqslant \sqrt{\log(2\tau/\delta)/2}$, we get

$$\begin{aligned}
e^{-2\rho^2} &\leqslant e^{-2\frac{\log(2\tau/\delta)}{2}} \\
&= e^{-\log(2\tau/\delta)} \\
&= e^{\log(\delta/2\tau)} \\
&= \delta/2\tau,
\end{aligned}$$

thus $2\tau \cdot e^{-2\rho^2} \leqslant \delta$; and $1 - 2\tau \cdot e^{-2\rho^2} \geqslant 1 - \delta$. In turn, this implies that $\Pr\left[\left|\grave{\ell}_\iota - 1/K\right| < \tau\rho\right] > 1 - \delta$. □

The proof of Theorem 4.3.2 is a direct consequence of Lemma 4.3.3 and Theorem 3.1.2. In our statement we make the assumption that $\grave{\ell}_\iota = 1/K$ for all ι, though this is not necessarily the case, as Lemma 4.3.3 permits a small deviation. For $\rho \leftarrow \epsilon$, we consider $\epsilon \ll 1/N$ so that the '$N\epsilon$ multiplicative error' in (3.3) is small. We note that [56, Theorem 1] considers sampling according to *approximate* block leverage scores.

Theorem 3.1.2 (ℓ_2-s.e. of the block leverage score sampling sketch, [56])**.** *The sketching matrix $\widetilde{\mathbf{S}}$ constructed by sampling blocks of $\mathbf{A}$ with replacement according to their normalized block leverage scores $\{\grave{\ell}_\iota\}_{\iota=1}^K$ and rescaling each sampled block by $\sqrt{1/\left(q\grave{\ell}_\iota\right)}$, guarantees a ℓ_2-s.e. of $\mathbf{A}$; as defined in (4.9). Specifically, for $\delta > 0$ and $q = \Theta(\frac{d}{\tau}\log(2d/\delta)/\epsilon^2)$:*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^T\widetilde{\mathbf{S}}^T\widetilde{\mathbf{S}}\mathbf{U}\|_2 \leqslant \epsilon\right] \geqslant 1-\delta.$$

Before we prove Proposition 4.3.1, we first derive (4.6). In [59], the optimal decoding vector of an approximate GCS was defined as

$$\mathbf{a}_{\mathcal{I}}^\star = \arg\min_{\mathbf{a}\in\mathbb{R}^{1\times q}} \left\{\|\mathbf{a}\mathbf{G}_{(\mathcal{I})} - \vec{\mathbf{1}}\|_2^2\right\}. \tag{3.4}$$

In the case where $q \geqslant K$, it follows that $\mathbf{a}_{\mathcal{I}}^\star = \vec{\mathbf{1}}\mathbf{G}_{(\mathcal{I})}^\dagger$. The error can then be quantified as

$$\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) \coloneqq \|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^\dagger\mathbf{G}_{(\mathcal{I})}\|_2 .$$

The optimal decoding vector (3.4) has also been considered in other schemes, *e.g.* [155, 253].

Let $\mathbf{g}^{[t]}$ be the matrix comprised of the transposed exact partial gradients at iteration t, *i.e.*

$$\mathbf{g}^{[t]} \coloneqq \begin{pmatrix} g_1^{[t]} & g_2^{[t]} & \dots & g_K^{[t]} \end{pmatrix}^T \in \mathbb{R}^{K\times d} .$$

Then, for a GCS $(\mathbf{G}, \mathbf{a}_{\mathcal{I}})$ satisfying $\mathbf{a}_{\mathcal{I}}\mathbf{G}_{(\mathcal{I})} = \vec{\mathbf{1}}$ for any $\mathcal{I}$, it follows that $\left(\mathbf{a}_{\mathcal{I}}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[t]} = \vec{\mathbf{1}}\mathbf{g}^{[t]} = \left(g^{[t]}\right)^T$. Hence, the gradient can be recovered exactly. Considering an optimal approximate scheme $(\mathbf{G}, \mathbf{a}_{\mathcal{I}}^\star)$ which recovers the gradient estimate $\grave{g}^{[t]} = \left(\mathbf{a}_{\mathcal{I}}^\star\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}$,

the error in the gradient approximation is

$$\begin{aligned}
\left\|g^{[s]} - \grave{g}^{[s]}\right\|_2 &= \left\|\left(\vec{\mathbf{1}} - \mathbf{a}_{\mathcal{I}}^{\star}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}\right\|_2 \\
&= \left\|\vec{\mathbf{1}}\left(\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right)\mathbf{g}^{[s]}\right\|_2 \\
&\leqslant \|\vec{\mathbf{1}}\|_2 \cdot \left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2 \cdot \left\|\mathbf{g}^{[s]}\right\|_2 \\
&\overset{\pounds}{\leqslant} \sqrt{K} \cdot \left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2 \cdot \left\|g^{[s]}\right\|_2 \\
&\overset{\$}{\leqslant} 2\sqrt{K} \cdot \underbrace{\left\|\mathbf{I}_K - \mathbf{G}_{(\mathcal{I})}^{\dagger}\mathbf{G}_{(\mathcal{I})}\right\|_2}_{\mathrm{err}(\mathbf{G}_{(\mathcal{I})})} \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2
\end{aligned}$$

where $\pounds$ follows from the facts that $\|\mathbf{g}^{[s]}\|_2 \leqslant \|g^{[s]}\|_2$ and $\|\vec{\mathbf{1}}\|_2 = \sqrt{K}$, and \$ from (4.2) and sub-multiplicativity of matrix norms. This concludes the derivation of (4.6).

Proof. [Proposition 4.3.1] Let $\hat{g}^{[t]}$ be the approximated gradient of our scheme at iteration t. Since we are considering linear regression, it follows that

$$\begin{aligned}
\left\|g^{[t]} - \hat{g}^{[t]}\right\|_2 &= 2\|\mathbf{A}^T(\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}) - \mathbf{A}^T(\mathbf{S}_{\mathbf{\Pi}}^T\mathbf{S}_{\mathbf{\Pi}})(\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b})\|_2 \\
&= 2\|\mathbf{A}^T(\mathbf{I}_N - \mathbf{S}_{\mathbf{\Pi}}^T\mathbf{S}_{\mathbf{\Pi}})(\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b})\|_2 \\
&\leqslant 2\|\mathbf{A}\|_2 \cdot \|\mathbf{I}_N - \mathbf{S}_{\mathbf{\Pi}}^T\mathbf{S}_{\mathbf{\Pi}}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\|_2 \\
&= 2\|\mathbf{A}\|_2 \cdot \|\mathbf{U}^T(\mathbf{I}_N - \mathbf{S}_{\mathbf{\Pi}}^T\mathbf{S}_{\mathbf{\Pi}})\mathbf{U}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2 \\
&= 2\|\mathbf{A}\|_2 \cdot \|\mathbf{I}_d - \mathbf{U}^T\mathbf{S}_{\mathbf{\Pi}}^T\mathbf{S}_{\mathbf{\Pi}}\mathbf{U}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[s]} - \mathbf{b}\|_2 \\
&\overset{\flat}{\leqslant} 2\epsilon \cdot \|\mathbf{A}\|_2 \cdot \|\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\|_2
\end{aligned}$$

where in $\flat$ we invoked the fact that $\mathbf{S}_{\mathbf{\Pi}}$ satisfies (4.9). Our approximate GC approach therefore (w.h.p.) satisfies (4.6), with $\mathrm{err}(\mathbf{G}_{(\mathcal{I})}) = \epsilon/\sqrt{K}$ □

3.2 Proofs of Section 4.4

In this appendix, we present two lemmas which we use to bound the entries of $\hat{\mathbf{V}} \coloneqq \hat{\mathbf{H}}\mathbf{D}\mathbf{U}$, and its *leverage scores* $\ell_i \coloneqq \|\hat{\mathbf{V}}_{(i)}\|_2^2$, for which $\sum_{i=1}^N \ell_i = d$. Leverage scores induce a sampling distribution which has proven to be useful in linear regression [91, 294, 201, 290] and GC [52]. From these lemmas, we deduce that the leverage scores of $\hat{\mathbf{H}}\mathbf{D}\mathbf{A}$ are close to being uniform, implying that the *block leverage scores*[221, 52] are also uniform, which is precisely what Lemma 3.2.3 states.

Lemma 3.2.2 is a variant of the Flattening Lemma [5, 201], a key result to

Hadamard based sketching algorithms, which justifies uniform sampling. In the proof, we make use of the Azuma-Hoeffding inequality; a concentration result for the values of martingales that have bounded differences. We also recall a matrix Chernoff bound, which we apply to prove our ℓ_2-s.e. guarantees. Finally, we present proofs of Proposition 3.2.1 and Theorems 4.3.1, 4.4.1.

Lemma 3.2.1 (Azuma-Hoeffding Inequality, [201]). *For zero mean random variable Z_i (or $Z_0, Z_1, \cdots, Z_m$ a martingale sequence of random variables), bounded above by $|Z_i| \leqslant \beta_i$ for all i with probability 1, we have*

$$\Pr\left[\left|\sum_{j=0}^{m} Z_j\right| > t\right] \leqslant 2\exp\left\{\frac{-t^2}{2\cdot\left(\sum_{j=0}^{m}\beta_j^2\right)}\right\}.$$

Theorem 3.2.1 (Matrix Chernoff Bound, [294, Fact 1]). *Let $\mathbf{X}_1, \cdots, \mathbf{X}_q$ be independent copies of a symmetric random matrix $\mathbf{X} \in \mathbb{R}^{d\times d}$, with $\mathbb{E}[\mathbf{X}] = 0, \|\mathbf{X}\|_2 \leqslant \gamma$, $\|\mathbb{E}[\mathbf{X}^T\mathbf{X}]\|_2 \leqslant \sigma^2$. Let $\mathbf{Z} = \frac{1}{q}\sum_{i=1}^{q}\mathbf{X}_i$. Then, $\forall\epsilon > 0$:*

$$\Pr\left[\|\mathbf{Z}\|_2 > \epsilon\right] \leqslant 2d\cdot\exp\left(-\frac{q\epsilon^2}{\sigma^2 + \gamma\epsilon/3}\right). \tag{3.5}$$

Lemma 3.2.2 (Flattening Lemma). *For $\mathbf{y} \in \mathbb{R}^N$ a fixed (orthonormal) column vector of $\mathbf{U}$, and $\mathbf{D} \in \{0, \pm 1\}^{N\times N}$ with random equi-probable diagonal entries of ± 1, we have:*

$$\Pr\left[\|\hat{\mathbf{H}}\mathbf{D}\cdot\mathbf{y}\|_\infty > C\sqrt{\log(Nd/\delta)/N}\right] \leqslant \frac{\delta}{2d} \tag{3.6}$$

for $0 < C \leqslant \sqrt{2 + \log(16)/\log(Nd/\delta)}$ a constant.

Proof. [Lemma 3.2.2] Fix i and define $Z_j = \hat{\mathbf{H}}_{ij}\mathbf{D}_{jj}\mathbf{y}_j$ for each $j \in \mathbb{N}_N$, which are independent random variables. Since $\mathbf{D}_{jj} = \vec{D}_j$ are i.i.d. entries with zero mean, so are Z_j. Furthermore $|Z_j| \leqslant |\hat{\mathbf{H}}_{ij}|\cdot|\mathbf{D}_{jj}|\cdot|\mathbf{y}_j| = \frac{|\mathbf{y}_j|}{\sqrt{N}}$, and note that

$$\sum_{j=1}^{N} Z_j = (\hat{\mathbf{H}}\mathbf{D}\mathbf{y})_i = \sum_{j=1}^{N}\hat{\mathbf{H}}_{ij}\mathbf{D}_{jj}\mathbf{y}_j = \langle\hat{\mathbf{H}}_{(i)}\odot\overbrace{\mathrm{diag}(\mathbf{D})}^{\vec{D}}, \mathbf{y}\rangle$$

where $\odot$ is the Hadamard product. By Lemma 3.2.1

$$\Pr\left[\Big|\sum_{j=1}^{N} Z_j\Big| > \rho\right] \leqslant 2\exp\left\{\frac{-\rho^2/2}{\sum_{j=1}^{N}(\mathbf{y}_j/\sqrt{N})^2}\right\}$$
$$= 2\exp\left\{\frac{-N\rho^2}{2\cdot\langle\mathbf{y},\mathbf{y}\rangle}\right\} \overset{\flat}{=} 2\cdot e^{-N\rho^2/2} \tag{3.7}$$

where $\flat$ follows from the fact that $\mathbf{y}$ is a column of $\mathbf{U}$. By setting $\rho = C\sqrt{\frac{\log(Nd/\delta)}{N}}$, we get

$$\Pr\left[\Big|\sum_{j=1}^{N} Z_j\Big| > C\sqrt{\frac{\log(Nd/\delta)}{N}}\right] \leqslant 2\exp\left\{-\frac{C^2\log(Nd/\delta)}{2}\right\}$$
$$= 2\left(\frac{\delta}{Nd}\right)^{C^2/2} \overset{\natural}{\leqslant} \frac{\delta}{2Nd}$$

where $\natural$ follows from the upper bound on C. By applying the union bound over all $i \in \mathbb{N}_N$, we attain (3.6). □

Lemma 3.2.3. *For all $i \in \mathbb{N}_N$ and $\{\mathbf{e}_i\}_{i=1}^N$ the standard basis:*

$$\Pr\left[\sqrt{\ell_i} \leqslant C\sqrt{d\log(Nd/\delta)/N}\right] \geqslant 1-\delta/2$$

for $\ell_i = \|\hat{\mathbf{V}}_{(i)}\|_2^2$ the i^{th} leverage score of $\hat{\mathbf{V}} = \hat{\mathbf{H}}\mathbf{D}\mathbf{U}$.

Proof. [Lemma 3.2.3] It is straightforward that the columns of $\hat{\mathbf{V}}$ form an orthonormal basis of $\mathbf{A}$, thus Lemma 3.2.2 implies that for $j \in \mathbb{N}_d$

$$\Pr\left[\|\hat{\mathbf{V}}\cdot\mathbf{e}_j\|_\infty > C\sqrt{\log(Nd/\delta)/N}\right] \leqslant \frac{\delta}{2d}\ .$$

By applying the union bound over all entries of $\hat{\mathbf{V}}^{(j)} = \hat{\mathbf{V}}\cdot\mathbf{e}_j$

$$\Pr\left[\overbrace{|\mathbf{e}_i^T\cdot\hat{\mathbf{V}}\cdot\mathbf{e}_j|}^{|(\hat{\mathbf{H}}\mathbf{D}\mathbf{U})_{ij}|} > C\sqrt{\frac{\log(Nd/\delta)}{N}}\right] \leqslant d\cdot\frac{\delta}{2d} = \delta/2\ . \tag{3.8}$$

We manipulate the argument of the above bound to obtain

$$\|\mathbf{e}_i^T\cdot\hat{\mathbf{V}}\|_2 = \Big(\sum_{j=1}^{d}(\hat{\mathbf{H}}\mathbf{D}\mathbf{U})_{ij}^2\Big)^{1/2} > C\sqrt{d\cdot\frac{\log(Nd/\delta)}{N}}\ ,$$

which can be viewed as a scaling of the random variable entries of $\hat{\mathbf{V}}$. The probability of the complementary event is therefore

$$\Pr\left[\|\mathbf{e}_i^T \cdot \hat{\mathbf{V}}\|_2 \leqslant C\sqrt{d\log(Nd/\delta)/N}\right] \geqslant 1 - \delta/2$$

and the proof is complete. □

Remark 3.2.1. *The complementary probable event of* (3.8) *can be interpreted as 'every entry of* $\hat{\mathbf{V}}$ *is small in absolute value'.*

Proof. [Lemma 4.4.1] For $\alpha \coloneqq \eta d \cdot \log(Nd/\delta)/N$

$$\Pr\left[\tilde{\ell}_\iota \leqslant \tau \cdot \alpha\right] > \Pr\left[\{\ell_j \leqslant \alpha : \forall j \in \mathcal{K}_\iota\}\right] \overset{\diamondsuit}{>} (1 - \delta/2)^\tau$$

where $\eta = C^2$ and $\diamondsuit$ follows from Lemma 3.2.3. By the binomial approximation, we have $(1 - \delta/2)^\tau \approx 1 - \tau\delta/2$. □

Define the symmetric matrices

$$\mathbf{X}_i = \left(\mathbf{I}_d - \frac{N}{\tau} \cdot \hat{\mathbf{V}}_{(\mathcal{K}^i)}^T \hat{\mathbf{V}}_{(\mathcal{K}^i)}\right) = \left(\mathbf{I}_d - K \cdot \hat{\mathbf{V}}_{(\mathcal{K}^i)}^T \hat{\mathbf{V}}_{(\mathcal{K}^i)}\right) \tag{3.9}$$

where $\hat{\mathbf{V}}_{(\mathcal{K}^i)} = \hat{\mathbf{V}}_{(\mathcal{K}_\iota)}$ is the submatrix of $\hat{\mathbf{V}}$ corresponding to the i^{th} sampling trial of our algorithm. Let $\mathbf{X}$ be the matrix r.v. of which the $\mathbf{X}_i$'s are independent copies. Note that the realizations $\mathbf{X}_i$ of $\mathbf{X}$ correspond to the sampling blocks of the event in (4.9). To apply Theorem 3.2.1, we show that the $\mathbf{X}_i$'s have zero mean, and we bound

their ℓ_2-norm and variance. Their ℓ_2-norms are upper bounded by

$$\begin{aligned}
\|\mathbf{X}_i\|_2 &\leqslant \|\mathbf{I}_d\|_2 + \|\frac{N}{\tau} \cdot \hat{\mathbf{V}}_{(\mathcal{K}^i)}^T \hat{\mathbf{V}}_{(\mathcal{K}^i)}\|_2 \\
&= 1 + \frac{N}{\tau} \cdot \|\hat{\mathbf{V}}_{(\mathcal{K}_\iota)}\|_2^2 \\
&\leqslant 1 + \frac{N}{\tau} \cdot \max_{\iota \in \mathbb{N}_K} \left\{ \|\mathbf{I}_{(\mathcal{K}_\iota)} \cdot \hat{\mathbf{V}}\|_2^2 \right\} \\
&\leqslant 1 + \frac{N}{\tau} \cdot \max_{\iota \in \mathbb{N}_K} \left\{ \|\mathbf{I}_{(\mathcal{K}_\iota)} \cdot \hat{\mathbf{V}}\|_F^2 \right\} \\
&\overset{\$}{\leqslant} 1 + \frac{N}{\tau} \cdot \left(|\mathcal{K}_\iota| \cdot \max_{j \in \mathbb{N}_N} \left\{ \|\mathbf{e}_j^T \cdot \hat{\mathbf{V}}\|_2^2 \right\} \right) \\
&\leqslant 1 + \frac{N}{\tau} \cdot \left(\tau \cdot (\eta \cdot d \log(Nd/\delta)/N) \right) && \text{[Lemma 3.2.2]} \\
&= 1 + \eta \cdot d \log(Nd/\delta) && (3.10) \\
&= 1 + N\alpha
\end{aligned}$$

for $\alpha = \eta d \cdot \log(Nd/\delta)/N$ where in \$ we used the fact that

$$\|\mathbf{I}_{(\mathcal{K}_\iota)} \cdot \hat{\mathbf{V}}\|_F^2 = \sum_{j \in \mathcal{K}_\iota} \|\mathbf{e}_j^T \cdot \hat{\mathbf{V}}\|_2^2 \leqslant |\mathcal{K}_\iota| \cdot \max_{j \in \mathcal{K}_\iota} \left\{ \|\mathbf{e}_j^T \cdot \hat{\mathbf{V}}\|_2^2 \right\} \ .$$

From the above derivation, it follows that

$$\begin{aligned}
\|\hat{\mathbf{V}}_{(\mathcal{K}^i)}\|_2^2 &= \|\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T \hat{\mathbf{V}}_{(\mathcal{K}^i)}\|_2 \\
&\leqslant \frac{\tau}{N} \cdot (1 + \eta \cdot d \log(Nd/\delta) - \|\mathbf{I}_d\|_2) \\
&= \tau \eta d/N \cdot \log(Nd/\delta) \\
&= \tau\alpha
\end{aligned}$$

for all $\iota \in \mathbb{N}_K$. By setting $\tau = 1$, we get an upper bound on the squared ℓ_2-norm of the rows of $\hat{\mathbf{V}}$:

$$\|\hat{\mathbf{V}}_l\|_2^2 = \|\hat{\mathbf{V}}_l \hat{\mathbf{V}}_l^T\|_2 = \|\hat{\mathbf{V}}_l^T \hat{\mathbf{V}}_l\|_2 \leqslant \alpha \tag{3.11}$$

where $\hat{\mathbf{V}}_l = \hat{\mathbf{V}}_{(l)}$, for all $l \in \mathbb{N}_N$.

