Imperial College London

Department of Electrical and Electronic Engineering

Efficient, Secure and Private Distributed Computation and Machine Learning

by

Burak Hasırcıoğlu

January 2024

Supervised by

Prof. Deniz Gündüz

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London and the Diploma of Imperial College London

Abstract

Distributed computation is the process of breaking down computational tasks and executing them across multiple nodes, rather than relying on a single machine. This approach is essential in the age of big data, where the large volume and variety of data can overwhelm individual machines. However, preserving privacy remains a critical concern in distributed computation, as there is a risk that adversarial nodes may attempt further analysis of the data to learn about the individuals owning the data. Additionally, the involvement of multiple nodes in the process can lead to arbitrary delays in computation if any of the nodes' service is unreliable. This dissertation aims to address specific research problems in the areas of efficiency, security, and privacy in distributed computation and federated learning.

The research presented in this dissertation is two-fold. First, we delve into the realm of coded computation, a framework that has been extensively used to address reliability and security issues in distributed computation. Our focus is on proposing a novel coding scheme called bivariate polynomial codes that is designed to address two major problems in distributed matrix multiplication, namely, straggler mitigation and data security. Unlike prior polynomial coding schemes, our scheme efficiently utilizes all worker nodes, including stragglers, in a storage- and upload-cost-efficient manner.

Then, we shift our focus to private federated learning, which is a technique allowing for private and collaborative training of models on decentralized data sources without requiring to transfer the data to a central location. We first investigate privacy amplification via client sampling using over-the-air computation to provide anonymity. We then theoretically analyse the joint effect of client and local dataset sampling on privacy amplification in conventional federated learning settings. Finally, we propose a compression scheme based on subtractive dithering, which provides differential privacy guarantees and is communication-efficient for private federated learning.

Acknowledgements

I am deeply grateful to my supervisor, Deniz Gündüz, for his unwavering support since the beginning of my PhD journey. In addition to his invaluable guidance and advice, he empowered me to explore and pursue areas that captured my interest, providing the freedom and encouragement to do so. His continued support, both in technical and personal aspects, has been pivotal in shaping my research and career. Furthermore, beyond being an exceptional supervisor, he has been a remarkable colleague with whom I could brainstorm, and our discussions consistently broadened my vision.

I extend my sincere thanks to my examiners, Salim El Rouayheb and Cong Ling, for their valuable time, constructive comments and a thorough evaluation of my dissertation.

I am also grateful to Jesús Gómez Vilardebó for the insightful discussions and guidance while working on coded computation together. It has been a pleasure to collaborate with him.

My gratitude also goes to Barış Coşkun and Ali Torkamani for supervising my project during my internship at Amazon Web Services and also for their kind and warm support.

I am thankful for the funding from The Scientific and Technological Research Council of Türkiye (TÜBİTAK) through the 2213 Overseas Graduate Scholarship Program for the first two years of my PhD studies. I appreciate the friendly assistance of the management and staff of TÜBİTAK and TÜBİTAK BİDEB in navigating the official procedures.

I am also thankful to the Department of Electrical and Electronic Engineering (EEE) of Imperial College London for the funding starting from my third year. I am grateful to our research group administrator, Joan O'Brien, and all the EEE department staff for their help and patience.

I would like to take to opportunity to express my gratitude to all the teachers and professors who have contributed to my academic journey from primary school to my PhD. Their guidance has been invaluable.

I would like to warmly thank Emre, Can, Ecenaz, Murat, Zaid, Mihajlo, David, Nitish, Mohammad Malekzadeh, Selim, Hakan, Yasin, Roy, Mohammad Amiri, Sreejith, Junlin, Waleed, Rui, Szymon, Anastasiia, Yassine, Yuxuan, Mikolaj, Edoardo, Marcello, Kate, Chenghong, Haotian, Li, Vivian, Qianqian, Samuel, Joan, Yulin, Touraj, Anouar and

Mahdi for their companionship and friendship in London. I also appreciate the support and friendship of Buğra, Ecenaz, Joseph, Ashish, Abhi, Thanos, and Mikhail during my stay in New York. Special thanks to Cenk for his consistent visits to each city where I was located and (mostly) relaxing conversations in person and virtually. I am also grateful for the friendship and enjoyable chats with Semih and Serkan.

Last but not least, I want to express my deepest gratitude to my family: my wife Betül, my sister Hatice, my mother Emir and my father Mustafa for their unconditional love and support throughout my life. Without their emotional support and encouragement, none of my achievements would have been possible.

Copyright

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Declaration of Originality

I, Burak Hasırcıoğlu, declare that this thesis titled, "Efficient, Secure and Private Distributed Machine Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Imperial College of Science, Technology and Medicine.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Burak Hasırcıoğlu

Date: April 12, 2024

Contents

List of Figures

List of Tables

Glossary

ACT average computation time. 103, 105, 106, 108–110

B-PROC Bivariate Product Coding. 11, 59, 61, 62, 68, 69, 71–73, 75, 88, 91

BPC-HO Bivariate Polynomial Coding with Horizontal Computation Order. 65, 68–74, 91, 96, 164

BPC-NZO Bivariate Polynomial Coding with N-zig-zag Computation Order. 66, 68–74, 91, 96, 164

BPC-VO Bivariate Polynomial Coding with Vertical Computation Order. 64, 68–74, 91, 96, 164

BPC-ZZO Bivariate Polynomial Coding with Z-zig-zag Computation Order. 67–74, 91, 96, 164

C-CDF complementary cumulative distribution function. 136

CDF cumulative distribution function. 46, 159

CNN convolutional neural network. 144

CS Centralized Shuffling. 138–141

CSA cross subspace alignment. 31

CSI channel state information. 121, 123, 125

DP differential privacy. 41–47, 120, 121, 125–128, 130–133, 136–138, 141, 142, 144, 151–153, 155–157, 159–161

DP-DSGD differentially private distributed stochastic gradient descent. 133, 135, 136, 139, 141, 142, 145

FedAvg federated averaging. 39

FL federated learning. 7, 37–41, 46, 47, 120, 121, 126, 127, 129–131, 133, 134, 152–156, 160, 161, 163, 165–167

GASP Gap Additive Secure Polynomial. 9, 34, 93, 101, 102

GM Gaussian Mechanism. 126

HE homomorphic encryption. 41

i.i.d. independently and identically distributed. 40, 122, 124, 134, 135, 137

MAC multiple access channel. 121, 122

ML machine learning. 37, 39, 41, 155

MM-GASP multi-message GASP. 11, 101–110, 119

OAC over-the-air computation. 120, 121, 126–128, 131, 165, 166

OLS Only Local Sampling. 137–140, 144–146, 152

PS parameter server. 9, 37–41, 120–125, 128–131, 133–137, 139, 142, 143, 154–160, 162, 166, 167

RDP Rényi differential privacy. 45, 46, 126, 127

S-MPC secure multi-party computation. 41

S-UPC Secure Univariate Polynomial Codes. 33, 34, 36, 93

SBP Secure Bivariate Polynomial. 11, 93, 97, 100, 102–110, 119

SDMM secure distributed matrix multiplication. 9, 92–97, 101, 119

SGD stochastic gradient descent. 38, 121, 122, 130, 145, 146, 156

TEE trusted execution environments. 155

UCB upload cost budget. 103, 105–110

UPC Univariate Polynomial Codes. 8, 28–31, 33, 36, 54–56, 92

UPC-PC Univariate Polynomial Codes with Partial Computations. 55–57, 73–75

WCS Weak Client Sampling. 138–146

To my family.

Chapter 1

Introduction

The concept of distributed computation has taken on an increasingly crucial role in our data-driven age. Essentially, it refers to the process of breaking down computational tasks and executing them across multiple machines or nodes, rather than relying on a single machine. This approach has proven particularly essential in the age of big data, where the sheer volume, velocity, and variety of data can overwhelm individual machines.

By utilizing distributed computation, large-scale data processing tasks can be efficiently handled, including those required for machine learning applications. This approach enables faster computations, enhances system reliability, and maximizes resource utilization. In turn, it has become a pivotal component across a range of sectors, from academia to industry.

Distributed computation takes various forms, each with its unique characteristics. One such form involves a single client who employs multiple compute nodes to complete a task. In this scenario, the client is responsible for providing all the necessary data to the compute nodes to obtain the final result. In contrast, the second form entails a central node that only coordinates the computations, while many clients utilize their own data to compute, which is typical in federated learning.

However, regardless of the distributed computation form adopted, preserving privacy remains a critical concern. In the former setting, where nodes receive the data, there is a risk that adversarial nodes may attempt further analysis of the data to learn sensitive information about the individual owning the data. Conversely, in the latter form, sharing the result of local processing with the central node may reveal information about the local dataset, even though local computations keep the data private. As a result, while distributed computation is essential to process large-scale data at reasonable speeds, it poses a significant risk of privacy breaches. Thus, privacy in distributed computations is a critical challenge that needs addressing.

Distributed computation is primarily designed to enhance processing speed. However, it is paradoxical that the involvement of multiple nodes in the process can lead to arbitrary delays in computation if any of the nodes are unreliable in terms of their service. The primary challenges in such systems are slow machines and their failures, which affect the dependability of the overall system. Therefore, in distributed computation, another crucial problem to address is the issue of failed or slow nodes by ensuring that their absence can be easily compensated by other nodes. This is essential to maintain the efficiency and effectiveness of distributed computation systems.

This dissertation aims to address specific research problems in the areas of efficiency, security, and privacy in distributed computation and federated learning. The research is two-fold, with the first part of the dissertation focusing on coded computation, a framework used to address reliability and security in distributed computation. Specifically, the research focuses on realizing distributed matrix multiplication, which is one of the fundamental blocks of many data processing and machine learning frameworks. Chapters 3 and 4 present the results of this research.

The second part of the dissertation focuses on private federated learning. Federated learning can be seen as an application of distributed computation to the machine learning domain. It is a paradigm that enables the collaborative training of models on decentralized data sources without the need to transfer the data to a central location. This approach has gained significant attention in recent years due to its potential to speed up learning and address privacy concerns associated with centralized data storage and processing. Hence, we focus on techniques to guarantee and enhance the privacy of clients in various settings, including wireless edge training and communication efficiency. The research in this area is particularly important given the increasing use of federated learning in various applications, including healthcare and finance, where privacy concerns are paramount. In Chapters 5 to 7, we present the results of our research on this topic.

In the following sections, we provide an overview of the dissertation, presenting an outline and short summaries of the problems considered, and list the papers published during this Ph.D. course, relating them to the relevant chapters.

1.1 Overview

Firstly, in Chapter 2, we present the necessary background and related work, which are necessary to follow our results in the subsequent chapters. Specifically, in Section 2.1, we introduce the notion of coded computation together with the most popular solution in this area, namely polynomial codes. We also present several previous schemes from earlier literature that use coded computation to mitigate stragglers, i.e., slow workers, and to improve security. Then, in Section 2.2, we give an introduction to federated learning and discuss some of the main challenges faced in this area by referring to related work. Finally, in Section 2.3, we introduce differential privacy and its common variants. We also discuss Gaussian mechanism, one of the most prominent techniques used to achieve differential privacy guarantees.

In Chapter 3, we present our work on bivariate polynomial codes for straggler mitigation in distributed matrix multiplication. We motivate our work by showing that previously proposed univariate polynomial codes are suboptimal when it comes to effectively utilizing all the resources in the distributed computation systems. That is because they cannot adapt the multi-message schemes, which are essential to fully utilize the slow computing nodes in the system. Then, to address these challenges, we introduce a novel coding technique called bivariate polynomial codes. We theoretically show that the proposed scheme efficiently utilizes the storage capacities of the worker nodes and reduces the upload cost from the client to worker nodes significantly. We further demonstrate, via experiments, that the proposed scheme speeds up the target matrix multiplication task significantly under fixed storage capacities of the worker nodes.

The work presented in Chapter 3 has led to two conference and one journal publications, which are listed below:

- Hasircioglu, B., Gómez-Vilardebó, J., and Gündüz, D. (2020). Bivariate Polynomial Coding for Straggler Exploitation with Heterogeneous Workers. *2020 IEEE International Symposium on Information Theory (ISIT)*, 251-256.
- Hasircioglu, B., Gómez-Vilardebó, J., and Gündüz, D. (2020). Bivariate Hermitian Polynomial Coding for Efficient Distributed Matrix Multiplication. *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 1-6.
- Hasircioglu, B., Gómez-Vilardebó, J., and Gunduz, D. (2020). Bivariate Polynomial Coding for Efficient Distributed Matrix Multiplication. *IEEE Journal on Selected Areas in Information Theory*, 2, 814-829.

Motivated by the significant improvement brought by the bivariate polynomial codes to the straggler mitigation problem, in Chapter 4, we extend the setting such that the security of the input data is ensured as well as the scheme still provides straggler mitigation and efficient straggler utilisation. Such an extension of the requirements imposes the condition that the employed coding scheme is defined over a finite field. Hence, in this section, we extend bivariate polynomial codes over a finite field and provide all necessary proofs for the new setting, including the security proof. We further present extensive experimental results demonstrating that the proposed scheme is superior to the existing schemes in the literature in terms of overall computation time under fixed storage or upload constraints.

Throughout this dissertation, we use the term "security" to refer to the prevention of any information leakage to the worker nodes that are hired for distributed computations. When computations are deemed "secure," it implies that the final result is not impacted by this security guarantee. In other words, the resulting computation is exactly the same as if no security measures had been taken. This differs from the concept of privacy, which is used in federated learning in the subsequent chapters, in the sense that privacy-preserving algorithms perturb the output irreversibly.

The results presented in Chapter 4 has led to the following publications:

- Hasircioglu, B., Gómez-Vilardebó, J., and Gunduz, D. (2021). Speeding Up Private Distributed Matrix Multiplication via Bivariate Polynomial Codes. *2021 IEEE International Symposium on Information Theory (ISIT)*, 1853-1858.
- Hasircioglu, B., Gómez-Vilardebó, J., and Gündüz, D. (2021). Bivariate Polynomial Codes for Secure Distributed Matrix Multiplication. *IEEE Journal on Selected Areas in Communications*, 40, 955-967.

In Chapter 5, we shift the focus of the dissertation to federated learning. Unlike conventional federated learning, in this chapter, we consider a federated learning setting over a wireless medium for edge devices. The chapter aims to use the interference phenomena of the signals transmitted wirelessly to provide anonymity for the participating clients, thereby enhancing the privacy guarantees of federated learning. We demonstrate that when anonymity is ensured through certain techniques, the differential privacy guarantees of the client's data against the parameter server are significantly amplified by employing device and local dataset sampling.

The work presented in Chapter 5 has led to the following publication:

- Hasircioglu, B., and Gündüz, D. (2020). Private Wireless Federated Learning with Anonymous Over-the-Air Computation. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5195-5199.

In Chapter 6, we focus on conventional federated learning and investigate how jointly sampling participating devices and their local datasets can amplify privacy. We point out that when we sample participating devices before sampling their local datasets, the data sampling becomes non-uniform. This non-uniform data sampling invalidates the analysis of the privacy amplification effects of dataset sampling provided in previous work. Therefore, in this chapter, we provide a theoretical analysis of the privacy amplification effects of non-uniform sampling and present some experimental results that support our theoretical claims.

The work presented in Chapter 6 has led to the following pre-print, which is still under submission process:

- Hasircioglu, B., and Gunduz, D. (2022). Privacy Amplification via Random Participation in Federated Learning. *ArXiv*, abs/2205.01556.

In Chapter 7, we study the communication efficiency of differentially private federated learning, and we propose a compression technique that uses subtractive dithering and quantization. The technique ensures that the distortion due to quantization provides differential privacy of the client data against other clients. Additionally, the privacy guarantees of our method remain valid even for the deployed model. We have both theoretically and experimentally shown that the accuracy of the final model trained using our technique matches that of the full-precision differentially private federated learning setting.

The work presented in Chapter 7 has led to the following publication:

- Hasircioglu, B., and Gunduz, D. (2024). Communication Efficient Private Federated Learning Using Dithering. *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

Finally, in Chapter 8, we conclude the dissertation by summarizing the achievements of this Ph.D. course and by discussing the future directions of the problems studied.

Moreover, we would like to note that, during this Ph.D. course, the following papers were also published but their content is not included in this dissertation:

- Malekzadeh, M., Hasircioglu, B., Mital, N., Katarya, K., Ozfatura, M.E., and Gunduz, D. (2021). Dopamine: Differentially Private Federated Learning on Medical Data. The *Second AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI-21).*
- Yilmaz, S.F., Hasircioglu, B., and Gunduz, D. (2022). Over-the-Air Ensemble Inference with Model Privacy. *2022 IEEE International Symposium on Information Theory (ISIT)*, 1265-1270.

1.2 Notation

Throughout the dissertation, we denote the set of consecutive integers, i.e., $\{a, a+1, a+2, \cdots, b-2, b-1, b\}$, by $[a:b]$. For matrices and vectors, we use bold lowercase letters for vectors and bold uppercase letters for matrices. We use calligraphic letters for sets. All other notation is introduced in the related chapters where necessary.

Chapter 2

Background

2.1 Coded Computation

Modern machine learning and computational tasks rely heavily on distributed computing to handle the enormous datasets and model sizes that require computational power beyond that of a single machine. To tackle the most demanding computational tasks, such as matrix multiplication, multiple dedicated servers known as *workers* are utilized. However, computational processes can be hindered by *stragglers* which experience unpredictable service times due to partial or complete failures, or concurrent computations assigned to them, ultimately creating a bottleneck for distributed computation.

Apart from computational challenges, data security poses another obstacle in distributed computing. Since computational tasks require sharing data chunks with worker nodes, sensitive information may be contained in these chunks that the data owner needs to keep private. Even partial sharing of this data with workers can result in a privacy breach. In some cases, even if individual workers are not able to extract any information from the data provided to them, a collaborative effort from workers exchanging information with other workers, referred to as collusion, may result in a leakage.

Coded computation is a powerful framework that leverages concepts from channel coding to address the challenges of distributed computation, such as straggler mitigation and security. The core idea behind coded computation is to introduce redundancy into computation tasks, allowing for the failure or slowness of some worker nodes to be compensated for by other worker nodes. This redundancy can also be used to enhance security by ensuring that computations assigned to a worker do not reveal any sensitive information, but when all necessary workers respond, the collective result is meaningful. As such, coded computation is closely related to multi-party computation, which also aims to protect the privacy of data during distributed computation. By using coded computation, researchers and practitioners can improve the efficiency, reliability, and security of distributed computation systems, making them more suitable for various applications in fields such as machine learning, data analytics, and cryptography.

In this dissertation, we primarily concentrate on matrix multiplication, utilizing this operation to illustrate both existing and proposed coded computation solutions. We have chosen to focus on matrix multiplication due to its pivotal role in numerous engineering and data science challenges. For instance, in the realm of scientific computing, matrix multiplication is employed to solve extensive linear systems of equations. Similarly, in pattern recognition, procedures such as principal component analysis and linear discriminant analysis incorporate matrix multiplication as a key subroutine. Furthermore, in machine learning, operations like matrix factorization and neural network training fundamentally rely on matrix multiplication. Therefore, the development of techniques to expedite this widely used subroutine not only enhances computational speed but also bolsters data security in a multitude of engineering tasks.

The next subsection is dedicated to explaining the concept of coded computation by providing a general framework. Then, we will delve into polynomial codes, which are a specific type of error-correcting codes that can be employed to improve the computational efficiency and security of matrix multiplication. We will explore the different variations of polynomial codes, highlighting their advantages and disadvantages. Overall, this section aims to provide a thorough understanding of how coded computation and polynomial codes can be leveraged to optimize the performance of distributed computing systems.

2.1.1 A framework for coded computation

The process of distributed matrix multiplication through coded computation can be broken down into three phases: encoding, computation, and decoding. Suppose a client wishes to multiply two large matrices, $\mathbf{A}$ and $\mathbf{B}$.

The first step is to encode matrices $\mathbf{A}$ and $\mathbf{B}$ using an encoder function, $E(\mathbf{A}, \mathbf{B})$, resulting in p and r coded matrices of $\mathbf{A}$ and $\mathbf{B}$, respectively, denoted as $\tilde{\mathbf{A}}_1$, $\tilde{\mathbf{A}}_2$, $\cdots$, $\tilde{\mathbf{A}}_p$, and $\tilde{\mathbf{B}}_1$, $\tilde{\mathbf{B}}_2$, $\cdots$, $\tilde{\mathbf{B}}_r$. The dimensions of coded matrices and the specific parameters p and r depend on the coding schemes used, which vary and will be discussed in detail later on. Assuming the client has N worker nodes available, the encoded matrices are then assigned to these workers. Each worker node i is sent a subset of coded matrices of $\mathbf{A}$ and $\mathbf{B}$, indexed by the sets $\mathcal{I}_{A,i}$ and $\mathcal{I}_{B,i}$. Specifically, to worker i, the client sends the coded matrices $\tilde{\mathbf{A}}_j$, $j \in \mathcal{I}_{A,i}$ and $\tilde{\mathbf{B}}_k$, $k \in \mathcal{I}_{B,i}$.

During the computation phase, each worker node performs the local computations assigned to it by using the communicated matrix partitions. This process can be represented by a function, denoted as $C\left(\{\tilde{\mathbf{A}}_j \mid j \in \mathcal{I}_{A,i}\}, \{\tilde{\mathbf{B}}_k \mid k \in \mathcal{I}_{B,i}\}\right)$, which generates a set of results for client i, referred to as $\mathcal{C}_i$. Once the computations are completed, each element in the result set $\mathcal{C}_i$ is sent back to the client.

Finally, during the decoding phase, the client waits for the workers to send a sufficient number of their responses, which is denoted as the recovery threshold, R_{th}. These responses are represented by $\mathcal{R} \subset \{\cup_{i\in[1:N]}\mathcal{C}_i\}$ such that $|\mathcal{R}| = R_{th}$. Once the required number of responses have been received, the client decodes the final computation it needs from these responses. Specifically, the decoding function D will be applied to $\mathcal{R}$ to decode $\mathbf{AB}$, i.e., $\mathbf{AB} = D(\mathcal{R})$.

To counteract the impact of stragglers, it's crucial to assign redundant computations to the workers. This redundancy ensures that slow or failed workers can be offset by other computations, implying that the cardinality of $\{\cup_{i\in[1:N]}\mathcal{C}_i\}$ should exceed that of $\mathcal{R}$, and their difference characterizes the straggler tolerance capability of the scheme. A similar principle is also applicable to data security scenarios. In these instances, redundant random matrices are incorporated and encoded alongside the original matrices. These random matrices serve to obscure the original matrices from the workers, albeit at the cost of a slight increase in the recovery threshold. Straggler mitigation and security measures can also be implemented concurrently to achieve both reliable and secure matrix multiplication. In the rest of this section, we will introduce polynomial codes and explore their applications in straggler tolerance and security scenarios.

2.1.2 Polynomial Codes

Polynomial codes are a class of codes that are employed in coded computation as an underlying coding scheme. In this family of codes, the encoding operation $E(\mathbf{A}, \mathbf{B})$ first partitions the matrices $\mathbf{A}$ and $\mathbf{B}$ into smaller submatrices and then using these submatrices, two encoding polynomials $\mathbf{A}(x)$ and $\mathbf{B}(x)$ are generated. In this background chapter, we only consider univariate polynomial codes but in the following chapters, we extend it to bivariate cases as well. Hence, throughout this chapter, both encoding polynomials are expressed in terms of the same variable x. In a general form, when $\mathbf{A}$ is divided into K, and $\mathbf{B}$ into L submatrices, these polynomials can be expressed as follows.

$$\mathbf{A}(x) = \mathbf{A}_1 x^{\alpha_1} + \mathbf{A}_2 x^{\alpha_2} + \cdots + \mathbf{A}_K x^{\alpha_K} \tag{2.1}$$

$$\mathbf{B}(x) = \mathbf{B}_1 x^{\beta_1} + \mathbf{B}_2 x^{\beta_2} + \cdots + \mathbf{B}_L x^{\beta_L}. \tag{2.2}$$

Note that we do not specify yet how the matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned. Indeed, there are several ways of doing that such as row-wise and column-wise partitioning, which we elaborate on later. Moreover, we keep the exponents of the variable in a general form, i.e., α_i's and β_j's. The way of choosing these exponents depends on the specific coding scheme employed.

To generate $\tilde{\mathbf{A}}_i$'s and $\tilde{\mathbf{B}}_i$'s, we evaluate encoding polynomials at a set of evaluation points x_i, where i belongs to the range $[1:p]$. Each evaluation is then considered as a coded matrix, i.e., $\tilde{\mathbf{A}}_i = \mathbf{A}(x_i)$ and $\tilde{\mathbf{B}}_i = \mathbf{B}(x_i)$. It is important to note that since we use univariate polynomials, p is equal to r in this case since evaluating the encoding polynomials at different evaluation points would not be useful for our purposes.

In the computation phase, each subtask is defined as a multiplication between $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$. Each worker may be assigned to compute one or more of such subtasks.