Next, we compute $\mathbf{E} \coloneqq \mathbb{E}[\mathbf{X}^T\mathbf{X} + \mathbf{I}_d]$ and its eigenvalues. By the definition of $\mathbf{X}$

and its realizations:

$$\begin{aligned}\mathbf{X}_i^T\mathbf{X}_i &= \left(\mathbf{I}_d - N/\tau \cdot \hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\right)^T \cdot \left(\mathbf{I}_d - N/\tau \cdot \hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\right) \\ &= \mathbf{I}_d - 2\cdot\frac{N}{\tau}\cdot\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)} + \left(\frac{N}{\tau}\right)^2\cdot\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\end{aligned}$$

thus $\mathbf{E}$ is evaluated as follows:

$$\begin{aligned}\mathbb{E}[\mathbf{X}^T\mathbf{X} + \mathbf{I}_d] &= 2\mathbf{I}_d - 2\cdot(N/\tau)\cdot\mathbb{E}\left[\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\right] + (N/\tau)^2\cdot\mathbb{E}\left[\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\hat{\mathbf{V}}_{(\mathcal{K}^i)}^T\hat{\mathbf{V}}_{(\mathcal{K}^i)}\right] \\ &= 2\mathbf{I}_d - 2\cdot(N/\tau)\cdot\left(\textstyle\sum_{j=1}^K K^{-1}\cdot\hat{\mathbf{V}}_{(\mathcal{K}_j)}^T\hat{\mathbf{V}}_{(\mathcal{K}_j)}\right) + \\ &\qquad + (N/\tau)^2\cdot\left(\textstyle\sum_{j=1}^K K^{-1}\cdot\hat{\mathbf{V}}_{(\mathcal{K}_j)}^T\left(\hat{\mathbf{V}}_{(\mathcal{K}_j)}\hat{\mathbf{V}}_{(\mathcal{K}_j)}^T\right)\hat{\mathbf{V}}_{(\mathcal{K}_j)}\right) \\ &= 2\mathbf{I}_d - 2\cdot\left(\textstyle\sum_{l=1}^N\hat{\mathbf{V}}_l^T\hat{\mathbf{V}}_l\right) + (N/\tau)\cdot\left(\textstyle\sum_{l=1}^N\hat{\mathbf{V}}_l^T\left(\hat{\mathbf{V}}_l\hat{\mathbf{V}}_l^T\right)\hat{\mathbf{V}}_l\right) \\ &= K\cdot\left(\textstyle\sum_{l=1}^N\langle\hat{\mathbf{V}}_l,\hat{\mathbf{V}}_l\rangle\cdot\hat{\mathbf{V}}_l^T\hat{\mathbf{V}}_l\right)\end{aligned}$$

where in the last equality we invoked $\sum_{l=1}^N\hat{\mathbf{V}}_l^T\hat{\mathbf{V}}_l = \mathbf{I}_d$.

In order to bound the variance of the matrix random variable $\mathbf{X}$, we bound the largest eigenvalue of $\mathbf{E}$; by comparing it to the matrix

$$\mathbf{F} = K\alpha\cdot\left(\sum_{l=1}^N\hat{\mathbf{V}}_l^T\hat{\mathbf{V}}_l\right) = K\alpha\cdot\mathbf{I}_d$$

whose eigenvalue $K\alpha$ is of algebraic multiplicity d. It is clear that $\mathbf{E}$ and $\mathbf{F}$ are both real and symmetric; thus they admit an eigendecomposition of the form $\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. Note

also that for all $\mathbf{y} \in \mathbb{R}^d$:

$$\begin{aligned}
\mathbf{y}^T\mathbf{E}\mathbf{y} &= K \cdot \mathbf{y}^T \left(\sum_{l=1}^{N} \hat{\mathbf{V}}_l^T \left(\hat{\mathbf{V}}_l \hat{\mathbf{V}}_l^T \right) \hat{\mathbf{V}}_l \right) \mathbf{y} \\
&\overset{\sharp}{=} K \cdot \sum_{l=1}^{N} \langle \mathbf{y}, \hat{\mathbf{V}}_l \rangle^2 \cdot \|\hat{\mathbf{V}}_l\|_2^2 \\
&\overset{\flat}{\leqslant} K\alpha \cdot \sum_{l=1}^{N} \langle \mathbf{y}, \hat{\mathbf{V}}_l \rangle^2 \\
&= K\alpha \cdot \sum_{l=1}^{N} \mathbf{y}^T \hat{\mathbf{V}}_l^T \cdot \hat{\mathbf{V}}_l \mathbf{y} \\
&= \mathbf{y}^T \left(K\alpha \cdot \sum_{l=1}^{N} \hat{\mathbf{V}}_l^T \cdot \hat{\mathbf{V}}_l \right) \mathbf{y} \\
&= \mathbf{y}^T \mathbf{F} \mathbf{y}
\end{aligned} \tag{3.13}$$

where in $\flat$ we invoked (3.11). By $\sharp$ we conclude that $\mathbf{y}^T\mathbf{E}\mathbf{y} \geqslant 0$, thus $\mathbf{F} \succeq \mathbf{E} \succeq 0$.

Let $\mathbf{w}_i, \mathbf{z}_i$ be the unit-norm eigenvectors of $\mathbf{E}, \mathbf{F}$ corresponding to their respective i^{th} largest eigenvalue. Then

$$\mathbf{w}_i^T \left(\mathbf{Q}_\mathbf{E} \mathbf{\Lambda}_\mathbf{E} \mathbf{Q}_\mathbf{E}^T \right) \mathbf{w}_i = \mathbf{e}_i^T \cdot \mathbf{\Lambda}_\mathbf{E} \cdot \mathbf{e}_i = \lambda_i$$

and by (3.13) we bound this as follows:

$$\lambda_i = \mathbf{w}_i^T \mathbf{E} \mathbf{w}_i \leqslant K\alpha \cdot \sum_{l=1}^{N} \langle \mathbf{w}_i, \hat{\mathbf{V}}_l \rangle^2 \ .$$

Since

$$\mathbf{w}_1 = \arg \max_{\substack{\mathbf{v} \in \mathbb{R}^d \\ \|\mathbf{v}\|_2 = 1}} \left\{ \mathbf{v}^T \mathbf{E} \mathbf{v} \right\} \implies \|\mathbf{E}\|_2 = \lambda_1 = \mathbf{w}_1^T \mathbf{E} \mathbf{w}_1 \ ,$$

and $\mathbf{F} \succeq \mathbf{E} \geqslant 0$, it follows that

$$\|\mathbf{E}\|_2 = \mathbf{w}_1^T \mathbf{E} \mathbf{w}_1 \leqslant \mathbf{w}_1^T \mathbf{F} \mathbf{w}_1 \leqslant \arg \max_{\substack{\mathbf{v} \in \mathbb{R}^d \\ \|\mathbf{v}\|_2 = 1}} \left\{ \mathbf{v}^T \mathbf{F} \mathbf{v} \right\} = \|\mathbf{F}\|_2 = K\alpha \ .$$

In turn, this gives us

$$\begin{aligned}\|\mathbb{E}[\mathbf{X}^T\mathbf{X}]\|_2 &= \|\mathbf{E} - \mathbf{I}_d\|_2 \\ &\leqslant \|\mathbf{E}\|_2 + \|\mathbf{I}_d\|_2 \\ &\leqslant \|\mathbf{F}\|_2 + 1 \\ &= K\alpha + 1 \\ &\leqslant \eta K \frac{d}{N}\log(Nd/\delta) + 1 \\ &= \eta\frac{d}{\tau}\log(Nd/\delta) + 1 \end{aligned} \tag{3.14}$$

hence $\|\mathbb{E}[\mathbf{X}^T\mathbf{X}]\|_2 = O\big(\frac{d}{\tau}\log(Nd/\delta)\big)$.

We now have everything we need to apply Theorem 3.2.1.

Proposition 3.2.1. *The block-SRHT* $\mathbf{S}_{\hat{\mathbf{\Pi}}}$ *guarantees*

$$\Pr\left[\|\mathbf{I}_d - \mathbf{U}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}\mathbf{U}\|_2 > \epsilon\right] \leqslant 2d\cdot\exp\left\{\frac{-\epsilon^2\cdot q}{\Theta\left(\frac{d}{\tau}\cdot\log(Nd/\delta)\right)}\right\}$$

for any $\epsilon > 0$, *and* $q = r/\tau > d/\tau$.

Proof. [Proposition 3.2.1] Let $\{\mathbf{X}_i\}_{i=1}^q$ as defined in (3.9) denote q block samples. Let $j(i)$ denote the index of the submatrix which was sampled at the i^{th} random trial, *i.e.* $\mathcal{K}_{j(i)} = \mathcal{K}^i_{j(i)}$. Then

$$\begin{aligned}\mathbf{Z} &= \frac{1}{q}\sum_{i=1}^{t}\mathbf{X}_{j(i)} \\ &= \frac{1}{q}\cdot\sum_{i=1}^{q}\left(\mathbf{I}_d - \frac{N}{\tau}\cdot\hat{\mathbf{V}}^T_{(\mathcal{K}_{j(i)})}\hat{\mathbf{V}}_{(\mathcal{K}_{j(i)})}\right) \\ &= \mathbf{I}_d - \sum_{i=1}^{q}\left(\sqrt{N/r}\cdot\hat{\mathbf{V}}_{(\mathcal{K}_{j(i)})}\right)^T\cdot\left(\sqrt{N/r}\cdot\hat{\mathbf{V}}_{(\mathcal{K}_{j(i)})}\right) \\ &= \mathbf{I}_d - \sum_{i=1}^{q}\left(\sqrt{N/r}\cdot\mathbf{I}_{(\mathcal{K}_{j(i)})}\cdot\hat{\mathbf{V}}\right)^T\cdot\left(\sqrt{N/r}\cdot\mathbf{I}_{(\mathcal{K}_{j(i)})}\cdot\hat{\mathbf{V}}\right) \\ &= \mathbf{I}_d - \left(\widetilde{\mathbf{\Omega}}\hat{\mathbf{H}}\mathbf{D}\mathbf{U}\right)^T\cdot\left(\widetilde{\mathbf{\Omega}}\hat{\mathbf{H}}\mathbf{D}\mathbf{U}\right) \\ &= \mathbf{I}_d - \mathbf{U}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}^T\mathbf{S}_{\hat{\mathbf{\Pi}}}\mathbf{U}\ .\end{aligned}$$

We apply Lemma 3.2.1 by fixing the terms we bounded: (3.10) $\gamma = \eta d\log(Nd/\delta)+1$, (3.14) $\sigma^2 = \eta\frac{d}{\tau}\log(Nd/\delta)+1$, and fix q and ϵ. The denominator of the exponent

in (3.5) is then

$$\begin{aligned}\big(\eta d/\tau \cdot \log(Nd/\delta) + 1\big) + \big((\eta d \log(Nd/\delta) + 1) \cdot \epsilon/3\big) = \\ = \eta d/\tau \cdot \log(Nd/\delta) \cdot \big(1 + \epsilon\tau/3\big) + (1 + \epsilon/3) \\ = \Theta\left(\frac{d}{\tau}\log(Nd/\delta)\right)\end{aligned}$$

and the proof is complete. □

Proof. [Theorem 4.4.1] By substituting q in the bound of Proposition 3.2.1 and taking the complementary event, we attain the statement. □

3.2.1 The Hadamard Transform

Remark 3.2.2. *The Hadamard matrix is a real analog of the discrete Fourier matrix, and there exist matrix multiplications algorithms for the Hadamard transform which resemble the FFT algorithm. Recall that the Fourier matrix represents the characters of the cyclic group of order N. In this case, $\hat{\mathbf{H}}_N$ represents the characters of the group $(\mathbb{Z}_2^N, +)$, where $\mathbb{Z}_2^n \cong \mathbb{Z}_N$. For both of these transforms, it is precisely through this algebraic structure which one can achieve a matrix-vector multiplication in $\mathcal{O}(N \log N)$ arithmetic operations.*

Recall that the characters of a group G, form an orthonormal basis of the vector space of functions over the Boolean hypercube, *i.e.* $\mathcal{F}_n = \{f : \{0,1\}^n \to \mathbb{R}\}$, and it is the Fourier basis. Furthermore, when working over groups of characteristic 2, *e.g.* $\mathbb{F}_{2^q} \cong \mathbb{F}_2^q$ for $q \in \mathbb{Z}_+$, we can move everything so that the underlying field is $\mathbb{R}$. Specifically, we map the elements of the binary field to $\mathbb{R}$ by applying $f(y) = 1 - 2y$. This gives us $f : \{0,1\} \mapsto \{+1,-1\} \subseteq \mathbb{R}$, and we can work with addition and multiplication over $\mathbb{R}$.

We note that there is a bijective correspondence between the characters of $\mathbb{Z}_m$ and the m^{th} root of unity, which is precisely how we get an orthonormal (Fourier) basis. In the case where m is not a power of two, we have a basis with complex elements, which violates (c) in the list of properties we seek to satisfy. This is why the Hadamard matrix is appropriate for our application, and why we do not consider a general discrete Fourier transform.

3.2.2 Recursive Kronecker Products of Orthonormal Matrices

In this subsection, we show that multiplying a vector of length N with $\mathbf{\Pi} = \mathbf{\Pi}_k^{\otimes \lceil \log_k(N) \rceil}$ for $\mathbf{\Pi}_k \in O_k(\mathbb{R})$ and $k \in \mathbb{Z}_{>2}$, takes $\mathcal{O}(Nk^2 \log_k N)$ elementary operations. Therefore, multiplying $\mathbf{A} \in \mathbb{R}^{N \times d}$ with $\mathbf{\Pi}$ takes $\mathcal{O}(Ndk^2 \log_k N)$ operations. We follow a similar analysis to that of [220, Section 6.10.2].

For $C(N)$ the number of elementary operations involved in carrying out the above matrix-vector multiplication, the basic recursion relationship is

$$C(N) = k(C/k) + NC(1) \tag{3.15}$$

where $C(1) = \zeta k^2$, for $\zeta > 0$ a constant.

For $p = \lceil \log_k(N) \rceil$, we have the following relationship:

$$T(p) = \frac{C(N)}{N} \quad \Longrightarrow \quad C(N) = NT(p). \tag{3.16}$$

Then, $p - 1 = \lceil \log_k(N/k) \rceil$, which gives us

$$T(p-1) = \frac{C(N/k)}{N/k} = k\frac{C(N/k)}{N} \quad \Longrightarrow \quad NT(p-1) = kC(N/k). \tag{3.17}$$

By substituting (3.17) into (3.15), we get

$$C(N) = NT(p) = NT(p-1) + NC(1),$$

thus $T(p) = T(p-1) + C(1)$, which implies that $T(p)$ is linear. Therefore $T(p) = pC(1) = \zeta k^2 p$, and from (3.16) we conclude that the total number of elementary operations is

$$C(N) = NT(p) = N\zeta k^2 p = N\zeta k^2 \lceil \log_k(N) \rceil = \mathcal{O}(Nk^2 \log_k N).$$

3.3 Proofs of Section 4.5

In this appendix, we present the proofs of Proposition 4.5.1 and Corollary 4.5.1.

Proof. [Proposition 4.5.1] Note that the optimization problem (4.14) is equivalent to

$$\xi_t^\star = \arg\min_{\xi \in \mathbb{R}} \left\{ \|\mathbf{A}\mathbf{x}^{[t+1]} - \mathbf{b}\|_2^2 \right\}. \tag{3.18}$$

If we cannot decrease further, the optimal solution to (3.18) will be 0, and we can never have $\xi_t < 0$, as this would imply that

$$\|\mathbf{A}\mathbf{x}^{[t+1]} - \mathbf{b}\|_2^2 = \|\mathbf{A}(\mathbf{x}^{[t]} - \xi_t \cdot g^{[t]})\mathbf{b}\|_2^2 > \|\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\|_2^2$$

which contradicts the fact that we are minimizing the objective function of (3.18). Specifically, if $\xi_t < 0$, we get an ascent step in (4.3), and a step-size $\xi_t = 0$ achieves a lower value. It therefore suffices to prove the given statement by solving (3.18).

We will first derive (4.15) for $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$, and then show it is the same for the optimization problems $L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ and $L_{\mathbf{G}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$.

Recall that $\mathbf{x}^{[t+1]} \leftarrow \mathbf{x}^{[t]} - \xi_t \cdot g_{ls}^{[t]}$ for the least squares objective $L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$. From here onward, we denote the gradient update of the underlying objective function by g_t. We then reformulate the objective function of (4.14) as follows

$$\Delta_{t+1} \coloneqq \|\mathbf{A}\mathbf{x}^{[t+1]} - \mathbf{b}\|_2^2 = \|\mathbf{A}(\mathbf{x}^{[t]} - \xi \cdot g^{[t]}) - \mathbf{b}\|_2^2 \,.$$

By expanding the above expression, we get

$$\begin{aligned}\Delta_{t+1} = \xi^2 \cdot \|\mathbf{A}g_t\|_2^2 - 2\xi \cdot \left(g_t^T \mathbf{A}^T \mathbf{A}\mathbf{x}^{[t]} - g_t^T \mathbf{A}^T \mathbf{b}\right) + \\ + \left(\|\mathbf{A}\mathbf{x}^{[t]}\|_2^2 - 2\langle \mathbf{A}\mathbf{x}^{[t]}, \mathbf{b}\rangle + \langle \mathbf{b}, \mathbf{b}\rangle\right)\end{aligned}$$

and by setting $\frac{\partial \Delta_{t+1}}{\partial \xi} = 0$ and solving for ξ, it follows that

$$\frac{\partial \Delta_{t+1}}{\partial \xi} = 2\xi \cdot \left(g_t^T \mathbf{A}^T \mathbf{A} g_t\right) - 2 \cdot \left(g_t^T \mathbf{A}^T\right)\left(\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\right) = 0$$

$$\implies \quad \xi_t^\star = \frac{\langle \mathbf{A}g_t, \mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\rangle}{\|\mathbf{A}g_t\|_2^2} \,, \tag{3.19}$$

which is the updated step-size we use at the next iteration. Since $\partial^2 \Delta_{t+1}/\partial \xi^2 = 2\|\mathbf{A}g_t\|_2^2 \geqslant 0$, we know that Δ_{t+1} is convex. Therefore, $\xi_t^\star$ derived in (3.19) is indeed the minimizer of Δ_{t+1}.

Now consider SD with the objective function $L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$. The only thing that changes in the derivation, is that now we have $(\grave{\mathbf{A}} = \mathbf{\Pi}\mathbf{A}, \grave{\mathbf{b}} = \mathbf{\Pi}\mathbf{b})$ instead of $(\mathbf{A}, \mathbf{b})$. By replacing $(\grave{\mathbf{A}}, \grave{\mathbf{b}}) \leftarrow (\mathbf{A}, \mathbf{b})$ in (3.19), it follows that

$$\frac{\langle \grave{\mathbf{A}}g_t, \grave{\mathbf{A}}\mathbf{x}^{[t]} - \grave{\mathbf{b}}\rangle}{\|\grave{\mathbf{A}}g_t\|_2^2} = \frac{\langle \mathbf{A}g_t, \mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\rangle}{\|\mathbf{A}g_t\|_2^2} \tag{3.20}$$

as $\grave{\mathbf{A}}^T\grave{\mathbf{A}} = \mathbf{A}^T\mathbf{A}$ and $\grave{\mathbf{A}}^T\grave{\mathbf{b}} = \mathbf{A}^T\mathbf{b}$, since $\mathbf{\Pi} \in O_N(\mathbb{R})$. The step-sizes for the corresponding iterations are therefore identical.