Finally, in the decoding phase, the client aims to interpolate the polynomial that is the multiplication of the encoding polynomials,

$$\mathbf{A}(x)\mathbf{B}(x) = \sum_{i\in[1:K]} \sum_{j\in[1:L]} \mathbf{A}_i\mathbf{B}_j x^{\alpha_i+\beta_j}. \tag{2.3}$$

To interpolate it, the number of evaluations required is equal to the number of coefficients of this polynomial. Hence, the recovery threshold, R_{th} becomes the number of coefficients of $\mathbf{A}(x)\mathbf{B}(x)$, and it depends on K, L and the choice of α_i's and β_j's.

Next, we investigate some primary polynomial coding schemes proposed to mitigate stragglers.

2.1.3 Polynomial Codes for Straggler Mitigation

Assuming all workers start computing simultaneously, we define *computation time* as the time from the start until the client collects sufficiently many computations (i.e., as many as R_{th}) that allow for decoding $\mathbf{AB}$. For brevity, this definition excludes communication time, as well as encoding and decoding times. Stragglers can increase computation time arbitrarily and hence, by employing polynomial codes, we aim to combat their effects. As we discussed earlier, it can be done by assigning redundant computations to worker nodes. In other words, stragglers can be considered as random erasures, and computation time can be improved by utilizing these redundant computations in the case of slow or failed worker nodes, similar to channel coding techniques for erasure channels.

In the following, we present fundamental polynomial coding techniques differing in their matrix partitioning routines, which results in differences in their several performance metrics. Since, in general, the matrices to be multiplied for various scientific tasks are in double precision by default, in this subsection, we assume that the matrices and the polynomials under interest are all in real numbers, i.e., $x \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{r \times s}$ and $\mathbf{B} \in \mathbb{R}^{s \times c}$.

2.1.3.1 Univariate Polynomial Codes (UPC)

In [2], polynomial codes are proposed for combating stragglers during the distributed multiplication of matrices $\mathbf{A}$ and $\mathbf{B}$. In this scheme, a client partitions $\mathbf{A}$ row-wise and $\mathbf{B}$ column-wise. That is, $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T & \cdots & \mathbf{A}_K^T \end{bmatrix}^T$ and $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_L \end{bmatrix}$. In this dissertation, we name this scheme as UPC. Given the partitions of $\mathbf{A}$ and $\mathbf{B}$, their multiplication $\mathbf{AB}$ can be expressed in terms of their submatrices as

$$\mathbf{AB} = \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_1\mathbf{B}_2 & \cdots & \mathbf{A}_1\mathbf{B}_L \\ \mathbf{A}_2\mathbf{B}_1 & \mathbf{A}_2\mathbf{B}_2 & \cdots & \mathbf{A}_2\mathbf{B}_L \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{A}_K\mathbf{B}_1 & \mathbf{A}_K\mathbf{B}_2 & \cdots & \mathbf{A}_K\mathbf{B}_L \end{bmatrix}. \tag{2.4}$$

This implies all pairwise multiplications between submatrices of $\mathbf{A}$ and $\mathbf{B}$ are required to be known for decoding $\mathbf{AB}$. To keep, all the pairwise multiplication decodable, the following encoding polynomials are constructed.

$$\mathbf{A}(x) = \mathbf{A}_1 + \mathbf{A}_2 x + \cdots + \mathbf{A}_K x^{K-1}, \tag{2.5}$$

$$\mathbf{B}(x) = \mathbf{B}_1 + \mathbf{B}_2 x^K + \cdots + \mathbf{B}_i x^{(i-1)K} + \cdots + \mathbf{B}_L x^{(L-1)K}. \tag{2.6}$$

Hence, the exponents of the encoding polynomials are designed such that $\alpha_i = i - 1$, $i \in [1 : K]$ and $\beta_j = (j-1)K$, $j \in [1 : L]$.

To worker i, the client sends $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$, $i \in [1 : N]$, for some distinct $x_i \in \mathbb{R}$. Thus, every worker receives one coded partition of $\mathbf{A}$ and one coded partition of $\mathbf{B}$, and worker i is assigned to compute $\mathbf{A}(x_i)\mathbf{B}(x_i)$. After completion of its assigned task, worker i communicates the result to the client.

The encoding polynomials impose the following polynomial to be interpolated by the client

$$\mathbf{A}(x)\mathbf{B}(x) = \sum_{i=1}^{K}\sum_{j=1}^{L} \mathbf{A}_i\mathbf{B}_j x^{i-1+K(j-1)}. \tag{2.7}$$

$\mathbf{A}(x)\mathbf{B}(x)$ has a degree $KL - 1$, and hence, it has KL distinct coefficients. This means R_{th}, which is the number of computations required to decode $\mathbf{AB}$, or equivalently, the number of workers that complete their assigned tasks in this case, is KL. Hence, the fastest KL workers' responses are enough to decode the desired multiplication $\mathbf{AB}$.

Observe that with $N > R_{th}$, this scheme can tolerate up to $N - R_{th}$ stragglers. It helps to reduce the average computation time thanks to the parallelization afforded by redundant workers.

In [2], it has been shown that this scheme is optimal in terms of the *download rate*, which is defined as the ratio of the total number of bits needed to be downloaded from the workers to the number of bits needed to represent the result of the multiplication. This is indeed not difficult to see since any coded submatrix of $\mathbf{AB}$ is $\tilde{\mathbf{A}}_i\tilde{\mathbf{B}}_i \in \mathbb{R}^{\frac{r}{K}\times\frac{c}{L}}$ and only KL of such coded multiplications are required to be downloaded from the workers. Hence, in total, the client needs to download the same number of bits to represent $\mathbf{AB} \in \mathbb{R}^{r\times c}$, which results in a download rate of 1.

2.1.3.2 MatDot Codes

Instead of the matrix partitioning employed in UPC, alternatively, the matrices $\mathbf{A}$ and $\mathbf{B}$ can be partitioned in a column-wise and row-wise manner, respectively. In [3], a coding scheme employing such partitioning called *MatDot codes* is proposed. In this coding scheme, both matrices are partitioned into K submatrices, i.e., $K = L$ and the partitioning is carried as $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_K\end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix}\mathbf{B}_1^T & \mathbf{B}_2^T & \cdots & \mathbf{B}_K^T\end{bmatrix}^T$. Hence, their multiplication can be expressed as $\mathbf{AB} = \sum_{i\in[1:K]} \mathbf{A}_i\mathbf{B}_i$. Note that unlike UPC, in MatDot codes, it is not required to know all pairwise multiplications $\mathbf{A}_i\mathbf{B}_i$ separately but instead, only their sum is sufficient. Using this fact, the encoding polynomials are constructed as follows.

$$\mathbf{A}(x) = \sum_{i\in[1:K]} \mathbf{A}_i x^{i-1} = \mathbf{A}_1 + \mathbf{A}_2 x + \cdots + \mathbf{A}_K x^{K-1} \tag{2.8}$$

$$\mathbf{B}(x) = \sum_{j\in[1:K]} \mathbf{B}_j x^{K-j} = \mathbf{B}_1 x^{K-1} + \mathbf{B}_2 x^{K-2} + \cdots + \mathbf{B}_{K-1} x + \mathbf{B}_K. \tag{2.9}$$

That is, the exponents are designed to be $\alpha_i = i - 1$ and $\beta_j = K - j$. Similarly to UPC to worker i, the client sends $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$ using distinct x_i's for each worker. After computing $\mathbf{A}(x_i)\mathbf{B}(x_i)$, client i communicates its result to the client. From the worker's responses, the client aims to interpolate the polynomial

$$\mathbf{A}(x)\mathbf{B}(x) = \sum_{i\in[1:K]} \sum_{j\in[1:K]} \mathbf{A}_i\mathbf{B}_j x^{i+K-j-1}, \tag{2.10}$$

which has a degree $2K-2$. Note that to decode $\mathbf{AB}$, only the coefficient of the monomial x^{K-1} is necessary. However, to solve for the required coefficient, one needs to interpolate all coefficients from the worker's responses. Hence, $R_{th} = 2K - 1$.

The authors of [3] show that, compared to [2], MatDot codes improve the recovery threshold. This is because, compared to KL, which has a quadratic dependency on K if $K = L$, now, MatDot codes require only $2K - 1$ workers to respond. However, in terms of the amount of computation each worker should carry out, referred to as the *computation cost*, and the download rate, UPC of [2] outperforms MatDot codes. That is because the matrix partitioning scheme employed in MatDot codes generates matrices with larger dimensions if the same K parameter is used in both schemes. This results in $\tilde{\mathbf{A}}_i\tilde{\mathbf{B}}_i \in \mathbb{R}^{r\times c}$, implying that, from each worker, the client must download the same number of bits to represent the whole $\mathbf{AB}$. Compared to UPC, this is a dramatic increase. Similarly, this also implies that the amount of computation conducted by each worker node is much larger. Still, for the scenarios in which the amount of computation assigned to a worker and the download cost are not a concern, MatDot codes may be useful since they have a smaller R_{th}.

2.1.3.3 PolyDot Codes

In [3], the authors also propose PolyDot codes as an interpolation between UPC and MatDot codes. PolyDot codes trade-off between the recovery threshold and the computation/download costs via a hybrid approach to the partitioning techniques. That is, both matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned both row-wise and column-wise as follows.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,K_c} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{A}_{K_r,1} & \mathbf{A}_{K_r,2} & \cdots & \mathbf{A}_{K_r,K_c} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} & \cdots & \mathbf{B}_{1,L_c} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{B}_{L_r,1} & \mathbf{B}_{L_r,2} & \cdots & \mathbf{B}_{L_r,L_c}. \end{bmatrix} \tag{2.11}$$

where K_c, K_r, L_c and L_r are design parameters such that $K_c = L_r$. Then, the desired multiplication becomes

$$\mathbf{AB} = \begin{bmatrix} \sum_{i\in[K_c]} \mathbf{A}_{1,i}\mathbf{B}_{i,1} & \sum_{i\in[K_c]} \mathbf{A}_{1,i}\mathbf{B}_{i,2} & \cdots & \sum_{i\in[K_c]} \mathbf{A}_{1,i}\mathbf{B}_{i,L_c} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i\in[K_c]} \mathbf{A}_{K_r,i}\mathbf{B}_{i,1} & \sum_{i\in[K_c]} \mathbf{A}_{K_r,i}\mathbf{B}_{i,2} & \cdots & \sum_{i\in[K_c]} \mathbf{A}_{K_r,i}\mathbf{B}_{i,L_c} \end{bmatrix}. \tag{2.12}$$

The encoding polynomials of PolyDot codes are constructed as

$$\mathbf{A}(x) = \sum_{i\in[K_r]}\sum_{j\in[K_c]} \mathbf{A}_{i,j} x^{i-1+K_r(j-1)} \tag{2.13}$$

$$\mathbf{B}(x) = \sum_{k\in[K_c]}\sum_{l\in[L_c]} \mathbf{B}_{k,l} x^{K_r(K_c-k)+K_r(2K_c-1)(l-1)}. \tag{2.14}$$

As before, each worker i is sent distinct evaluations $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$, and then worker i sends the result $\mathbf{A}(x_i)\mathbf{B}(x_i)$ back to the client. Hence, the client aims to interpolate

$$\mathbf{A}(x)\mathbf{B}(x) = \sum_{i\in[K_r]}\sum_{j\in[K_c]}\sum_{k\in[K_c]}\sum_{l\in[L_c]} \mathbf{A}_{i,j}\mathbf{B}_{k,l} x^{i-1+K_r(K_c-k+j-1)+K_r(2K_c-1)(l-1)}, \tag{2.15}$$

which has a degree of $2K_rK_cL_c - K_rL_c - 1$, implying $R_{th} = 2K_rK_cL_c - K_rL_c$.

Observe that in PolyDot codes, similarly to MatDot codes, not all pairwise multiplications between $\mathbf{A}_{i,j}$ and $\mathbf{B}_{k,l}$ are required to decode $\mathbf{AB}$, but also similarly to UPC, not all such multiplications are accumulated under the same monomial. In this sense, PolyDot codes are a generalization of these two coding schemes. Note that by choosing $K_r = L_c = 1$, PolyDot codes reduce to MatDot codes and by choosing $K_c = 1$, they reduce to UPC. Hence, by tweaking K_r, K_c and L_c, the recovery threshold can be traded off for the computation and download costs.

2.1.3.4 Related Literature

Numerous studies have investigated the case where both $\mathbf{A}$ and $\mathbf{B}$ can be partitioned row-wise and column-wise simultaneously. One notable example is the proposal of entangled polynomial codes in [4], which improves the recovery threshold of PolyDot codes [3] while maintaining a fixed computation cost and download rate. Additionally, [5] introduces generalized PolyDot codes that achieve the same recovery threshold as the entangled polynomial codes presented in [4].

Another area of research, as detailed in [6], pertains to the batch multiplication of matrices, specifically $\mathbf{A}_i\mathbf{B}_i$ for $i \in [1 : L]$ where $L > 1$. This study proposes cross subspace alignment (CSA) codes that provide an improved trade-off between upload-download cost when compared to the separate application of entangled polynomial codes for each multiplication task within the batch.

Polynomial codes pose a challenge in terms of numerical stability, given that polynomial interpolation forms the basis of the decoding process. As a result, ensuring numerical stability is of utmost importance for practical implementations. In order to address this concern, several schemes have been proposed for distributed coded matrix multiplication in [7–10] that offer numerically stable polynomial coding solutions. These solutions can be considered as important contributions towards enhancing the robustness of polynomial codes in practical applications.

2.1.4 Polynomial Codes for Security

In this section, we provide an overview of the polynomial coding techniques proposed for secure distributed matrix multiplication. It is worth noting that the matrices used in such computations often contain sensitive information that the client may not want to disclose. While the polynomial coding techniques presented in previous sections typically involve sharing coded matrix partitions instead of the matrices themselves, even partially sharing coded matrices can lead to information leakage to the workers. In certain scenarios, a group of workers can collude to exchange information with one another, thereby gaining insights into the multiplied matrices. Such collusion can result in information leakage even if no information is revealed to individual workers. In coded computation literature, it is common to assume an upper bound on the number of workers that can collude, denoted by T. This upper bound implies that among N available workers, no more than T can collaborate to share their received coded matrix partitions and gain access to sensitive data.

In order to tackle all these challenges, in distributed matrix multiplication, a stringent security model is employed, in which no leak of information about the matrices is allowed under the honest but curious workers. An honest but curious worker adheres to the protocol but may utilize the received coded matrices to gain knowledge about the original matrices $\mathbf{A}$ and $\mathbf{B}$. Given that at most T workers can collude, it is required that a mutual information constraint be satisfied, which ensures that there is no information leakage from the matrices. Specifically, this constraint is expressed as follows:

$$I\left(\mathbf{A},\mathbf{B};\{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i \mid i\in\mathcal{T},\mathcal{T}\subset[1:N],|\mathcal{T}|=T\}\right)=0, \tag{2.16}$$

where $I(\cdot)$ is the mutual information, and $\mathcal{T}$ is any subset with cardinality T of available worker indices. Hence, polynomial codes for data security aim to achieve this mutual information constraint while introducing as low delays as possible.

Due to the no-leakage nature of the mutual information constraint, similar to the secure multi-party computation literature, in secure polynomial coding, the matrices and the polynomials are assumed to be in a finite field, $\mathbb{F}$. That is, we have $x\in\mathbb{F}$, $\mathbf{A}\in\mathbb{F}^{r\times s}$ and $\mathbf{B}\in\mathbb{F}^{s\times c}$. Hence, before encoding the matrices, we assume they are properly quantized such that they are mapped to some element of the underlying finite field.

2.1.4.1 Secure Univariate Polynomial Codes (S-UPC)

A direct extension of UPC codes to the secure distributed matrix multiplication is proposed in [11], which we refer to as Secure Univariate Polynomial Codes (S-UPC) in this dissertation. In this work, to hide the matrices from the workers, random matrix partitions are created, and linearly encoded together with the true matrix partitions using polynomial codes. That is, as in UPC, matrices are partitioned as $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1^T & \mathbf{A}_2^T & \cdots & \mathbf{A}_K^T\end{bmatrix}^T$ and $\mathbf{B} = \begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_L\end{bmatrix}$, where $\mathbf{A}_i \in \mathbb{F}^{\frac{r}{K}\times s}$ and $\mathbf{B}_j \in \mathbb{F}^{s\times\frac{c}{L}}$. In addition, to mask $\mathbf{A}$ and $\mathbf{B}$, the client uniform randomly draws T random matrix partitions independently, $\mathbf{R}_i \in \mathbb{F}^{\frac{r}{K}\times s}$ and $\mathbf{S}_i \in \mathbb{F}^{s\times\frac{c}{L}}$, $i \in [1:T]$, respectively. Then, the encoding polynomials are constructed as follows.

$$\mathbf{A}(x) = \sum_{i\in[1:K]} \mathbf{A}_i x^{i-1} + \sum_{i\in[1:T]} \mathbf{R}_i x^{K+i-1} \tag{2.17}$$

$$\mathbf{B}(x) = \sum_{i\in[1:L]} \mathbf{B}_i x^{(K+T)(i-1)} + \sum_{i\in[1:T]} \mathbf{S}_i x^{(K+T)(L+i-1)}. \tag{2.18}$$

The client sends to the worker i, the evaluations $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$ such that the evaluation points x_i's are distinct for different workers. The worker i computes $\mathbf{A}(x_i)\mathbf{B}(x_i)$ and communicates the result to the client.

In [11], it has been shown that the sets $\{\mathbf{A}(x_i) \mid i \in \mathcal{T}, |\mathcal{T}| = T\}$ and $\{\mathbf{B}(x_i) \mid i \in \mathcal{T}, |\mathcal{T}| = T\}$ do not leak any information about the matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, implying Equation (2.16) is satisfied. This is because linear combinations of T independent uniformly random $\mathbf{R}_i$'s and $\mathbf{S}_i$'s are added to the encoded matrix partitions and at least $T+1$ linear equations are required to eliminate the effect of these random matrix partitions. Hence, the system is secure against the collusion of T workers.

From the responses collected from the workers, the client aims to interpolate

$$\begin{aligned}\mathbf{A}(x)\mathbf{B}(x) = &\sum_{i\in[1:K]}\sum_{j\in[1:L]} \mathbf{A}_i\mathbf{B}_j x^{i-1+(K+T)(j-1)} + \sum_{i\in[1:T]}\sum_{j\in[1:L]} \mathbf{R}_i\mathbf{B}_j x^{K+i-1+(K+T)(j-1)} \\ &+ \sum_{i\in[1:K]}\sum_{j\in[1:T]} \mathbf{A}_i\mathbf{S}_j x^{i-1+(K+T)(L+j-1)} + \sum_{i\in[1:T]}\sum_{j\in[1:T]} \mathbf{R}_i\mathbf{S}_j x^{K+i-1+(K+T)(L+j-1)}.\end{aligned} \tag{2.19}$$

Hence, $\mathbf{A}(x)\mathbf{B}(x)$ has a degree of $(K+T)(L+T)-1$, implying $R_{th} = (K+T)(L+T)$ workers are required to respond to interpolate it and to decode $\mathbf{AB}$.

Compared to the R_{th} of UPC, which is KL, R_{th} of S-UPC has an extra term $T(K+L)+T^2$. That is because of the random matrix partitions introduced to guarantee security. However, note that the ultimate goal of the coded distributed matrix multiplication is to decode $\mathbf{AB}$. Hence, only the multiplications $\mathbf{A}_i\mathbf{B}_j$ are required. All other products involving random matrix partitions do not have to be necessarily decodable. Hence, it is not required that all distinct $\mathbf{R}_i\mathbf{B}_j$'s, $\mathbf{A}_i\mathbf{S}_j$'s and $\mathbf{R}_i\mathbf{S}_j$'s are the coefficients of distinct monomials, implying that there is room for improving the R_{th}.

2.1.4.2 Gap Additive Secure Polynomial (GASP) Codes

The recovery threshold of S-UPC has been improved in subsequent works [12], [13], by carefully choosing the degrees of the encoding monomials so that the resultant decoding polynomial contains the minimum number of additional coefficients. Since GASP codes proposed in [13] provide fewer additional coefficients via a more sophisticated method for choosing the degrees of the monomials, in this section, we prefer to present this work.

If we extend the general model presented in Equation (2.1) and Equation (2.2) to the secure case, we obtain

$$\mathbf{A}(x) = \sum_{i\in[1:K]} \mathbf{A}_i x^{\alpha_i} + \sum_{i\in[1:T]} \mathbf{R}_i x^{\alpha_{K+i}} \tag{2.20}$$

$$\mathbf{B}(x) = \sum_{i\in[1:L]} \mathbf{B}_i x^{\beta_i} + \sum_{i\in[1:T]} \mathbf{S}_i x^{\beta_{L+i}}. \tag{2.21}$$

Like S-UPC, in GASP codes, the $\mathbf{A}$ is partitioned row-wise and $\mathbf{B}$ is partitioned column-wise. The aim of GASP codes is to design α_i's and β_i's such that all the pairwise multiplications between actual matrix partitions $\mathbf{A}_i\mathbf{B}_j$ are decodable at the $\mathbf{A}(x)\mathbf{B}(x)$, while minimizing the number of monomials whose coefficients include any random matrix partition, i.e., $\mathbf{R}_i$'s or $\mathbf{S}_i$'s. In this way, R_{th} is aimed to be improved and fewer workers are needed to realize the task.

For this purpose, different strategies to choose α_i's and β_i's are proposed in [13] depending on the values of K, L and T. For $T \geq \min\{K, L\}$, the strategy is called GASP$_{\text{big}}$, and given by

$$\alpha_i = \begin{cases} i-1 & \text{if } i \in [1:K] \\ K+L+t-1 & \text{if } i = K+t, t \in [1:T] \end{cases}$$

$$\beta_i = \begin{cases} K(i-1) & \text{if } i \in [1:L] \\ K+L+t-1 & \text{if } i = L+t, t \in [1:T] \end{cases} \tag{2.22}$$

when $K \geq L$, and by

$$\alpha_i = \begin{cases} L(i-1) & \text{if } i \in [1:K] \\ K+L+t-1 & \text{if } i = K+t, t \in [1:T] \end{cases} \tag{2.23}$$

$$\beta_i = \begin{cases} i-1 & \text{if } i \in [1:L] \\ K+L+t-1 & \text{if } i = L+t, t \in [1:T] \end{cases} \tag{2.24}$$

otherwise. On the other hand, when $T < \min\{K, L\}$, the exponents are chosen as follows, which is named as $\text{GASP}_{\text{small}}$,

$$\alpha_i = \begin{cases} i-1 & \text{if } i \in [1:K] \\ KL+K(t-1) & \text{if } i = K+t, t \in [1:T] \end{cases} \tag{2.25}$$

$$\beta_i = \begin{cases} K(i-1) & \text{if } i \in [1:L] \\ KL+t-1 & \text{if } i = L+t, t \in [1:T] \end{cases} \tag{2.26}$$

when $K \leq L$, and

$$\alpha_i = \begin{cases} L(i-1) & \text{if } i \in [1:K] \\ KL+t-1 & \text{if } i = K+t, t \in [1:T] \end{cases} \tag{2.27}$$

$$\beta_i = \begin{cases} i-1 & \text{if } i \in [1:L] \\ KL+L(t-1) & \text{if } i = L+t, t \in [1:T] \end{cases} \tag{2.28}$$

otherwise.

It is important to note that the reason behind selecting α_i's and β_i's as specified is to create gaps in the degrees of the monomials in $A(x)B(x)$. These gaps result in monomials with zero coefficients. In [13], the authors demonstrate that the presence of such zero-coefficient monomials helps to decrease the recovery threshold and they do not pose any issue in terms of decodability.

Similarly to the previous schemes, using these exponents, the client sends to the client i the evaluations $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$. Note that since $\mathbf{R}_i$'s and $\mathbf{S}_i$'s are generated independently and uniform randomly, and each $\mathbf{R}_i$ and $\mathbf{S}_i$ has a corresponding monomial at $\mathbf{A}(x)$ and $\mathbf{B}(x)$, respectively, the evaluations $\mathbf{A}(x_i)$ and $\mathbf{B}(x_i)$ do not leak any information about $\mathbf{A}$ and $\mathbf{B}$ as long as the maximum number of colluding workers does not exceed T. A formal proof is provided in [13].

From the worker's responses, the client interpolates

$$\begin{aligned}\mathbf{A}(x)\mathbf{B}(x) = &\sum_{i\in[1:K]}\sum_{j\in[1:L]}\mathbf{A}_i\mathbf{B}_j x^{\alpha_i+\beta_j} + \sum_{i\in[1:K]}\sum_{j\in[1:T]}\mathbf{A}_i\mathbf{S}_j x^{\alpha_i+\beta_{L+j}} \\ &+ \sum_{i\in[1:T]}\sum_{j\in[1:L]}\mathbf{R}_i\mathbf{B}_j x^{\alpha_{K+i}+\beta_j} + \sum_{i\in[1:T]}\sum_{j\in[1:T]}\mathbf{R}_i\mathbf{S}_j x^{\alpha_{K+i}+\beta_{L+j}}. \quad (2.29)\end{aligned}$$

It has been shown in [13], the recovery threshold becomes

$$R_{th}^{GASP}(K,L,T) = \begin{cases} KL+K+L, & 1=T<L\le K \\ KL+K+L+T^2+T-3, & 1<T<L\le K \\ (K+T)(L+1)-1, & L\le T<K \\ 2KL+2T-1, & L\le K\le T. \end{cases} \quad (2.30)$$

Note that this recovery threshold considerably improves that of S-UPC due to the sophisticated design of exponents of the monomials in the encoding polynomials.

2.1.4.3 Related Literature

Similar to the relation between UPC, MatDot and PolyDot codes, in the secure case, via employing different matrix partitioning techniques, the recovery threshold can be improved. In [14], the extension of the PolyDot and entangled polynomial codes to secure distributed matrix multiplication is studied, and the trade-off between the download rate and the recovery threshold is explored. In [15], CSA codes [6] are used for the secure distributed batch multiplication problem, and the scheme is shown to be optimal in terms of the download rate. [16] is another work CSA codes are utilized for secure distributed matrix multiplication and similarly to [6], it has been shown that a low download rate can be achieved at the price of a large upload rate. In these works, lower recovery threshold values than [13] are obtained by using different matrix partitioning techniques but this comes at the expense of a considerable increase in the other metrics like download rate and upload rate.

Another novel coding approach is proposed in [17] for distributed matrix multiplication, based on polynomial evaluation at the roots of unity in a finite field. It has constant time decoding complexity and a lower recovery threshold than the aforementioned traditional polynomial-type coding approaches. However, in this approach, the sub-tasks assigned to the workers are not *one-to-any replaceable*, i.e., the failure of one worker cannot be compensated by any other worker. Hence, these codes lack straggler mitigation capability. Since, in this dissertation, we focus on codes which both have straggler mitigation and security properties simultaneously, we do not discuss this scheme in more detail.

2.2 Federated Learning

Machine learning (ML) has traditionally relied on a centralized approach, where data is stored and processed in a central entity or a cluster belonging to a single organization. This framework operates under the assumption that the central entity is the owner of the data and can freely access and distribute it in any way that is convenient for learning purposes. However, in modern ML systems, there are situations where the assumptions of such a centralized model are hard to satisfy.

One of the main challenges in centralised ML is the collection of data. In many cases, the amount of data available is massive, making it difficult to collect and store in a single entity. This requires substantial communication and storage resources that may not be feasible. Additionally, if the data originates from independent parties, privacy concerns may prevent direct sharing with a central entity.

Another challenge is the central entity's computational resources. As the demand for processing power increases, the central entity may not have the necessary resources to handle the massive demand. This is especially true in scenarios where the data is constantly changing, and the model needs to be updated frequently.

To tackle the challenges of centralized ML settings, it is imperative to explore alternative approaches. FL has emerged as a promising solution in recent years, owing to its ability to overcome the limitations of traditional methods [18]. FL is a framework that enables multiple clients to collaboratively train a model with the aid of a central server, known as the parameter server (PS), without the need to share their local datasets. Instead, clients share their local model updates with the PS after each training round, and the PS aggregates these updates to generate an improved global model.

FL is typically categorized into two settings based on the application scenario: cross-device and cross-silo settings. In the cross-device setting, there are usually a large number of clients, ranging from thousands to millions, and only a small fraction of them are available to participate in each round. These clients are typically edge devices with limited capabilities, such as being battery-powered, having poor communication resources, and limited computation power. Personal AI assistants, like Alexa [19], mobile device keyboards, like Gboard [20], and vehicular networks [21] are examples of real-world applications of this setting. In contrast, in the cross-silo setting, there are usually fewer clients, typically tens to hundreds, and they have better capabilities than edge devices. They are mostly available to participate during the learning rounds, and are more reliable than edge devices. Collaborative learning among different institutions for the common good, such as collaboration among financial institutions, or training a diagnostic model on patient data from different hospitals [22, 23] are among typical use cases for this setting.

In the following, we present a more formal description of the FL framework by introducing basic terminology, which is necessary for the rest of the dissertation.

2.2.1 FL framework

In this section, we aim to explain the general framework of FL. The FL framework involves N clients participating in a learning task. Each client has a local dataset $\mathcal{D}_i$. The aim is to train a global model $f_{\mathbf{w}}(\cdot)$ with parameters $\mathbf{w} \in \mathbb{R}^m$, where m is an integer, using the local datasets of the clients. Each client, represented by $i \in [1:N]$, has a local loss function $\ell_i(\cdot,\cdot)$ that measures the discrepancy between the model output and the ground truth for each sample. The goal of training is to find the optimal model parameter w that minimizes the average of the local loss functions across all clients, which is given by

$$\mathbf{w} = \arg\min \frac{1}{N} \sum_{i\in[1:N]} \frac{1}{|\mathcal{D}_i|} \sum_{d\in\mathcal{D}_i} \ell_i(f_{\mathbf{w}}(d), d). \tag{2.31}$$

The training process comprises several rounds T, which can either be predetermined or dependent on certain system parameters, as well as the precision of the current model predictions. In each of the rounds, denoted by $t \in [1:T]$, the PS begins by broadcasting the current global model $\mathbf{w}^t$ to all available clients, represented by $\mathcal{P}_t \subset [1:N]$. Subsequently, each client $i \in \mathcal{P}_t$ uses its local dataset $\mathcal{D}_i$ to compute a model update $\mathbf{g}_i^t$. The update is the result of a local optimization procedure, such as stochastic gradient descent (SGD), which can involve one or several local iterations, depending on the problem's nature and constraints. Afterwards, each participating client sends its local update $\mathbf{g}_i^t$ to the PS. Once the PS has collected all the responses from each client $i \in \mathcal{P}_t$, it aggregates the local updates and updates the global model as follows

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \cdot \mathrm{Agg}(\mathbf{g}_i^t \mid i \in \mathcal{P}_t), \tag{2.32}$$

where $\mathrm{Agg}(\cdot)$ is the function aggregating the model updates, and η_t is the learning rate at round t. The aggregating function can be customized based on the requirements of the problem or the optimization algorithm used for training. The simplest form of this function is basic averaging, which is commonly known as federated averaging (FedAvg) [18]. However, it might not perform well in certain cases where there is a considerable difference in the data distributions among the clients or when there are adversarial clients present. In such scenarios, some modifications to the function might be necessary.

In some settings, for efficiency purposes, without waiting for all the available workers' responses, the PS may decide to proceed to the update phase. For example, if there are some stragglers in the system and sufficiently many responses are collected, the PS may ignore the rest of the clients and proceed with what it already received. Although Equation (2.32) does not involve such a scenario, it is worth noting that such cases are possible and in these cases, $\mathcal{P}_t$ is replaced by the set of responded clients' indices.

2.2.2 Challenges in FL

In this subsection, we briefly describe the main challenges in FL, which are communication efficiency, heterogeneity across clients and data privacy.

2.2.2.1 Communication efficiency

It's important to recall that one of the main reasons why the FL framework was created is to tackle the communication bottleneck of central ML systems. In these systems, local datasets of the clients need to be uploaded to a central entity, which can be quite time-consuming. While FL solves this issue partially by only requiring model updates from the clients to be uploaded to the PS, it still involves several rounds of training, which means that the communication cost of uploading model updates should be as low as possible. This is especially important in cross-device settings where clients have limited resources, and high communication overhead may cause excessive power usage and generate large delays.

There are two main approaches to improving communication efficiency in FL: minimising the number of communication rounds and compressing client updates.

The first approach involves clients not sending updates to the PS after every local iteration. Instead, they update their own local model by selecting a subset of data from their local dataset. In the following local iteration, they use their updated local model without any communication with the PS. The number of local iterations required varies depending on the problem and the convergence properties of the optimization algorithm used. If the number of iterations is too large, then the local models may diverge from each other, leading to a decrease in the accuracy of the global model. However, if the number of iterations is properly selected, it can speed up the convergence of the global model, resulting in fewer global communication rounds and less communication cost. This approach is an effective way of reducing communication overhead and conserving resources while still achieving high accuracy.

Another popular approach to reduce the communication cost of FL is compressing the client updates. This is especially important when the model size of the trained model is large. In each global round, the update with the same size of the model is transmitted from each participating client to the PS, creating a large communication overhead. As a solution, techniques involving low-precision gradients, such as quantization and gradient sparsification, are actively researched. These techniques are effective in making FL communication-efficient, and they include solutions such as [24–29].

2.2.3 Heterogenity across clients

In FL, there can be various sources of heterogeneity across clients. Firstly, since each client may represent different users or user groups, and these users' behaviours may differ depending on factors such as their location, time zone, daily routines, or socioeconomic status, the data stored on the clients may have non-identical distributions. This is known as non-independently and identically distributed (i.i.d.). data in FL. In addition to non-i.i.d. data, such heterogeneities in users may also result in heterogeneities at the devices themselves, implying differences in their computational power, communication capabilities, availability rate, and power constraints, among others.

The issue of non-i.i.d. data is a critical challenge faced by machine learning models in achieving good global performance. This problem arises due to the highly heterogeneous distribution of data present across clients, which can cause local models to diverge from each other and result in a poorly performing global model. As an example, consider a classification problem where a specific label exists only in the dataset of one client. In such a scenario, the local models of other clients may not learn to classify such examples accurately, leading to a minimal contribution of the client having that label during the averaging process. In order to tackle the non-i.i.d. data problem, several solutions have been proposed, such as data centralization, personalisation, multi-task learning, and meta-learning. However, this topic remains an active area of research with ongoing developments and advancements. For a comprehensive review of the works related to this subject, we refer the readers to [22].

On the other hand, the heterogeneity of the devices leads to challenges such as failures, errors, and straggling behaviour. These challenges can result in inaccurate results or a significant delay in the training process. Several potential solutions have been proposed to address these issues, including asynchronous training, device sampling, and gradient coding techniques [30–34]. and the topic is still an active research area to develop more effective and efficient approaches for FL in the face of device heterogeneity.

2.2.4 Data privacy

The preservation of privacy is a crucial concern in the realm of ML, as the very solutions depend on data that can potentially reveal sensitive information about the individuals who own it. Therefore, while training ML models, it is of paramount importance to prevent any sensitive features from the training set from leaking. One of the core promises of FL is preserving privacy, as data never leaves clients and only local model updates are shared. Unfortunately, research has shown that such updates, their average, and even the final trained model are sufficient to reveal sensitive information about the training set [35–43]. Hence, privacy concerns remain a significant area of research.

To ensure privacy in FL, various techniques can be employed, including secure multi-party computation (S-MPC), homomorphic encryption (HE), and differential privacy (DP) mechanisms. S-MPC-based solutions allow the PS to learn the summation of the client updates without revealing any individual client update. This technique has been successfully implemented in various studies, [44], to eliminate attacks based on individual client updates. Similarly, HE enables the computation of functions of encrypted data, as demonstrated in [45, 46] to preserve privacy.

However, in both approaches, the final aggregated update may still be visible to the PS, which opens up the possibility of attacks utilizing aggregated models and the final global model. Thus, while S-MPC and HE are effective in preventing attacks based on individual client updates, these techniques do not offer complete protection against attacks utilizing aggregated models.

DP is a strong privacy notion that adds irreversible disturbance to the training procedure, ensuring all local and aggregate updates, and the final deployed model have some degree of protection. Though this technique adds noise to the process and may hurt the accuracy, it remains an active research topic to develop mechanisms that satisfy a desired level of privacy while having a less harmful effect on accuracy. It is worth noting that the techniques based on S-MPC and HE can be used in conjunction with DP mechanisms to further enhance their privacy guarantees.

In this dissertation, we mainly focus on DP mechanisms. Hence, in the next section, we provide a detailed background on this topic.

2.3 Differential Privacy

Differential privacy (DP) has emerged as a widely accepted and highly regarded privacy notion in various privacy-preserving data analysis tasks. It serves as a gold standard for privacy preservation in such scenarios. When a data-processing algorithm is designed to respect the privacy of users whose data is contained in the input dataset, the output must not reveal substantial information about any individual in the dataset. In order to formalize this, DP quantifies the degree of indistinguishability, from the algorithm's output, between the presence and the absence of a single individual in the input dataset.

DP is a robust and rigorous concept of privacy that provides a strong foundation for privacy-preserving data analysis. Unlike other privacy definitions that rely on specific attack scenarios or assumptions about adversaries, DP is based on an information-theoretic characterization that holds true for any attack model or adversary with arbitrary computational power. In essence, DP mathematically quantifies the privacy protection provided by a data analysis algorithm. Specifically, it provides an upper bound on the increase in an adversary's certainty about whether a particular individual is present in the input dataset after the algorithm has run, regardless of the adversary's background knowledge. Furthermore, DP exhibits a composition property that allows it to quantify the total amount of privacy loss from multiple accesses to a dataset. This makes it a preferred notion for measuring privacy leakage in practical applications. Since its introduction by Dwork et al. in [47], DP has become a cornerstone of privacy research and has found numerous applications in various fields such as healthcare, finance, and social sciences.

In the following, we give the formal definition of DP.

2.3.1 Pure differential privacy

Definition 2.1. A randomized mechanism $M : \mathbb{D} \to \mathbb{S}$ is ε-differentially private (ε-DP) if

$$\Pr[M(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[M(\mathcal{D}') \in \mathcal{S}], \tag{2.33}$$

for all neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$, i.e., sets differing in only one element, and $\forall \mathcal{S} \subset \mathbb{S}$, where $\varepsilon > 0$.

This notion is also referred to as *pure differential privacy.* The definition refers to a concept that datasets $\mathcal{D}$ and $\mathcal{D}'$ are considered neighbours if they differ in only one element. The meaning of this, however, depends on the context. For instance, if the goal is to protect the privacy of a user, then the datasets differ by only one user. Furthermore, this difference can occur in two possible ways. The first option is that the cardinalities of the datasets may differ by one. That is, $\mathcal{D}$ can be obtained by removing one element from $\mathcal{D}'$, or vice versa. This relationship is called the *add-remove* relation. Alternatively, $\mathcal{D}$ can be obtained by replacing one user in $\mathcal{D}'$ with another user, or vice versa. This relationship is called the *replacement* relation.

Note that the definition of DP suggests that the relation in Equation (2.33) is valid for all $\mathcal{D}$ and $\mathcal{D}'$ pairs and for all possible outputs $\mathcal{S}$. Hence, the notion of DP is indeed two-sided, implying

$$e^{-\varepsilon} \Pr[M(\mathcal{D}') \in \mathcal{S}] \leq \Pr[M(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[M(\mathcal{D}') \in \mathcal{S}]. \tag{2.34}$$

Hence, an algorithm or mechanism that satisfies DP does not leak information about an individual that would increase the odds of an adversary correctly identifying one user's participation by more than a multiplicative factor of e^{ε}. Hence in DP, smaller ε values imply better privacy protection. Moreover, since this guarantee is for the participation of a user, any other specific attribute of the user is protected at least by the parameter ε. This is because the presence of a user in a dataset is its simplest feature, and learning any other feature of a user first requires learning its presence in the dataset.

There are many examples of DP mechanisms proposed in the literature including randomized response [48], Laplace mechanism [47] and staircase mechanism [49]. We do not discuss them here since we do not utilize them in the following chapters and hence, they are out of the scope of this dissertation. However, depending on the nature of the problem, in general, satisfying ε-DP exactly requires adding too much noise to the output, hence, harming the accuracy.

In the next subsection, we present a relaxation of pure DP, which allows the use of much smaller noise variances compared to pure DP despite a small probability of failure.

2.3.2 Approximate differential privacy

Definition 2.2. A randomized mechanism $M : \mathbb{D} \to \mathbb{S}$ is (ε, δ)-DP if

$$\Pr[M(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[M(\mathcal{D}') \in \mathcal{S}] + \delta, \tag{2.35}$$

for all neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$, and $\forall \mathcal{S} \subset \mathbb{S}$, where $\varepsilon > 0$ and $\delta \in [0, 1)$.

Compared to the definition of pure DP, in the above definition, we have δ, which, loosely speaking, characterises the failure probability of pure DP. This is clearly a relaxation of pure DP and setting $\delta = 0$ reduces the guarantee to pure DP, which is $(\varepsilon, 0)$-DP.

The exact characterization of δ is better understood via an alternative but equivalent definition of (ε, δ)-DP, which is expressed in terms of *hockey stick divergence*, which we introduce next.

Definition 2.3 ([50]). We define the *hockey stick divergence* between two probability measures μ and μ' as

$$D_\alpha(\mu||\mu') \triangleq \int_Z \left[d\mu(z) - \alpha d\mu'(z)\right]_+ d(z) \tag{2.36}$$

where $[\cdot]_+ \triangleq \max\{0, \cdot\}$.

Based on hockey stick divergence, (ε, δ)-DP can be expressed as follows.

Theorem 2.1 ([51], Theorem 1 in [52]). *A mechanism M is (ε, δ)-DP if and only if*

$$\sup_{\mathcal{D}, \mathcal{D}'} D_\alpha \left(M(\mathcal{D})||M(\mathcal{D}')\right) \leq \delta, \tag{2.37}$$

where $\mathcal{D}$ and $\mathcal{D}'$ are two neighbouring datasets and $\alpha = e^\varepsilon$.

Theorem 2.1 characterizes the trade-off between ε and δ, and hence, δ can be considered as a function of ε. Therefore, δ is not exactly the failure probability but rather a measure of how much the condition $\Pr[M(\mathcal{D}) \in \mathcal{S}] \leq e^\varepsilon \Pr[M(\mathcal{D}') \in \mathcal{S}]$ is violated by using hockey stick divergence. Hence, δ provides more information than just the probability of failure. It also indicates the severity of the failure. Hence, like ε, smaller δ values are more desirable in DP.

DP has several desirable properties which makes it an appropriate choice for many privacy-preserving data processing tasks, including post-processing property and composition. When a mechanism provides an output using a mechanism with some DP guarantees, then transformation of this output via any other function still provides at least as strong DP guarantees as the original output as long as the dataset is not accessed again by this transformation.

Another intriguing property of DP is composition. In certain scenarios where a particular dataset needs to be accessed multiple times T, it is important to consider the amount of total privacy lost with each access. For example, in machine learning (ML) models, a dataset might be accessed multiple times during the training process, and with each access, the mechanism learns and outputs some information about the dataset. In such cases, it becomes essential to quantify the loss of privacy incurred due to these repeated accesses. The naive approach to quantifying privacy loss is to use the $(T \cdot \varepsilon, T \cdot \delta)$ [53]. However, this characterization is too loose for large values of T. To address this limitation, the advanced composition theorem [53, Theorem 3.20], based on concentration inequalities, provides a tighter characterization.

Luckily, the use of Rényi divergence has been shown to lead to even more significant improvements in composition. In this regard, in the following subsection, we present a formal definition of Rényi differential privacy (RDP) and proceed to discuss its noteworthy composition properties.

2.3.3 Rényi differential privacy (RDP)

Definition 2.4. Rényi divergence between two probability distributions is defined as

$$R_\alpha(P||Q) \triangleq \frac{1}{\alpha-1}\log \mathbb{E}_{x\sim Q(x)}\left(\frac{P(x)}{Q(x)}\right)^\alpha, \tag{2.38}$$

where $\alpha \in [1,\infty)$, and R_α for $\alpha = 1$ and $\alpha = \infty$ is defined by continuity.

Definition 2.5. (α,ε)-Rényi DP [54]: A randomized mechanism $M : \mathbb{D} \to \mathbb{S}$ satisfies (α,ε)-RDP if

$$R_\alpha\left(\Pr\left(M(\mathcal{D}) = s\right) || \Pr\left(M(\mathcal{D}') = s\right)\right) \le \varepsilon, \tag{2.39}$$

where $\mathcal{D}$ and $\mathcal{D}'$ are neighbouring datasets.

As we stated in the previous subsection, RDP has a quite useful composition property, which we state in the next lemma.

Lemma 2.1 ([54]). *If a dataset is accessed T times with (α,ε_i)-RDP mechanisms where $i \in [1:T]$, then the overall composite application of these mechanisms satisfy $(\alpha, \sum_{i\in[1:T]}\varepsilon_i)$-RDP.*

In addition to being quite simple to compute, for some important mechanisms RDP-based composition is tighter than advanced composition theorem.

Next, we present a lemma which provides a conversion formula from RDP to approximate DP. Note that this result is not the optimal conversion but it is close to the optimal and easier to characterise. The optimal conversion is proposed in [55].

Lemma 2.2. *RDP to DP [54]: A mechanism satisfying (α,ε)-RDP also satisfies $(\varepsilon + \log(1/\delta)/(\alpha-1),\delta)$-DP.*

2.3.4 Gaussian Mechanism

In this subsection, we present the Gaussian mechanism, which is widely used to achieve approximate DP guarantees for various data processing tasks. The application of this mechanism is quite simple, which consists of adding a Gaussian noise with a specified mean and variance. Thanks to the mathematically desirable properties of Gaussian distribution, it makes analysis of the resulting DP guarantees also easier. The following theorem formally characterizes the DP guarantees that can be achieved using the Gaussian mechanism.

Theorem 2.2 (Theorem 8 in [36]). *Let $f : \mathbb{D} \to \mathbb{R}^m$ be a function satisfying $||f(\mathcal{D}) - f(\mathcal{D}')||_2 \leq C$, for all neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$, where $||\cdot||_2$ denotes the L_2 norm. A mechanism $M(\mathcal{D}) = f(\mathcal{D}) + \mathcal{N}(0, \sigma^2)$ is (ε, δ)-DP if and only if*

$$\Phi\left(\frac{C}{2\sigma} - \frac{\varepsilon\sigma}{C}\right) - e^{\varepsilon}\Phi\left(-\frac{C}{2\sigma} - \frac{\varepsilon\sigma}{C}\right) \leq \delta, \tag{2.40}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution.

In the following theorem, we also present the RDP guarantees of the Gaussian mechanism.

Theorem 2.3 (Proposition 7 in [54]). *The Gaussian mechanism with sensitivity C satisfies $(\alpha, \alpha C^2/\sigma^2)$-RDP.*

Therefore, the Gaussian mechanism is a versatile tool that can be applied to a wide range of tasks, thanks to its robust RDP guarantees and tight composition guarantees. Moreover, RDP guarantees can be easily converted to DP guarantees, further enhancing the mechanism's applicability. Of particular interest is its use in differentially private deep learning, which has numerous potential applications. In the upcoming chapters, we will explore the application of the Gaussian mechanism to various problems in the context of FL, highlighting its effectiveness and versatility.

2.3.5 Local and Central Differential Privacy

In this section, we discuss two contextual models of differential privacy. Although both models adhere to the formal definitions of differential privacy introduced earlier in this chapter, they specify the context in which the privacy guarantees are valid.

In the local model of differential privacy, an entity that wants to keep its data private doesn't rely on any other party to protect its privacy. Hence, before releasing a response to any query about its local dataset, it randomizes it properly using a suitable method to achieve the desired DP guarantees, e.g., Gaussian mechanism. The DP guarantees achieved by such a local randomization are called *local differential privacy* guarantees [53].

On the other hand, in the central model of differential privacy, we assume a federated setting, where there is a central entity and several participating entities that trust the central entity. The primary goal is to keep the datasets of the participating entities private. When given a query, such as requesting gradients in an iteration of FL, the participating entities can either locally randomize their individual responses or provide noiseless responses. The central entity can access these individual responses to generate an aggregate response. Before releasing it to third parties or other participating entities, it can further randomize the aggregate response or release only the aggregation if the individual responses are already locally randomized. In this setting, any adversary can only see the aggregate response, and the differential privacy guarantees that the aggregate response has against an adversary are referred to as *central differential privacy* guarantees [53].

Chapter 3

Bivariate Polynomial Coding for Straggler Mitigation

3.1 Abstract

Coded computing is an effective technique to mitigate stragglers in large-scale and distributed matrix multiplication. In particular, univariate polynomial codes have been shown effective in straggler mitigation by making the computation time depend only on the fastest workers. However, these schemes completely ignore the work done by the straggling workers resulting in a waste of computational resources. To reduce the amount of work left unfinished at workers, one can further decompose the matrix multiplication task into smaller sub-tasks, and assign multiple sub-tasks to each worker, possibly heterogeneously, to better fit their particular storage and computation capacities. In this chapter, we present a novel family of *bivariate polynomial codes* to efficiently exploit the work carried out by straggling workers. We show that bivariate polynomial codes bring significant advantages in terms of upload communication costs and storage efficiency, measured in terms of the number of sub-tasks that can be computed per worker. We propose two variants of bivariate polynomial codes. The first one exploits the fact that bivariate interpolation is always possible on a rectangular grid of evaluation points. We obtain such points at the cost of adding some redundant computations. For the second scheme, we relax the decoding constraints and require decodability for almost all choices of the evaluation points. We present interpolation sets satisfying such decodability conditions for certain storage configurations of workers. Our numerical results show that bivariate polynomial coding considerably reduces the average computation time of distributed matrix multiplication. We believe this work opens up a new class of previously unexplored coding schemes for efficient coded distributed computation.

3.2 Introduction

In this chapter, we delve into the crucial problem of mitigating stragglers in distributed matrix multiplication. Stragglers, which are slow or failed worker nodes, can significantly impede the performance of the overall system. To address this issue, one promising approach is to assign redundant computations to the worker nodes. Under this framework, the stragglers are treated as random erasures, and the computation time can be improved by creating redundant computations using coding techniques similar to those used for erasure channels.