Moreover, the only difference between the objective functions $L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ and $L_{\mathbf{G}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ is the factor of $\sqrt{N/r}$. Let $\tilde{\mathbf{A}} = \mathbf{G}\mathbf{A}$ and $\tilde{\mathbf{b}} = \mathbf{G}\mathbf{b}$. Therefore, the step-size at iteration $t+1$ when considering the objective function $L_{\mathbf{G}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ is

$$\begin{aligned}\frac{\langle \tilde{\mathbf{A}} g_t, \tilde{\mathbf{A}}\mathbf{x}^{[t]} - \tilde{\mathbf{b}}\rangle}{\|\tilde{\mathbf{A}} g_t\|_2^2} &= \frac{N/r}{N/r} \cdot \frac{\langle \grave{\mathbf{A}} g_t, \grave{\mathbf{A}}\mathbf{x}^{[t]} - \grave{\mathbf{b}}\rangle}{\|\tilde{\mathbf{A}} g_t\|_2^2} \\ &\overset{\diamondsuit}{=} \frac{\langle \mathbf{A} g_t, \mathbf{A}\mathbf{x}^{[t]} - \mathbf{b}\rangle}{\|\mathbf{A} g_t\|_2^2}\end{aligned}$$

where $\diamondsuit$ follows from (3.20). □

Proof. [Corollary 4.5.1] We want to show that $\xi_t^\star$ according to (4.14), is a solution to (4.5.1). We know that the only difference in the induced sketching matrices $\mathbf{S}_{\mathbf{\Pi}}^{[t]}$ at each iteration are the resulting index sets $\mathcal{S}^{[t]}$, and the corresponding sampling and rescaling matrices $\widetilde{\mathbf{\Omega}}_{[t]}$.

To prove the given statement, since $\mathbf{S}_{\mathbf{\Pi}}^{[t]} = \widetilde{\mathbf{\Omega}}_{[t]}\mathbf{\Pi}$; and by Proposition (4.5.1) $\xi_t^\star$ is a solution to

$$\arg\min_{\xi\in\mathbb{R}} \left\{ \left\| \mathbf{\Pi}\left(\mathbf{A}\hat{\mathbf{x}}^{[t+1]} - \mathbf{b}\right)\right\|_2^2 \right\},$$

it suffices to show that $\mathbb{E}\left[\widetilde{\mathbf{\Omega}}_{[t]}^T\widetilde{\mathbf{\Omega}}_{[t]}\right] = \mathbf{I}_N$. This was proven in Lemma 4.3.1. Hence, the proof is complete. □

3.4 Proofs of Section 4.6

In this appendix, we present the proofs of Theorems 4.6.1 and 4.6.2, and Corollary 4.6.1. We also present a counterexample to perfect secrecy of the SRHT.

Proof. [Theorem 4.6.1] Denote the application of $\mathbf{\Pi}$ to a matrix $\mathbf{M}$ by $\mathsf{Enc}_{\mathbf{\Pi}}(\mathbf{M}) = \mathbf{\Pi}\mathbf{M}$. We will prove secrecy of this scheme, which then implies that a subsampled version of the transformed information is also secure. Let $\grave{\mathbf{A}} = \mathsf{Enc}_{\mathbf{\Pi}}(\mathbf{A})$ and $\grave{\mathbf{b}} = \mathsf{Enc}_{\mathbf{\Pi}}(\mathbf{b})$.

The adversaries' goal is to reveal $\mathbf{A}$. To prove that $\mathsf{Enc}_{\mathbf{\Pi}}$ is a well-defined security scheme, we need to show that an adversary cannot learn recover $\mathbf{A}$; with only knowledge of $(\grave{\mathbf{A}}, \grave{\mathbf{b}})$.

For a contradiction, assume an adversary is able to recover $\mathbf{A}$ after only observing $(\grave{\mathbf{A}}, \grave{\mathbf{b}})$. This means that it was able to obtain $\mathbf{\Pi}^{-1}$, as the only way to recover $\mathbf{A}$

from $\grave{\mathbf{A}}$ is by inverting the transformation of $\mathbf{\Pi}$: $\mathbf{A} = \mathbf{\Pi}^{-1} \cdot \grave{\mathbf{A}}$. This contradicts the fact that only $(\grave{\mathbf{A}}, \grave{\mathbf{b}})$ were observed. Thus, $\mathsf{Enc}_{\mathbf{\Pi}}$ is a well-defined security scheme.

It remains to prove perfect secrecy according to Definition 4.2.2. Observe that for any $\bar{\mathbf{U}} \in \mathcal{M}$ and $\bar{\mathbf{Q}} \in \mathcal{C}$

$$\Pr_{\mathbf{\Pi} \xleftarrow{U} \mathcal{K}} \left[\mathsf{Enc}_{\mathbf{\Pi}}(\bar{\mathbf{U}}) = \bar{\mathbf{Q}}\right] = \Pr_{\mathbf{\Pi} \xleftarrow{U} \mathcal{K}} \left[\mathbf{\Pi} \cdot \bar{\mathbf{U}} = \bar{\mathbf{Q}}\right] = \tag{3.21}$$

$$= \Pr_{\mathbf{\Pi} \xleftarrow{U} \mathcal{K}} \left[\mathbf{\Pi} = \bar{\mathbf{Q}} \cdot \bar{\mathbf{U}}^{-1}\right] \overset{\sharp}{=} \frac{1}{|\tilde{O}_{\mathbf{A}}|} = \frac{1}{|\mathcal{K}|} \tag{3.22}$$

where $\sharp$ follows from the fact that $\bar{\mathbf{Q}} \cdot \bar{\mathbf{U}}^{-1}$ is fixed. Hence, for any $\mathbf{U}_0, \mathbf{U}_1 \in \mathcal{M}$ and $\bar{\mathbf{Q}} \in \mathcal{C}$ we have

$$\Pr_{\mathbf{\Pi} \xleftarrow{U} \mathcal{K}} \left[\mathsf{Enc}_{\mathbf{\Pi}}(\mathbf{U}_0) = \bar{\mathbf{Q}}\right] = \frac{1}{|\mathcal{K}|} = \Pr_{\mathbf{\Pi} \xleftarrow{U} \mathcal{K}} \left[\mathsf{Enc}_{\mathbf{\Pi}}(\mathbf{U}_1) = \bar{\mathbf{Q}}\right]$$

as required by Definition 4.2.2. This completes the proof. □

We note that through the SVD of $\grave{\mathbf{A}}$, the adversaries can learn the singular values and right singular vectors of $\mathbf{A}$, since

$$\grave{\mathbf{A}} = (\mathbf{\Pi} \cdot \mathbf{U}_{\mathbf{A}}) \cdot \mathbf{\Sigma}_{\mathbf{A}} \cdot \mathbf{V}_{\mathbf{A}}^T = \mathbf{U}_{\grave{\mathbf{A}}} \cdot \mathbf{\Sigma}_{\mathbf{A}} \cdot \mathbf{V}_{\mathbf{A}}^T \,. \tag{3.23}$$

Recall that the singular values are unique and, for distinct positive singular values, the corresponding left and right singular vectors are also unique up to a sign change of both columns. We assume w.l.o.g. that $\mathbf{V}_{\grave{\mathbf{A}}} = \mathbf{V}_{\mathbf{A}}$ and $\mathbf{U}_{\grave{\mathbf{A}}} = \mathbf{\Pi} \cdot \mathbf{U}_{\mathbf{A}}$.

Geometrically, the encoding $\mathsf{Enc}_{\mathbf{\Pi}}$ changes the orthonormal basis of $\mathbf{U}_{\mathbf{A}}$ to $\mathbf{U}_{\grave{\mathbf{A}}}$, by rotating it or reflecting it; when $\det(\mathbf{\Pi})$ is +1 or -1 respectively. Of course, there are infinitely many ways to do so, which is what we are relying the security of this approach on.

Furthermore, unless $\mathbf{U}_{\mathbf{A}}$ has some special structure (*e.g.*, triangular, symmetric, etc.), one cannot use an off-the-shelf factorization to reveal $\mathbf{U}_{\mathbf{A}}$. Even though a lot can be revealed about $\mathbf{A}$, *i.e.* $\Sigma_{\mathbf{A}}$ and $\mathbf{V}_{\mathbf{A}}$, we showed that it is not possible to reveal $\mathbf{U}_{\mathbf{A}}$; hence nor $\mathbf{A}$, without knowledge of $\mathbf{\Pi}$.

Proof. [Corollary 4.6.1] The proof is identical to that of Lemma 3.2.2. The only difference is that the random variable entries $\tilde{Z}_j = \tilde{\mathbf{H}}_{ij}\mathbf{D}_{jj}\mathbf{y}_j$ for $j \in \mathbb{N}_N$ and the fixed

i now differ, though they still meet the same upper bound

$$|\tilde{Z}_j| \leqslant |\tilde{\mathbf{H}}_{ij}| \cdot |\mathbf{D}_{jj}| \cdot |\mathbf{y}_j| = \frac{|\mathbf{y}_j|}{\sqrt{N}} \ .$$

Since (3.7) holds true, the guarantees implied by flattening lemma also do, thus the sketching properties of the SRHT are maintained. □

Remark 3.4.1. *Since the Lemma 3.2.2 and Corollary 4.6.1 give the same result for the block-SRHT and garbled block-SRHT respectively, it follows that Theorem 4.4.1 also holds for the garbled block-SRHT.*

Proof. [Theorem 4.6.2] Assume w.l.o.g. that a computationally bounded adversary observes $\tilde{\mathbf{\Pi}}\mathbf{A}$, for which $\widetilde{\mathbf{A}}_r = \mathbf{S}_{\mathbf{\Pi}} \cdot \mathbf{A} = \widetilde{\mathbf{\Omega}} \cdot (\tilde{\mathbf{\Pi}}\mathbf{A})$ is the resulting sketch of Algorithm 7, for $\tilde{\mathbf{\Pi}} \in \tilde{H}_N$. To invert the transformation of $\tilde{\mathbf{\Pi}}$, the adversary needs knowledge of the components of $\tilde{\mathbf{\Pi}}$, *i.e.* $\hat{\mathbf{H}}$ and $\mathbf{P}$. Assume for a contradiction that there exists a probabilistic polynomial-time algorithm which, is able to recover $\mathbf{A}$ from $\tilde{\mathbf{\Pi}}\mathbf{A}$. This means that it has revealed $\mathbf{P}$, so that it can compute

$$\overbrace{(\mathbf{D}\hat{\mathbf{H}}\mathbf{P}^T)}^{\tilde{\mathbf{\Pi}}^T = \tilde{\mathbf{\Pi}}^{-1}} \cdot (\mathbf{P}\hat{\mathbf{H}}\mathbf{D}) \cdot \mathbf{A} = \tilde{\mathbf{\Pi}}^{-1} \cdot \tilde{\mathbf{\Pi}} \cdot \mathbf{A} = \mathbf{A} \ ,$$

which contradicts the assumption that the permutation $\mathbf{P}$ is a s-PRP. Specifically, recovering $\mathbf{A}$ by observing $\tilde{\mathbf{\Pi}}\mathbf{A}$ requires finding $\mathbf{P}$ in polynomial time. □

Finally, we show that $\hat{g}^{[t]} = g_{ls}^{[t]}$, which we claimed in Subsection 4.6.2. Since $\mathbf{\Pi} \in O_N(\mathbb{R})$ for the suggested projections (except that random Rademacher projection), we have $\mathbf{\Pi}^T\mathbf{\Pi} = \mathbf{I}_N$. It then follows that

$$\begin{aligned} \hat{g}^{[t]} &= 2 \sum_{j=1}^{K} \tilde{\mathbf{A}}_j^T \left(\tilde{\mathbf{A}}_j \mathbf{x}^{[t]} - \tilde{\mathbf{b}}_j \right) \\ &= (\mathbf{\Pi}\mathbf{A})^T \left(\mathbf{\Pi}\mathbf{A}\mathbf{x}^{[t]} - \mathbf{\Pi}\mathbf{b} \right) \\ &= \mathbf{A}^T \left(\mathbf{\Pi}^T\mathbf{\Pi} \right) \left(\mathbf{A}\mathbf{x}^{[t]} - \mathbf{b} \right) \\ &= g_{ls}^{[t]} \end{aligned} \tag{3.24}$$

and this completes the derivation.

3.4.1 Counterexample to Perfect Secrecy of the SRHT

Here, we present an explicit example for the SRHT (which also applies to the block-SRHT), which contradicts Definition 4.2.2. Therefore, the SRHT cannot provide perfect secrecy.

Consider the simple case where $N = 2$, and assume that $\hat{\mathbf{H}}_2 \in \tilde{O}_{\mathbf{A}}$. Since $(\tilde{O}_{\mathbf{A}}, \cdot)$ is a multiplicative subgroup of $\mathrm{GL}_2(\mathbb{R})$, we have $\mathbf{I}_2 \in \tilde{O}_{\mathbf{A}}$. Let $\mathbf{U}_0 = \mathbf{I}_2$ and $\mathbf{U}_1 = \hat{\mathbf{H}}_2$.

For d_1, d_2 i.i.d. Rademacher random variables and

$$\mathbf{D} = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix},$$

it follows that

$$\mathbf{C}_0 = \left(\hat{\mathbf{H}}_2\mathbf{D}\right) \cdot \mathbf{U}_0 = \hat{\mathbf{H}}_2\mathbf{D} = \frac{1}{2}\begin{pmatrix} d_1 & -d_2 \\ d_1 & d_2 \end{pmatrix}$$

and

$$\begin{aligned} \mathbf{C}_1 = \left(\hat{\mathbf{H}}_2\mathbf{D}\right) \cdot \mathbf{U}_1 &= \frac{1}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} d_1 & -d_1 \\ d_2 & d_2 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} d_1 - d_2 & -d_1 - d_2 \\ d_1 + d_2 & -d_1 + d_2 \end{pmatrix}. \end{aligned}$$

It is clear that $\mathbf{C}_0$ always has precisely two distinct entries, while $\mathbf{C}_1$ has three distinct entries; with 0 appearing twice for any pair $d_1, d_2 \in \{\pm 1\}$. Therefore, depending on the observed transformed matrix, we can disregard one of $\mathbf{U}_0$ and $\mathbf{U}_1$ as being a potential choice for $\mathbf{\Pi}$.

For instance, if $\bar{\mathbf{C}}$ is the observed matrix and it has two zero entries, then

$$\Pr_{\mathbf{\Pi} \overset{U}{\leftarrow} H_2}\left[\mathbf{\Pi} \cdot \mathbf{U}_1 = \bar{\mathbf{C}}\right] > \Pr_{\mathbf{\Pi} \overset{U}{\leftarrow} H_2}\left[\mathbf{\Pi} \cdot \mathbf{U}_0 = \bar{\mathbf{C}}\right] = 0$$

which contradicts (4.8).

Note that even if we apply a permutation, as in the case of the garbled block-SRHT, we still get the same conclusion. Hence, the garbled block-SRHT also does not achieve perfect secrecy.

3.4.2 Analogy with the One-Time Pad

It is worth noting that the encryption resulting by the multiplication with $\mathbf{\Pi}$; under the assumptions made in Theorem 4.6.1, bares a strong resemblance with the one-time pad (OTP), which is the optimum cryptosystem with theoretically perfect secrecy. This is not surprising, as it is one of the few known perfectly secret encryption schemes.

The main difference between the two, is that the spaces we work over are the multiplicative group $(\tilde{O}_{\mathbf{A}}, \cdot)$ whose identity is $\mathbf{I}_N$ in Theorem 4.6.1, and the additive group $\left((\mathbb{Z}_2)^{\ell}, +\right)$ in the OTP; whose identity is the zero vector of length ℓ.

As in the OTP, we make the assumption that $\mathcal{K}, \mathcal{M}, \mathcal{C}$ are all equal to the group we are working over; $\tilde{O}_{\mathbf{A}}$, which it is closed under multiplication. In the OTP, a message is revealed by applying the key on the ciphertext: if $c = m \oplus k$ for k drawn from $\mathcal{K}$, then $c \oplus k = m$. Analogously here, for $\mathbf{\Pi}$ drawn from $\tilde{O}_{\mathbf{A}}$: if $\bar{\mathbf{C}} = \mathbf{\Pi} \cdot \mathbf{U}_{\mathbf{A}}$, then $\bar{\mathbf{C}}^T \cdot \mathbf{\Pi} = (\mathbf{U}_{\mathbf{A}}^T \cdot \mathbf{\Pi}^T) \cdot \mathbf{\Pi} = \mathbf{U}_{\mathbf{A}}^T$. An important difference here is that the multiplication is not commutative.

Also, for two distinct messages m_0, m_1 which are encrypted with the same key k to c_0, c_1 respectively, it follows that $c_0 \oplus c_1 = m_1 \oplus m_2$ which reveals the XOR of the two messages. In our case, for the bases $\mathbf{U}_0, \mathbf{U}_1$ encrypted to $\mathbf{C}_0 = \mathbf{\Pi}\mathbf{U}_0$ and $\mathbf{C}_0 = \mathbf{\Pi}\mathbf{U}_1$ with the same projection matrix $\mathbf{\Pi}$, it follows that $\mathbf{C}_0^T \cdot \mathbf{C}_1 = \mathbf{U}_0^T \cdot \mathbf{U}_1$.

3.5 Orthonormal Encryption for Distributive Tasks

In this appendix, we discuss how applying a random projection $\mathbf{\Pi}$ can be utilized in existing CC schemes, both approximate and exact, to securely recover other matrix operations. The main idea is that after we apply and arbitrary $\mathbf{\Pi}$ to the underlying matrix or matrices, the analysis and corresponding conclusion of Theorem 4.6.1 still applies. Once the information is encrypted through $\mathbf{\Pi}$, *e.g.* $\grave{\mathbf{A}} = \mathbf{\Pi}\mathbf{A}, \grave{\mathbf{b}} = \mathbf{\Pi}\mathbf{b}$, we then carry out the CC scheme of choice, and we will recover the same result as if no encryption took place, without requiring an additional decryption step for the least squares problem and matrix multiplication, and does not increasing the system's redundancy. The drawback of this approach is the additional encryption step which corresponds to matrix multiplication. Fast matrix multiplication can be used to secure the data [11, 276], which is faster than computing $\mathbf{x}_{ls}^{\star} = \mathbf{A}^{\dagger}\mathbf{x}$.

We show how this approach is applied to GCSs for linear and logistic regression through SD, as well as *coded matrix multiplication* (CMM) schemes, and an approximate *matrix inversion* CC scheme; which is a non-linear operation [51]. In this

scheme we utilize the structure of the gradient of the respective objective functions.

What was discussed above resembles *Homomorphic Encryption* [117, 118, 37], which allows computations to be performed over encrypted data; and has been used to enable privacy-preserving machine learning. Two main drawbacks of homomorphic encryption though is that it leads to many orders of magnitude slow down in training, and it allows collusion between a larger number of workers [265]. Moreover, the privacy guarantees of homomorphic encryption rely on computational assumptions, while our approach is information-theoretic. Furthermore, we use (random) orthogonal matrices for encrypting the data, which has been studied in the context of image-processing and message encryption [3, 4, 10, 163, 242, 254].

3.5.1 Securing Linear Regression

As pointed out in 4.6.2 and (3.24), for the modified objective function

$$L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]}) \coloneqq \|\mathbf{\Pi}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2 = \|\grave{\mathbf{A}}\mathbf{x} - \grave{\mathbf{b}}\|_2^2 \ ,$$

we have $\nabla_{\mathbf{x}} L_{\mathbf{\Pi}}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]}) = \nabla_{\mathbf{x}} L_{ls}(\mathbf{A}, \mathbf{b}; \mathbf{x}^{[t]})$ for all t, *i.e.* $\hat{g}^{[t]} = g_{ls}^{[t]}$. It is clear that for $\mathbf{\Pi}$ an orthonormal matrix, there is no need to reverse the transformation to uncover the partial gradients.

Consider any GCS; exact or approximate, *e.g.* [25, 42, 47, 52, 58, 59, 62, 134, 144, 155, 223, 239, 279, 289, 292, 298]. If the workers are given partitions of $(\grave{\mathbf{A}},\grave{\mathbf{b}})$, they will have no knowledge of $(\mathbf{A}, \mathbf{b})$, unless they learn $\mathbf{\Pi}$. From the conclusion of Theorem 4.6.1, this is not a concern. The workers therefore locally recover the partial gradients of the block pairs they were assigned, and perform the encoding of the GCS which is being deployed, once these are encoded and communicated to the central server. After sufficiently many encodings are received, *i.e.* when the threshold recovery is met, the central server can then recover the gradient at the given iteration.

3.5.2 Securing Logistic Regression

Another widely used algorithm whose solution is accelerated though gradient methods is logistic regression, which yields a linear classifier [212]. Applying a random orthonormal matrix can secure the information when GCSs are used to solve logistic regression, though at each iteration the central server will have to apply two encryptions. At each iteration t, the workers are collectively given $\grave{\mathbf{A}} = \mathbf{\Pi}_1 \cdot \mathbf{A}$, $\grave{\mathbf{a}}_i = \mathbf{\Pi}_2 \cdot \left[1 \ \mathbf{a}_i^T\right]^T$ and $\grave{\mathbf{x}}^{[t]} = \mathbf{\Pi}_2 \cdot \mathbf{x}^{[t]}$, for $\mathbf{\Pi}_1 \in \tilde{O}_N(\mathbb{R})$ and $\mathbf{\Pi}_2 \in \tilde{O}_{d+1}(\mathbb{R})$. The

gradient update to be recovered is

$$\hat{g}^{[t+1]} = \grave{\mathbf{A}}^T(\boldsymbol{\mu} - \mathbf{b}) \;\text{ for }\; \boldsymbol{\mu}_i = \Big(1 + \exp\big(-\overbrace{\langle \hat{\mathbf{x}}^{[t]}, \hat{\mathbf{a}}_i \rangle}^{\langle \mathbf{x}^{[t]}, [1 \;\; \mathbf{a}_i^T] \rangle}\big)\Big)^{-1} .$$

Thus $\hat{g}^{[t+1]} = \boldsymbol{\Pi}_1^T \cdot g^{[t+1]}$, so we apply $\boldsymbol{\Pi}_1$ to recover $g^{[t+1]}$. In this problem, the labels $\mathbf{b}_i \in \{0, 1\}$ are not hidden. This is not a concern, as nothing can be inferred from these alone.

3.5.3 Securing Matrix Multiplication

Consider the matrices $\mathbf{A}_1 \in \mathbb{R}^{L \times N}$ and $\mathbf{A}_2 \in \mathbb{R}^{N \times M}$ whose multiplication is to be computed by a CMM scheme. For $\boldsymbol{\Pi} \in \tilde{O}_N(\mathbb{R})$, by carrying out the CMM scheme on $\grave{\mathbf{A}}_1 = \mathbf{A}_1 \cdot \boldsymbol{\Pi}$ and $\grave{\mathbf{A}}_2 = \boldsymbol{\Pi} \cdot \mathbf{A}_2$, we recover

$$\grave{\mathbf{A}}_1 \cdot \grave{\mathbf{A}}_2 = \mathbf{A}_1 \cdot (\boldsymbol{\Pi}^T \boldsymbol{\Pi}) \cdot \grave{\mathbf{A}}_2 = \mathbf{A}_1 \cdot \mathbf{A}_2 \;, \tag{3.25}$$

and a security guarantee analogous to Theorem 4.6.1 holds. This encryption is useful when $N \ll L, M$, as otherwise the cost of encrypting the two matrices could be higher than that of performing the multiplication.

3.5.4 Securing Distributive Matrix Inversion

In [51] a CC scheme was used to recover an approximation of the inverse of a matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, by requesting from the workers to approximate as part of their computation a subset of the optimization problems

$$\check{\mathbf{b}}_i = \arg\min_{\mathbf{b} \in \mathbb{R}^N} \left\{ \|\mathbf{A}\mathbf{b} - \mathbf{e}_i\|_2^2 \right\} \tag{3.26}$$

for each $i \in \mathbb{N}_N$, where $\{\mathbf{e}_i\}_{i=1}^N$ is the standard basis of $\mathbb{R}^N$. The solutions $\{\check{\mathbf{b}}\}_{i=1}^N$ comprise the columns of the inverse's estimate $\check{\mathbf{A}}^{-1}$, *i.e.* $\mathbf{A}^{-1} \approx \check{\mathbf{A}}^{-1} = \big[\check{\mathbf{b}}_1 \;\cdots\; \check{\mathbf{b}}_N\big]$, as

$$\mathbf{I}_N = \mathbf{A}\mathbf{A}^{-1} \approx \mathbf{A}\check{\mathbf{A}}^{-1} = \mathbf{A}\big[\check{\mathbf{b}}_1 \;\cdots\; \check{\mathbf{b}}_N\big] = \big[\mathbf{A}\check{\mathbf{b}}_1 \;\cdots\; \mathbf{A}\check{\mathbf{b}}_N\big] \;.$$

In this scheme, each worker has knowledge of the entire matrix $\mathbf{A}$. In our approach, instead of sharing $\mathbf{A}$ we share $\grave{\mathbf{A}} \coloneqq \mathbf{A} \cdot \boldsymbol{\Pi}^T$, for a randomly chosen $\boldsymbol{\Pi} \in \tilde{O}_N(\mathbb{R})$. The workers then approximate

$$\breve{\mathbf{b}}_i = \arg\min_{\mathbf{b} \in \mathbb{R}^N} \left\{ \|\grave{\mathbf{A}}\mathbf{b} - \mathbf{e}_i\|_2^2 \right\} \;, \tag{3.27}$$

thus $\grave{\mathbf{A}}^{-1} = \mathbf{\Pi} \cdot \check{\mathbf{A}}^{-1} = \left[\breve{\mathbf{b}}_1 \ \cdots \ \breve{\mathbf{b}}_N\right]$.

As in the case of logistic regression, here we also need an additional decryption step: $\mathbf{\Pi}^T \cdot (\mathbf{\Pi} \cdot \check{\mathbf{A}}^{-1}) = \check{\mathbf{A}}^{-1}$ at the end of the process, to recover the approximation.

Appendix D

Appendix to Chapter V

4.1 Additional Material and Background

In this appendix, we include material and background which was used in our derivations. First, we recall what an ϵ-*optimal solution/point* is, which was used in the proof of Proposition 5.4.2. Next, we state the MDS theorem and the BCH Bound. We then give a brief overview of the GC scheme from [134], to show how it differs from our CMIM. We also explicitly give their construction of a balanced mask matrix $\mathbf{M} \in \{0,1\}^{n\times k}$, which we use for the construction of the BRS generator matrices. Lastly, we illustrate a simple example of the encoding matrix.

Definition 4.1.1 ([15])**.** *A point $\bar{x}$ is said to be an* ***ϵ-optimal solution/point*** *to a minimization problem with objective function $f(x)$, if for any x, it holds that $f(x) \geqslant f(\bar{x}) - \epsilon$, where $\epsilon \geqslant 0$. When $\epsilon = 0$, an ϵ-optimal solution is an exact minimizer.*

The MDS theorem establishes the properties of MDS codes, which achieve the Singleton bound [154] and provide optimal error-correction capabilities for linear codes over finite fields.

Theorem 4.1.1 (MDS Theorem — [191])**.** *Let $\mathcal{C}$ be a linear $[n,k,d]$ code over $\mathbb{F}_q$, with $\mathbf{G},\mathbf{K}$ the generator and parity-check matrices. Then, the following are equivalent:*

1. *$\mathcal{C}$ is a MDS code, i.e. $d = n-k+1$*
2. *every set of $n-k$ columns of $\mathbf{K}$ is linearly independent*
3. *every set of k columns of $\mathbf{G}$ is linearly independent*
4. *$\mathcal{C}^{\perp}$ is a MDS code.*

The BCH bound is a theoretical limit that provides a lower bound on the minimum distance of a binary cyclic code. This bound was derived independently by the inventors of the BCH code [29, 141].

Theorem 4.1.2 (BCH Bound — [135],[207]). *Let* $p(\mathsf{x}) \in \mathbb{F}_q[\mathsf{x}]\backslash\{0\}$ *with* t *cyclically consecutive roots, i.e.* $p(\alpha^{j+\iota}) = 0$ *for all* $\iota \in \mathbb{N}_t$. *Then, at least* $t+1$ *coefficients of* $p(\mathsf{x})$ *are nonzero.*

Algorithm 12: MaskMatrix(n, k, d) [134]

Input: $n, k, d \in \mathbb{Z}_+$ s.t. $n > d, k$ and $w = \frac{kd}{n}$
Output: row-balanced mask matrix $\mathbf{M} \in \{0,1\}^{n\times k}$
$\mathbf{M} \leftarrow \mathbf{0}_{n\times k}$
for $j = 0$ *to* $k-1$ **do**
  for $i = 0$ *to* $d-1$ **do**
    $\iota \leftarrow (i + jd + 1) \bmod n$
    $\mathbf{M}_{r,\iota} \leftarrow 1$
  end
end
return M

4.1.1 Generator Matrix Example

For an example, consider the case where $n = 9$, $k = 6$ and $d = 6$, thus $w = \frac{kd}{n} = 4$. Then, Algorithm 12 produces

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \in \{0,1\}^{9\times 6} .$$

For our CCM, this means that the i^{th} worker computes the blocks indexed by $\mathrm{supp}(\mathbf{M}_{(i)})$, *e.g.* $\mathrm{supp}(\mathbf{M}_{(1)}) = \{1, 2, 4, 5\}$. We denote the indices of the respective task allocations by $\mathcal{J}_i = \mathrm{supp}(\mathbf{M}_{(i)})$. The entries of the generator matrix $\mathbf{G}$ are the evaluations of

the constructed polynomials (5.3) at each of the evaluation points $\mathcal{B} = \{\beta_i\}_{i=1}^n$, *i.e.* $\mathbf{G}_{ij} = p_j(\beta_i)$. This results in:

$$\mathbf{G} = \begin{pmatrix} p_1(\beta_1) & p_2(\beta_1) & 0 & p_4(\beta_1) & p_5(\beta_1) & 0 \\ p_1(\beta_2) & p_2(\beta_2) & 0 & p_4(\beta_2) & p_5(\beta_2) & 0 \\ p_1(\beta_3) & p_2(\beta_3) & 0 & p_4(\beta_3) & p_5(\beta_3) & 0 \\ p_1(\beta_4) & 0 & p_3(\beta_4) & p_4(\beta_4) & 0 & p_6(\beta_4) \\ p_1(\beta_5) & 0 & p_3(\beta_5) & p_4(\beta_5) & 0 & p_6(\beta_5) \\ p_1(\beta_6) & 0 & p_3(\beta_6) & p_4(\beta_6) & 0 & p_6(\beta_6) \\ 0 & p_2(\beta_7) & p_3(\beta_7) & 0 & p_5(\beta_7) & p_6(\beta_7) \\ 0 & p_2(\beta_8) & p_3(\beta_8) & 0 & p_5(\beta_8) & p_6(\beta_8) \\ 0 & p_2(\beta_9) & p_3(\beta_9) & 0 & p_5(\beta_9) & p_6(\beta_9) \end{pmatrix}.$$

4.2 Distributed Pseudoinverse

For full-rank rectangular matrices $\mathbf{A} \in \mathbb{R}^{N\times M}$ where $N > M$, one resorts to the left Moore–Penrose pseudoinverse $\mathbf{A}^\dagger \in \mathbb{R}^{M\times N}$, for which $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}_M$. In Algorithm 13, we present how to approximate the left pseudoinverse of $\mathbf{A}$, by using the fact that $\mathbf{A}^\dagger = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$; since $\mathbf{A}^\top\mathbf{A} \in \mathrm{GL}_M(\mathbb{R})$. The right pseudoinverse $\mathbf{A}^\dagger = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}$ of $\mathbf{A} \in \mathbb{R}^{M\times N}$ where $M < N$, can be obtained by a modification of Algorithm 13.

Just like the inverse, the pseudoinverse of a matrix also appears in a variety of applications. Computing the pseudoinverse of $\mathbf{A} \in \mathbb{R}^{N\times M}$ for $N > M$ is even more cumbersome, as it requires inverting the Gram matrix $\mathbf{A}^\top\mathbf{A}$. For this appendix, we consider a full-rank matrix $\mathbf{A}$.

One could naively attempt to modify Algorithm 8 in order to retrieve $\mathbf{A}^\dagger$ such that $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}_M$, by approximating the rows of $\mathbf{A}^\dagger$. This would *not* work, as the underlying optimization problems would not be strictly convex. Instead, we use Algorithm 13 to estimate the rows of $\mathbf{B}^{-1} \coloneqq (\mathbf{A}^\top\mathbf{A})^{-1}$, and then multiply the estimate $\widehat{\mathbf{B}^{-1}}$ by $\mathbf{A}^\top$. This gives us the approximation $\widehat{\mathbf{A}^\dagger} = \widehat{\mathbf{B}^{-1}} \cdot \mathbf{A}^\top$.

The drawback of Algorithm 13 is that it requires two additional matrix multiplications, $\mathbf{A}^\top\mathbf{A}$ and $\widehat{\mathbf{B}^{-1}}\mathbf{A}^\top$. We overcome this barrier by using a CMM scheme twice, to recover $\widehat{\mathbf{A}^\dagger}$ in a two or three-round communication CC approach. These are discussed in below.

Bounds on $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}})$ and $\mathrm{err}_{rF}(\widehat{\mathbf{A}^{-1}})$ can be established for both algorithms, specific to the black-box least squares solver being utilized.

Algorithm 13: Estimating $\mathbf{A}^\dagger$

Input: full-rank $\mathbf{A} \in \mathbb{R}^{N\times M}$ where $N > M$
$\mathbf{B} \leftarrow \mathbf{A}^\top\mathbf{A}$
for *i=1 to M* **do**
  $\hat{\mathbf{c}}_i = \arg\min_{\mathbf{c}\in\mathbb{R}^{1\times M}} \left\{ g_i(\mathbf{c}) \coloneqq \|\mathbf{cB} - \mathbf{e}_i^\top\|_2^2 \right\}$
  $\hat{\mathbf{b}}_i \leftarrow \hat{\mathbf{c}}_i \cdot \mathbf{A}^\top$
end
return $\widehat{\mathbf{A}^\dagger} \leftarrow \left[\hat{\mathbf{b}}_1^\top \ \cdots \ \hat{\mathbf{b}}_M^\top\right]^\top$ ▷ $\widehat{\mathbf{A}^\dagger}_{(i)} = \hat{\mathbf{b}}_i$

Corollary 4.2.1. *For full-rank* $\mathbf{A} \in \mathbb{R}^{N\times M}$ *with* $N > M$*, we have* $\mathrm{err}_F(\widehat{\mathbf{A}^\dagger}) \leqslant \frac{\sqrt{M}\epsilon\cdot\kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^3}$ *and* $\mathrm{err}_{rF}(\widehat{\mathbf{A}^\dagger}) \leqslant \frac{\sqrt{M}\epsilon\cdot\kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^2}$ *when using SD to solve the subroutine optimization problems of Algorithm 13, with termination criteria* $\|\nabla g_i(\mathbf{c}^{[t]})\|_2 \leqslant \epsilon$*.*

Proof. From (5.10), it follows that

$$\|\mathbf{B}^{-1}\mathbf{e}_i - \hat{\mathbf{c}}_i^\top\|_2 \leqslant \frac{\epsilon/\sqrt{2}}{\sigma_{\min}(\mathbf{B})^2} = \frac{\epsilon/\sqrt{2}}{\sigma_{\min}(\mathbf{A})^4} =: \delta \ .$$

The above bound implies that for each summand of the Frobenius error; $\|\hat{\mathbf{b}}_i - \mathbf{A}^\dagger_{(i)}\|_2 = \|\hat{\mathbf{c}}_i\mathbf{A}^\top - \mathbf{e}_i^\top\cdot\mathbf{B}^{-1}\mathbf{A}^\top\|_2$, we have $\|\hat{\mathbf{b}}_i - \mathbf{A}^\dagger_{(i)}\|_2 \leqslant \delta\|\mathbf{A}^\top\|_2$. Summing the right hand side M times, we get that

$$\begin{aligned}\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})^2 &\leqslant M\cdot(\delta\|\mathbf{A}^\top\|_2)^2 \\ &= \frac{M\epsilon^2\cdot\sigma_{\max}(\mathbf{A})^2}{2\sigma_{\min}(\mathbf{A})^8} \\ &= \frac{M\epsilon^2\cdot\kappa_2^2}{2\sigma_{\min}(\mathbf{A})^6} \ .\end{aligned}$$

By taking the square root, we have shown the first claim.

Since $1/\sigma_{\min}(\mathbf{A}) = \|\mathbf{A}^\dagger\|_2 \leqslant \|\mathbf{A}^\dagger\|_F$, it then follows that

$$\mathrm{err}_{rF}(\widehat{\mathbf{A}^\dagger}) = \frac{\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})}{\|\mathbf{A}^\dagger\|_F} \leqslant \frac{\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})}{\|\mathbf{A}^\dagger\|_2} = \frac{\sqrt{M}\epsilon\cdot\kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^2} \ ,$$

which completes the proof. □

4.2.1 Pseudoinverse from Polynomial CMM

One approach to leverage Algorithm 13 in a two-round communication scheme is to first compute $\mathbf{B} = \mathbf{A}^\top \mathbf{A}$ through a CMM scheme, then share $\mathbf{B}$ with all the workers who estimate the rows of $\widehat{\mathbf{B}^{-1}}$, and finally use another CMM to locally encode the estimated columns with blocks of $\mathbf{A}^\top$; to recover $\widehat{\mathbf{A}^\dagger} = \widehat{\mathbf{B}^{-1}} \cdot \mathbf{A}^\top$. Even though there are only two rounds of communication, the fact that we have a local encoding by the workers results in a higher communication load overall. An alternative approach which circumvents this issue, uses three-rounds of communication.

For this approach, we use the polynomial CMM scheme from [303] twice, along with our coded matrix inversion scheme. This CMM has a reduced communication load, and minimal computation is required by the workers. To have a consistent recovery threshold across our communication rounds, we partition $\mathbf{A}$ as in (5.11) into $\bar{k} = \sqrt{n-s} = \sqrt{k}$ blocks. Each block is of size $N \times \bar{\mathcal{T}}$, for $\bar{\mathcal{T}} = \frac{M}{k}$. The encodings from [303] of the partitions $\{\mathbf{A}_j\}_{j=1}^{\bar{k}}$ for carefully selected parameters $a, b \in \mathbb{Z}_+$ and distinct elements $\gamma_i \in \mathbb{F}_q$, are

$$\tilde{\mathbf{A}}_i^a = \sum_{j=1}^{k} \mathbf{A}_j \gamma_i^{(j-1)a} \quad \text{and} \quad \tilde{\mathbf{A}}_i^b = \sum_{j=1}^{k} \mathbf{A}_j \gamma_i^{(j-1)b}$$

for each worker indexed by i. Thus, each encoding is comprised of $N\bar{\mathcal{T}}$ symbols. The workers compute the product of their respective encodings $(\tilde{\mathbf{A}}_i^a)^\top \cdot \tilde{\mathbf{A}}_i^b$. The decoding step corresponds to an interpolation step, which is achievable when $\bar{k}^2 = k$ many workers respond[1], which is the optimal recovery threshold for CMM. Any fast polynomial interpolation or RS decoding algorithm can be used for this step, to recover $\mathbf{B}$.

Next, the master shares $\mathbf{B}$ with all the workers (from 5.5.1, this is necessary), who are requested to estimate the *column-blocks* of $\widehat{\mathbf{B}^{-1}}$

$$\widehat{\mathbf{B}^{-1}} = \begin{bmatrix} \bar{\mathcal{B}}_1 & \cdots & \bar{\mathcal{B}}_k \end{bmatrix} \quad \text{where } \bar{\mathcal{B}}_j \in \mathbb{R}^{M \times \bar{\mathcal{T}}} \ \forall j \in \mathbb{N}_k \tag{4.1}$$

according to Algorithm 8. We can then recover $\widehat{\mathbf{B}^{-1}}$ by our BRS based scheme, once k workers send their encoding.

[1] We select $\bar{k} = \sqrt{k}$ in the partitioning of $\mathbf{A}$ in (5.11) when deploying this CMM, to attain the same recovery threshold as our inversion scheme.

For the final round, we encode $\widehat{\mathbf{B}^{-1}}$ as

$$\tilde{\mathbf{B}}_i^a = \sum_{j=1}^{k} \bar{\mathcal{B}}_j \gamma_i^{(j-1)a}$$

which are sent to the respective workers. The workers already have in their possession the encodings $\tilde{\mathbf{A}}_i^b$. We then carry out the polynomial CMM where each worker is requested to send back $(\tilde{\mathbf{B}}_i^a)^\top \cdot \tilde{\mathbf{A}}_i^b$. The master server can then recover $\widehat{\mathbf{A}^\dagger}$.

Theorem 4.2.1. *Consider* $\mathbf{G} \in \mathbb{F}^{n\times k}$ *as in Theorem 5.5.1. By using any CMM, we can devise a matrix pseudoinverse CCM by utilizing Algorithm 13, in two-rounds of communication. By using polynomial CMM [303], we achieve this with a reduced communication load and minimal computation, in three-rounds of communication.*

Appendix E

Appendix to Chapter VI — Graph Sparsification by Approximate Matrix Multiplication

5.1 Introduction and Related Work

Large graphs, networks and their associated Laplacian are prevalent in many applications and domains of modern signal processing, statistics and engineering, *e.g.* spectral clustering [158, 287], community detection [72] and graph learning [152]. Their size makes them hard to store and process, which is why it is preferred to instead work with a good approximation or sketch of the graph. Algorithms for approximating large graphs have been developed through the study of *spectral graph theory*, which deals with the eigenvalues and eigenvectors of matrices naturally associated with graphs. A standard approach is by sampling edges or vertices of these graphs, with judiciously chosen sampling distributions.