In Chapter 2, we provide a comprehensive review of various coding schemes that have been proposed to mitigate the straggler problem, including univariate polynomial codes (UPC) [2], MatDot codes, and PolyDot codes [3]. Additionally, we discuss further advancements in the field, including solutions proposed in [4], [5], and [6].

However, in all of these approaches, the client node receives the result of the work assigned to a worker only after it has been completed in its entirety, regardless of the scale of the computations allocated. Workers that fail to complete their assigned tasks by the time as many computations as the recovery threshold are collected by the client are treated as erasures. Consequently, all the work done by these workers is deemed invalid and ignored by the client. This approach is sub-optimal, particularly when the speeds of the workers are similar. In such cases, the ignored workers may have already completed a substantial part of the assigned task, resulting in wasted computational resources. Therefore, there is a need to investigate more efficient approaches that account for the partially completed work by the workers.

To exploit the partially completed work done by stragglers, a multi-message approach is considered in several works such as [56–58]. These approaches involve dividing workers' tasks into smaller sub-tasks, and communicating the result of each sub-task to the master as soon as it is completed without waiting for the other tasks.

The same approach can be also employed in polynomial coding schemes. However, unfortunately, it turns out that the univariate polynomial coding schemes, such as [2–6, 59] do not scale well with the number of assigned tasks per worker in terms of storage efficiency and upload costs. As we show in the following sections, under fixed storage capacities at the workers, in univariate polynomial coding, dividing a task into sub-tasks by a given factor reduces the fraction of work that can be done by the workers in the same factor, resulting in inefficient use of workers' storage capacity and upload costs.

The approaches in [56, 58] are based on uncoded computation and a hybrid of uncoded and coded computation, respectively. It has been shown that uncoded computation may be more beneficial if the workers' computation speeds are similar. However, in scenarios where workers have heterogeneous computation speeds, as encountered in serverless computing, peer-to-peer applications, or edge computing, coded computation with multi-message communication is still relevant and helps speed up the target task.

On the other hand, the approach in [57] is based on fully coded computation, where a coding scheme called product codes is employed. Product codes are constructed based on the product of two MDS codes. They partially address the scalability issues of univariate polynomial coding schemes, which do not scale well with the number of assigned tasks per worker in terms of storage efficiency and upload costs. However, computations at workers in product codes are not one-to-any replaceable, which results in poor performance in various scenarios. Moreover, both univariate polynomial codes and product codes impose certain constraints preventing fully heterogeneous workloads across workers.

In [60], a hierarchical coding framework for the straggler exploitation problem is proposed, also taking into account the decoding complexity. This work is extended to matrix-vector and matrix-matrix multiplications in [61]. However, the benefits of hierarchical coding are significant mainly if the decoding time is comparable to the computation time.

Another line of work which has the potential to efficiently address the straggler exploitation problem is rateless codes. Previously, the use of rateless codes for distributed matrix-matrix and matrix-vector multiplication is proposed in [62–64]. Rateless codes are not based on polynomial codes and the matrix partitions are encoded by using random coefficients. Hence, unlike univariate polynomial codes, they do not suffer from inefficiencies in terms of upload cost and storage when it comes to straggler exploitation while also providing low numerical errors. However, these types of codes have non-zero, and sometimes notably high, failure probabilities in their decoding due to their stochastic nature in the encoding phase [64], and this might limit their usage in some settings.

Hence, although the issue of straggling workers and their efficient utilization in distributed computing has been a subject of active research, with several coding schemes proposed to tackle the problem, there is still a continued need for scalable and efficient solutions.

To address these challenges, in this chapter, we present bivariate polynomial codes. Bivariate polynomial codes aim to improve the computation time of distributed matrix-matrix multiplication under limited storage at the workers while improving the upload cost from the client to the workers. The main contributions of this chapter can be summarized as follows:

- We first show the limitation of univariate polynomial codes in terms of both computational and storage efficiency when extended to the multi-message setting.
- We introduce bivariate polynomial coding schemes to address these limitations. Interpolation of bivariate polynomials cannot be guaranteed by simply requiring all evaluation points to be distinct. Here, we introduce the concepts of regular (always invertible), and almost regular (almost always invertible) interpolation matrices.
- We first extend the product coding scheme of [57] to bivariate polynomial coding, which leads to a regular interpolation matrix by imposing a particular rectangular grid structure on the interpolation points. This strategy attains maximum storage efficiency, but the computation efficiency can be limited due to redundant computations.
- We propose two novel bivariate coding schemes. We demonstrate that unlike univariate schemes, for bivariate coding, the order by which the computations are done at the workers has a non-trivial impact on decodability; and hence, we impose a special computation order for the tasks assigned to each worker. These schemes achieve maximum computation efficiency by completely avoiding redundant computations. Their storage efficiency is limited, yet higher than that of univariate schemes. We further propose two alternative bivariate polynomial codes with higher storage efficiency at the cost of a slight decrease in computation efficiency.
- We numerically validate our findings assuming a shifted exponential model for computation speeds, and show the superiority of the proposed bivariate schemes compared to univariate alternatives and product codes.
- While polynomial codes have been extensively studied with numerous applications in practice, to the best of our knowledge, our work provides the first examples of bivariate polynomial code constructions with superior performance compared to their univariate counterparts.

3.3 System Model and Problem Formulation

Our system, as illustrated in Figure 3.1, involves a master server that aims to multiply two matrices $\mathbf{A} \in \mathbb{R}^{r \times s}$ and $\mathbf{B} \in \mathbb{R}^{s \times c}$, where r, s, and c are positive integers, by offloading partial computations to N workers with heterogeneous storage capacities and computation speeds.

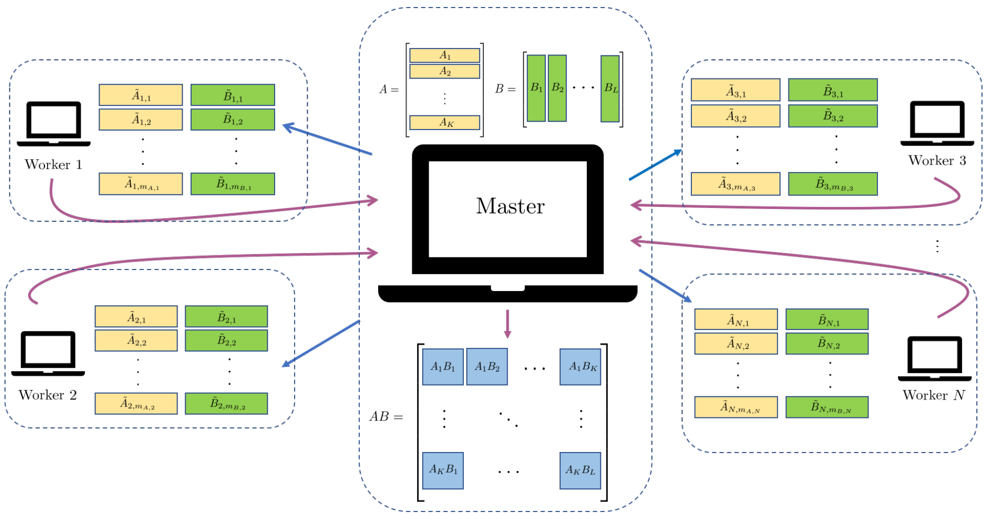


FIGURE 3.1: Overview of distributed matrix multiplication employing N workers.

The master server divides matrix $\mathbf{A}$ row-wise and matrix $\mathbf{B}$ column-wise into K and L partitions respectively. Thus, $\mathbf{A} = [\mathbf{A}_1^T\ \mathbf{A}_2^T \dots\ \mathbf{A}_K^T]^T$ and $\mathbf{B} = [\mathbf{B}_1\ \mathbf{B}_2 \dots\ \mathbf{B}_L]$, where $\mathbf{A}_i \in \mathbb{R}^{\frac{r}{K} \times s}$ for $i \in [1:K]$ and $\mathbf{B}_j \in \mathbb{R}^{s \times \frac{c}{L}}$ for $j \in [1:L]$.

Based on $\mathbf{A}$ and $\mathbf{B}$, The master server generates and sends to each worker $i \in [1:N]$, coded matrix partitions $\tilde{\mathbf{A}}_{i,k} \in \mathbb{R}^{\frac{r}{K} \times s}$ for $k \in [1:m_{A,i}]$, $\tilde{\mathbf{B}}_{i,l} \in \mathbb{R}^{s \times \frac{c}{L}}$ for $l \in [1:m_{B,i}]$ where $m_{A,i}$ and $m_{B,i}$ are positive integers. Hence, each worker i is assumed to store a fraction of $\mathbf{A}$ and $\mathbf{B}$, denoted by $M_{A,i} = \frac{m_{A,i}}{K}$ and $M_{B,i} = \frac{m_{B,i}}{L}$, respectively. In general, the way these coded matrix partitions are generated depends on the specific coding scheme employed. In polynomial coding schemes, they are obtained as linear combinations of the original matrix partitions.

Depending on the coding scheme employed, worker i can compute all, or a subset of the products of coded matrix partitions assigned to it, i.e., $\tilde{\mathbf{A}}_{i,k}\tilde{\mathbf{B}}_{i,l}$, $k \in [1:m_{A,i}]$, $l \in [1:m_{B,i}]$ in a prescribed order, which is also specific to the coding scheme. We denote by η_i the maximum number of computations worker i can provide, which can be possibly used by the master for decoding $\mathbf{AB}$. Thus, $\eta_i \leq m_{A,i}m_{B,i}$, and the specific value of η_i depends on the coding scheme.

In order to exploit the partial work done by straggling workers, the results of these individual products are sent to the master as soon as they are finished. The master collects the responses from the workers until the received set of computations allows the master to uniquely recover $\mathbf{AB}$. Then, the master instructs all the workers to stop computing and decodes $\mathbf{AB}$. Note that the recovery threshold, which is defined as the minimum number of computations that guarantee the decodability of $\mathbf{AB}$, does not have to be a fixed quantity in our setting. Depending on the coding scheme, R_{th} can be a function of the collected computations by the master.

As is common in the related literature, we specify the storage capacity at workers separately for each of the two matrices, i.e., $M_{A,i}$ and $M_{B,i}$. However, in practice, it is more appropriate to assume a total storage capacity at each worker, which can be freely allocated between the partitions of the two matrices. Since they are composed of the same number of real numbers, we assume that the rows of $\mathbf{A}$ and the columns of $\mathbf{B}$ require the same amount of storage. We define the storage capacity of worker i, denoted by $s_i \in \mathbb{N}^+$, as the sum of the total number of rows of $\mathbf{A}$ and the total number of columns of $\mathbf{B}$ that it can store. Accordingly, for a given K, L, and s_i, we allocate $m_{A,i}$ and $m_{B,i}$ to maximize η_i subject to $M_{A,i}r + M_{B,i}c = s_i$. Defining $C_{\text{part}} \triangleq \frac{1}{KL}$ as the fraction of work corresponding to a single partial product $\tilde{A}_{i,j}\tilde{B}_{i,l}$, the maximum fraction of work that can be done by worker i is given by

$$C_{\max,i} \triangleq \eta_i C_{\text{part}} = \frac{\eta_i}{m_{A,i}m_{B,i}} M_{A,i}M_{B,i}. \tag{3.1}$$

Under the same storage constraints, a code that can provide more fraction of work uses its storage more efficiently; hence, $C_{\max,i}$ will be used to measure the *storage efficiency.*

We define C_{wasted} as the worst-case fraction of wasted computations with respect to the full product, $\mathbf{AB}$. There are two sources of wasted computations. Firstly, depending on the coding scheme, some of the computations completed by the workers may not be used in decoding $\mathbf{AB}$. Secondly, when R_{th} is reached, the master instructs all the workers to stop their computations and the ongoing computations of the workers are wasted. We assume that the communication time for the stop signal to reach from the master to the workers is short enough that the workers receive this instruction before finishing their ongoing computations. In the following sections, we compute the fraction of the wasted computations of the second type based on this assumption. If this assumption does not hold, the wasted computations of the second type may increase.

For a fixed N and fixed storage capacities s_i's at workers, our objective is to minimize the average computation time of $\mathbf{AB}$. This depends on the statistics of the computation speeds of the workers and is difficult to obtain in closed form. Instead, we use $C_{max,i}$ and C_{wasted} as proxies for the performance of a code. These metrics do not depend on the worker's speeds and provide general indicators on the code performance. Note that, especially in heterogeneous settings, in which some workers may be much faster than others, the higher fraction of work provided by faster workers helps to finish the task earlier. Therefore, storage efficiency, or $C_{\max,i}$, is a factor to be optimized to improve the average computation time. Moreover, low C_{wasted} implies that more of the available computation capacity across the workers is exploited towards completing the desired computation. Therefore, to minimize the average computation time, we are interested in maximizing $C_{\max,i}$ and minimizing C_{wasted}. Table 3.1 summarizes the key code parameters $C_{\text{max,i}}$, C_{wasted} and system constraints for the schemes considered in this work. A detailed discussion on these parameters is postponed to the later sections.

Scheme	$C_{\max,i}$	C_{wasted}	System constraints
UPC	M_AM_B	$NM_AM_B - 1$	$m_{A,i} = m_{B,i} = 1$
UPC-PC	$\frac{M_{A,i}M_{B,i}}{m_i}$	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_i^2}$	$m_{A,i} = m_{B,i} = m_i \in [1 : \min(K, L)]$
B-PROC	M_AM_B	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_{A,i}m_{B,i}}$ $+(n_AM_B - 1)(1 - \frac{M_A}{m_A})$ $+(n_BM_A - 1)(1 - \frac{M_B}{m_B})$	$N = n_An_B$ $m_{A,i} = m_A$, $m_{B,i} = m_B$ $K \leq n_Am_B$, $L \leq n_Bm_A$
BPC-VO	$M_{A,i}M_{B,i}$	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_{A,i}m_{B,i}}$	$m_{A,i} = 1$ and $m_{B,i} \leq L$ **or** $m_{A,i} \geq 1$ and $m_{B,i} = L$
BPC-HO	$M_{A,i}M_{B,i}$	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_{A,i}m_{B,i}}$	$m_{B,i} = 1$ and $m_{A,i} \leq K$ **or** $m_{B,i} \geq 1$ and $m_{A,i} = K$
BPC-NZO	$M_{A,i}M_{B,i}$	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_{A,i}m_{B,i}}$ $+(\mu_B - 2)(\frac{L}{\mu_B} - 1)\frac{1}{KL}$	$\mu_B \mid L$, $\mu_B \mid m_{B,i}$, $m_{A,i} = K$ and $m_{B,i} \leq L$ **or** $m_{B,i} = \mu_B$,$\mu_B \mid L$, $m_{B,i} \leq L$ and $m_{A,i} \leq K$ **or** $m_{A,i} = 1$, $m_{B,i} < \mu_B$ and $\mu_B \mid L$
BPC-ZZO	$M_{A,i}M_{B,i}$	$\sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_{A,i}m_{B,i}}$ $+(\mu_A - 2)(\frac{K}{\mu_A} - 1)\frac{1}{KL}$	$\mu_A \mid K$, $\mu_A \mid m_{A,i}$, $m_{A,i} \leq K$ and $m_{B,i} = L$ **or** $m_{A,i} = \mu_A$,$\mu_A \mid K$, $m_{A,i} \leq K$ and $m_{B,i} \leq L$ **or** $\mu_A \mid K$, $m_{A,i} < \mu_A$ and $m_{B,i} = 1$

TABLE 3.1: Comparison of the key performance metrics and system constraints of polynomial coding schemes for straggler mitigation.

3.4 Univariate Schemes

In this section, we review the codes based on univariate polynomial interpolation and their performance metrics.

3.4.1 Performance metrics of UPC

First, we investigate the performance metrics of UPC, which is discussed in detail in Section 2.1.3.1. Recall that in the UPC, the encoding polynomials $\mathbf{A}(x)$ and $\mathbf{B}(x)$ given in Equation (2.5) and Equation (2.6), respectively, are evaluated at a distinct $x_i \in \mathbb{R}$ and sent to worker i by the master. Thus, every worker receives one coded partition of $\mathbf{A}$ and one partition of $\mathbf{B}$, i.e., $m_{A,i} = m_{B,i} = 1$ and $M_{A,i} = M_A = 1/K$, $M_{B,i} = M_B = 1/L$, $\forall i \in [1 : N]$.

Worker i is assigned only one computation $A(x_i)B(x_i)$ and hence, it computes and returns the result to the master, implying $\eta_i = m_{A,i}m_{B,i} = 1$. Once the master receives any $R_{th} = KL$ results, it can interpolate the polynomial $\mathbf{A}(x)\mathbf{B}(x)$ of degree $KL-1$. Observe that the coefficients of the interpolated polynomial correspond to the KL sub-products

$\mathbf{A}_i\mathbf{B}_j$, $\forall i \in [1:K]$, $\forall j \in [1:L]$ of $\mathbf{AB}$. Finally, notice that

$$C_{\max,i} = C_{\text{part}} = \frac{1}{KL} = M_A M_B. \tag{3.2}$$

Observe that with $N > R_{th}$, this scheme can tolerate up to $N - R_{th}$ stragglers. It helps to reduce the average computation time thanks to the parallelization afforded by redundant workers. However, all the work done by the $N - R_{\text{th}}$ slowest workers is ignored. In the worst case, where the $N - R_{\text{th}} + 1$ slowest workers finish simultaneously, we have

$$C_{\text{wasted}} = (N - R_{\text{th}})C_{\text{part}} = (N - R_{\text{th}})\frac{1}{KL} = NM_A M_B - 1. \tag{3.3}$$

We observe from Equation (3.3) that without changing the number of workers N or the storage capacities of workers M_A, M_B, it is not possible to improve C_{wasted}, and thus reduce the amount of work lost at workers.

3.4.2 Univariate Polynomial Codes with Partial Computations (UPC-PC)

In this section, to exploit the partial work done at slower workers, we present an extension of UPC called Univariate Polynomial Codes with Partial Computations (UPC-PC), which is based on the multi-message approach and also allows heterogeneous storage capacities at workers. In Univariate Polynomial Codes with Partial Computations (UPC-PC) the encoding polynomials are constructed the same as in UPC, i.e., as in Equation (2.5) and Equation (2.6).

UPC-PC is based on the idea of dividing the task assigned to a worker into smaller sub-tasks, represented by larger K and L values, and assigning multiple tasks to each worker, thereby allowing them to store several partitions. Specifically, we allow worker i to store $m_i = m_{A,i} = m_{B,i}$ coded partitions of $\mathbf{A}$ and $\mathbf{B}$, i.e., $M_{A,i} = m_i/K$ and $M_{B,i} = m_i/L$. For worker i, the master evaluates $\mathbf{A}(x)$ and $\mathbf{B}(x)$ at m_i different points $\{x_{i,1}, \ldots, x_{i,m_i}\}$ such that $x_{i,k} \neq x_{j,l}$ if $(i,k) \neq (j,l), \forall i,j \in [1:N]$ and $\forall k,l \in [1:m_i]$. Worker i computes $\mathbf{A}(x_{i,j})\mathbf{B}(x_{i,j})$ consecutively for $j \in [1:m_i]$ and sends each result to the master as soon as it is completed. Observe that multiplications are only allowed between the same-point evaluations of $\mathbf{A}(x)$ and $\mathbf{B}(x)$, i.e, at $x_{i,j}$, and thus $\eta_i = m_i$, $\forall i \in [1:N]$.

The master can interpolate $\mathbf{A}(x)\mathbf{B}(x)$ as soon as it receives $R_{th} = KL$ responses from the workers. Thus,

$$C_{\text{part}} = \frac{1}{KL} = \frac{M_{A,i}M_{B,i}}{m_i^2}, \tag{3.4}$$

$$C_{\max,i} = m_i C_{\text{part}} = \frac{M_{A,i}M_{B,i}}{m_i}. \tag{3.5}$$

The total fraction of wasted work in the worst case, in which all the workers were up to finish its ongoing partial multiplication once the R_{th}-th result is received by the master, is given by

$$C_{\text{wasted}} = (N-1)\, C_{\text{part}} = (N-1)\, \frac{1}{KL} = \sum_{i=1}^{N-1} \frac{M_{A,i}M_{B,i}}{m_i^2}. \tag{3.6}$$

In contrast to UPC, UPC-PC can improve C_{wasted} by increasing K and L while maintaining N or $M_{A,i}$ and $M_{B,i}$ constant, as shown in Equation (3.6). That is because once the master collects sufficiently many computations from the workers, all the not-yet-completed tasks at the stragglers correspond to a lesser fraction of the full task since now these computations are smaller subtasks compared to UPC. This implies that the smaller the sub-tasks executed at workers, the smaller the work that can be lost at a straggler. On the other hand, since only the multiplications between the same point evaluations of $\mathbf{A}(x)$ and $\mathbf{B}(x)$ are allowed, UPC-PC makes quite an inefficient use of the storage capacity at workers. For example, even if a worker has enough storage to fully store $\mathbf{A}$ and $\mathbf{B}$, i.e., $M_{A,i} = 1$ and $M_{B,i} = 1$, it, alone, can only provide $\min\{K, L\}$ partial computations. Indeed, for a fixed storage capacity at the workers, i.e., $M_{A,i}$ and $M_{B,i}$ are kept constant, the maximum fraction of work done at a worker, $C_{\max,i}$, decreases while K and L increase to improve C_{wasted}, which results in less efficient use of the storage capacities of the workers.

In the next section, we introduce the bivariate polynomial codes to address this problem.

3.5 Bivariate Polynomial Codes

Bivariate polynomial codes for straggler mitigation share many similarities with their univariate counterparts UPC and UPC-PC. In both cases, matrix $\mathbf{A}$ is partitioned row-wise and matrix $\mathbf{B}$ is partitioned column-wise. However, in bivariate polynomial codes, two variables are used to construct the encoding polynomials, as opposed to one variable in univariate codes. Specifically, the encoding polynomials are generated as follows:

$$\mathbf{A}(x) = \mathbf{A}_1 + \mathbf{A}_2 x + \cdots + \mathbf{A}_K x^{K-1}, \tag{3.7}$$

$$\mathbf{B}(y) = \mathbf{B}_1 + \mathbf{B}_2 y + \cdots + \mathbf{B}_L y^{L-1}. \tag{3.8}$$

Depending on the coding scheme, coded matrix partitions $\tilde{A}_{i,k}$ and $\tilde{B}_{i,l}$ are either direct evaluations of the encoding polynomials $\mathbf{A}(x)$ and $\mathbf{B}(y)$, respectively, or the evaluations of their derivatives. After receiving these, the workers calculate their subtasks by multiplying the coded matrix partitions $\tilde{A}_{i,k}$ and $\tilde{B}_{i,l}$'s, and obtain evaluations of the bivariate polynomial

$$\mathbf{A}(x)\mathbf{B}(y) = \sum_{i=1}^{K}\sum_{j=1}^{L} \mathbf{A}_i \mathbf{B}_j x^{i-1} y^{j-1} \tag{3.9}$$

or of its derivatives. From the results communicated by the workers, finally, the master interpolates the bivariate polynomial $\mathbf{A}(x)\mathbf{B}(y)$.

Bivariate coding schemes not only enable heterogeneous storage capacities across workers but also allow for different numbers of stored coded partitions of $\mathbf{A}$ and $\mathbf{B}$ for each worker, i.e., $m_{A,i} \neq m_{B,i}$ in general. The maximum number of computations a worker can generate is $\eta_i = m_{A,i} m_{B,i}$, resulting in $C_{\max,i} = m_{A,i} m_{B,i} C_{\text{part}} = M_{A,i} M_{B,i}$.

In contrast to univariate polynomial coding schemes, the maximum amount of work done at worker i, $C_{\max,i}$, does not decrease with $m_{A,i}$ and $m_{B,i}$ for a given storage capacity $M_{A,i}$ and $M_{B,i}$ in bivariate schemes. In univariate schemes, workers can only use each evaluation of $\mathbf{A}(x)$ and $\mathbf{B}(x)$ for one partial computation, resulting in storage inefficiency. For instance, $A(x_{i,k})B(x_{i,l})$, for $k \neq l$, cannot be used to interpolate $\mathbf{A}(x)\mathbf{B}(x)$ in a univariate scheme. Bivariate polynomial coding eliminates this limitation and allows workers to provide additional useful computations at no additional storage cost. Furthermore, similar to UPC-PC, bivariate polynomial codes can leverage the computational power of the stragglers.

As previously discussed, $\mathbf{A}(x)\mathbf{B}(y)$ has KL coefficients, and thus, KL partial computations are required to interpolate it. However, in bivariate polynomial coding schemes, additional computations may be necessary to ensure decodability in some cases. We will discuss the decodability conditions that lead to this issue in the following sections. For now, we note that in bivariate schemes, the number of computations required to guarantee decodability satisfies $R_{th} \geq KL$. Consequently, we have a new source of wasted computations: $R_{th} - KL$ redundant computations that have been received by the master to ensure decodability but are not used for the actual interpolation, as well as all ongoing computations at the workers when the R_{th}-th computation is received by the master. When calculating C_{wasted}, we consider the worst-case scenario and assume that all ongoing computations at the remaining $N-1$ workers are close to completion, and thus, we count them as wasted computations. As a result, the maximum fraction of wasted computations is given by:

$$
\begin{aligned}
C_{\text{wasted}} &= (N-1)\,C_{\text{part}} + (R_{th} - KL)C_{\text{part}} \\
&= (N-1)\frac{1}{KL} + \frac{R_{th}}{KL} - 1. \qquad (3.10)
\end{aligned}
$$

Before presenting the bivariate schemes, next, we introduce some basic concepts and definitions from polynomial interpolation theory.