Our main contribution, is bridging a connection between *randomized numerical linear algebra* (RandNLA) and approximate *matrix multiplication* (MM), with Laplacian *spectral sparsifiers* of *weighted* graphs $G = (V, E, w)$. The resulting algorithm is intuitive and simple, and has been considered in independent works. Our analysis though is more straightforward and shorter than other analyses considering the same and similar sparsifiers, *e.g.* [252]. Lastly, we introduce an alternative measure for spectral sparsifiers, which captures additive approximation errors.

Spectral sparsifiers are of importance, as they preserve eigenvector centrality [274], cuts in a graph [23], flows in networks modeled by graphs [122], and maintain the structure of the original graph. By viewing the Laplacian $\boldsymbol{L}$ of G as the outer-product of its boundary matrix $\boldsymbol{B} \in \mathbb{R}_{\geqslant 0}^{E \times V}$, we use CR matrix multiplication (CR−MM) to approximate the Laplacian $\tilde{\boldsymbol{L}} \approx \boldsymbol{L}$. This turns out be equivalent to sampling and re-weighting edges from G, with sampling probabilities proportional to the edges'

weights. The resulting Laplacian $\tilde{\boldsymbol{L}}$ is an unbiased estimate of minimum variance, and represents the sketched graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$. Unlike most other spectral sparsifiers whose guarantees depend entirely on the number of vertices n, ours depends on the edge weights w. We provide a concrete example in Figure E.1.

What we present also draws connections between sampling according to the Frobenius norm of vectors, and *leverage scores*; which has been extensively studied in the context of linear systems and ℓ_2-subspace embeddings [96, 91]. Sparsifying Laplacians through sampling is the appropriate intermediate application, between MM and subspace embeddings.

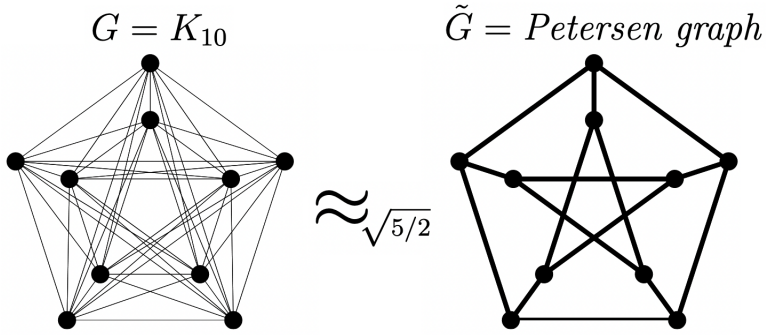


Figure E.1: The Petersen graph is a $\sqrt{5/2}$-approximation of K_{10} [275].

5.1.1 Related Work

The main idea behind the sparsifier we study is simple and intuitive. By using a primitive which has extensively been studied; approximate multiplication, as a surrogate to analysing the proposed spectral sparsifier, we present a simple analysis which yields more concise statements regarding the resulting sparsifier, compared to related work [252]; which considers Gaussian smoothing. Our guarantees differ from previous works, and we draw connections to RandNLA.

Along similar lines, connections between *effective resistances* and leverage scores have been previously established [92, 273]. Spectral sparsification has also been used in linear algebra to obtain deterministic and randomized algorithms for low-rank matrix approximations [32, 39]. In this work, we obtain results in the converse direction.

The state-of-the-art approach to spectral sparsification is to sample edges according to effective resistances [271, 273]. This approach leads to a nearly-linear time algorithm that produces high-quality sparsifiers of weighted graphs. A drawback of this approach is the computational complexity of determining the resistances, which

requires either a spectral decomposition of $\boldsymbol{L}$, or directly computing $\boldsymbol{L}^\dagger$. In what we propose, the sampling distribution is already known through w, and the sampling can be done pass-efficiently only inquiring $O(1)$ additional storage space [201, Algorithm 1]. This makes our method algorithmically superior to sampling according to effective resistances, as computing them requires $O(|E| \cdot |V|^2)$ operations.

Furthermore, through leverage scores, sampling according to the effective resistances relates to the notion of an ℓ_2-*subspace embedding*. As contrasted to the objective of [273], we use approximate multiplication; to obtain minimum variance unbiased estimators.

5.1.2 Preliminaries

Recall that the ***Laplacian*** of $G = (V, E, w)$ a weighted undirected graph with $|V| = n$, $|E| = m$, weights $w_{i,j}$ for each edge $(i, j) \in E$ is

$$\boldsymbol{L}_{ij} = \begin{cases} \sum_{(i,\ell)\in E} w_{i,j} & \text{if } i = j \\ -w_{i,j} & \text{if } i \neq j \end{cases} \qquad \text{for } i, j \in V. \tag{5.1}$$

Equivalently, it is expressed as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, for $\boldsymbol{D}, \boldsymbol{A} \in \mathbb{N}_0^{n\times n}$ respectively the degree and adjacency matrices of G. This can also be expressed as the Gram matrix of the ***boundary matrix***[1] $\boldsymbol{B} \in \mathbb{R}^{E\times V}$. Once we determine an arbitrary positive orientation (i, j) of the edges in E, the boundary matrix associated with the orientation is defined as

$$\boldsymbol{B}_{(i,j),v} = \begin{cases} -\sqrt{w_{i,j}} & \text{if } v = i \\ \sqrt{w_{i,j}} & \text{if } v = j \\ 0 & \text{o.w.} \end{cases} \qquad \text{for } (i, j) \in E \text{ and } v \in V.$$

For an edge $e = (u, v)$, the orientation is represented in the ***incidence vector*** $\chi_e = \mathbf{e}_u - \mathbf{e}_v$; for $\mathbf{e}_i \in \mathbb{R}^V$ the standard basis vectors. We define the ***weighted incidence vector*** as $\tilde{\chi}_e = \sqrt{w_e} \cdot \chi_e$. The Laplacian of G is then

$$\boldsymbol{L} = \boldsymbol{B}^T\boldsymbol{B} = \sum_{e\in E} \tilde{\chi}_e\tilde{\chi}_e^T = \sum_{e\in E} w_e \cdot \chi_e\chi_e^T \in \mathbb{R}^{V\times V}. \tag{5.2}$$

[1] The transpose of the boundary matrix of G, is also known as the *incidence matrix* of G.

5.1.3 Approximate Matrix Multiplication

Consider the two matrices $A \in \mathbb{R}^{L\times N}$ and $B \in \mathbb{R}^{N\times M}$, for which we want to approximate the product AB. It is known that the product may be approximated by sampling with replacement (s.w.r.) columns of A and rows of B, where the row-column sampling probabilities are proportional to their Euclidean norms. That is, we sample with replacement r pairs $(A^{(i)}, B_{(i)})$ for $i \in \mathbb{N}_N \coloneqq \{1, \cdots, N\}$ and $r < N$ ($A^{(i)}$=i^{th} column of A, and $B_{(i)}$=i^{th} row of B), with probability

$$p_i = \frac{\|A^{(i)}\|_2 \cdot \|B_{(i)}\|_2}{\sum_{l=1}^{N} \|A^{(l)}\|_2 \cdot \|B_{(l)}\|_2} \tag{5.3}$$

and sum a rescaling of the samples' outer-products:

$$AB \approx \frac{1}{r} \cdot \left(\sum_{j\in\mathcal{S}} \frac{1}{p_j} A^{(j)} B_{(j)} \right) = \sum_{j\in\mathcal{S}} \frac{A^{(j)}}{\sqrt{rp_j}} \cdot \frac{B_{(j)}}{\sqrt{rp_j}} =: Y \tag{5.4}$$

where $\mathcal{S}$ is the multiset consisting of the indices (possibly repeated) of the sampled pairs, hence $|\mathcal{S}| = r$. We denote the corresponding "compressed versions" of the input matrices by $C \in \mathbb{R}^{L\times r}$ and $R \in \mathbb{R}^{r\times M}$ respectively. This approximation satisfies $\|AB - CR\|_F = O(\|A\|_F \|B\|_F / \sqrt{r})$. Further details on this algorithm may be found in [88, 89, 90, 201, 294]. We have the following known results for the $CR-$MM algorithm.

Theorem 5.1.1 (Section 3.2 [201]). *The estimator $Y = CR$ from (5.4) is unbiased, while the sampling probabilities $\{p_i\}_{i=1}^N$ minimize the variance, i.e.*

$$\{p_i\}_{i=1}^N = \underset{\sum_{i=1}^N p_i = 1}{\arg\min} \left\{ \mathrm{Var}(Y) = \mathbb{E}\left[\|AB - CR\|_F^2 \right] \right\} \tag{5.5}$$

and it is an ϵ-multiplicative error approximation of the matrix product, with high probability. Specifically, for $\delta \geqslant 0$ and $r \geqslant \frac{1}{\delta^2\epsilon^2}$ the number of sampling trials which take place

$$\Pr\left[\|AB - CR\|_F \leqslant \epsilon \cdot \|A\|_F \|B\|_F \right] \geqslant 1 - \delta \tag{5.6}$$

for any $\epsilon > 0$.

Theorem 5.1.2 (Theorem 8 [201]). *Let $A \in \mathbb{R}^{L\times N}$ with $\sigma_{\max}(A) = \|A\|_2 \leqslant 1$, and approximate the product $Y \approx AA^T$ using $CR-$MM. Let $\epsilon \in (0, 1)$ be an accuracy*

parameter, and assume that $\|A\|_F^2 \geqslant 1/24$. *If*

$$r \geqslant \frac{96\|A\|_F^2}{\epsilon^2} \ln\left(\frac{96\|A\|_F^2}{\epsilon^2\sqrt{\delta}}\right) \geqslant \frac{4}{\epsilon^2} \ln\left(\frac{4}{\epsilon^2\sqrt{\delta}}\right)$$

for $r \leqslant N$, *then*

$$\Pr\left[\|AA^T - Y\|_2 \leqslant \epsilon\right] \geqslant 1 - \delta. \tag{5.7}$$

Below, we provide the pseudocode of the $CR-$MM algorithm.

Algorithm 14: CR matrix multiplication

Input: Matrices $A \in \mathbb{R}^{L\times N}$ and $B \in \mathbb{R}^{N\times M}$
Output: Approximate product $Y \approx AB$
Determine: Distribution $\{p_i\}_{i=1}^N$, according to (5.3)
Initialize: $Y = \mathbf{0}_{L\times M}$
for $i \leftarrow$ *1 to* r **do**
  sample $j \in \mathbb{N}_N$ with replacement, according to $\{p_i\}_{i=1}^N$
  $Y \leftarrow Y + \frac{1}{rp_j} \cdot A^{(j)}B_{(j)}$
end

5.2 Spectral Sparsification

First, recall that an ε-***spectral sparsifier*** for $\varepsilon \in (0,1)$ of G with Laplacian $\boldsymbol{L}$, is a sketched graph $\tilde{G}$ whose Laplacian $\tilde{\boldsymbol{L}}$ satisfies

$$(1-\varepsilon)\mathbf{x}^T\tilde{\boldsymbol{L}}\mathbf{x} \leqslant \mathbf{x}^T\boldsymbol{L}\mathbf{x} \leqslant (1+\varepsilon)\mathbf{x}^T\tilde{\boldsymbol{L}}\mathbf{x} \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad (1-\varepsilon)\|\tilde{\boldsymbol{B}}\mathbf{x}\|_2^2 \leqslant \|\boldsymbol{B}\mathbf{x}\|_2^2 \leqslant (1+\varepsilon)\|\tilde{\boldsymbol{B}}\mathbf{x}\|_2^2 \tag{5.8}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$. This implies that the approximated graph $\tilde{G}$ preserves the total weight of any cut between the factors of $1\pm\varepsilon$, hence also allowing a good approximation to its max-flow. A natural definition to consider, is that of when the approximation error is *additive*.

Definition 5.2.1. *An* ***additive*** ε***-sparsifier*** *of* G *with Laplacian* $\boldsymbol{L}$, *is a sketched graph* $\tilde{G}$ *whose Laplacian* $\tilde{\boldsymbol{L}}$ *satisfies*

$$\boldsymbol{x}^T\big(\tilde{\boldsymbol{L}} - \varepsilon\cdot\boldsymbol{I}_n\big)\boldsymbol{x} \leqslant \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \leqslant \boldsymbol{x}^T\big(\tilde{\boldsymbol{L}} + \varepsilon\cdot\boldsymbol{I}_n\big)\boldsymbol{x} \quad \Longleftrightarrow \quad \Big|\boldsymbol{x}^T(\boldsymbol{L}-\tilde{\boldsymbol{L}})\boldsymbol{x}\Big| \leqslant \varepsilon\cdot\|\boldsymbol{x}\|_2^2\ ,$$

for all $\boldsymbol{x} \in \mathbb{R}^n$.

We distinguish between the two types of sparsifiers, by referring to those satisfying (5.8) as ***multiplicative***. It is worth pointing out that row/column sampling algorithms whose approximations are in terms of the Frobenius norm; *e.g.* (5.6), naturally yield additive sparsifiers, while those which are in terms of the Euclidean norm; *e.g.* (5.7), admit multiplicative sparsifiers.

5.2.1 Spectral Sparsifier from $CR-$MM

We propose approximating $\boldsymbol{L}$ by using the $CR-$MM algorithm on $\boldsymbol{B}^T\boldsymbol{B}$. Let $W = \|\boldsymbol{B}\|_F^2/2 = \sum_{e' \in E} w_{e'}$. The resulting sampling probability of $e \in E$ according to (5.3), is

$$p_e \propto \|\boldsymbol{B}_{(e)}\|_2^2 = 2w_e \quad \Longrightarrow \quad p_e = w_e/W \ . \tag{5.9}$$

Thus, we are sampling edges proportionally to their weights. The resulting procedure is presented in Algorithm 15, where at each iteration we have a rank-1 update. We carry out a total of r sampling trials and rescale the updates, to reduce the variance of the estimator. Moreover, for $\boldsymbol{\Pi} = \mathrm{diag}(w_e/W)$, let $x_e = \sqrt{\boldsymbol{\Pi}}\chi_e$. Then $p_e = \|x_e\|_2^2$.

In simple words, we carry out r sampling trials with replacement on E, and each time e' is sampled, its new weight is increased by $\frac{W}{r}$. Furthermore, we note that the sampling procedure results in a diagonal sketching matrix $\boldsymbol{S}$, where $\boldsymbol{S}_{e,e} = \frac{\#\ e\ \text{is sampled}}{rp_e}$. Hence $\tilde{\boldsymbol{L}} = \boldsymbol{B}^T\boldsymbol{S}\boldsymbol{B}$ and $\tilde{\boldsymbol{B}} = \sqrt{\boldsymbol{S}}\boldsymbol{B}$.

Algorithm 15: CR spectral sparsifier

Input: A weighted simple undirected graph $G = (V, E, w)$, number of sampling trials r
Output: Laplacian $\tilde{\boldsymbol{L}}$, of sparsified $\tilde{G} = (V, \tilde{E}, \tilde{w})$
Determine: Boundary matrix $\boldsymbol{B} \in \mathbb{R}^{E\times V}$ of G, distribution $\{p_e = w_e/W\}_{e\in E}$
Initialize: $\tilde{\boldsymbol{L}} = \boldsymbol{0}_{V\times V}$
for $i \leftarrow$ *1 to* r **do**
  sample w.r. $e' \in E$, according to $\{p_e\}_{e\in E}$
  $\tilde{\boldsymbol{L}} \leftarrow \tilde{\boldsymbol{L}} + \frac{W}{rw_{e'}} \cdot \tilde{\chi}_{e'}\tilde{\chi}_{e'}^T = \tilde{\boldsymbol{L}} + \frac{W}{r} \cdot \chi_{e'}\chi_{e'}^T$
end

Proposition 5.2.1. *Given a weighted simple undirected graph $G = (V, E, w)$, Algorithm 15 produces an additive ε-spectral sparsifier of minimum variance; for $\varepsilon = 2W\epsilon$ and ϵ the CR accuracy parameter, with probability $1-\delta$ and $r \geqslant \frac{1}{\delta^2\epsilon^2}$.*

Proof. From (5.6), for $\Delta \coloneqq \boldsymbol{L} - \tilde{\boldsymbol{L}} \succeq 0$ we have with high probability $\|\Delta\|_F =$

$\|\boldsymbol{L}-\tilde{\boldsymbol{L}}\|_F \leqslant \epsilon\|\boldsymbol{B}\|_F^2 = 2W\epsilon$, and in turn:

$$\begin{aligned}\boldsymbol{x}^T(\boldsymbol{L}-\tilde{\boldsymbol{L}})\boldsymbol{x} &\overset{\sharp}{=} \boldsymbol{x}^T\Delta\boldsymbol{x}\\ &= \|\boldsymbol{x}^T\Delta\boldsymbol{x}\|_F\\ &\leqslant \|\Delta\|_F\cdot\|\boldsymbol{x}\|_2^2\\ &= \big(2W\epsilon\big)\cdot\|\boldsymbol{x}\|_2^2,\end{aligned}$$

which implies that

$$\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \leqslant \boldsymbol{x}^T\Big(\tilde{\boldsymbol{L}}+\boldsymbol{I}_n\cdot\big(2W\epsilon\big)\Big)\,\boldsymbol{x}.$$

In the case where $\Delta \preceq 0$, continuing from $\sharp$ we have

$$-\boldsymbol{x}^T\Delta\boldsymbol{x} = \|\boldsymbol{x}^T\Delta\boldsymbol{x}\|_F \leqslant \big(2W\epsilon\big)\cdot\|\boldsymbol{x}\|_2^2 \quad\Longrightarrow\quad \boldsymbol{x}^T\Big(\tilde{\boldsymbol{L}}-\boldsymbol{I}_n\cdot\big(2W\epsilon\big)\Big)\,\boldsymbol{x} \leqslant \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}.$$

All in all we have

$$\boldsymbol{x}^T\Big(\tilde{\boldsymbol{L}}-\mathbf{I}_n\big(2W\epsilon\big)\Big)\,\boldsymbol{x} \leqslant \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \leqslant \boldsymbol{x}^T\Big(\tilde{\boldsymbol{L}}+\mathbf{I}_n\big(2W\epsilon\big)\Big)\,\boldsymbol{x}$$

$$\Longleftrightarrow\qquad \Big|\boldsymbol{x}^T(\boldsymbol{L}-\tilde{\boldsymbol{L}})\boldsymbol{x}\Big| \leqslant \big(2W\epsilon\big)\cdot\|\boldsymbol{x}\|_2^2\ , \tag{5.10}$$

which is an additive ε-spectral sparsifier; for $\varepsilon = 2W\epsilon$.

By Theorem 5.1.1, the resulting estimator is of minimum variance. If $r \geqslant \frac{1}{\delta^2\epsilon^2}$ sampling trials are carried out, by (5.6) it follows that we attain such a sparsifier with probability at least $1-\delta$. □

5.2.2 Multiplicative Spectral Sparsifier

The case where $A = B^T$ in the $CR-$MM algorithm has also been studied as a special case, as it appears in numerous applications. This restriction allows us to use statements from random matrix theory [219], to get stronger spectral norm bounds, *e.g.* Theorem 5.1.2 [96, Theorem 4], [201, Theorem 8].

We will use Theorem 5.1.2 to show that Algorithm 15 is also a multiplicative spectral sparsifier. First, we recall an equivalent definition of a multiplicative ε-spectral sparsifier, based on spectral norm.