Definition 3.1. The interpolation of a bivariate polynomial of the form $\mathbf{A}(x)\mathbf{B}(y)$ can be expressed as a system of linear equations, where the unknowns are $\mathbf{A}_i\mathbf{B}_j$'s, where $i \in [K], j \in [L]$. We define the *interpolation matrix* as the coefficient matrix of this linear system, denoted by $\mathbf{M}$. Depending on the coding scheme, the entries of $\mathbf{M}$ are determined by the monomials of $\mathbf{A}(x)\mathbf{B}(y)$ or their derivatives evaluated at specific points. Hence, each row corresponds to a different evaluation of $\mathbf{A}(x)\mathbf{B}(y)$.

The rules imposed by coding schemes on computations, such as the order in which computations are conducted and the types of computations assigned to workers, may cause $\det(\mathbf{M})$ to become an identically zero polynomial, regardless of the chosen evaluation points. This is an undesirable situation, and it is important to demonstrate that a proposed scheme does not exhibit such behaviour. To this end, we introduce two notions that ensure that undesirable structures are not imposed on the interpolation matrix.

Definition 3.2. [65, Definition 3.1.3] An interpolation scheme is considered *regular* if $\det(\mathbf{M}) \neq 0$ for every set of distinct and non-zero evaluation points. On the other hand, if $\det(\mathbf{M}) \neq 0$ for almost all choices of the evaluation points, the interpolation scheme is referred to as *almost regular*. The property of almost regularity implies that $\det(\mathbf{M})$ is not the zero polynomial in general, and the set of evaluation points for which $\det(\mathbf{M}) = 0$ has Lebesgue measure zero in $\mathbb{R}^2$. The notion of almost regularity is weaker than regularity but still guarantees that the interpolation matrix does not exhibit undesirable structures almost surely.

To gain insight into the practical implications of almost regularity, we consider a scenario where evaluation points are uniformly sampled from the interval $[l, u]$, where $l, u \in \mathbb{R}$ and $l < u$. Since the Lebesgue measure of the set of evaluation points that result in $\det(\mathbf{M}) = 0$ is zero, the probability of sampling such evaluation points is exactly zero. This is due to the use of an uncountable set to sample evaluation points, and the fact that there are infinitely many possible choices of evaluation points. Even if there are countably many bad choices of evaluation points, the invertibility of $\mathbf{M}$ is guaranteed almost surely.

Univariate polynomial interpolation is regular when the evaluation points are distinct and non-zero, as the corresponding interpolation matrix is a Vandermonde matrix, which is known to be invertible. However, for bivariate interpolation, there are few known sufficient conditions for regularity. In the following section, we explore one such case.

3.5.1 Bivariate Polynomial Interpolation on Rectangular Grids

It is well-established that the interpolation of $\mathbf{A}(x)\mathbf{B}(y)$, where $\mathbf{A}(x)$ and $\mathbf{B}(y)$ have degrees $K-1$ and $L-1$, respectively, is regular for any rectangular grid of evaluation points $x_1, x_2, \ldots, x_K \times y_1, y_2, \ldots, y_L$, provided that all x_i's and y_i's are distinct. In the following subsection, we present an interpolation scheme that leverages this fact called B-PROC. The scheme was originally introduced in [57] using product codes, and we present it here using bivariate polynomial codes, which are equivalent to the product-code form in terms of all the performance metrics considered in this study. Furthermore, we generalize the scheme to allow for $m_A \neq m_B$ and $n_A \neq n_B$, where $N = n_A n_B$.

3.5.1.1 Bivariate Product Coding (B-PROC)

In this coding scheme, we impose that all workers can store the same number of $\mathbf{A}$ partitions, i.e., $m_{A,i} = m_A$ for all $i \in [1 : N]$, and the same number of $\mathbf{B}$ partitions, i.e., $m_{B,i} = m_B$ for all $i \in [1 : N]$. Additionally, the number of workers must satisfy the factorization $N = n_A n_B$, where $K \leq m_A n_A$ and $L \leq m_B n_B$. For encoding, the master generates n_A disjoint sets of evaluation points for $\mathbf{A}(x)$, denoted by $\mathcal{X}i = xi, 1, x_{i,2}, \ldots, x_{i,m_A}$ for $i \in [1 : n_A]$, and n_B disjoint sets of evaluation points for $\mathbf{B}(y)$, denoted by $\mathcal{Y}j = yj, 1, y_{j,2}, \ldots, y_{j,m_B}$ for $j \in [1 : n_B]$, where all elements are distinct. The master then enumerates the workers as (i, j), where $i \in [1 : n_A]$ and $j \in [1 : n_B]$, and sends $\mathbf{A}(x_{i,k})$ for $k \in [1 : m_A]$ and $\mathbf{B}(y_{j,l})$ for $l \in [1 : m_B]$ to worker (i, j). Worker (i, j) can compute all the products $\mathbf{A}(x_{i,k})\mathbf{B}(y_{j,l})$ for $k \in [1 : m_A]$ and $l \in [1 : m_B]$.

The set of evaluation points at workers forms a rectangular grid of size $m_A n_A \times m_B n_B$. Notably, n_A workers have the evaluation $\mathbf{B}(\hat{y})$ for any $\hat{y} \in \mathcal{Y}j$, and each of them can compute mA distinct evaluations of the univariate polynomial $\mathbf{A}(x)\mathbf{B}(\hat{y})$, which has degree $K-1$ with respect to x. Once the first K of these evaluations are received at the master, $\mathbf{A}(x)\mathbf{B}(\hat{y})$ can be interpolated. Similarly, for a given $\hat{x} \in \mathcal{X}_i$, $\mathbf{A}(\hat{x})\mathbf{B}(y)$ can be interpolated from any L evaluations, as it is a univariate polynomial in y with degree $L-1$. Consequently, once we have the evaluations of $\mathbf{A}(x)\mathbf{B}(y)$ on any rectangular grid of size $K \times L$, either directly received from the workers or via univariate interpolation, the bivariate interpolation problem can be solved.

However, computations that are simultaneously received from workers and that are already interpolated from previous results are redundant. Such computations may increase the recovery threshold. To minimize such computations, [57] proposed different heuristics to schedule computations at the workers for the specific case of $m_A = m_B$ and $n_A = n_B$. In the following example, we illustrate a case where more than KL computations are collected by the master, but decoding is still not possible by using random computation orders at the workers.

Example 3.1. Consider a scenario where both matrices $\mathbf{A}$ and $\mathbf{B}$ are divided into $K = L = 10$ partitions, and there are $N = 15$ workers, each capable of storing $M_A = 3/10$ of $\mathbf{A}$ and $M_B = 5/10$ of $\mathbf{B}$. We take $n_A = 5$ and $n_B = 3$, and worker (i, j) stores $\{\mathbf{A}(x_{i,1}), \mathbf{A}(x_{i,2}), \mathbf{A}(x_{i,3})\}$ and $\{\mathbf{B}(y_{j,1}), \mathbf{B}(y_{j,2}), \mathbf{B}(y_{j,3}), \mathbf{B}(y_{j,4}), \mathbf{B}(y_{j,5})\}$.

Assuming that the order of computations is random within a worker, Figure 3.2 shows an instance of the received responses from the workers. Each worker is represented by a 3×5 rectangle, and each filled circle represents a received computation by the master. To illustrate how a worker's finished computations look, worker $(4, 2)$ is emphasized in the figure. All elements in the columns and rows coloured green can be interpolated, i.e., decoded, using the received responses on the same column or row. Note that there are columns/rows coloured green even if they have less than 10 computations, such as the column of $x_{4,1}$. These rows and columns can be decoded after decoding rows and columns with at least 10 computations by utilizing all elements in these columns and rows after decoding. Since there must be at least 10 green columns and 10 green rows to decode $\mathbf{A}(x)\mathbf{B}(y)$, the received responses in our example are insufficient, even though the master has received $110 > KL = 100$ responses.

Next, we aim to compute the total fraction of work wasted in the worst-case scenario, which is dependent on the heuristics employed for computation order. To achieve this, we consider a uniform random computation order at the workers, as reported to perform well in [57], and also discussed in Example 3.1. Under this computation order, the computations can be received in any order by the master. In the worst-case scenario, which is depicted in Figure 3.3 for the case in Example 3.1, there may be $K - 1$ fully computed columns, where each column contains exactly $n_B m_B$ computations, except for one column with exactly L computations. Consequently, $n_B m_B - L$ computations in each of the $K - 1$ fully computed columns are wasted. Similarly, there may be $L - 1$ fully computed rows and one row with exactly K computations. In this case, $n_A m_A - K$ computations in each of the $L - 1$ fully computed rows are wasted. Therefore, the total number of wasted computations is given by $(n_B m_B - L)(K - 1) + (n_A m_A - K)(L - 1)$.

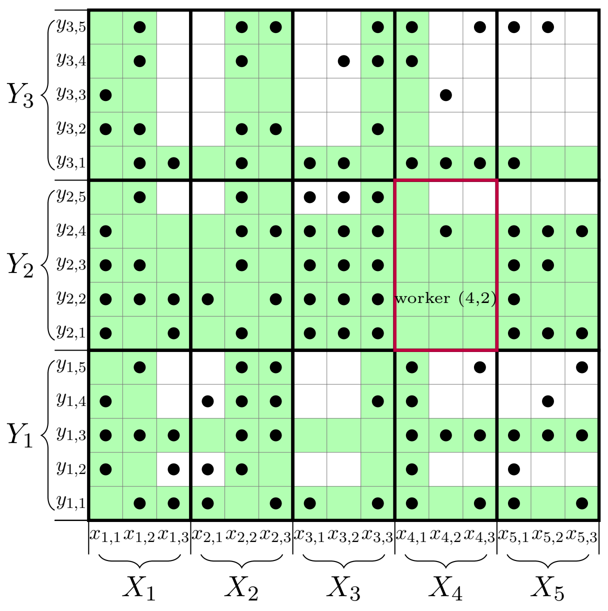

Y_3
$y_{3,5}$
$y_{3,4}$
$y_{3,3}$
$y_{3,2}$
$y_{3,1}$
Y_2
$y_{2,5}$
$y_{2,4}$
$y_{2,3}$
$y_{2,2}$
$y_{2,1}$
Y_1
$y_{1,5}$
$y_{1,4}$
$y_{1,3}$
$y_{1,2}$
$y_{1,1}$
worker (4,2)
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{5,1}$ $x_{5,2}$ $x_{5,3}$
X_1 X_2 X_3 X_4 X_5

FIGURE 3.2: An example instance of the received responses at the master for B-PROC under random computation order.

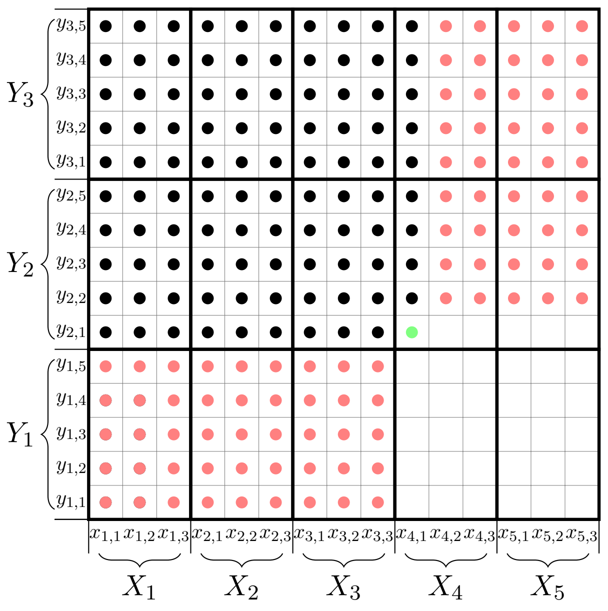

Y_3
$y_{3,5}$
$y_{3,4}$
$y_{3,3}$
$y_{3,2}$
$y_{3,1}$
Y_2
$y_{2,5}$
$y_{2,4}$
$y_{2,3}$
$y_{2,2}$
$y_{2,1}$
Y_1
$y_{1,5}$
$y_{1,4}$
$y_{1,3}$
$y_{1,2}$
$y_{1,1}$
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{5,1}$ $x_{5,2}$ $x_{5,3}$
X_1 X_2 X_3 X_4 X_5

FIGURE 3.3: A worst-case instance of the received responses at the master for B-PROC.

Finally, using this information the worst-case R_{th} of B-PROC is determined as

$$R_{th}^{B-PROC} = KL + (n_B m_B - L)(K-1) + (n_A m_A - K)(L-1). \tag{3.11}$$

It is important to note that this expression represents a worst-case value. Depending on the received responses, the actual number of computations required to guarantee decodability may be much lower. By substituting Equation (3.11) into Equation (3.10), we can determine the fraction of wasted computations for B-PROC in the worst-case scenario. The resulting expression is as follows:

$$\begin{aligned} C_{\text{wasted, B-PROC}} &= (N-1)\frac{1}{KL} + [(n_B m_B - L)(K-1) \\ &\quad + (n_A m_A - K)(L-1)]\frac{1}{KL} \\ &= (N-1)\frac{1}{KL} + (n_B M_B - 1)\left(1 - \frac{M_A}{m_A}\right) \\ &\quad + (n_A M_A - 1)\left(1 - \frac{M_B}{m_B}\right). \end{aligned} \tag{3.12}$$

It is worth noting that this expression is highly dependent on how n_A and n_B are allocated. Increasing the storage, i.e., M_A and M_B, while K and L remain constant, or increasing K and L while the storage remains constant, both increase C_{wasted}. However, it is important to keep in mind that the value of C_{wasted} calculated here is for the worst-case scenario, and the actual situation may not be as bad most of the time.

B-PROC requires additional constraints on the system, i.e., $N = n_A n_B$, $K \leq n_A m_A$, $L \leq n_B m_B$ and homogeneous storage capacities at workers, and yet it is not possible to ensure that the first KL results arriving at the master form a regular interpolation problem. To address these issues, in the next subsection, we propose novel bivariate polynomial codes. However, showing the regularity of these schemes is a hard problem, if not impossible. Therefore, instead, we use the notion of almost regularity, which is a relaxation of regularity and propose almost regular bivariate interpolation schemes.

3.5.2 Almost Regular Bivariate Interpolation Schemes

In this section, we propose several almost regular bivariate interpolation schemes such that the polynomial $\mathbf{A}(x)\mathbf{B}(y)$ is interpolated from its evaluations as well as its derivatives. Such an interpolation is known as *Hermite interpolation* in the literature [66, Chapter 3.6].

3.5.2.1 Encoding Phase

All the almost regular interpolation schemes described in this section have the following encoding procedure in common. Consider the polynomials in Equation (3.7) and Equation (3.8). To each worker $i \in [1:N]$ the master assigns a distinct evaluation point $(x_i, y_i) \in \mathbb{R}^2$, and sends the evaluations of $\mathbf{A}(x)$ and $\mathbf{B}(y)$, and their derivatives up to order $m_{A,i}-1$ and $m_{B,i}-1$, respectively, at (x_i, y_i). That is, the set of coded matrix partitions sent to the worker i by the master are

$$\mathcal{A}_i \triangleq \left\{ \mathbf{A}(x_i), \frac{d\mathbf{A}(x_i)}{dx}, \ldots, \frac{d^{(m_{A,i}-1)}\mathbf{A}(x_i)}{dx^{(m_{A,i}-1)}} \right\}, \tag{3.13}$$

$$\mathcal{B}_i \triangleq \left\{ \mathbf{B}(y_i), \frac{d\mathbf{B}(y_i)}{dy}, \ldots, \frac{d^{(m_{B,i}-1)}\mathbf{B}(y_i)}{dy^{(m_{B,i}-1)}} \right\}. \tag{3.14}$$

For brevity, in the sequel, we use $\partial_k \mathbf{A}(x_i)$ and $\partial_l \mathbf{B}(y_i)$ to denote $\frac{d^k}{dx^k}\mathbf{A}(x_i)$and $\frac{d^l}{dy^l}\mathbf{B}(y_i)$, respectively.

3.5.2.2 Computation Phase

Although the encoding procedures are the same for all proposed almost regular interpolation schemes, they differ in the order in which the assigned computations are conducted. In general, after receiving $\mathcal{A}_i$ and $\mathcal{B}_i$ from the master, each worker i begins computing all the cross products between elements in $\mathcal{A}_i$ and $\mathcal{B}_i$, one by one, and sends the result of each computation to the master as soon as it is completed. To ensure decodability, each worker must follow a specific computation order while conducting their assigned multiplications. To specify such computation orders, we define a quantity called the *priority score* for each scheme, whose value is uniquely defined for each assigned computation. The computations with a lower priority score are conducted before the ones with a higher score.

Specifically, for any worker i, each computation $\partial_k \mathbf{A}(x_i)\partial_l \mathbf{B}(y_i)$ for $k \in [0:K-1]$ and $l \in [0:L-1]$ is assigned a priority score denoted as $S(k,l)$. Worker i computes $\partial_k \mathbf{A}(x_i)\partial_l \mathbf{B}(y_i)$ only after all the computations $\partial_{\tilde{k}} \mathbf{A}(x_i)\partial_{\tilde{l}} \mathbf{B}(y_i)$, where $\tilde{k} \in [0:K-1]$ and $\tilde{l} \in [0:L-1]$ such that $S(\tilde{k},\tilde{l}) < S(k,l)$, have already been computed. It is important to note that priority scores $S(k,l)$ are defined for computations that may not be available at worker i, i.e., $K > k \geq m_{A,i}$ or $L > l \geq m_{B,i}$. Whenever such a computation has the lowest priority score among all the remaining computations at worker i, the worker must discard all the remaining computations and stop.

To better characterize the assigned computations to the workers, we define the notion of *derivative order space* in the following.

Definition 3.3. Derivative Order Space. The derivative order space of a bivariate polynomial $\mathbf{A}(x)\mathbf{B}(y)$ is defined as the 2-dimensional space of all its possible derivative orders. When $\mathbf{A}(x)$ and $\mathbf{B}(y)$ have degrees $K-1$ and $L-1$, respectively, the derivative order space becomes $\{(k,l) : 0 \leq k < K, 0 \leq l < L\}$, where the tuple (k,l) represents the derivative $\partial_k \mathbf{A}(x) \partial_l \mathbf{B}(y)$.

Hence, in each worker, elements of the derivative order space of $\mathbf{A}(x)\mathbf{B}(y)$ correspond to computations assigned to that worker.

In the following, we introduce four different almost regular interpolation schemes which differ in their computation orders.

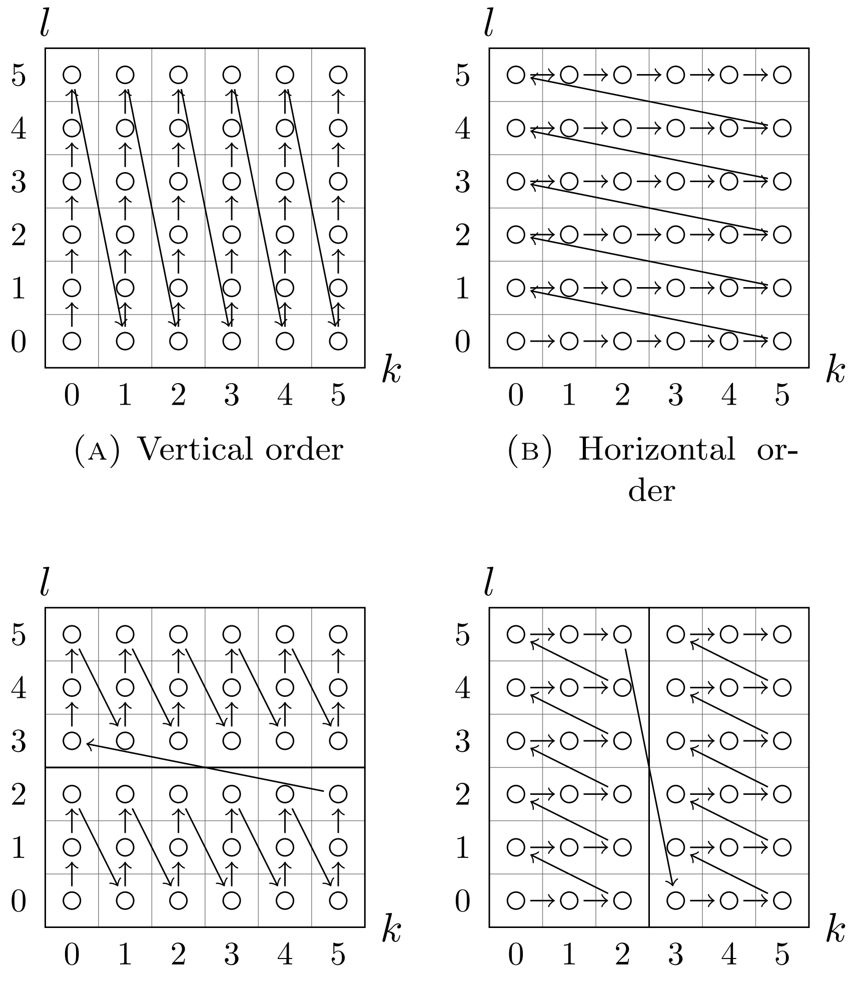
3.5.2.3 Bivariate Polynomial Coding with Vertical Computation Order (BPC-VO)

The Bivariate Polynomial Coding with Vertical Computation Order (BPC-VO) scheme is named after the *vertical computation order* that workers follow, as illustrated in Figure 3.4a for $K = L = 6$. In the vertical computation order, a worker completes the computations in a column k in the derivative order space, i.e., all the computations in $\{\partial_k \mathbf{A}(x_i)\mathbf{B}(y_i), \partial_k \mathbf{A}(x_i)\partial_1 \mathbf{B}(y_i), \ldots, \partial_k \mathbf{A}(x_i)\partial_{L-1}\mathbf{B}(y_i)\}$ before moving on to the computations from column $k+1$. Specifically, for worker i, the priority score for the computation $\partial_k \mathbf{A}(x_i)\partial_l \mathbf{B}(y_i)$, where $k \in [0 : K-1]$ and $l \in [0 : L-1]$, is defined as

$$S_{\mathcal{V}}(k,l) \triangleq (K-1)L\left(\left\lceil \frac{l}{L} \right\rceil - 1\right) + L(k-1) + l. \tag{3.15}$$

Due to the storage constraints, since only the computations $\partial_k \mathbf{A}(x_i)\partial_l \mathbf{B}(y_i)$, $k \in [0 : m_{A,i} - 1]$, and $l \in [0 : m_{B,i} - 1]$ can be computed by worker i, in order to satisfy the vertical computation order without discarding any computations, worker i can store either:

1. a single coded partition of $\mathbf{A}$, and any number of coded partitions of $\mathbf{B}$ less than L, i.e., $m_{A,i} = 1$ and $1 \leq m_{B,i} < L$, or
2. coded partitions of $\mathbf{B}$ equivalent to the full matrix $\mathbf{B}$ in size, and not more than K coded partitions of $\mathbf{A}$, i.e., $1 \leq m_{A,i} \leq K$ and $m_{B,i} = L$.



(A) Vertical order

(B) Horizontal order

(C) N-zig-zag order

(D) Z-zig-zag order

FIGURE 3.4: Illustration of computation orders the workers follow for the proposed almost regular interpolation schemes.

3.5.2.4 Bivariate Polynomial Coding with Horizontal Computation Order (BPC-HO)

In Bivariate Polynomial Coding with Horizontal Computation Order (BPC-HO) scheme, workers follow the *horizontal computation order*, as illustrated in Figure 3.4b for $K = L = 6$. In the horizontal computation order, a worker first completes the computations in a row l in the derivative order space, i.e., all the computations in $\{\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i), \partial_1\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i), \ldots, \partial_{K-1}\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i)\}$ before moving on to the computations from row $l+1$. Specifically, for any worker i, the priority score for the computation $\partial_k\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i)$, $k \in [0 : K-1]$, $l \in [0 : L-1]$ is defined as

$$S_{\mathcal{H}}(k,l) \triangleq K(L-1)\left(\left\lceil \frac{k}{K} \right\rceil - 1\right) + K(l-1) + k. \tag{3.16}$$

Similar to the vertical computation order, due to the storage constraints, since only computations $\partial_k\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i)$ $k \in [0 : m_{A,i} - 1]$, and $l \in [0 : m_{B,i} - 1]$ can be computed by worker i, in order to satisfy the horizontal computation order without discarding any computations, worker i can store either:

1. a single coded partition of $\mathbf{B}$, and any number of coded partitions of $\mathbf{A}$ not more than K, i.e., $1 \leq m_{A,i} \leq K$ and $m_{B,i} = 1$, or
2. coded partitions of $\mathbf{A}$ equivalent to the full matrix $\mathbf{A}$, and any number of coded partitions of $\mathbf{B}$ not more than L, i.e., $m_{A,i} = K$ and $1 \leq m_{B,i} \leq L$.