Definition 5.2.2. *For a weighted graph with Laplacian $\boldsymbol{L}$ and $\varepsilon > 0$, a sketched graph $\tilde{G}$ of G with Laplacian $\tilde{\boldsymbol{L}}$ and* ***isotropic boundary matrix*** $\tilde{\boldsymbol{B}}_{\text{iso}} \coloneqq \tilde{\boldsymbol{B}}\boldsymbol{L}^{-1/2}$

for $\boldsymbol{L}^{-1/2} \coloneqq \sqrt{\boldsymbol{L}^{\dagger}}$*, is a* ***multiplicative*** ε***-spectral sparsifier*** *if*

$$\|\boldsymbol{I}_n - \tilde{\boldsymbol{B}}_{\text{iso}}^T \tilde{\boldsymbol{B}}_{\text{iso}}\|_2 = \|\boldsymbol{L}^{-T/2}(\boldsymbol{L} - \tilde{\boldsymbol{L}})\boldsymbol{L}^{-1/2}\|_2 \leqslant \varepsilon \ . \tag{5.11}$$

Proposition 5.2.2. *Let* $G = (V, E, w)$ *be a weighted simple undirected graph with* $W = \sum_{e' \in E} w_{e'} \geqslant \sigma_{\max}^2(\boldsymbol{B})/48$*, and* $\epsilon \in (0, 1)$ *an accuracy parameter.*[2] *Algorithm 15 produces a multiplicative* ε*-spectral sparsifier* $\tilde{G}$ *for* $\varepsilon = \kappa_2(\boldsymbol{L}) \cdot \epsilon$ *with high probability, for* r *sufficiently large.*[3]

Proof. Denote the sketch of Algorithm 15 by $\boldsymbol{B}^T\boldsymbol{B} \approx \tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}$, and define $\bar{\boldsymbol{B}} \coloneqq \boldsymbol{B}/\sigma_{\max}(\boldsymbol{B})$; $\hat{\boldsymbol{B}} \coloneqq \tilde{\boldsymbol{B}}/\sigma_{\max}(\boldsymbol{B})$. Let $\bar{\boldsymbol{B}}^T \leftarrow A$ in Theorem 5.1.2, thus $\bar{\boldsymbol{B}}^T\bar{\boldsymbol{B}} \approx \hat{\boldsymbol{B}}^T\hat{\boldsymbol{B}}$. The first condition of Theorem 5.1.2 is met, as $\|\bar{\boldsymbol{B}}\|_2 = \|\boldsymbol{B}\|_2/\sigma_{\max}(\boldsymbol{B}) = 1$. Since $\|\bar{\boldsymbol{B}}\|_F^2 = \|\boldsymbol{B}\|_F^2/\sigma_{\max}^2(\boldsymbol{B}) = 2W/\sigma_{\max}^2(\boldsymbol{B})$, by our assumption on W it follows that $\|\bar{\boldsymbol{B}}\|_F^2 = \frac{2W}{\sigma_{\max}^2(\boldsymbol{B})} \geqslant \frac{2\sigma_{\max}^2(\boldsymbol{B})}{48\sigma_{\max}^2(\boldsymbol{B})} = 1/24$. Hence, the condition $\|\bar{\boldsymbol{B}}\|_F^2 \geqslant 1/24$ is also met.

Let $\theta = \sigma_{\max}(\boldsymbol{L})\epsilon = \sigma_{\max}^2(\boldsymbol{B})\epsilon$. From (5.7) it follows that:

$$\begin{aligned}
\Pr\left[\|\boldsymbol{L} - \tilde{\boldsymbol{L}}\|_2 \leqslant \sigma_{\max}(\boldsymbol{L})\epsilon\right] &= \Pr\left[\frac{\|\boldsymbol{B}^T\boldsymbol{B} - \tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}\|_2}{\sigma_{\max}^2(\boldsymbol{B})} \leqslant \epsilon\right] \\
&= \Pr\left[\|\bar{\boldsymbol{B}}^T\bar{\boldsymbol{B}} - \hat{\boldsymbol{B}}^T\hat{\boldsymbol{B}}\|_2 \leqslant \epsilon\right] \\
&\leqslant 1 - \delta \ .
\end{aligned}$$

We now appropriately apply $\boldsymbol{L}^{-1/2}$, in order to invoke (5.11):

$$\begin{aligned}
\|\boldsymbol{I}_n - \tilde{\boldsymbol{B}}_{\text{iso}}^T \tilde{\boldsymbol{B}}_{\text{iso}}\|_2 &= \|\boldsymbol{I}_n - \boldsymbol{L}^{-1/2} \cdot (\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}}) \cdot \boldsymbol{L}^{-1/2}\|_2 \\
&= \|\boldsymbol{I}_n - \boldsymbol{L}^{-1/2} \cdot \tilde{\boldsymbol{L}} \cdot \boldsymbol{L}^{-1/2}\|_2 \\
&= \|\boldsymbol{L}^{-1/2}(\boldsymbol{L} - \tilde{\boldsymbol{L}})\boldsymbol{L}^{-1/2}\|_2 \\
&\leqslant \theta \cdot \|\boldsymbol{L}^{-1/2}\|_2^2 \\
&= \frac{\theta}{\sigma_{\min}(\boldsymbol{L})} \\
&= \kappa_2(\boldsymbol{L}) \cdot \epsilon \ .
\end{aligned}$$

Therefore

$$\Pr\left[\|\boldsymbol{I}_n - \tilde{\boldsymbol{B}}_{\text{iso}}^T \tilde{\boldsymbol{B}}_{\text{iso}}\|_2 \leqslant \kappa_2(\boldsymbol{L}) \cdot \epsilon\right] \geqslant 1 - \delta$$

[2] $\lambda_{\max}(\boldsymbol{L}) = \sigma_{\max}(\boldsymbol{L}) = \sigma_{\max}^2(\boldsymbol{B})$

[3] The ***condition number*** of $\boldsymbol{L}$ is denoted by $\kappa_2(\boldsymbol{L}) = \|\boldsymbol{L}\|_2\|\boldsymbol{L}^{\dagger}\|_2 = \sigma_{\max}(\boldsymbol{L})/\sigma_{\min}(\boldsymbol{L})$ [271, 286, 162]. Since the smallest singular of $\boldsymbol{L}$ for G connected is 0, by $\sigma_{\min}(\boldsymbol{L})$ we denote the second smallest singular, which is equal to $1/\|\boldsymbol{L}^{\dagger}\|_2$. Also note that $\boldsymbol{L}^{-T/2} = \boldsymbol{L}^{-1/2}$.

for $r \geqslant 6\gamma^2_{\epsilon,\boldsymbol{B}} \ln\left(\gamma^2_{\epsilon,\boldsymbol{B}}/\sqrt{\delta}\right)$, where $\gamma_{\epsilon,\boldsymbol{B}} = \frac{8W}{\epsilon \cdot \sigma_{\max}(\boldsymbol{B})}$ and $\delta \in (0,1]$. This completes the proof. □

We note that since the objective here is to sparsify the graph, and since we do so by s.w.r., the condition $r \leqslant N$ assumed in Theorem 5.1.2 can be violated, as we will get heavier resulting edges for unstructured graphs, rather than more edges. All guarantees will still hold true.

5.2.3 Comparison to the Effective Resistances Approach

Let $\tilde{x}_e = \boldsymbol{L}^{-1/2}\chi_e$, for each $e \in E$. Then, the effective resistances are defined as $r_e = \|\tilde{x}_e\|_2^2$. It is therefore clear that the only difference between the proposed algorithm and that of sparsifying through effective resistances, is that the former is rescaled according to $\boldsymbol{\Pi}$ rather than $\boldsymbol{L}^\dagger$. The main benefit in our approach, is that the sampling distribution can be determined directly through w. We note also that $\{r_e\}_{e\in E}$ can be *approximated* in nearly-linear time [273].

The analysis of the proposed random sampling algorithm invokes Theorem 5.1.2, whose proof relies on a Chernoff bound on sums of Hermitian matrices [219]. Use of this bound is new in random sampling for Laplacian sparsification, and specifically applies to our proposed spectral method using sampling with replacement. This is to be compared with the use of other conventional Chernoff bounds [283] and concentration of measure [248] approaches. Intriguingly, unlike [273]; our approach does not require spectral information of $\boldsymbol{L}$.

The benefit of using the bound of [219], is that it can be applied to directly approximate the *intersection* of two different graphs on V. Specifically, we use $CR-$MM to approximate $\boldsymbol{L}_{1,2} = \boldsymbol{B}_1^T\boldsymbol{B}_2$, for $\boldsymbol{B}_1, \boldsymbol{B}_2$ the boundary matrices of the two graphs. A spectral guarantee, is not available, to our knowledge, for the case where the error of the underlying approximate MM is in terms of the Euclidean norm. Therefore, the techniques of [273] on sampling according to effective resistances does not apply. Definition 5.2.1 on the other hand quantifies the approximation error we get for Laplacians of such intersection graphs.

Another benefit of our approach, is that it permits spectral sparsification of dynamic graphs in which edges are continuously added to G, *i.e.* for $G^t = (V, E^t, w^t)$ after some time we get $G^{t+1} = (V, E^{t+1}, w^{t+1})$; where $E^{t+1} = E^t \cup \{e_{t+1}\}$ for $e_{t+1} \notin E^t$. This can be done through the SELECT algorithm, with $O(1)$ additional storage space, without altering the distribution according to the weights when sampling from $E^t \subset E^{t+1}$ [201, Lemma 1]. This is not possible when sampling according to ef-

fective resistances, as the addition of a single edge e_{t+1} could drastically alter the corresponding distribution on $E^t \subset E^{t+1}$.

5.3 Experiment

We compared s.w.r. according to $\{p_e\}_{e\in E}$ (via $CR-$MM) which is already known through w, and $\{r_e\}_{e\in E}$ (ER); which requires $O(mn^2)$ operations to calculate. Even though our main benefit is algorithmic, empirically our approach performs just as well; in terms of the error characterization (5.11). We considered the barbell graph on $n = 2713$ vertices, and assigned weights to each of the $m = 7864$ edges randomly from $\mathbb{N}_{100}$. We sparsified the graph for $r = 3500 + 500\nu$; for each $\nu \in \mathbb{N}_{13}$. In Figure E.2 we present the adjacency matrices of G and $\tilde{G}$, to distinguish the difference of G and $\tilde{G}$ for $r = 4000$. In Figures E.3 and E.4, we show the sparsification rate and error for each r.

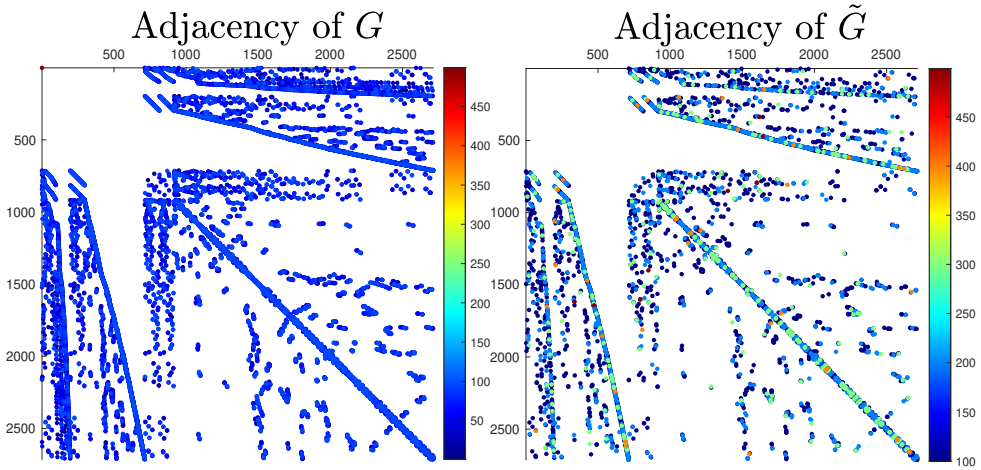


Figure E.2: Adjacency matrices of G and $\tilde{G}$, for $r = 4000$.

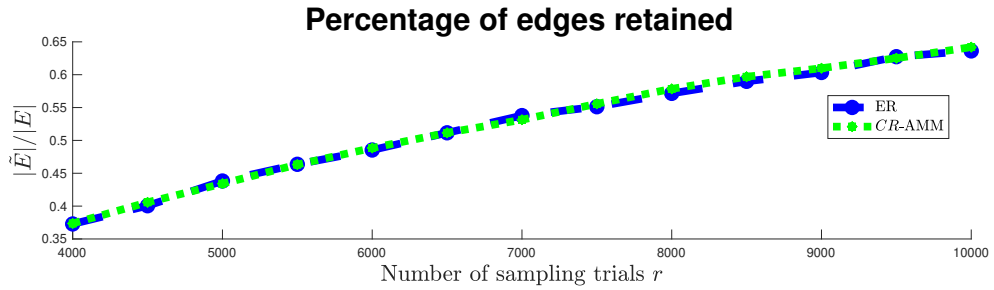


Figure E.3: Percentage of retained edges, after sparsification.

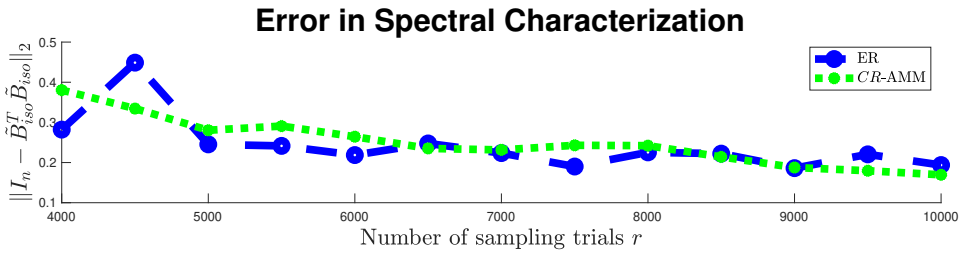


Figure E.4: Error in terms of (5.11), for varying r.

5.4 Future Directions

In this appendix, we proposed a graph sparsifier that approximates Laplacian through the use of $CR-$MM; a sampling with replacement technique, adapted from RandNLA. Applications of the proposed method to spectral clustering through *block* sampling [53, 217] would be worthwhile future work. Specifically, cliques of a given graph may be determined by approximating their Laplacians. The proposed computationally efficient spectral approximation may permit the identification of highly connected vertices without the need to traverse through the entire graph.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Dimitris Achlioptas. Database-friendly Random Projections: Johnson-Lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003. 5

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. k-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006. 169

[3] Jawad Ahmad, Muazzam A Khan, Seong Oun Hwang, and Jan Sher Khan. A compression sensing and noise-tolerant image encryption scheme based on chaotic maps and orthogonal matrices. *Neural computing and applications*, 28(1):953–967, 2017. 211

[4] Jawad Ahmad, Muazzam Ali Khan, Fawad Ahmed, and Jan Sher Khan. A novel image encryption scheme based on orthogonal matrix, skew tent map, and xor operation. *Neural Computing and Applications*, 30(12):3847–3857, 2018. 211

[5] Nir Ailon and Bernard Chazelle. Approximate Nearest Neighbors and the Fast Johnson–Lindenstrauss Transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, 2006. 5, 60, 79, 147, 177, 195

[6] Nir Ailon and Bernard Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009. 60

[7] Nir Ailon and Edo Liberty. Fast Dimension Reduction Using Rademacher Series on Dual BCH Codes. *Discrete & Computational Geometry*, 42:615–630, 2009. 5

[8] Nir Ailon and Edo Liberty. An Almost Optimal Unrestricted Fast Johnson-Lindenstrauss Transform. *ACM Transactions on Algorithms (TALG)*, 9(3):1–12, 2013. 5

[9] Miklós Ajtai. Generating Hard Instances of Lattice Problems. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery. 2

[10] Salamudeen Alhassan, Mohammed Muniru Iddrisu, and Mohammed Ibrahim Daabo. Perceptual video encryption using orthogonal matrix. *International Journal of Computer Mathematics: Computer Systems Theory*, 4(3-4):129–139, 2019. 211

[11] Josh Alman and Virginia Vassilevska Williams. A Refined Laser Method and Faster Matrix Multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021. 47, 113, 133, 210

[12] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998. 2

[13] Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998. 2

[14] Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 616–623, 1999. 6

[15] Fusheng Bai, Zhiyou Wu, and Daoli Zhu. Sequential Lagrange multiplier condition for ϵ-optimal solution in convex programming. *Optimization*, 57(5):669–680, 2008. 214

[16] Oleg Balabanov, Matthias Beaupère, Laura Grigori, and Victor Lederer. Block subsampled randomized Hadamard transform for low-rank approximation on distributed architectures. *arXiv preprint arXiv:2210.11295*, 2022. 177

[17] Frank Ban, David P. Woodruff, and Richard Zhang. Regularized Weighted Low Rank Approximation. In *Advances in Neural Information Processing Systems*, pages 4059–4069, 2019. 169

[18] Burak Bartan and Mert Pilanci. Polar Coded Distributed Matrix Multiplication. *arXiv preprint arXiv:1901.06811*, 2019. 14, 165

[19] Burak Bartan and Mert Pilanci. Straggler Resilient Serverless Computing Based on Polar Codes. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 276–283. IEEE, 2019. 77, 150

[20] Burak Bartan and Mert Pilanci. Distributed sketching for randomized optimization: Exact characterization, concentration, and lower bounds. *IEEE Transactions on Information Theory*, 69(6):3850–3879, 2023. 49, 79, 80

[21] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan Sparsifiers. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 255–262, 2009. 6

[22] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computations. In *Proceedings of the* 20^{th} *Annual ACM Symposium on the Theory of Computing, 1988*, pages 1–10, 1988. 3

[23] András A Benczúr and David R Karger. Approximating $s-t$ Minimum Cuts in $\tilde{O}(n^2)$ Time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55, 1996. 220

[24] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. In Stanley R. Petrick, Jean E. Sammet, Robert G. Tobey, and Joel Moses, editors, *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation, SYMSAC 1971, Los Angeles, California, USA, March 23-25, 1971*, page 223. ACM, 1971. 2

[25] Rawad Bitar, Mary Wootters, and Salim El Rouayheb. Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning. *IEEE Journal on Selected Areas in Information Theory*, 1:277–291, 2020. 12, 77, 79, 150, 211

[26] Å. Björck and V. Pereyra. Solution of Vandermonde Systems of Equations. *Mathematics of Computation*, 24:893–903, 1970. 113

[27] George R. Blakley. Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1899. 1, 122

[28] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020. 6

[29] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On A Class of Error Correcting Binary Group Codes. *Information and Control*, 3(1):68–79, 1960. 215

[30] Jean Bourgain, Stephen J. Dilworth, Kevin Ford, Sergei Konyagin, and Denka Kutzarova. Explicit Constructions of RIP Matrices and Related Problems. *CoRR*, abs/1008.4535, 2010. 6

[31] Christos Boutsidis. *Topics in Matrix Sampling Algorithms*. PhD thesis, Rensselaer Polytechnic Institute, 2011. 6

[32] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014. 221

[33] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the Subsampled Randomized Hadamard Transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013. 79

[34] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. An Improved Approximation Algorithm for the Column Subset Selection Problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM, 2009. 77, 151

[35] Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized Dimensionality Reduction for k-means Clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2014. 2, 48

[36] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004. 115, 116

[37] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. 211

[38] Sébastien Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. 72, 88, 117

[39] Martin Ayalde Camacho. *Spectral Sparsification: The Barrier Method and its Applications*. Harvard College, 2014. 221

[40] Emmanuel J. Candes, Justin Romberg, and Terence Tao. Robust Uncertainty Principles: Exact Signal Reconstruction From Highly Incomplete Frequency Information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006. 5

[41] Emmanuel J. Candes and Terence Tao. Decoding by Linear Programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005. 5

[42] Hankun Cao, Qifa Yan, Xiaohu Tang, and Guojun Han. Adaptive Gradient Coding. *IEEE/ACM Transactions on Networking*, 30(2):717–734, 2022. 79, 93, 211

[43] Wei-Ting Chang and Ravi Tandon. Random Sampling for Distributed Coded Matrix Multiplication. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8187–8191. IEEE, 2019. 78, 100, 151

[44] Neophytos Charalambides. Beyond the Guruswami-Sudan (and Parvaresh-Vardy) Radii: Folded Reed-Solomon, Multiplicity and Derivative Codes. *arXiv preprint arXiv:2003.05400*, 2020. 132

[45] Neophytos Charalambides. Dimensionality Reduction for k-means Clustering. *arXiv preprint arXiv:2007.13185*, 2020. 2, 48

[46] Neophytos Charalambides and Alfred O. Hero III. Graph Sparsification by Approximate Matrix Multiplication. In *2023 IEEE Statistical Signal Processing Workshop (SSP)*, page to appear. IEEE, 2023. 11, 134

[47] Neophytos Charalambides, Hessam Mahdavifar, and Alfred O. Hero III. Numerically Stable Binary Gradient Coding. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2622–2627, 2020. 8, 12, 49, 52, 79, 82, 104, 211

[48] Neophytos Charalambides, Hessam Mahdavifar, and Alfred O. Hero III. Numerically Stable Binary Coded Computations. *arXiv preprint arXiv:2109.10484*, 2021. 8, 49, 78