3.5.2.5 Bivariate Polynomial Coding with N-zig-zag Computation Order (BPC-NZO)

In Bivariate Polynomial Coding with N-zig-zag Computation Order (BPC-NZO) scheme, we relax the vertical computation order by dividing the derivative order space into L/μ_B equal horizontal *blocks*, where μ_B is a design parameter such that $\mu_B \mid L$. For the computation $\partial_k\mathbf{A}(x_i)\partial_l\mathbf{B}(y_i)$, $k \in [0 : K - 1]$ and $l \in [0 : L - 1]$, we define the priority score

$$S_{\mathcal{N}}(k,l) \triangleq (K-1)\mu_B\left(\left\lceil \frac{l}{\mu_B} \right\rceil - 1\right) + \mu_B(k-1) + l, \tag{3.17}$$

which we refer to as *N-zig-zag order*. In Figure 3.4c, we illustrate the N-zig-zag order for $K = L = 6$ and $\mu_B = 3$. For this computation order, we simply apply vertical computation order inside each horizontal block in the derivative order space starting from the lowermost block. Only when all the computations in a block are completed, the computations from the next block can start.

Although it is more relaxed than the vertical computation order, in order to satisfy the N-zig-zag order without discarding any computations at worker i, one of the following conditions must be imposed on $m_{A,i}$ and $m_{B,i}$:

1. $m_{B,i}$ is a positive integer multiple of μ_B, and $m_{A,i} = K$, or
2. $m_{B,i} = \mu_B$ and $1 \leq m_{A,i} \leq K$, or,
3. $m_{A,i} = 1$ and $1 \leq m_{B,i} \leq \mu_B$

Observe that by setting $\mu_B = L$, the N-zig-zag order reduces to the vertical computation order.

3.5.2.6 Bivariate Polynomial Coding with Z-zig-zag Computation Order (BPC-ZZO)

In Bivariate Polynomial Coding with Z-zig-zag Computation Order (BPC-ZZO) scheme, we relax the horizontal computation order by dividing the derivative order space into K/μ_A equal vertical blocks, where μ_A is a design parameter such that $\mu_A \mid L$. For the computation $\partial_k \mathbf{A}(x_i)\partial_l \mathbf{B}(y_i)$, $k \in [0 : K-1]$ and $l \in [0 : L-1]$, we define the priority score as

$$S_{\mathcal{Z}}(k,l) \triangleq (L-1)\mu_A \left(\left\lceil \frac{k}{\mu_A} \right\rceil - 1 \right) + \mu_A(l-1) + k, \tag{3.18}$$

which we refer to as *Z-zig-zag order.* In Figure 3.4d, we visualize the Z-zig-zag computation order when $K = L = 6$ and $\mu_A = 3$. For this computation order, we apply horizontal computation order inside each vertical block starting from the leftmost block. Similarly to N-zig-zag order, only when all the computations in a block are completed, the computations from the next block can start.

In order to satisfy the Z-zig-zag computation order without discarding any computations at worker i, we need to impose one of the following constraints on $m_{A,i}$ and $m_{B,i}$:

1. $m_{A,i}$ is a positive integer multiple of μ_A, and $m_{B,i} = L$, or
2. $m_{A,i} = \mu_A$ and $1 \leq m_{B,i} \leq L$, or
3. $m_{B,i} = 1$ and $1 \leq m_{A,i} \leq \mu_A$

Observe that setting $\mu_A = K$, we recover the horizontal computation order conditions.

3.5.2.7 Decoding Phase

Common to all almost regular interpolation schemes defined in this section, the master receives responses from the workers and decodes $\mathbf{AB}$ by solving a bivariate polynomial interpolation problem. That is, $\mathbf{A}(x)\mathbf{B}(y)$ is interpolated from the evaluations of $\mathbf{A}(x)\mathbf{B}(y)$ and its derivatives. Since the degree of $\mathbf{A}(x)\mathbf{B}(y)$ is KL, to solve the interpolation problem, the master needs at least KL computations returned from the workers. The condition for such an interpolation uniquely exists is that the interpolation matrix $\mathbf{M}$ is invertible.

To better illustrate the concept of interpolation matrix, in Equation (3.19), we provide an example interpolation matrix for a case in which the responses are received from all N workers. In this example, the master received 3 responses from worker 1, and 2 responses from worker N. Note that, in the observed rows, the derivatives are taken with respect

$$\mathbf{M} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^{K-1} & \cdots & x_1^{K-1}y_1^{L-1} \\ 0 & 1 & 2x_1 & 3x_1^2 & \cdots & (K-1)\,x_1^{K-2} & \cdots & (K-1)\,x_1^{K-2}y_1^{L-1} \\ 0 & 0 & 2 & 6x_1 & \cdots & (K-1)(K-2)x_1^{K-3} & \cdots & (K-1)(K-2)x_1^{K-3}y_1^{L-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 & \cdots & x_N^{K-1} & \cdots & x_N^{K-1}y_N^{L-1} \\ 0 & 1 & 2x_N & 3x_N^2 & \cdots & (K-1)\,x_N^{K-2} & \cdots & (K-1)\,x_N^{K-2}y_N^{L-1} \end{bmatrix}. \tag{3.19}$$

to x, and hence, it can be concluded that this interpolation matrix belongs to either BPC-HO scheme or BPC-ZZO scheme.

The next theorem and the following corollary characterise the number of computations needed to guarantee the invertibility of the interpolation matrix, and hence, the unique interpolation of $\mathbf{A}(x)\mathbf{B}(y)$, in the worst-case scenario.

Theorem 3.1. *a) For BPC-NZO, the worst-case recovery threshold is* $R_{th}^{NZO} \triangleq KL + \max\left\{0, (\mu_B - 2)(\frac{L}{\mu_B} - 1)\right\}$.

b) For BPC-ZZO, the worst-case recovery threshold is $R_{th}^{ZZO} \triangleq KL + \max\left\{0, (\mu_A - 2)(\frac{K}{\mu_A} - 1)\right\}$.

Thus, if the number of computations received by the master is at least R_{th}^{NZO} *and* R_{th}^{ZZO} *for BPC-NZO and BPC-ZZO, respectively, then* $\det(\mathbf{M}) \neq 0$ *for almost all choices of the evaluation points.*

The proof of Theorem 3.1 is given in Section 3.7.

Corollary 3.1. *BPC-VO and BPC-HO can be obtained by setting* $\mu_B = L$ *and* $\mu_A = K$ *in BPC-NZO and BPC-ZZO, respectively. Therefore, the recovery thresholds of BPC-VO and BPC-HO are* $R_{th}^{VO} = R_{th}^{HO} \triangleq KL$, *meaning any* KL *computations received by the master results in* $\det(\mathbf{M}) \neq 0$ *for almost all choices of the evaluation points.*

According to Corollary 3.1, for BPC-VO and BPC-HO, every partial computation sent by the workers to the master is useful, i.e., $R_{th} = KL$. Therefore, for these schemes, the computations are one-to-any replaceable. Thus, according to Equation (3.10), we have

$$C_{\text{wasted, BPC-VO}} = C_{\text{wasted, BPC-HO}} = (N-1)\frac{1}{KL}. \tag{3.20}$$

Compared to $C_{\text{wasted, B-PROC}}$ given in Equation (3.12), $C_{\text{wasted, BPC-VO}}$ and $C_{\text{wasted, BPC-HO}}$ are much lower, and this is the main advantage of BPC-VO and BPC-HO over B-PROC. On the other hand, while in the BPC-HO and BPC-VO, η_i is limited by the constraints imposed on $m_{A,i}$ and $m_{B,i}$, in B-PROC, all the available storage can be fully exploited. Therefore, B-PROC has a better storage efficiency $C_{\max,i}$ compared to BPC-VO and BPC-HO. The main motivation of introducing BPC-NZO and BPC-ZZO is to relax these constraints. According to Theorem 3.1, this can be done at the cost of potentially introducing redundant computations; however, the number of redundant computations needed is still much less than those needed for B-PROC. Specifically, from Equation (3.10), we obtain

$$\begin{aligned} C_{\text{wasted, BPC-NZO}} &= (N-1)\frac{1}{KL} + (\mu_B - 2)\left(\frac{L}{\mu_B} - 1\right)\frac{1}{KL}, \\ C_{\text{wasted, BPC-ZZO}} &= (N-1)\frac{1}{KL} + (\mu_A - 2)\left(\frac{K}{\mu_A} - 1\right)\frac{1}{KL}. \end{aligned} \tag{3.21}$$

The following example further illustrates the storage efficiency of bivariate polynomial codes.

Example 3.2. Let us consider a scenario where $K = L = 8$, i.e., the size of partitions of $\mathbf{A}$ and $\mathbf{B}$ are equal, and each worker can store 8 coded matrix partitions in total, i.e., $m_{A,i} + m_{B,i} = 8$. Firstly, we note that univariate schemes can only carry out $\eta_i = m_{A,i} = m_{B,i} = 4$ computations. On the other hand, while keeping $m_{A,i} = m_{B,i} = 4$, B-PROC can achieve $\eta_i = 16$ computations. However, in BPC-VO and BPC-HO, the same worker can generate at most $\eta_i = 7$ computations by setting $m_{A,i} = 1, m_{B,i} = 7$ for BPC-VO, or $m_{A,i} = 7, m_{B,i} = 1$ for BPC-HO since it is not possible to satisfy condition 2 of BPC-VO and BPC-HO under this storage capacity. Finally, for BPC-NZO or BPC-ZZO, by setting $\mu_A = \mu_B = 4$ and $m_{A,i} = m_{B,i} = 4$, we can also reach $\eta_i = 16$. It is worth noting that, although it is the case in this specific example, BPC-NZO and BPC-ZZO may not always achieve the same storage efficiency as B-PROC. Still, they can usually perform very close to it.

Remark 3.1. Note that R_{th}^{NZO} and R_{th}^{ZZO} provided in Theorem 3.1 are worst-case values. Depending on the number of computations each worker sends to the master, smaller values, even KL computations may be enough. In Section 3.7, Lemma 3.3 presents certain conditions under which the computations received from the workers are useful. If the number of computations provided by all workers satisfies these conditions, then all computations are useful and KL computations are enough. Otherwise, we need to discard some computations and since we need to compensate for these discarded computations, the recovery threshold may increase up to the values presented in Theorem 3.1. In Section 3.7, the discussion following Lemma 3.3 explains what kind of computations we discard to guarantee almost regular decodability.

Remark 3.2. When the conditions of Theorem 3.1 are satisfied, the bivariate polynomial interpolation problem has a unique solution. The interpolation problem can be solved by inverting the interpolation matrix and multiplying it with the vector of responses collected from the workers. This has a complexity of $\mathcal{O}(rc(KL)^2)$. However, such an interpolation strategy may result in large numerical errors, and hence, more sophisticated methods, such as Newton interpolation, may be needed in practice [67,68]. Moreover, given the complexity of univariate polynomial interpolation can be reduced to $\mathcal{O}(n \log^2 n \log\log n)$, where n is the degree of the polynomial [69], by choosing the interpolation points carefully, we conjecture that the complexity of our decoding scheme can be also reduced to almost linear complexity. Although investigating this aspect is beyond the scope of this dissertation, we note that it is an interesting future research direction.

Remark 3.3. We note the similarity between the proposed bivariate polynomial codes and Reed-Muller codes [70,71]. In Reed-Muller codes, codewords are generated by evaluating a multivariate polynomial in a similar way to our scheme. This means that our scheme can be considered a variant of Reed-Solomon codes. However, there is a difference in the structure of the decoding polynomial $A(x)B(y)$. In our coding scheme, it is generated by multiplying two other polynomials, $A(x)$ and $B(y)$, which results in all the monomials $x^{i-1}y^{j-1}$, where i is in the range $[1:K]$ and j is in the range $[1:L]$.

On the other hand, bivariate Reed-Muller codes rely on generator polynomials that have monomials of the form x^iy^j, such that $i+j \leq r$, where r is the code parameter determining the polynomial degree. Another difference is that the generator polynomial of Reed-Muller codes is evaluated at every possible element of the field in which the code is defined, which determines the block length of the code. However, in our matrix multiplication problem, it is not feasible to evaluate the polynomial $A(x)B(y)$ since we aim to cover a large real space. Even if we use a finite field, as we will do in Chapter 4, our field size is typically much larger than the total number of evaluations of the decoding polynomial that we obtain from the workers, in order to cover large real intervals.

3.5.2.8 Selecting between computation orders

In this subsection, we provide a guideline on how to choose the most appropriate computation order to employ depending on the dimensions, i.e., c and r and the number of partitions, i.e., K and L, of the multiplied matrices.

When the partitions of $\mathbf{B}$ are smaller than those of $\mathbf{A}$, i.e., $c/L < r/K$, under a fixed storage capacity, reducing $m_{A,i}$ by 1 will increase $m_{B,i}$ at least by 1. Since, in this case, the constraints of vertical-type computation orders BPC-VO and BPC-NZO can be satisfied more easily than those of BPC-HO and BPC-ZZO, the schemes having a vertical-type computation order should be chosen. On the other hand, when $r/K \leq c/L$, we should prefer horizontal ordering schemes BPC-HO or BPC-ZZO.

Choosing between BPC-HO and BPC-ZZO when $r/K < c/L$, or between BPC-VO and BPC-NNO when $c/L < r/K$, depends on the storage capacity per worker and is discussed further in Section 3.6.

3.5.2.9 Alternative formulation of almost regular interpolation schemes

The reason we formulate almost regular interpolation schemes in terms of Hermite interpolation is to shorten the proof of Theorem 3.1. Alternatively, instead of interpolating $\mathbf{A}(x)\mathbf{B}(y)$ from the evaluations of its derivatives, i.e., Hermite interpolation, almost regular interpolation schemes can also be formulated as the interpolation of $\mathbf{A}(x)\mathbf{B}(y)$ directly from its evaluations, as done in B-PROC. Such an approach is equivalent to the Hermite interpolation-based formulation, under the almost regularity condition. A more technical discussion on this topic is presented in Section 3.11.

3.6 Numerical Results

The purpose of this section is to compare the schemes presented throughout this chapter in terms of their average computation time. We focus solely on computation time since the bivariate polynomials to be interpolated in B-PROC, BPC-VO, BPC-HO, BPC-NZO, and BPC-ZZO schemes have the same number of coefficients. Therefore, variations in the encoding and decoding times of these schemes are considered negligible. Additionally, we assume that the communication time is negligible.

We model the computation speed of the workers by the shifted exponential model [59,72], which is commonly used in the literature to analyze the performance of coded computation schemes. In this model, the probability that a worker finishes at least p computations by time t is

$$F(p,t) = \begin{cases} 1 - e^{-\lambda(\frac{t}{p} - \nu)}, & \text{if } t \geq p\nu \\ 0, & \text{otherwise.} \end{cases} \tag{3.22}$$

Thus, the probability that a worker completes exactly p computations by time t is given by $P(p,t) = F(p,t) - F(p+1,t)$ assuming $F(0,t) = 1$, and $F(p_{max}+1,t) = 0$, where p_{max} is the maximum number of computations a worker can complete. In Equation (3.22), ν represents the minimum duration required to complete one computation, while the scale parameter λ controls the variance of computation times. A smaller value of λ indicates more variance and thus more heterogeneous computation speeds among the workers.

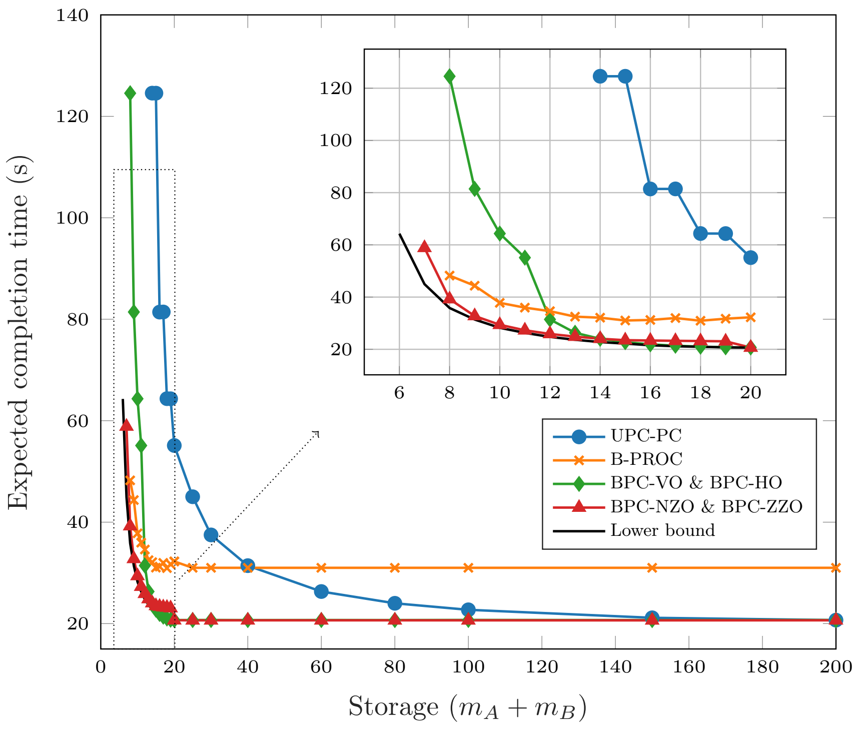


FIGURE 3.5: Average computation times of univariate and bivariate polynomial codes as a function of available storage when partitions of **A** and **B** have equal sizes.

Throughout our experiments, to cover more heterogeneous cases, we choose $\nu = 0.01$ and $\lambda = 0.1$.

To compute the expected computation time for each scheme under different storage availability, we run Monte Carlo simulations. We consider two scenarios: one in which the partitions of **A** and **B** have equal sizes, i.e., $\frac{c}{L} = \frac{r}{K}$, and another in which the partitions of **B** are twice as large as the partitions of **A**, i.e., $\frac{c}{L} = \frac{2r}{K}$. In both cases, we assume that the workers have the same storage capacity, as required by B-PROC. Thus, $M_{A,i} = M_A$ and $M_{B,i} = M_B$, $\forall i \in [1:N]$. We set $K = L = 10$ and assume $N = 15$. In both scenarios, we set $\mu_B = 5$ and $\mu_A = 5$ for BPC-NZO and BPC-ZZO, respectively. For each storage value, we run 10^4 experiments. The results of the first scenario and the second scenario are given in Figure 3.5 and Figure 3.6, respectively. For each scheme, the minimum storage required to complete $KL = 100$ computations with $N = 15$ workers is different. Therefore, we plot each scheme starting from a different minimum storage value.

We first consider the scenario in which the partitions of **A** and **B** have equal size. In this case, since we also have $K = L$ and $\mu_A = \mu_B = 5$, there is no difference between BPC-HO and BPC-VO, and also no difference between BPC-NZO and BPC-ZZO. In Figure 3.5, we observe that BPC-NZO and BPC-ZZO result in a much lower expected computation time than the other schemes for low storage capacities. Even though we allow partial computations, the univariate polynomial coding, which is UPC-PC, performs far worse than all the others due to inefficient use of the storage resulting in a much smaller fraction of work done per worker compared to other schemes.

In B-PROC, despite the optimality in the storage allocation between $m_{A,i}$ and $m_{B,i}$, we see that the higher number of useless computations aggravates the average computation time. For the same reason, increasing the storage capacity does not improve the average computation time beyond a certain point. While simulating B-PROC, we use a random computation order at the workers, which is reported to perform well in [57], and stop the computation as soon as the master is able to decode.

On the other hand, for BPC-NZO and BPC-ZZO, we consider the worst-case scenario, in which the master starts decoding only after $(\mu_B - 2)(\frac{L}{\mu_B} - 1)$ computations for BPC-NZO or $(\mu_A - 2)(\frac{K}{\mu_A} - 1)$ computations for BPC-ZZO are collected. Thus, we can expect the performance of BPC-NZO and BPC-ZZO to be even better than what we observe in Figure 3.5.

We also observe that BPC-VO and BPC-HO perform significantly better than B-PROC and UPC-PC for intermediate and large storage values. For this storage regime, we also observe that BPC-HO and BPC-VO perform slightly better than BPC-ZZO and BPC-NZO due to the first constraint of these schemes. For instance, in BPC-NZO, when $m_{A,i} = K$, we need $m_{B,i}$ to be a multiple of μ_B. Therefore, increasing storage capacity while keeping $m_{A,i} = K$ improves the expected computation time at some specific storage values. This is the reason for the improvement we observe in the expected computation time of the BPC-NZO in Figure 3.5 when the storage is 20.

On the other hand, in the low storage regime, there is significant performance degradation of BPC-VO and BPC-HO due to the restrictive constraints of these schemes at small storage values. Since in BPC-NZO and BPC-ZZO, the constraints of BPC-VO and BPC-HO are relaxed especially for small storage values, we observe that BPC-NZO and BPC-ZZO are superior in low storage regimes.

To evaluate the performance of our schemes, we also plot a lower bound on the average computation time of any bivariate polynomial-based coding scheme, assuming there are no redundant computations and the storage allocation between $m_{A,i}$ and $m_{B,i}$ is optimal. We observe that the average computation time of the proposed bivariate schemes is quite close to this lower bound, especially for the intermediate and high storage regimes. In a low storage regime, we observe that BPC-NZO and BPC-ZZO perform close to the lower bound, although for very low values of storage, the gap between BPC-NZO/ZZO and the lower bound increases. This is due to the third constraint of these schemes, which forbids optimal storage allocation between $m_{A,i}$ and $m_{B,i}$. This suggests that there might be

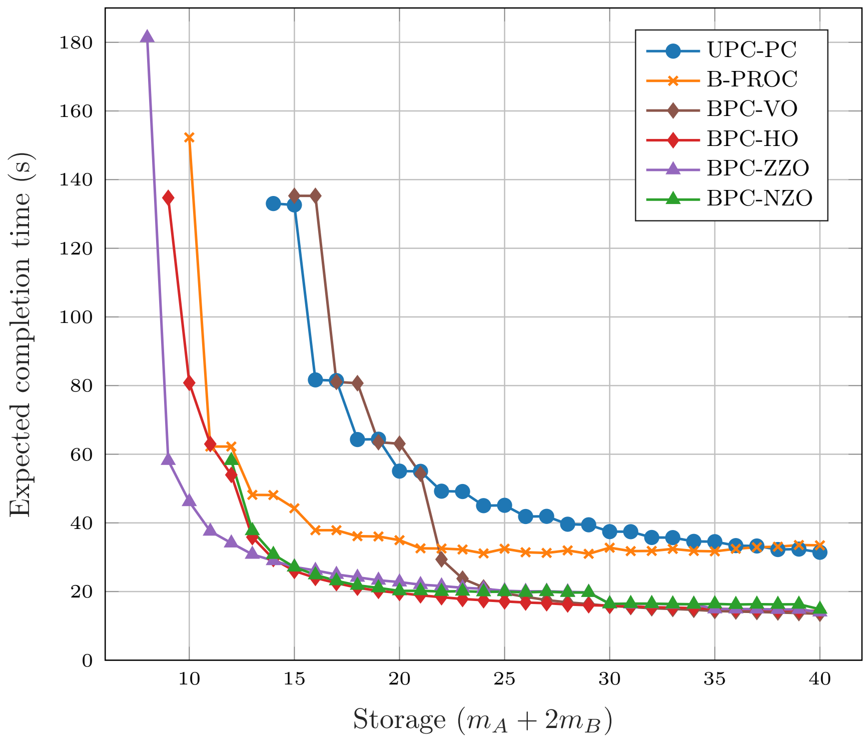


FIGURE 3.6: Average computation times of univariate and bivariate polynomial codes as a function of storage capacity, when partitions of **B** are twice larger than partitions of **A**.

still room for improvement in the trade-off between the expected computation time and the storage capacities of the workers.

Next, we consider the scenario in which the partitions of **B** are twice as large as those of **A**. In Figure 3.6, we observe that neither BPC-NZO and BPC-ZZO nor BPC-VO and BPC-HO are equivalent. Referring to the discussion in Section 3.5.2, since the partitions of **B** are larger in this case, decreasing $m_{B,i}$ by one may increase $m_{A,i}$ more than one, making it easier to satisfy the constraints of horizontal-type schemes, i.e., BPC-ZZO and BPC-HO. Therefore, we expect they perform better than the schemes with vertical-type order. We verify this in Figure 3.6, in which the performances of BPC-ZZO and BPC-HO are superior, especially in the low storage regime.

We also observe that for low and intermediate values of storage, the performance of BPC-VO degrades close to that of UPC-PC. This is because computations are done column-by-column in BPC-VO, as shown in Figure 3.4a, and assigning one more computation requires twice as much storage availability compared to BPC-HO and BPC-ZZO. BPC-NZO suffers from the same problem, but since the constraints are relaxed in the zig-zag order, i.e., the computation grid is divided into blocks, its performance stays reasonable. We observe that it performs similarly to BPC-HO in the intermediate and large storage values. Finally, similar to Figure 3.5, we observe that for the large storage regime, almost regular schemes perform similarly to each other and much better than B-PROC and UPC-PC.

3.7 Proof of Theorem 3.1

In this section, we provide the proof of Theorem 3.1. First, without loss of generality, we assume $[1:n], 1 \leq n \leq N$ is the set of workers which provide at least one computation by the time the master collects sufficient responses to decode $\mathbf{AB}$. Consider an interpolation matrix $\mathbf{M}$ as defined in Definition 3.1. To prove its invertibility, we use Taylor series expansion of $\det(\mathbf{M})$. Note that $\det(\mathbf{M})$ is a polynomial in the evaluation points $z_i \triangleq (x_i, y_i), i \in [1:n]$. We can write the Taylor series expansion of $\det(\mathbf{M})$ around (x_i, y_i) by taking the evaluation point (x_j, y_j) as the variable, as:

$$\det(\mathbf{M}) = \sum_{(\alpha_1,\alpha_2)\in\mathbb{N}^2} \frac{1}{\alpha_1!\alpha_2!}(x_j - x_i)^{\alpha_1}(y_j - y_i)^{\alpha_2} D_{\alpha_1,\alpha_2}(\tilde{Z}), \tag{3.23}$$

where $\tilde{Z} \triangleq \{(x_k, y_k), k \in [1:n]\} \setminus \{(x_j, y_j)\}$, and

$$D_{\alpha_1,\alpha_2}(\tilde{Z}) \triangleq \frac{\partial^{\alpha_1+\alpha_2}}{\partial x_j^{\alpha_1} \partial y_j^{\alpha_2}} \det(\mathbf{M})\bigg|_{x_j=x_i, y_j=y_i}. \tag{3.24}$$

We call (x_i, y_i) the *pivot* node and (x_j, y_j) the *variable* node in this expansion.

When none of the monomials in the set $\{x^{\alpha_1}y^{\alpha_2} \mid (\alpha_1, \alpha_2) \in \mathbb{N}^2\}$ can be expressed as a linear combination of any of the other monomials in the set, then the monomials are said to be *linearly independent.* In this sense, the monomials $(x_j - x_i)^{\alpha_1}(y_j - y_i)^{\alpha_2}$ in Equation (3.23) are linearly independent for different (α_1, α_2) pairs, as long as there is no dependence between x_i, x_j and y_i, y_j. Consequently, $\det(\mathbf{M}) = 0$ for all values of $(x_j, y_j) \in \mathbb{R}^2$, if and only if $D_{\alpha_1,\alpha_2}(\tilde{Z}) = 0, \forall(\alpha_1, \alpha_2) \in \mathbb{N}^2$. That is, to show that $\mathbf{M}$ is non-singular, it suffices to show that there exists an (α_1, α_2) pair such that $D_{\alpha_1,\alpha_2}(\tilde{Z})$ is nonzero.

Let us choose some (α_1, α_2) pair, and analyse $D_{\alpha_1,\alpha_2}(\tilde{Z})$. Notice that, $D_{\alpha_1,\alpha_2}(\tilde{Z})$ is a polynomial in the evaluation points that compose $\tilde{Z}$. Specifically, it does not depend on x_j and y_j since the derivatives were taken with respect to these variables, and then evaluated at $x_j = x_i$ and $y_j = y_i$. We call this procedure the *coalescence* of the evaluation points (x_i, y_i) and (x_j, y_j) into (x_i, y_i). Next, to show $D_{\alpha_1,\alpha_2}(\tilde{Z}) \neq 0$, we do a new coalescence, i.e., we write the Taylor series expansion of $D_{\alpha_1,\alpha_2}(\tilde{Z})$ on a new variable point, choose a new (α_1, α_2) pair, and coalescence them into (x_i, y_i). Our proof technique is based on such recursive Taylor series expansions until all evaluation points are coalesced into one. We will present a technique to choose (α_1, α_2) pairs for each of these recursive steps, which guarantees to obtain a non-zero polynomial at the final coalescence step.

In the following, we first present some preliminaries which are needed to present our technique for choosing (α_1, α_2).

3.7.1 Preliminaries

In order to choose an appropriate (α_1, α_2) pair at each coalescence step, we need to analyze $D_{\alpha_1,\alpha_2}(\tilde{Z})$. Since $D_{\alpha_1,\alpha_2}(\tilde{Z})$ is derived from the Taylor series expansion of a determinant, in some cases, we can write it, again, in terms of the determinants of other matrices, which turns out to be more insightful than using its polynomial form. Before showing this, we introduce the notions of *derivative set* and *shift*, which will be useful in the rest of the proof.

Definition 3.4. Associated to every evaluation point $z_i \triangleq (x_i, y_i), i \in [1 : n]$, there may be one or more rows in $\mathbf{M}$ each corresponding to a different derivative order of $\mathbf{A}(x)\mathbf{B}(y)$ evaluated at z_i, i.e., different computation received from worker i. We define the *derivative set*, $\mathcal{U}_{z_i,M}$, of node z_i as the multiset[1] of derivative orders associated with z_i in $\mathbf{M}$, i.e., we say $(d_x, d_y) \in \mathcal{U}_{z_i,M}$, if $\mathbf{M}$ has a row corresponding the evaluation $\partial_{d_x}\mathbf{A}(x_i)\partial_{d_y}\mathbf{B}(y_i)$ or equivalently the master received the evaluation $\partial_{d_x}\mathbf{A}(x_i)\partial_{d_y}\mathbf{B}(y_i)$ from worker i.

Definition 3.5. Let $\mathbf{M} \in \mathbb{R}^{KL\times KL}$ be an interpolation matrix such that at least one of its rows depends on (x_j, y_j), and let r_i denote the i^{th} row in $\mathbf{M}$. We define a *simple shift*[2] as

$$\partial_{i,x_j}\mathbf{M} \triangleq \left[r_1^T, \dots, \frac{\partial}{\partial x_j} r_i^T, \dots, r_{KL}^T\right]^T$$

or

$$\partial_{i,y_j}\mathbf{M} \triangleq \left[r_1^T, \dots, \frac{\partial}{\partial y_j} r_i^T, \dots, r_{KL}^T\right]^T.$$

Assume that the i^{th} row of $\mathbf{M}$ corresponds to $\partial_{d_{i,x}}\mathbf{A}(x_j)\partial_{d_{i,y}}\mathbf{B}(y_j)$. Then, the derivative sets of node z_i associated to matrices $\partial_{i,x_j}\mathbf{M}$ and $\partial_{i,y_j}\mathbf{M}$ are shifted versions of the ones associated to $\mathbf{M}$, in the sense that, $\mathcal{U}_{z_j,\partial_{i,x_j}\mathbf{M}} = \{(d_{i,x}+1, d_{i,y})\} \cup \mathcal{U}_{z_j,\mathbf{M}} \setminus \{(d_{i,x}, d_{i,y})\}$ and $\mathcal{U}_{z_j,\partial_{i,y_j}\mathbf{M}} = \{(d_{i,x}, d_{i,y}+1)\} \cup \mathcal{U}_{z_j,\mathbf{M}} \setminus \{(d_{i,x}, d_{i,y})\}$. Note that if the multiplicity of any element in a derivative set is greater than one, then the corresponding interpolation

[1]We use the notion of multiset instead of the notion of set as we allow multiple instances for each of its elements. The number of instances of a given element is called the multiplicity of that element in the multiset.

[2]The term *shift* referring to derivatives of interpolation matrices highlights the fact that derivatives applied to interpolation matrices correspond to shifts in their derivative sets when depicted in the derivative order space, as shown in Figure 3.7

matrix has at least two identical rows making the matrix singular. ∂_{i,x_j} is called a *regular simple shift*, if all elements in $\mathcal{U}_{z_j,\partial_{i,x_j}\mathbf{M}}$ have a multiplicity of one. Similarly, ∂_{i,y_j} is a regular simple shift if all elements in $\mathcal{U}_{z_j,\partial_{i,y_j}\mathbf{M}}$ have a multiplicity of one.

Definition 3.6. Let $\boldsymbol{k}$ and $\boldsymbol{l}$ be vectors such that $\boldsymbol{k} \in \{0, 1, \cdots, K-1\}^{R_{th}}$ and $\boldsymbol{l} \in \{0, 1, \cdots, L-1\}^{R_{th}}$. We define *the composition of simple shifts* as

$$\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M} = \partial_{1,x_j}^{\boldsymbol{k}(1)}\partial_{2,x_j}^{\boldsymbol{k}(2)}\cdots\partial_{R_{th},x_j}^{\boldsymbol{k}(R_{th})}\partial_{1,y_j}^{\boldsymbol{l}(1)}\partial_{2,y_j}^{\boldsymbol{l}(2)}\cdots\partial_{R_{th},y_j}^{\boldsymbol{l}(R_{th})}\mathbf{M}. \tag{3.25}$$

That is, the i^{th} element of $\boldsymbol{k}$ denotes the order of the derivative of i^{th} row of $\mathbf{M}$ with respect to the variable x_j, and the i^{th} element of $\boldsymbol{l}$ denotes the order of the derivative of i^{th} row of $\mathbf{M}$ with respect to the variable y_j. In fact, Equation (3.25) is not the only way to compute $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}$ since the derivative operation is commutative. One can compute $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}$ in any other order. Each of these possible orders is referred to as a derivative path. If a derivative path involves only regular simple shifts, i.e. after each derivative there are not two equal rows, then it is called a *regular derivative path.* We denote the number of regular derivative paths by $C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M})$.

The following lemma provides an expression for the derivatives of the determinant of an interpolation matrix in terms of a weighted sum of determinants of other interpolation matrices.

Lemma 3.1. *Let $\boldsymbol{k} \in [0 : K-1]^{KL}$, $\boldsymbol{l} \in [0 : L-1]^{KL}$ and $\alpha_1 = \sum_{i=1}^{KL}\boldsymbol{k}(i)$ and $\alpha_2 = \sum_{i=1}^{KL}\boldsymbol{l}(i)$. Then, we have*

$$\frac{\partial^{\alpha_1+\alpha_2}}{\partial x_j^{\alpha_1}\partial y_j^{\alpha_2}}\det(\mathbf{M})\bigg|_{x_j=x_i,y_j=y_i} = \sum_{(\boldsymbol{k},\boldsymbol{l})\in\mathcal{R}_{\mathbf{M}}(\alpha_1,\alpha_2)} C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M})\det\left(\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}\right)\bigg|_{x_j=x_i,y_j=y_i} \tag{3.26}$$

where $\mathcal{R}_{\mathbf{M}}(\alpha_1, \alpha_2)$ is the set of $(\boldsymbol{k}, \boldsymbol{l})$ pairs satisfying $C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}) \neq 0$, i.e., there is at least one derivative path for which $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}$ can be applied by using only regular simple shifts.

We defer the proof of Lemma 3.1 to Section 3.8.

3.7.2 Choosing an (α_1, α_2) pair in a coalescence

Recall that our objective is to find an (α_1, α_2) pair for each step in the successive coalescence procedure. While Lemma 3.1 is an important step in this direction as it allows us to express $D_{\alpha_1,\alpha_2}(\tilde{Z})$ as a sum of determinants of interpolation matrices, it still does not provide us with a clear clue on how to choose (α_1, α_2) so that $D_{\alpha_1,\alpha_2}(\tilde{Z}) \neq 0$.

To address this issue, we define a structure over the derivative sets of the interpolation matrices, similar to the ones defined for the computation orders in Section 3.5.2. This

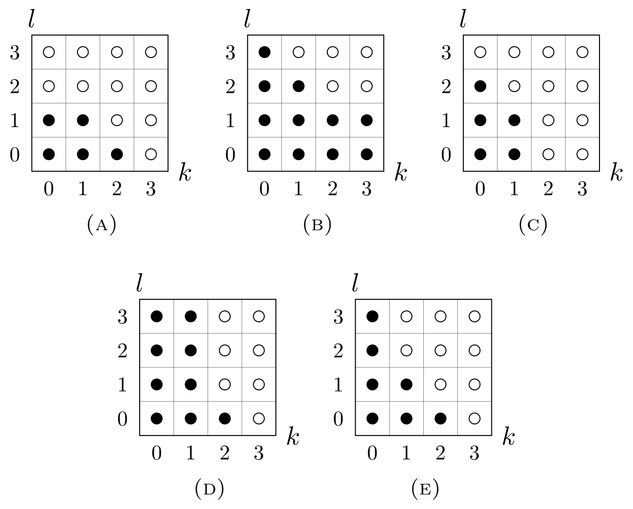


FIGURE 3.7: Illustrations of N-zig-zag ordered (a,b), Z-zig-zag ordered (c,d) and neither N-zig-zag ordered nor Z-zig-zag ordered derivative sets. (e).

structure will eventually help us to define the *quasi-unique shift* pairs (α_1, α_2), which satisfy the conditions needed for completing a coalescence procedure successfully.

Definition 3.7. A derivative set $\mathcal{U}_{z,\mathbf{M}}$ is said to be *N-zig-zag ordered* with parameter μ_B if $(i,j) \in \mathcal{U}_{z,\mathbf{M}}$ implies that all the derivatives with order (k,l) such that $S_{\mathcal{N}}(1,1) \geq S_{\mathcal{N}}(k+1,l+1) \geq S_{\mathcal{N}}(i+1,j+1)$ are also in $\mathcal{U}_{z,\mathbf{M}}$, where $S_{\mathcal{N}}(k,l) = (K-1)\mu_B \left(\left\lceil \frac{l}{\mu_B} \right\rceil - 1 \right) + \mu_B(k-1) + l$, which is the priority score for N-zig-zag order defined in Equation (3.17). Similarly, $\mathcal{U}_{z,\mathbf{M}}$ is *Z-zig-zag ordered* with parameter μ_A if $(i,j) \in \mathcal{U}_{z,\mathbf{M}}$ implies that all (k,l) such that $S_{\mathcal{Z}}(1,1) \geq S_{\mathcal{Z}}(k+1,l+1) \geq S_{\mathcal{Z}}(i+1,j+1)$ are in $\mathcal{U}_{z,\mathbf{M}}$, where $S_{\mathcal{Z}}(k,l) = (L-1)\mu_A \left(\left\lceil \frac{k}{\mu_A} \right\rceil - 1 \right) + \mu_A(l-1) + k$ as defined in Equation (3.18).

Example 3.3. Consider the derivative sets for $K = L = 4$. The derivative sets illustrated in Figure 3.7a and Figure 3.7b are N-zig-zag ordered for $\mu_B = 2$, and the sets in Figure 3.7c and Figure 3.7d are Z-zig-zag ordered for $\mu_A = 2$. The set in Figure 3.7e is neither N-zig-zag nor Z-zig-zag ordered.

For brevity, we will use the N-zig-zag order in the following discussion. This will allow us to prove part *a* of Theorem 3.1. The proof of part *b* follows similarly using the Z-zig-zag order instead, and thus, we omit it here.

Definition 3.8. Suppose that (x_j, y_j) is the variable node and (x_i, y_i) is the pivot node. Let $\mathcal{U}_{z_j,\mathbf{M}}$ obey the N-zig-zag order, and $(\boldsymbol{k}^*, \boldsymbol{l}^*) \in \mathcal{R}_{\mathbf{M}}(\alpha_1, \alpha_2)$. We define $\mathbf{M}^* \triangleq \nabla^{x_j,y_j}_{\boldsymbol{k}^*,\boldsymbol{l}^*}\mathbf{M}|x_j = x_i, y_j = y_i$. For an (α_1, α_2) pair, if $(\boldsymbol{k}^*, \boldsymbol{l}^*)$ is the only element of $\mathcal{R}_{\mathbf{M}}(\alpha_1, \alpha_2)$ such that $\mathcal{U}_{z_j,\mathbf{M}^*}$ obeys the N-zig-zag order, then (α_1, α_2) is called *quasi-unique*.

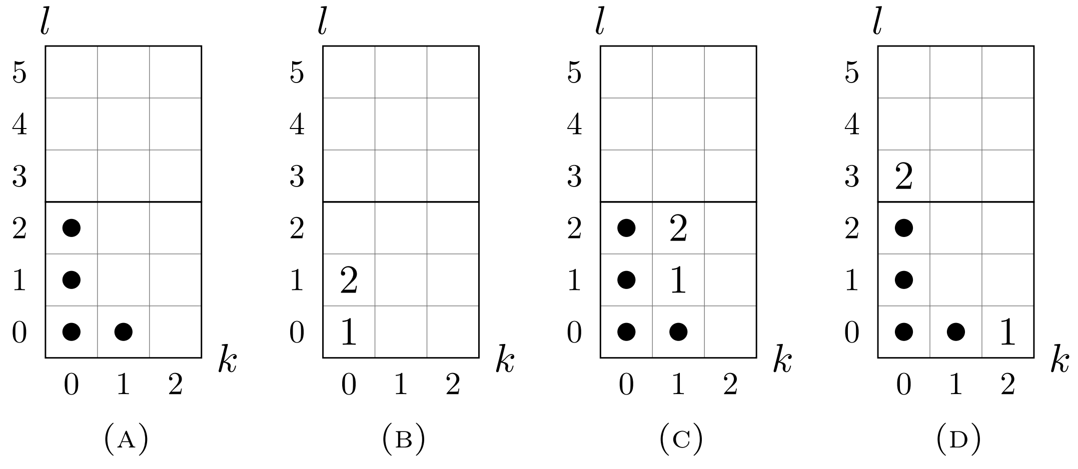


FIGURE 3.8: Depictions of derivative sets in Example 3.4.

Example 3.4. Let us consider $(\alpha_1, \alpha_2) = (2, 2)$ and take $K = 3$, $L = 6$, and $\mu_B = 3$. We assume that the derivative sets of the pivot and variable nodes are as depicted in the derivative order space in Figure 3.8a and Figure 3.8b, respectively. We observe that the interpolation matrix has two rows that depend on the variable node, which are the numbered elements in Figure 3.8b, and four rows that depend on the pivot node, which are the elements depicted by circles in Figure 3.8a. Without loss of generality, we assume rows 1 and 2 of the interpolation matrix depend on the variable node. In this example, we stick to the definition in Equation (3.24), i.e., we take derivatives of the interpolation matrix first with respect to the y component of the variable node and then the x component. Therefore, $\mathcal{R}_M(2,2) = \{([1,1,\mathbf{0}_{KL-2}],[1,1,\mathbf{0}_{KL-2}]),([2,0,\mathbf{0}_{KL-2}],[0,2,\mathbf{0}_{KL-2}])\}$, where $\mathbf{0}_{KL-2}$ is the all-zero vector with dimension $KL-2$. Note that there is no other $(\boldsymbol{k},\boldsymbol{l})$ pair such that $\nabla^{x_j,y_j}_{\boldsymbol{k},\boldsymbol{l}}$ can be applied by using only regular simple shifts. When we apply $\nabla^{x_j,y_j}_{\boldsymbol{k},\boldsymbol{l}}$ with $(\boldsymbol{k},\boldsymbol{l}) = ([1,1,\mathbf{0}_{KL-2}],[1,1,\mathbf{0}_{KL-2}])$, we obtain a derivative set as depicted in Figure 3.8c, and obtain the one in Figure 3.8d with $(\boldsymbol{k},\boldsymbol{l}) = ([2,0,\mathbf{0}_{KL-2}],[0,2,\mathbf{0}_{KL-2}])$. Note that the derivative set in Figure 3.8c obeys the N-zig-zag order while the one in Figure 3.8d does not. Since there is only one $(\boldsymbol{k},\boldsymbol{l})$ pair resulting in an N-zig-zag ordered derivative set, $(\alpha_1,\alpha_2) = (2,2)$ is quasi-unique.

Next, we describe, in detail, the first two iterations of the recursive coalescence procedure, and then generalize the result to any iteration. Without loss of generality, we choose (x_n, y_n) as the pivot node for all the coalescences in the recursion, and coalesce it with the variable node z_i in the i^{th} coalescence where $i \in [1:n-1]$. Let us define the set of remaining nodes before applying the j^{th} coalescence as $Z_j \triangleq \{(x_i, y_i) \mid i \in [j:n]\}$. For the first coalescence, let $\mathbf{M}_1 = \mathbf{M}$, and suppose we find a quasi-unique shift for order (α_1^*, α_2^*). We denote by $\mathbf{M}_2 \triangleq C_{\boldsymbol{k}^*,\boldsymbol{l}^*}(\mathbf{M}_1)\nabla^{x_1,y_1}_{\boldsymbol{k}^*,\boldsymbol{l}^*}\mathbf{M}_1\big|_{x_1=x_n,y_1=y_n}$ the unique matrix such that $\mathcal{U}_{z_n,\mathbf{M}_2}$ satisfies the N-zig-zag order, and define the set of matrices containing the rest of the interpolation matrices as

$$\Phi_2 \triangleq \left\{ C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}_1)\nabla^{x_1,y_1}_{\boldsymbol{k},\boldsymbol{l}}\,\mathbf{M}_1\big|_{x_1=x_n,y_1=y_n} \big| (\boldsymbol{k},\boldsymbol{l}) \in \mathcal{R}_{\mathbf{M}_1}(\alpha_1^*,\alpha_2^*) \setminus (\boldsymbol{k}^*,\boldsymbol{l}^*) \right\}. \tag{3.27}$$

Then, from Equation (3.26), we can write

$$D_2(Z_2) = \frac{\partial^{\alpha_1^*+\alpha_2^*}}{\partial x_1^{\alpha_1^*}\partial y_1^{\alpha_2^*}} \det(\mathbf{M}_1)\Bigg|_{x_1=x_n,y_1=y_n} = \det(\mathbf{M}_2) + \sum_{\bar{\mathbf{M}}\in\Phi_2} \det\left(\bar{\mathbf{M}}\right). \tag{3.28}$$

For the second coalescence, taking (x_n, y_n) as the pivot node and (x_2, y_2) as the variable node, we write the Taylor series expansion of $D_2(Z_2)$ as

$$D_2(Z_2) = \sum_{(\alpha_1,\alpha_2)\in\mathbb{N}^2} \frac{1}{\alpha_1!\alpha_2!}(x_2-x_n)^{\alpha_1}(y_2-y_n)^{\alpha_2} D_{\alpha_1,\alpha_2}(Z_3) \tag{3.29}$$

where

$$D_{\alpha_1,\alpha_2}(Z_3) = \frac{\partial^{\alpha_1+\alpha_2}}{\partial x_2^{\alpha_1}\partial y_2^{\alpha_2}} \det(\mathbf{M}_2)\Bigg|_{x_2=x_n,y_2=y_n} + \sum_{\bar{\mathbf{M}}\in\Phi_2} \frac{\partial^{\alpha_1+\alpha_2}}{\partial x_2^{\alpha_1}\partial y_2^{\alpha_2}} \det\left(\bar{\mathbf{M}}\right)\Bigg|_{x_2=x_n,y_2=y_n}. \tag{3.30}$$

Next, we apply Equation (3.26) to Equation (3.30). This time, we find a quasi-unique shift (α_1^*, α_2^*) by only considering matrix $\mathbf{M}_2$. Note that, the (α_1^*, α_2^*) pair is different for each recursion but for a clearer notation, we omit the recursion index. Since the choice of (α_1^*, α_2^*) only considers $\mathbf{M}_2$, it does not imply the existence of quasi-unique shifts for all the other matrices in Φ_2. We denote by $\mathbf{M}_3 \triangleq C_{\boldsymbol{k}^*,\boldsymbol{l}^*}(\mathbf{M}_2)\nabla^{x_1,y_1}_{\boldsymbol{k}^*,\boldsymbol{l}^*}\mathbf{M}_2\big|_{x_2=x_n,y_2=y_n}$ the unique matrix satisfying that $\mathcal{U}_{z_n,\mathbf{M}_3}$ follows the N-zig-zag order, and define the set of matrices containing the rest of weighted interpolation matrices, originated from $\mathbf{M}_2$ or from $\bar{\mathbf{M}} \in \Phi_2$, as

$$\begin{aligned}\Phi_3 \triangleq &\left\{ C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}_2)\nabla^{x_2,y_2}_{\boldsymbol{k},\boldsymbol{l}}\,\mathbf{M}_2\big|_{x_2=x_n,y_2=y_n} \big| (\boldsymbol{k},\boldsymbol{l}) \in \mathcal{R}_{\mathbf{M}_2}(\alpha_1^*,\alpha_2^*) \setminus (\boldsymbol{k}^*,\boldsymbol{l}^*) \right\} \\ &\cup \left\{ C_{\boldsymbol{k},\boldsymbol{l}}(\bar{\mathbf{M}})\nabla^{x_2,y_2}_{\boldsymbol{k},\boldsymbol{l}}\bar{\mathbf{M}}\big|_{x_2=x_n,y_2=y_n} \big| (\boldsymbol{k},\boldsymbol{l}) \in \mathcal{R}_{\mathbf{M}_2}(\alpha_1^*,\alpha_2^*), \bar{\mathbf{M}} \in \Phi_2 \right\}.\end{aligned}$$

Then, we can write

$$D_3(Z_3) = D_{\alpha_1^*,\alpha_2^*}(Z_3) = \det(\mathbf{M}_3) + \sum_{\bar{\mathbf{M}}\in\Phi_3} \det\left(\bar{\mathbf{M}}\right). \tag{3.31}$$

We follow the same procedure until all nodes are coalesced with the pivot node and we reach $D_n(Z_n)$. In general, the expressions in this procedure are defined recursively as follows.

$$\mathbf{M}_{i+1} \triangleq C_{\boldsymbol{k}^*,\boldsymbol{l}^*}(\mathbf{M}_i)\nabla^{x_i,y_i}_{\boldsymbol{k}^*,\boldsymbol{l}^*}\mathbf{M}_i\big|_{x_i=x_n,y_i=y_n}, i \in [1:n-1] \tag{3.32}$$

$$D_i(Z_i) \triangleq D_{\alpha_1^*,\alpha_2^*}(Z_i) = \det(\mathbf{M}_i) + \sum_{\bar{\mathbf{M}} \in \Phi_i} \det(\bar{\mathbf{M}}). \tag{3.33}$$

$$D_i(Z_i) = \sum_{(\alpha_1,\alpha_2)\in\mathbb{N}^2} \frac{1}{\alpha_1!\alpha_2!}(x_i - x_n)^{\alpha_1}(y_i - y_n)^{\alpha_2} D_{\alpha_1,\alpha_2}(Z_{i+1}). \tag{3.34}$$

$$\begin{aligned}\Phi_{i+1} \triangleq & \left\{ C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}_i)\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\mathbf{M}_i\big|_{x_i=x_n,y_i=y_n} \,|(\boldsymbol{k},\boldsymbol{l}) \in \mathcal{R}_{\mathbf{M}_i}(\alpha_1^*,\alpha_2^*) \setminus (\boldsymbol{k}^*,\boldsymbol{l}^*)\right\} \\ & \cup \left\{ C_{\boldsymbol{k},\boldsymbol{l}}(\bar{\mathbf{M}})\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\bar{\mathbf{M}}\big|_{x_i=x_n,y_i=y_n} \,|(\boldsymbol{k},\boldsymbol{l}) \in \mathcal{R}_{\mathbf{M}_i}(\alpha_1^*,\alpha_2^*), \bar{\mathbf{M}} \in \Phi_i \right\}.\end{aligned} \tag{3.35}$$

Lemma 3.2. *Consider the recursive Taylor series expansion procedure on a fixed pivot, (x_n, y_n). If, for every step $i \in [1:n]$, we can find a quasi-unique shift (α_1^*, α_2^*) for $\mathbf{M}_i$ in Equation* (3.32) *, then*

$$D_n(Z_n) = \det(\mathbf{M}_n), \tag{3.36}$$

where $\mathbf{M}_n$ depends only on $Z_n = \{(x_n, y_n)\}$. Therefore, the associated interpolation matrix $\mathbf{M}_n$, and hence, $\mathbf{M}_1$ are invertible for almost all choices of evaluation points.