[49] Neophytos Charalambides, Hessam Mahdavifar, Mert Pilanci, and Alfred O. Hero III. Orthonormal Sketches for Secure Coded Regression. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 826–831, 2022. 10, 12, 61, 102, 147, 177, 178

[50] Neophytos Charalambides, Hessam Mahdavifar, Mert Pilanci, and Alfred O. Hero III. Iterative Sketching for Secure Coded Regression. *arXiv preprint arXiv:2308.04185*, 2023. 10

[51] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Straggler Robust Distributed Matrix Inverse Approximation. *arXiv preprint arXiv:2003.02948*, 2020. 7, 14, 40, 41, 78, 210, 212

[52] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Weighted Gradient Coding with Leverage Score Sampling. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5215–5219. IEEE, 2020. 12, 42, 49, 50, 54, 62, 77, 79, 89, 134, 150, 195, 211

[53] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Approximate Weighted CR Coded Matrix Multiplication. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5095–5099, 2021. 14, 40, 41, 42, 49, 50, 58, 62, 77, 78, 100, 134, 135, 150, 151, 154, 157, 230

[54] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Secure Linear MDS Coded Matrix Inversion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8, 2022. ix, 10, 14, 103, 105, 108, 119, 120

[55] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Federated Coded Matrix Inversion. *arXiv preprint arXiv:2301.03539*, 2023. 10, 78

[56] Neophytos Charalambides, Mert Pilanci, and Alfred O. Hero III. Gradient Coding through Iterative Block Leverage Score Sampling. *arXiv preprint arXiv:2308.03096*, 2023. 9, 83, 89, 90, 194

[57] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding Frequent Items in Data Streams. *Theoretical Computer Science*, 312(1):3–15, 2004. 59

[58] Zachary Charles and Dimitris Papailiopoulos. Gradient Coding Using the Stochastic Block Model. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1998–2002, 2018. 12, 27, 77, 79, 150, 211

[59] Zachary Charles, Dimitris Papailiopoulos, and Jordan Ellenberg. Approximate gradient coding via sparse random graphs. *arXiv preprint arXiv:1711.06771*, 2017. 12, 73, 77, 79, 83, 150, 194, 211

[60] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty Unconditionally Secure Protocols. In *Proceedings of the* 20^{th} *Annual ACM Symposium on the Theory of Computing, 1988*, pages 11–19, 1988. 3

[61] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert De Haan, and Vinod Vaikuntanathan. Secure Computation from Random Error Correcting Codes. In *Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings 26*, pages 291–310. Springer, 2007. 2

[62] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018. 12, 79, 211

[63] Mahdi Cheraghchi. Nearly optimal robust secret sharing. *Designs, Codes and Cryptography*, 87:1777–1796, 2019. 122

[64] Yasuko Chikuse. *Statistics on Special Manifolds*. Lecture Notes in Statistics. Springer New York, 2012. 95

[65] Fan R.K. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Soc., 1997. 6

[66] Ali Civril. Column Subset Selection Problem is UG-hard. *Journal of Computer and System Sciences*, 80(4):849–859, 2014. 77, 151

[67] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality Reduction for k-Means Clustering and Low Rank Approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172, 2015. 2, 48

[68] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform Sampling for Matrix Approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015. 65

[69] Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal Approximate Matrix Product in Terms of Stable Rank. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. 134

[70] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987. 133

[71] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. 122

[72] Lorenzo Dall'Amico, Romain Couillet, and Nicolas Tremblay. Optimal Laplacian Regularization for Sparse Spectral Community Detection. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3237–3241. IEEE, 2020. 220

[73] Anindya Bijoy Das and Aditya Ramamoorthy. Coded sparse matrix computation schemes that leverage partial stragglers. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1570–1575. IEEE, 2021. 14

[74] Anindya Bijoy Das, Aditya Ramamoorthy, and Namrata Vaswani. Efficient and Robust Distributed Matrix Computations via Convolutional Coding. *IEEE Transactions on Information Theory*, 67(9):6266–6282, 2021. 14

[75] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A Sparse Johnson–Lindenstrauss Transform. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 341–350, 2010. 5

[76] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. 5

[77] Son Hoang Dau, Wentu Song, Zheng Dong, and Chau Yuen. Balanced Sparsest Generator Matrices for MDS Codes. In *2013 IEEE International Symposium on Information Theory*, pages 1889–1893, 2013. 112, 126

[78] Luca De Feo. Mathematics of Isogeny Based Cryptography. *arXiv e-prints*, pages arXiv–1711, 2017. 2

[79] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal Distributed Online Prediction Using Mini-Batches. *Journal of Machine Learning Research*, 13(1), 2012. 72, 88

[80] Michał Dereziński and Michael W. Mahoney. Determinantal Point Processes in Randomized Numerical Linear Algebra. *Notices of the American Mathematical Society*, 68(1):34–45, 2021. 6

[81] Sagar Dhakal, Saurav Prakash, Yair Yona, Shilpa Talwar, and Nageen Himayat. Coded Federated Learning. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2019. 103, 105, 106

[82] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1

[83] Irit Dinur. The PCP Theorem by Gap Amplification. *Journal of the ACM (JACM)*, 54(3):12–es, 2007. 2

[84] David L. Donoho. Compressed Sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006. 5

[85] Petros Drineas. *Randomized Algorithms for Matrix Operations*. PhD thesis, Yale University, 2003. 6

[86] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in Large Graphs and Matrices. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 291–299, 1999. 48, 58

[87] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering Large Graphs via the Singular Value Decomposition. *Machine learning*, 56(1-3):9–33, 2004. 48

[88] Petros Drineas and Ravi Kannan. Fast Monte-Carlo Algorithms for Approximate Matrix Multiplication. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, pages 452–459, 2001. 8, 41, 58, 134, 135, 223

[89] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006. 8, 41, 48, 54, 134, 135, 223

[90] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix. *SIAM Journal on computing*, 36(1):158–183, 2006. 8, 41, 48, 54, 134, 135, 223

[91] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast Approximation of Matrix Coherence and Statistical Leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012. 54, 59, 60, 65, 89, 136, 138, 145, 195, 221

[92] Petros Drineas and Michael W. Mahoney. Effective Resistances, Statistical Leverage, and Applications to Linear Equation Solving. *arXiv preprint arXiv:1005.3097*, 2010. 5, 6, 134, 147, 221

[93] Petros Drineas and Michael W. Mahoney. RandNLA: Randomized Numerical Linear Algebra. *Communications of the ACM*, 59(6):80–90, 2016. 6, 48, 78, 89

[94] Petros Drineas and Michael W. Mahoney. Lectures on Randomized Numerical Linear Algebra. *arXiv preprint arXiv:1712.08880*, 2017. 6

[95] Petros Drineas, Michael W. Mahoney, and Shan Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136, 2006. 60, 89

[96] Petros Drineas, Michael W. Mahoney, Shan Muthukrishnan, and Tamás Sarlós. Faster Least Squares Approximation. *Numerische mathematik*, 117(2):219–249, 2011. 53, 78, 79, 81, 89, 91, 134, 135, 136, 147, 177, 221, 226

[97] Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. Short-Dot: Computing Large Linear Transforms Distributedly Using Coded Short Dot Products. In *Advances In Neural Information Processing Systems*, pages 2100–2108, 2016. 14, 78

[98] Sanghamitra Dutta, Mohammad Fahim, Farzin Haddadpour, Haewon Jeong, Viveck Cadambe, and Pulkit Grover. On the Optimal Recovery Threshold of Coded Matrix Multiplication. *IEEE Transactions on Information Theory*, 66(1):278–301, 2019. 14, 34, 43, 108

[99] Carl Eckart and G. Marion Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936. 169

[100] Salim El Rouayheb and Kannan Ramchandran. Fractional repetition codes for repair in distributed storage systems. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1510–1517. IEEE, 2010. 13, 46, 150

[101] Tommy Elfving. Block-iterative methods for consistent and inconsistent linear equations. *Numerische Mathematik*, 35(1):1–12, 1980. 48, 86

[102] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 1

[103] Paul Erdős. Some Remarks on the Theory of Graphs. *Bulletin of the American Mathematical Society*, 53(4):292–294, 1947. 2

[104] Paul Erdős and Alfred Rényi. On random graphs. *Publicationes Mathematicae*, 6:290297, 1959. 2

[105] N. Benjamin Erichson, Ariana Mendible, Sophie Wihlborn, and J. Nathan Kutz. Randomized Nonnegative Matrix Factorization. *arXiv preprint arXiv:1711.02037*, 2017. 169

[106] Ali Eshragh, Fred Roosta, Asef Nazari, and Michael W. Mahoney. LSAR: Efficient Leverage Score Sampling Algorithm for the Analysis of Big Time Series Data. *Journal of Machine Learning Research*, 23(22):1–36, 2022. 54, 59, 84

[107] Mohammad Fahim and Viveck R. Cadambe. Lagrange Coded Computing with Sparsity Constraints. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 284–289, 2019. 107

[108] Mohammad Fahim and Viveck R Cadambe. Numerically Stable Polynomially Coded Computing. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 3017–3021. IEEE, 2019. 14

[109] Mohammad Fahim, Haewon Jeong, Farzin Haddadpour, Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. On the Optimal Recovery Threshold of Coded Matrix Multiplication. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1264–1270. IEEE, 2017. 14, 34, 43

[110] Ora Nova Fandina, Mikael Møller Høgsgaard, and Kasper Green Larsen. The Fast Johnson-Lindenstrauss Transform Is Even Faster. In *International Conference on Machine Learning*, pages 9689–9715. PMLR, 2023. 5

[111] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996. 2

[112] Nuwan S. Ferdinand and Stark C. Draper. Anytime coding for distributed computation. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 954–960. IEEE, 2016. 14, 58, 59, 77, 150

[113] Péter Frankl and Hiroaki Maehara. The Johnson-Lindenstrauss Lemma and the Sphericity of Some Graphs. *Journal of Combinatorial Theory, Series B*, 44(3):355–362, 1988. 5

[114] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015. 80

[115] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo Algorithms for finding Low-Rank Approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004. 6

[116] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962. 45

[117] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009. 211

[118] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009. 211

[119] Alex Gittens. *Topics in Randomized Numerical Linear Algebra*. PhD thesis, California Institute of Technology, 2013. 6

[120] Margalit Glasgow and Mary Wootters. Approximate Gradient Coding with Optimal Decoding. *IEEE Journal on Selected Areas in Information Theory*, 2(3):855–866, 2021. 77, 150

[121] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. 2

[122] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988. 220

[123] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984. 1, 2

[124] Gene H. Golub, Michael W. Mahoney, Petros Drineas, and Lek-Heng Lim. Bridging the Gap Between Numerical Linear Algebra, Theoretical Computer Science, and Data Applications. *SIAM News*, 39(8):1–3, 2006. 6

[125] Robert M. Gower. Sketch and Project: Randomized Iterative Methods for Linear Systems and Inverting Matrices. *arXiv preprint arXiv:1612.06013*, 2016. 120

[126] B. G. Greenberg and A. E. Sarhan. Matrix Inversion, Its Interest and Application in Analysis of Data. *Journal of the American Statistical Association*, 54(288):755–766, 1959. 102

[127] Vipul Gupta, Swanand Kadhe, Thomas Courtade, Michael W. Mahoney, and Kannan Ramchandran. OverSketched Newton: Fast Convex Optimization for Serverless Systems. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 288–297. IEEE, 2020. 49, 50, 58, 59, 101, 152

[128] Vipul Gupta, Shusen Wang, Thomas Courtade, and Kannan Ramchandran. OverSketch: Approximate Matrix Multiplication for the Cloud. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 298–304. IEEE, 2018. 49, 50, 58, 59

[129] Venkatesan Guruswami and Atri Rudra. Explicit Codes Achieving List Decoding Capacity: Error-Correction With Optimal Redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008. 132

[130] Venkatesan Guruswami and Alexander Vardy. Maximum-Likelihood Decoding of Reed-Solomon Codes is NP-hard. *IEEE Transactions on Information Theory*, 51(7):2249–2256, 2005. 131

[131] Martin H. Gutknecht. Block Krylov Space Methods for Linear Systems with Multiple Right-hand Sides: An Introduction. *Modern Mathematical Models,Methods and Algorithms for Real World Systems*, 2006. 48, 86

[132] Sukjong Ha, Jingjing Zhang, Osvaldo Simeone, and Joonhyuk Kang. Coded Federated Computing in Wireless Networks with Straggling Devices and Imperfect CSI. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2649–2653, 2019. 103

[133] Farzin Haddadpour and Viveck R Cadambe. Codes for Distributed Finite Alphabet Matrix-Vector Multiplication. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1625–1629. IEEE, 2018. 14

[134] Wael Halbawi, Navid Azizan, Fariborz Salehi, and Babak Hassibi. Improving Distributed Gradient Descent Using Reed-Solomon Codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2027–2031. IEEE, 2018. 12, 42, 79, 103, 107, 111, 112, 114, 165, 211, 214, 215

[135] Wael Halbawi, Zihan Liu, and Babak Hassibi. Balanced Reed-Solomon codes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 935–939. IEEE, 2016. 42, 103, 110, 112, 113, 215

[136] Wael Halbawi, Zihan Liu, and Babak Hassibi. Balanced Reed-Solomon Codes for all parameters. In *2016 IEEE Information Theory Workshop (ITW)*, pages 409–413. IEEE, 2016. 103, 110

[137] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011. 6, 48

[138] Keaton Hamm and Longxiu Huang. Perturbations of *CUR* Decompositions. *SIAM Journal on Matrix Analysis and Applications*, 42(1):351–375, 2021. 135

[139] Richard Wesley Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. 1

[140] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, USA, 2nd edition, 2002. 102

[141] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffers*, 2:147–156, 1959. 215

[142] James Hook. Max-plus statistical leverage scores. *arXiv preprint arXiv:1609.09519*, 2016. 136, 138, 146

[143] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander Graphs and Their Applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. 6

[144] Shunsuke Horii, Takahiro Yoshida, Manabu Kobayashi, and Toshiyasu Matsushima. Distributed Stochastic Gradient Descent Using LDGM Codes. *arXiv preprint arXiv:1901.04668*, 2019. 12, 77, 79, 150, 211

[145] Berivan Isik, Tsachy Weissman, and Albert No. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, pages 3821–3846. PMLR, 2022. 53

[146] Tayyebeh Jahani-Nezhad and Mohammad Ali Maddah-Ali. CodedSketch: Coded Distributed Computation of Approximated Matrix Multiplication. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2489–2493. IEEE, 2019. 14, 58, 59

[147] Tayyebeh Jahani-Nezhad and Mohammad Ali Maddah-Ali. CodedSketch: A Coding Scheme for Distributed Computation of Approximated Matrix Multiplication. *IEEE Transactions on Information Theory*, 67(6):4185–4196, 2021. 49

[148] Tayyebeh Jahani-Nezhad and Mohammad Ali Maddah-Ali. Optimal Communication-Computation Trade-Off in Heterogeneous Gradient Coding. *IEEE Journal on Selected Areas in Information Theory*, 2(3):1002–1011, 2021. 131, 153

[149] Mohammad V. Jamali, Mahdi Soleymani, and Hessam Mahdavifar. Coded Distributed Computing: Performance Limits and Code Designs. In *2019 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2019. 13, 14

[150] Michael Jauch, Peter D. Hoff, and David B. Dunson. Monte Carlo simulation on the Stiefel manifold via polar expansion. *Journal of Computational and Graphical Statistics*, 30(3):622–631, 2021. 90

[151] Haewon Jeong, Ateet Devulapalli, Viveck R Cadambe, and Flavio Calmon. ε-approximate coded matrix multiplication is nearly twice as efficient as exact multiplication. *arXiv preprint arXiv:2105.01973*, 2021. 14

[152] Bo Jiang, Yiyi Yu, Hamid Krim, and Spencer L Smith. Dynamic Graph Learning Based on Graph Laplacian. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1090–1094. IEEE, 2021. 220

[153] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Contemp. Math.*, volume 26, pages 189–206, 1984. 5, 48, 79

[154] Durga Datt Joshi. A Note on Upper Bounds for Minimum Distance Codes. *Information and Control*, 3(1):289–295, 1958. 214

[155] Swanand Kadhe, O Ozan Koyluoglu, and Kannan Ramchandran. Gradient coding based on block designs for mitigating adversarial stragglers. *arXiv preprint arXiv:1904.13373*, 2019. 12, 46, 73, 77, 79, 150, 194, 211

[156] Mohammad Mahdi Kamani, Farzin Haddadpour, Rana Forsati, and Mehrdad Mahdavi. Efficient Fair Principal Component Analysis. *arXiv preprint arXiv:1911.04931*, 2019. 169

[157] Daniel M. Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss Transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014. 5

[158] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On Clusterings: Good, Bad and Spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. 220

[159] Can Karakus, Yifan Sun, and Suhas Diggavi. Encoded Distributed Optimization. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2890–2894. IEEE, 2017. 80, 102

[160] Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. Redundancy Techniques for Straggler Mitigation in Distributed Optimization and Learning. *Journal of Machine Learning Research*, 20(72):1–47, 2019. 80

[161] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014. 84, 97

[162] Vijay Keswani. Laplacian Solvers and Graph Sparsification. Master's thesis, Indian Institute of Technology Kanpur, 2016. 227

[163] Fozia Hanif Khan, Rehan Shams, Farheen Qazi, and D Agha. Hill cipher key generation algorithm by using orthogonal matrix. *Int. J. Innov. Sci. Mod. Eng*, 3(3):5–7, 2015. 211

[164] Shahrzad Kiani and Stark C. Draper. Successive Approximation Coding for Distributed Matrix Multiplication. *IEEE Journal on Selected Areas in Information Theory*, 3(2):286–305, 2022. 108

[165] Shahrzad Kiani and Stark C. Draper. Successive Approximation for Coded Matrix Multiplication. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 838–843. IEEE, 2022. 108

[166] Shahrzad Kiani, Nuwan Ferdinand, and Stark C. Draper. Exploitation of Stragglers in Coded Computation. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1988–1992. IEEE, 2018. 78

[167] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv preprint arXiv:1610.02527*, 2016. 104

[168] Manfred Krause. A Simple Proof of the Gale-Ryser Theorem. *The American Mathematical Monthly*, 103(4):335–337, 1996. 112

[169] Siddhartha Kumar, Reent Schlegel, Eirik Rosnes, and Alexandre Graell i Amat. Coding for Straggler Mitigation in Federated Learning. *arXiv preprint arXiv:2109.15226*, 2021. 103, 105

[170] H. T. Kung. Fast evaluation and interpolation. *Carnegie Mellon University, Tech. Rep.*, 1973. 43

[171] Rasmus Kyng. *Approximate Gaussian Elimination*. PhD thesis, Yale University, 2017. 131, 153

[172] Rasmus Kyng and Sushant Sachdeva. Approximate Gaussian Elimination for Laplacians – Fast, Sparse, and Simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582. IEEE, 2016. 131, 153

[173] Jonathan Lacotte, Sifan Liu, Edgar Dobriban, and Mert Pilanci. Optimal Iterative Sketching with the Subsampled Randomized Hadamard Transform. *Advances in Neural Information Processing Systems*, 33, 2020. 49, 79, 101, 152

[174] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638. IEEE, 2017. 5

[175] Can M. Le. Edge sampling using local network information. *Journal of Machine Learning Research*, 22(88):1–29, 2021. 6

[176] François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303, 2014. 133

[177] Daniel D. Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorization. In *Advances in neural information processing systems*, pages 556–562, 2001. 168

[178] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding Up Distributed Machine Learning Using Codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2018. 2, 12, 14, 38, 49, 56, 66, 78, 102, 165

[179] Kangwook Lee, Changho Suh, and Kannan Ramchandran. High-Dimensional Coded Matrix Multiplication. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2418–2422. IEEE, 2017. 14, 33, 78

[180] Yin Tat Lee. Probabilistic Spectral Sparsification In Sublinear Time. *arXiv preprint arXiv:1401.0085*, 2013. 6

[181] Songze Li and Salman Avestimehr. Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning. *Foundations and Trends® in Communications and Information Theory*, 17(1):1–148, 2020. 4, 13, 17, 49, 51, 78, 102

[182] Songze Li, Seyed Mohammadreza Mousavi Kalan, Qian Yu, Mahdi Soltanolkotabi, and A. Salman Avestimehr. Polynomially Coded Regression: Optimal Straggler Mitigation via Data Encoding. *arXiv preprint arXiv:1805.09934*, 2018. 17, 33, 168

[183] Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. Coded Distributed Computing: Straggling Servers and Multistage Dataflows. In *54th Annual Allerton Conference*, pages 164–171. IEEE, 2016. 12, 78

[184] Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. A unified coding framework for distributed computing with straggling servers. *arXiv preprint arXiv:1609.01690*, 2016. 12

[185] Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. Coding for Distributed Fog Computing. *IEEE Commun. Mag.*, 55(4):34–40, 2017. 12, 78

[186] Songze Li, Seyed Mohammadreza Mousavi Kalan, A. Salman Avestimehr, and Mahdi Soltanolkotabi. Near-Optimal Straggler Mitigation for Distributed Gradient Methods. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 857–866, 2018. 12

[187] Edo Liberty. *Accelerated Dense Random Projections*. PhD thesis, Yale University, 2009. 5

[188] Edo Liberty, Nir Ailon, and Amit Singer. Dense Fast Random Projections and Lean Walsh Transforms. In *Proceedings of the 11th international workshop, APPROX 2008, and 12th international workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, pages 512–522, 2008. 5

[189] Chih-Jen Lin. On the Convergence of Multiplicative Update Algorithms for Nonnegative Matrix Factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596, 2007. 169

[190] Chih-Jen Lin. Projected Gradient Methods for Non-negative Matrix Factorization. *Neural computation*, 19(10):2756–2779, 2007. 169

[191] San Ling and Chaoping Xing. *Coding Theory: A First Course*. Cambridge University Press, 2004. 126, 214

[192] Hsuan-Po Liu, Mahdi Soleymani, and Hessam Mahdavifar. Differentially Private Coded Computing. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2189–2194, 2023. 80

[193] László Lovász. Eigenvalues of graphs. *Lecture notes*, 2007. 6

[194] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman. Efficient Erasure Correcting Codes. *IEEE Transactions on Information Theory*, 47(2):569, 2001. 2

[195] Ping Ma, Michael W. Mahoney, and Bin Yu. A Statistical Perspective on Algorithmic Leveraging. *The Journal of Machine Learning Research*, 16(1):861–911, 2015. 54, 59, 134

[196] David J.C. MacKay and Radford M. Neal. Near Shannon Limit Performance of Low Density Parity Check Codes. *Electronics Letters*, 32(18):1645–1646, 1996. 45

[197] Malik Magdon-Ismail. Row Sampling for Matrix Algorithms via a Non-Commutative Bernstein Bound. *arXiv preprint arXiv:1008.0587*, 2010. 137

[198] Michael W. Mahoney. Lecture Notes on Spectral Graph Methods. *arXiv preprint arXiv:1608.04845*, 2011. 5, 6

[199] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011. 6

[200] Michael W. Mahoney. Algorithmic and Statistical Perspectives on Large-Scale Data Analysis. *Combinatorial Scientific Computing*, pages 427–469, 2012. 6, 47, 48

[201] Michael W. Mahoney. Lecture Notes on Randomized Linear Algebra. *arXiv preprint arXiv:1608.04481*, 2016. 5, 6, 48, 54, 59, 60, 77, 134, 135, 136, 138, 146, 151, 192, 195, 196, 222, 223, 226, 228

[202] Michael W. Mahoney, Lek-Heng Lim, and Gunnar E Carlsson. Algorithmic and Statistical Challenges in Modern Large-Scale Data Analysis are the Focus of MMDS 2008. *ACM SIGKDD Explorations Newsletter*, 10(2):57–60, 2008. 5, 47

[203] Ankur Mallick, Malhar Chaudhari, and Gauri Joshi. Rateless Codes for Near-Perfect Load Balancing in Distributed Matrix-Vector Multiplication. *arXiv preprint arXiv:1804.10331*, 2018. 12

[204] Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020. 6

[205] Jiří Matoušek. On Variants of the Johnson–Lindenstrauss Lemma. *Random Structures & Algorithms*, 33(2):142–156, 2008. 5

[206] Robert J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN Progress Report.*, 33:114–116, 1978. 123

[207] Robert J. McEliece. *Theory of Information and Coding*. Cambridge University Press, USA, 2nd edition, 2001. 112, 215

[208] Mitzenmacher Michael and Upfal Eli. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017. 2

[209] Gary L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of computer and system sciences*, 13(3):300–317, 1976. 2

[210] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995. 2

[211] Alexander Munteanu, Simon Omlor, and David Woodruff. Oblivious Sketching for Logistic Regression. In *International Conference on Machine Learning*, pages 7861–7871. PMLR, 2021. 61

[212] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. 211

[213] Riley Murray, James Demmel, Michael W. Mahoney, N Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E Lopes, et al. Randomized Numerical Linear Algebra: A Perspective on the Field With an Eye to Software. *arXiv preprint arXiv:2302.11474*, 2023. 5, 6

[214] Cameron Musco and Christopher Musco. Projection-Cost-Preserving Sketches: Proof Strategies and Constructions. *arXiv preprint arXiv:2004.08434*, 2020. 6, 48

[215] Deanna Needell and Joel A. Tropp. Paved with Good Intentions: Analysis of a Randomized Block Kaczmarz Method. *Linear Algebra and its Applications*, 441:199–221, 2014. 48, 86

[216] Jer Shyuan Ng, Wei Yang Bryan Lim, Nguyen Cong Luong, Zehui Xiong, Alia Asheralieva, Dusit Niyato, Cyril Leung, and Chunyan Miao. A Comprehensive Survey on Coded Distributed Computing: Fundamentals, Challenges, and Networking Applications. *IEEE Communications Surveys & Tutorials*, 23(3):1800–1837, 2021. 4, 49

[217] Chengmei Niu and Hanyu Li. Optimal Sampling Algorithms for Block Matrix Multiplication. *Journal of Computational and Applied Mathematics*, 425:115063, 2023. 134, 135, 154, 230

[218] Ryan O'Donnell. Probability and Computing. `http://www.cs.cmu.edu/~odonnell/papers/probability-and-computing-lecture-notes.pdf`, 2009. 2

[219] Roberto Oliveira. Sums of random Hermitian matrices and an inequality by Rudelson. *Electronic Communications in Probability*, 15:203–212, 2010. 136, 226, 228

[220] Brad Osgood. The Fourier Transform and its Applications. *Stanford University, Lecture Notes*, 2009. 91, 204

[221] Urvashi Oswal, Swayambhoo Jain, Kevin S Xu, and Brian Eriksson. Block *CUR*: Decomposing Matrices Using Groups of Columns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 360–376. Springer, 2018. 49, 54, 77, 89, 151, 195

[222] Emre Ozfatura, Baturalp Buyukates, Deniz Gunduz, and Sennur Ulukus. Age-Based Coded Computation for Bias Reduction in Distributed Learning. *arXiv preprint arXiv:2006.01816*, 2020. 78

[223] Emre Ozfatura, Deniz Gunduz, and Sennur Ulukus. Gradient Coding with Clustering and Multi-message Communication. *arXiv preprint arXiv:1903.01974*, 2019. 12, 13, 79, 211

[224] Emre Ozfatura, Sennur Ulukus, and Deniz Gunduz. Coded Distributed Computing with Partial Recovery. *arXiv preprint arXiv:2007.02191*, 2020. 78

[225] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent Semantic Indexing: A Probabilistic Analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168, 1998. 5

[226] Dimitris Papailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable Deterministic Leverage Score Sampling. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 997–1006, 2014. 50

[227] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. Locally Repairable Codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014. 44

[228] Farzad Parvaresh and Alexander Vardy. Correcting Errors Beyond the Guruswami-Sudan Radius in Polynomial Time. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 285–294. IEEE, 2005. 131

[229] Chris Peikert. A Decade of Lattice Cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016. 2

[230] Richard Peng and Santosh Vempala. Solving Sparse Linear Systems Faster than Matrix Multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 504–521. SIAM, 2021. 47

[231] Mert Pilanci and Martin J Wainwright. Iterative Hessian Sketch: Fast and Accurate Solution Approximation for Constrained Least-Squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016. 49, 61, 79, 101, 152, 174

[232] Mert Pilanci and Martin J Wainwright. Newton Sketch: A Linear-time Optimization Algorithm with Linear-Quadratic Convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017. 49

[233] Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Yair Yona, Shilpa Talwar, Salman Avestimehr, and Nageen Himayat. Coded Computing for Low-Latency Federated Learning over Wireless Edge Networks. *IEEE Journal on Selected Areas in Communications*, 39(1):233–250, 2020. 103, 105

[234] Michael O Rabin. Probabilistic Algorithm for Testing Primality. *Journal of number theory*, 12(1):128–138, 1980. 2

[235] Michael O. Rabin. Probabilistic Algorithms in Finite Fields. *SIAM Journal on computing*, 9(2):273–280, 1980. 2

[236] Aditya Ramamoorthy and Li Tang. Numerically stable coded matrix computations via circulant and rotation matrix embeddings. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1712–1717. IEEE, 2021. 14, 131, 153

[237] Aditya Ramamoorthy, Li Tang, and Pascal O Vontobel. Universally Decodable Matrices for Distributed Matrix-Vector Multiplication. *arXiv preprint arXiv:1901.10674*, 2019. 12, 78

[238] Garvesh Raskutti and Michael W. Mahoney. A Statistical Perspective on Randomized Sketching for Ordinary Least-Squares. *J. Mach. Learn. Res.*, 17:214:1–214:31, 2016. 4

[239] Netanel Raviv, Itzhak Tamo, Rashish Tandon, and Alexandros G Dimakis. Gradient Coding from Cyclic MDS Codes and Expander Graphs. *IEEE Transactions on Information Theory*, 66(12):7475–7489, 2020. 12, 77, 79, 127, 150, 211

[240] Elizaveta Rebrova and Deanna Needell. On block Gaussian sketching for the Kaczmarz method. *Numerical Algorithms*, pages 1–31, 2020. 48, 86

[241] Benjamin Recht. A Simpler Approach to Matrix Completion. *Journal of Machine Learning Research*, 12(12), 2011. 137

[242] K. Madhusudhan Reddy, Anirudh Itagi, Saransh Dabas, and Bonam Kamala Prakash. Image encryption using orthogonal hill cipher algorithm. *International Journal of Engineering & Technology*, 7(4.10):59–63, 2018. 211

[243] Irving S. Reed and Gustave Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. 110

[244] Oded Regev. New Lattice-Based Cryptographic Constructions. *Journal of the ACM (JACM)*, 51(6):899–942, 2004. 2

[245] Amirhossein Reisizadeh, Saurav Prakash, Ramtin Pedarsani, and Amir Salman Avestimehr. Coded Computation over Heterogeneous Clusters. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2408–2412, 2017. 12, 78

[246] Thomas J. Richardson, Amin Shokrollahi, and Rüdiger L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47(2):619–637, 2001. 45

[247] Ron Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 1, 123

[248] Mark Rudelson. Random Vectors in the Isotropic Position. *Journal of Functional Analysis*, 164:60–72, 1999. 228

[249] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. On Fast Leverage Score Sampling and Optimal Learning. In *Advances in Neural Information Processing Systems*, pages 5672–5682, 2018. 136, 138, 146

[250] Michael Rudow, Neophytos Charalambides, Alfred O. Hero III, and K.V. Rashmi. Compression-Informed Coded Computing. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 2177–2182, 2023. 14, 62, 78, 100, 151

[251] Michael Rudow, K.V. Rashmi, and Venkatesan Guruswami. A locality-based lens for coded computation. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1070–1075. IEEE, 2021. 12, 78

[252] Veeru Sadhanala, Yu-Xiang Wang, and Ryan Tibshirani. Graph Sparsification Approaches for Laplacian Smoothing. In *Artificial Intelligence and Statistics*, pages 1250–1259. PMLR, 2016. 220, 221

[253] Animesh Sakorikar and Lele Wang. Soft BIBD and Product Gradient Codes. *arXiv preprint arXiv:2105.05231*, 2022. 73, 77, 150, 194

[254] Yeray Cachón Santana. Orthogonal Matrix in Cryptography. *arXiv preprint arXiv:1401.5787*, 2014. 211

[255] Tamás Sarlós. Improved Approximation Algorithms for Large Matrices via Random Projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 143–152. IEEE, 2006. 49, 78, 134

[256] Reent Schlegel, Siddhartha Kumar, Eirik Rosnes, and Alexandre Graell i Amat. CodedPaddedFL and CodedSecAgg: Straggler Mitigation and Secure Aggregation in Federated Learning. *arXiv e-prints*, pages arXiv–2112, 2021. 103, 104, 105, 106

[257] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014. 72, 167, 169

[258] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979. 1, 122

[259] Claude Elwood Shannon. A Mathematical Theory of Communication. *The Bell system technical journal*, 27(3):379–423, 1948. 1, 2

[260] Claude Elwood Shannon. Communication Theory of Secrecy Systems. *The Bell System Technical Journal*, 28(4):656–715, 1949. 1

[261] Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. *Carnegie Mellon University, Tech. Rep.*, 1994. 115, 117

[262] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017. 80

[263] Mehrdad Showkatbakhsh, Can Karakus, and Suhas Diggavi. Privacy-Utility Trade-off of Linear Regression under Random Projections and Additive Noise. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 186–190. IEEE, 2018. 80

[264] Michael Sipser and Daniel A. Spielman. Expander Codes. *IEEE Transactions on Information Theory*, 42(6), 1996. 2

[265] Jinhyun So, Basak Guler, A. Salman Avestimehr, and Payman Mohassel. CodedPrivateML: A Fast and Privacy-Preserving Framework for Distributed Machine Learning. *arXiv preprint arXiv:1902.00641*, 2019. 211

[266] Mahdi Soleymani, Ramy E. Ali, Hessam Mahdavifar, and A. Salman Avestimehr. List-Decodable Coded Computing: Breaking the Adversarial Toleration Barrier. *IEEE Journal on Selected Areas in Information Theory*, 2(3):867–878, 2021. 107, 132

[267] Mahdi Soleymani, Hessam Mahdavifar, and A. Salman Avestimehr. Analog Lagrange Coded Computing. *IEEE Journal on Selected Areas in Information Theory*, 2(1):283–295, 2021. 12, 107

[268] Mahdi Soleymani, Hessam Mahdavifar, and A. Salman Avestimehr. Analog Secret Sharing with Applications to Private Distributed Learning. *IEEE Transactions on Information Forensics and Security*, 17:1893–1904, 2022. 122

[269] Kyungrak Son and Aditya Ramamoorthy. Coded matrix computation with gradient coding. *arXiv preprint arXiv:2304.13685*, 2023. 45, 150

[270] Daniel A. Spielman. Spectral Graph Theory and its Applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 29–38, 2007. 6

[271] Daniel A. Spielman. Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 2698–2722. World Scientific, 2010. 6, 147, 221, 227

[272] Daniel A. Spielman. Spectral Graph Theory. *Combinatorial scientific computing*, 18:18, 2012. 6

[273] Daniel A. Spielman and Nikhil Srivastava. Graph Sparsification by Effective Resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. 6, 134, 136, 138, 146, 147, 148, 221, 222, 228

[274] Daniel A. Spielman and Shang-Hua Teng. Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004. 6, 220

[275] Daniel A. Spielman and Shang-Hua Teng. Spectral Sparsification of Graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. ix, 6, 134, 221

[276] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969. 100, 133, 152, 210

[277] Adarsh M. Subramaniam, Anoosheh Heidarzadeh, and Krishna R. Narayanan. Random Khatri-Rao-Product Codes for Numerically-Stable Distributed Matrix Multiplication. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 253–259. IEEE, 2019. 14

[278] Adarsh M. Subramaniam, Anoosheh Heidarzadeh, Asit Kumar Pradhan, and Krishna R. Narayanan. Product Lagrange Coded Computing. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 197–202, 2020. 107

[279] Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient Coding: Avoiding Stragglers in Distributed Learning. In *International Conference on Machine Learning*, pages 3368–3376, 2017. 8, 12, 13, 14, 15, 16, 17, 20, 21, 24, 26, 27, 29, 30, 42, 44, 45, 49, 79, 82, 107, 111, 128, 149, 161, 165, 211

[280] Busra Tegin, Eduin E. Hernandez, Stefano Rini, and Tolga M. Duman. Straggler Mitigation through Unequal Error Protection for Distributed Matrix Multiplication. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6, 2021. 14

[281] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997. 115, 118, 121

[282] Joel A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011. 79

[283] Joel A. Tropp. User-Friendly Tail Bounds for Sums of Random Matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012. 228

[284] Santosh S Vempala. *The Random Projection Method*, volume 65. American Mathematical Soc., 2005. 78

[285] Bogdan Vioreanu. *Spectra of Multiplication Operators as a Numerical Tool*. PhD thesis, Yale University, 2012. ii

[286] Nisheeth K. Vishnoi. $Lx = b$. *Foundations and Trends® in Theoretical Computer Science*, 8(1-2):1–141, 2013. 227

[287] Ulrike Von Luxburg. A Tutorial on Spectral Clustering. *Statistics and computing*, 17(4):395–416, 2007. 220

[288] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. Low Latency via Redundancy. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 283–294. ACM, 2013. 12

[289] Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. ErasureHead: Distributed Gradient Descent without Delays Using Approximate Gradient Coding. *arXiv preprint arXiv:1901.09671*, 2019. 12, 79, 211

[290] Shusen Wang. A Practical Guide to Randomized Matrix Computations with MATLAB Implementations. *arXiv preprint arXiv:1505.07570*, 2015. 6, 48, 195

[291] Sinong Wang, Jiashang Liu, and Ness Shroff. Coded Sparse Matrix Multiplication. In *International Conference on Machine Learning*, pages 5152–5160. PMLR, 2018. 12

[292] Sinong Wang, Jiashang Liu, and Ness Shroff. Fundamental Limits of Approximate Gradient Coding. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–22, 2019. 12, 79, 211

[293] Stephen Jay Wiesner. *Experimental test of the rotational invariance of the weak interaction*. PhD thesis, Columbia University, 1972. 2

[294] David P. Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Theoretical Computer Science*, 10(1-2):1–157, 2014. 5, 6, 48, 54, 55, 59, 60, 78, 84, 89, 134, 135, 139, 171, 195, 196, 223

[295] Marvin Xhemrishi, Alexandre Graell i Amat, Eirik Rosnes, and Antonia Wachter-Zeh. Computational Code-Based Privacy in Coded Federated Learning. *arXiv preprint arXiv:2202.13798*, 2022. 103, 105, 106, 120

[296] Yaoqing Yang, Pulkit Grover, and Soummya Kar. Coded Distributed Computing for Inverse Problems. In *Advances in Neural Information Processing Systems*, volume 30, pages 709–719. Curran Associates, Inc., 2017. 107

[297] Yaoqing Yang, Pulkit Grover, and Soummya Kar. Computing linear transformations with unreliable components. *IEEE Trans. Inf. Theory*, 63(6):3729–3756, 2017. 12

[298] Min Ye and Emmanuel Abbe. Communication-Computation Efficient Gradient Coding. In *International Conference on Machine Learning*, pages 5610–5619. PMLR, 2018. 12, 79, 211

[299] Qian Yu. *Coded Computing: A Transformative Framework for Resilient, Secure, Private, and Communication Efficient Large Scale Distributed Computing*. PhD thesis, University of Southern California, 2020. 4

[300] Qian Yu and A. Salman Avestimehr. Harmonic Coding: An Optimal Linear Code for Privacy-Preserving Gradient-Type Computation. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1102–1106. IEEE, 2019. 80

[301] Qian Yu and A. Salman Avestimehr. Entangled Polynomial Codes for Secure, Private, and Batch Distributed Matrix Multiplication: Breaking the "Cubic" Barrier. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 245–250. IEEE, 2020. 14

[302] Qian Yu, Songze Li, Netanel Raviv, Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and A. Salman Avestimehr. Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1215–1225. PMLR, 2019. 12, 78, 102, 107, 122, 124

[303] Qian Yu, Mohammad Maddah-Ali, and Salman Avestimehr. Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication. In *Advances in Neural Information Processing Systems*, pages 4403–4413, 2017. 14, 33, 37, 38, 43, 102, 108, 126, 128, 218, 219

[304] Qian Yu, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding. *IEEE Transactions on Information Theory*, 66(3):1920–1933, 2020. 14, 44, 59, 108

[305] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving MapReduce Performance in Heterogeneous Environments. In *USENIX Association*, OSDI'08, page 29–42, 2008. 12, 49

[306] Shuheng Zhou, Larry Wasserman, and John Lafferty. Compressed Regression. In *Advances in Neural Information Processing Systems*, volume 20, 2008. 80, 147

[307] Jinbao Zhu and Songze Li. Generalized Lagrange Coded Computing: A Flexible Computation-Communication Tradeoff. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 832–837, 2022. 107