The proof of Lemma 3.2 is given in Section 3.9.

In the following lemma, we present a set of situations for which a quasi-unique shift exits in a coalescence step between a pivot node and a variable node. These are not the only situations for which quasi-unique shifts exit but are sufficient to derive the recovery threshold presented in Theorem 3.1, as shown in the next subsection.

Lemma 3.3. *Assume that in the i^{th} coalescence step we have the variable node $z_i = (x_i, y_i)$ and the pivot node $z_n = (x_n, y_n)$. Define $r_f \triangleq |\mathcal{U}_{z_i,\mathbf{M}_i}| \mod \mu_B$ and $l_e \triangleq \mu_B - |\mathcal{U}_{z_n,\mathbf{M}_i}| \mod \mu_B$. That is, when depicted in the derivative order space, r_f is the number of elements in the rightmost partially-occupied column of the derivative set of the variable node, and l_e is the number of empty places in the rightmost partially-occupied column of the derivative set of the pivot node. Then, if $|\mathcal{U}_{z_i,\mathbf{M}_i}| + |\mathcal{U}_{z_n,\mathbf{M}_i}| \leq \mu_B K$ or $|\mathcal{U}_{z_i,\mathbf{M}_i}| + |\mathcal{U}_{z_n,\mathbf{M}_i}| > \mu_B K$ and one of the following conditions is satisfied:*

1. $r_f = 0$,
2. $r_f = l_e$,
3. $l_e = 0$,

then there exists a quasi-unique shift for the coalescence of these nodes.

The proof of Lemma 3.3 is given in Section 3.10.

3.7.3 Derivation of the Recovery Threshold Expression

The existence of a quasi-unique shift depends on the joint structure of the derivative sets of the pivot and the variable nodes. If the derivative sets of the pivot node and the variable node satisfy the conditions in Lemma 3.3, then, in a coalescence step, i.e., recursive Taylor series expansion, it is possible to find a quasi-unique shift for this recursive step and we can proceed to the next recursion. Otherwise, by simply ignoring specific computations provided by the worker whose evaluation point corresponds to the variable node under consideration, we can have the structure of the remaining computations satisfy the conditions in Lemma 3.3. This adds an overhead of ignored computations to the recovery threshold expression. In the following lemma, we provide an upper bound on the total number of computations we may need to ignore throughout the whole recursion process by analysing the worst-case scenario.

Lemma 3.4. *Assume that the conditions of Lemma 3.3 hold in none of the coalescences in the recursive Taylor series expansion process. Then, in the worst case, by ignoring at most* $(\mu_B - 2)(\frac{L}{\mu_B} - 1)$ *computations throughout all the recursion steps suffices to guarantee decodability for almost all choices of evaluation points.*

Proof. Assume that none of the conditions of Lemma 3.3 hold. If $r_f > l_e$, we can satisfy condition 2, i.e., $r_f = l_e$, in Lemma 3.3 by ignoring $r_f - l_e$ computations received from the worker whose evaluation point is the variable node. Thus, in the worst case, we ignore $\max(r_f - l_e) = (\mu_B - 1) - 1 = \mu_B - 2$ computations since the minimum value of l_e not satisfying the conditions of Lemma 3.3 is 1 and the maximum value of r_f not satisfying the conditions of Lemma 3.3 is $\mu_B - 1$. On the other hand, if $r_f < l_e$, we can ignore r_f computations and satisfy the condition 1 in Lemma 3.3. Since $r_f < l_e < \mu_B$, in the worst case, we need to ignore $\max(r_f) = \mu_B - 2$ computations. Thus, in either case, the maximum number of computations we ignore is $\mu_B - 2$. Observe that generating a new block in the derivative order space as a result of a coalescence and not satisfying any of the conditions in Lemma 3.3 are possible only if when $|\mathcal{U}_{z_i,\mathbf{M}_i}| + |\mathcal{U}_{z_n,\mathbf{M}_i}| > \mu_B K$. Given that there are L/μ_B blocks in the whole derivative order space, the maximum number of coalescences for which $|\mathcal{U}_{z_i,\mathbf{M}_i}| + |\mathcal{U}_{z_n,\mathbf{M}_i}| > \mu_B K$ is at most $(L/\mu_B - 1)$. Thus, in the worst case, the total number of ignored computations is $(\mu_B - 2)(\frac{L}{\mu_B} - 1)$. □

Since the polynomial we need to interpolate, $\mathbf{A}(x)\mathbf{B}(y)$, has KL coefficients, in the worst case, the recovery threshold becomes $KL + (\mu_B - 2)(\frac{L}{\mu_B} - 1)$. This number guarantees the existence of a quasi-unique shift in every recursive Taylor series expansion, as shown in Lemma 3.2. Therefore, we can conclude that our original interpolation matrix is invertible for almost all choices of the evaluation points. This completes the proof of Theorem 3.1.

3.8 Proof of Lemma 3.1

Given an interpolation matrix $\mathbf{M}$, we first prove

$$\frac{\partial}{\partial x_j}\det(\mathbf{M}) = \sum_{i=1}^{KL}\det(\partial_{i,x_j}\mathbf{M}), \tag{3.37}$$

and

$$\frac{\partial}{\partial y_j}\det(\mathbf{M}) = \sum_{i=1}^{KL}\det(\partial_{i,y_j}\mathbf{M}). \tag{3.38}$$

They follow directly from the chain rule. Let $m_{i,j}$'s denote the elements of $\mathbf{M}$, and S_{KL} the set of all permutations of the columns of $\mathbf{M}$. We use the fact $\det(\mathbf{M}) = \sum_{\pi\in S_{KL}}\operatorname{sgn}(\pi)\prod_{i=1}^{KL} m_{i,\pi(i)}$ [73, Definition 7.4], where π is a permutation, and $\operatorname{sgn}(\pi)$ is its parity. Then,

$$\begin{aligned}\frac{\partial}{\partial x_j}\det(\mathbf{M}) &= \sum_{\pi\in S_{KL}}\operatorname{sgn}(\pi)\frac{\partial}{\partial x_j}\prod_{i=1}^{KL} m_{i,\pi(i)}\\ &= \sum_{\pi\in S_{KL}}\operatorname{sgn}(\pi)\sum_{i=1}^{KL}\left(\frac{\partial}{\partial x_j}m_{i,\pi(i)}\right)\prod_{j\in[1:KL]\backslash\{i\}} m_{j,\pi(j)}\\ &= \sum_{i=1}^{KL}\sum_{\pi\in S_n}\operatorname{sgn}(\pi)\left(\frac{\partial}{\partial x_j}m_{i,\pi(i)}\right)\prod_{j\in[1:KL]\backslash\{i\}} m_{j,\pi(j)}\\ &= \sum_{i=1}^{KL}\det(\partial_{i,x_j}\mathbf{M}).\end{aligned} \tag{3.39}$$

The proof of Equation (3.38) can be done similarly. Next, consider part of a derivative path $s \triangleq \partial_{i_l,y_j}\cdots\partial_{i_2,y_j}\partial_{i_1,y_j}$ of length $l < \alpha_2$ such that it has two identical rows or at least one zero row, resulting in $\det(s\mathbf{M}) = 0$. Now let us consider the other sequences having s as the suffix, i.e., the sequence s is applied before them. Applying Equation (3.38) $m \leq \alpha_2 - l$ times, we obtain

$$\frac{\partial}{\partial y_j^m}\det(s\mathbf{M}) = \sum_{i_{l+m}=1}^{KL}\cdots\sum_{i_{l+1}=1}^{KL}\det(\partial_{i_{l+m},y_j}\cdots\partial_{i_{l+1},y_j}s\mathbf{M}). \tag{3.40}$$

However, $\frac{\partial}{\partial y_j^m}\det(s\mathbf{M}) = 0$ since $\det(s\mathbf{M}) = 0$. The same applies to x directional derivatives as well. This implies that while taking the derivatives of $\det(\mathbf{M})$, i.e., applying $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}$, if we encounter a sub-sequence s resulting in $\det(s\mathbf{M}) = 0$, then the sum of determinants of all matrices having $s\mathbf{M}$ as the suffix adds up to zero. Thus, only the sequences in which all simple shifts are regular contribute to Equation (3.26), while applying $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}$. Given a $(\boldsymbol{k},\boldsymbol{l})$ pair, if $C_{\boldsymbol{k},\boldsymbol{l}}$ denotes the number of sequences composed of only regular simple shifts, we obtain Equation (3.26).

3.9 Proof of Lemma 3.2

We start by stating the following lemma that will be useful in the rest of the proof.

Lemma 3.5. *No derivative set corresponding to the evaluation points z_i in $\Phi_{i+1}, \forall i \in [1 : N-1]$, defined in Equation* (3.35) *obeys N-zig-zag order.*

Proof. From the definition of quasi-unique shift, it is clear that no elements of the first set in Equation (3.35) obey the N-zig-zag order. To show the same for the second set, consider the elements of the variable node at the coalescence step i. (α_1^*, α_2^*) is chosen such that there is only one $(\boldsymbol{k}^*, \boldsymbol{l}^*)$ such that when the elements of the variable node are shifted according to $(\boldsymbol{k}^*, \boldsymbol{l}^*)$, and evaluated at (x_n, y_n), the resulting derivative set obeys the N-zig-zag order. Assume now that no element in Φ_i obeys the N-zig-zag order. Take any $\bar{\mathbf{M}} \in \Phi_i$ and apply $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\bar{\mathbf{M}}$ for some $(\boldsymbol{k},\boldsymbol{l})$. If $(\boldsymbol{k},\boldsymbol{l}) \neq (\boldsymbol{k}^*, \boldsymbol{l}^*)$, then at least one of the elements of the variable node will be placed to a location whose priority score is larger than those of all the locations to which the elements of the variable node would be placed if $(\boldsymbol{k}^*, \boldsymbol{l}^*)$ were applied. This is because $\exists j$ such that $\boldsymbol{k}(j) > \boldsymbol{k}^*(j)$ or $\exists j$ such that $\boldsymbol{l}(j) > \boldsymbol{l}^*(j)$. In $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\bar{\mathbf{M}}|_{x_i=x_n,y_i=y_n}$, if the derivative set of (x_n, y_n) obeyed the N-zig-zag order, the variable node would have to have more elements than it originally had since the largest priority score whose corresponding location is occupied is larger for $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\bar{\mathbf{M}}|_{x_i=x_n,y_i=y_n}$ than $\nabla_{\boldsymbol{k}^*,\boldsymbol{l}^*}^{x_i,y_i}\mathbf{M}_i|_{x_i=x_n,y_i=y_n}$. Therefore, it is not possible that the derivative set of the pivot node for $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_i,y_i}\bar{\mathbf{M}}|_{x_i=x_n,y_i=y_n}$ obeys the N-zig-zag order when $(\boldsymbol{k},\boldsymbol{l}) \neq (\boldsymbol{k}^*, \boldsymbol{l}^*)$. On the other hand, if $(\boldsymbol{k},\boldsymbol{l}) = (\boldsymbol{k}^*, \boldsymbol{l}^*)$, since we assume that no element of Φ_i obeys the N-zig-zag order, the derivative set of the pivot node for $\nabla_{\boldsymbol{k}^*,\boldsymbol{l}^*}^{x_i,y_i}\bar{\mathbf{M}}|_{x_i=x_n,y_i=y_n}$ does not obey the N-zig-zag order. Since we know that no element in Φ_2 obeys the N-zig-zag order by definition, by induction, we conclude that none of the elements in $\Phi_{i+1}, \forall i \in [2 : N]$ obeys the N-zig-zag order. □

The rows of $\mathbf{M}_n$ only depend on the pivot node (x_n, y_n), and thus, the derivative set associated with (x_n, y_n) has KL elements and satisfies the N-zig-zag order. Similarly, for all matrices in Φ_n, the derivative sets of the pivot node have KL elements. However, as Lemma 3.5 suggests, in this case, no elements of Φ_n satisfy the N-zig-zag order. This implies that all matrices in Φ_n have at least one duplicate row, or a zero row. Therefore, $\sum_{\bar{\mathbf{M}}\in\Phi_n} \det(\bar{\mathbf{M}})|_{x_{n-1}=x_n,y_{n-1}=y_n} = 0$. This proves Equation (3.36). The proof of $\det(\mathbf{M}_n) \neq 0$ follows, directly, from the fact that, for $\mathbf{M}_n$, the derivative set of the pivot node obeys the N-zig-zag order. This means that each row of $\mathbf{M}_n$ uniquely corresponds to one of the computations $\partial_k\mathbf{A}(x_n)\partial_l\mathbf{B}(y_n)$, $\forall k \in [0 : K-1]$, $\forall l \in [0 : L-1]$. Therefore, $\mathbf{M}_n$ can be written as an upper triangular matrix, and therefore, invertible, implying $D_n(Z_n) \neq 0$. Remember that $D_{i+1}(Z_{i+1}) \neq 0$ implies $D_i(Z_i) \neq 0$ for all $i \in [0 : n-1]$ due to the linear independence between $(x_i - x_n)^{\alpha_1}(y_i - y_n)^{\alpha_2}$ for different (α_1, α_2) pairs. Thus, $D_n(Z_n) \neq 0$ implies $D_1(Z_1) \neq 0$ recursively, and thus, $\mathbf{M}_1$ is invertible. This proves the claim of the lemma.

3.10 Proof of Lemma 3.3

First, note that the existence of a quasi-unique shift is only related to the structure of the uppermost blocks of the pivot and variable nodes' derivative sets. Therefore, even if the derivative sets of the pivot and variable nodes occupy more than one block, in the derivative order space, it is sufficient to consider only the uppermost blocks since the fully occupied blocks can be handled only by additional y-directional derivatives. Thus, we proceed as if there exist only the uppermost blocks of the derivative sets of the pivot and variable nodes.

Our proof is based on determining some sufficient conditions for the existence of a quasi-unique shift, which will reduce to the conditions claimed in the lemma. We first state our problem visually in the derivative order space in terms of the derivative sets and the derivatives of the evaluations, then we find the sufficient conditions on this visual problem statement.

In this first part of the proof, we take all the y-directional derivatives before the x-directional ones. We depict the elements in the derivative set of the pivot node, $z_n = (x_n, y_n)$, in the derivative order space by the filled circles in Figure 3.9a. The unfilled circles, on the other hand, in Figure 3.9a represent the locations of the elements of the variable node, denoted by $z_i = (x_i, y_i)$, to be placed after the coalescence with the pivot node. If the sum of the number of elements in the derivative sets of the pivot and variable nodes is larger than the size of one block, i.e., $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| > \mu_B K$, the coalescence generates a new block. Since we are interested in finding quasi-unique shifts, the locations of the variable node's elements after coalescence are determined such that the resulting derivative set obeys the N-zig-zag order. Therefore, from the structure in the figure, we write $|\mathcal{U}_{z_i,\mathbf{M}_i}| = l_e + (c_{e,b} + c_{e,u})\mu_B + r_e$.

After determining the locations to which the elements of the variable node are placed, we no longer need the elements of the pivot node. Therefore, in Figure 3.9b, we remove the elements of the pivot node from the picture, and, instead, we depict the elements of

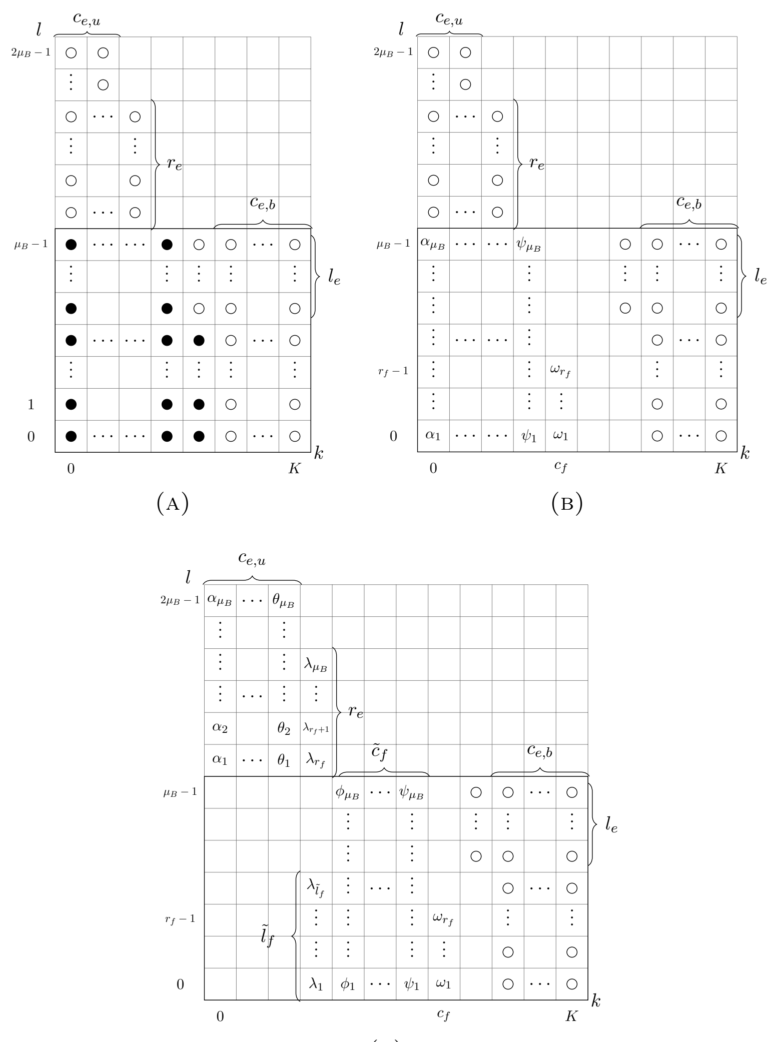


Figure 3.9: Visualization of the derivative sets of the pivot and variable nodes.

the variable node in their original places such that they obey N-zig-zag order. Note that in this proof, our goal is to find a quasi-unique shift (α_1^*, α_2^*) such that there is only one unique placement, characterized by $(\boldsymbol{k}^*, \boldsymbol{l}^*)$, of the elements of the variable node along with the elements of the pivot node. Therefore, we need to track the final location of each element of the variable node and make sure that to that location, it is not possible to place another element from the variable node. To distinguish the elements of the variable node, we denote each of them by Greek letters and their subscripts. Note that the letters used for this purpose should not be mixed with the other uses of the Greek letters throughout the chapter.

Given the depictions in Figure 3.9b, the next step is to determine y-directional shifts such that all elements of the variable node are placed to the correct row in the derivative order space. Since, according to Lemma 3.1, only regular simple shifts are considered, while taking y-directional derivatives, the sequence of the elements having the same x-directional derivative order cannot change. Therefore, for example, α_{μ_B} stays always above the elements denoted by $\alpha_i, i \in [1 : \mu_B - 1]$. Thanks to this property, shifts of the variable node elements via y-directional derivatives that will fill the locations denoted by unfilled circles in the new block are straightforward to determine. That is because shifting the block composed of the variable node's elements with the same shape as the locations to be filled in the new block uniquely determines the elements to be moved to the new block. After such shifts in the y-direction, the remaining elements of the variable node will fill the unfilled circles in the lower block. In Figure 3.9c, we both depict the shifted elements to the new block and the remaining elements together.

Whenever we fill a row composed of unfilled circles in the lower block, to have y-directional shifts which generate quasi-unique shifts, the elements to be placed there must be uniquely determined. For example, while filling the top l_e rows, for each row, there must be exactly $c_{e,b}+1$ columns among the elements of the variable node that are available to provide their top-most element. If there are more than that many candidate columns, then the elements to be placed in the top l_e rows are not unique. In the same way, after filling the top l_e rows, in the remaining rows, there must be exactly $c_{e,b}$ columns of the elements of the variable node that can provide their top-most element. Therefore, to guarantee this, a sufficient condition is to have $\tilde{c}_f = c_{e,b}$ and the remaining elements of the variable node have only one partially-occupied column with l_e elements. There might be several structures satisfying this condition. One of them is when $r_f = 0$, implying $\tilde{l}_f = l_e$. This is condition 1 in Lemma 3.3. Another way to satisfy the sufficient condition is to have $r_f = l_e$, implying $\tilde{l}_f = 0$, and this is condition 2 in Lemma 3.3.

After the elements are aligned with their final rows via y-directional derivatives, necessary x-directional shifts can be easily applied such that the elements of the variable node are finally placed in their intended locations. Again, due to Lemma 3.1, we consider only regular simple shifts and therefore, while taking x-directional derivatives, the sequence of the elements having the same y-directional derivative order cannot change.

To show other instances in which a quasi-unique shift exists, i.e., either $l_e = 0$ when $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| > \mu_B K$, or $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| \leq \mu_B K$, in the rest of the proof, we take all x-directional derivatives before y-directional derivatives. However, since we are taking x-directional derivatives first, we first align all the elements of the variable node that are to stay in the lower block with their intended columns. We start filling with the rightmost column of the lower block, which is column K. When $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| \leq \mu_B K$, there will not be any element at the upper block and column K will not be fully occupied, and instead it will contain only $\tilde{l}_e$ elements after the coalescence for $\tilde{l}_e < \mu_B$. Therefore, the rows of the elements of the variable node that will provide elements to these locations are uniquely determined, namely the rows $[0 : \tilde{l}_e - 1]$ of the elements of the variable node. On the other hand, if $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| > \mu_B K$, then $\tilde{l}_e = \mu_B$, and still the elements of the variable node to be placed in those locations are from the rows $[0 : \mu_B - 1]$. Hence, in both cases, the elements to be placed in the rightmost column of the lower block are uniquely determined. After the rightmost elements from the rows $[0 : \tilde{l}_e - 1]$ of the elements of the variable node are shifted to column K column via x-directional shifts, next, we fill the columns starting from column $K-1$ to column $K - c_{e,b} - 1$. Note that since each of these columns is intended to be fully occupied, they are directly filled with the remaining rightmost elements of each row via x-directional shifts. Finally, we fill column $K - c_{e,b}$, which has l_e locations intended to be occupied after the coalescence. If $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| \leq \mu_B K$, then the upper block is not generated and the number of remaining elements of the variable node is equal to l_e, each on different rows. Thanks to the property that the sequence of the elements having the same y-directional derivative orders cannot change by x-directional shifts, the elements to be placed to the l_e empty locations are uniquely determined. This proves that when $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| \leq \mu_B K$, a quasi-unique shift exists. On the other hand, when $|\mathcal{U}_{z_n,\mathbf{M}_i}| + |\mathcal{U}_{z_i,\mathbf{M}_i}| > \mu_B K$, a new block is generated, so there will be always more than l_e remaining elements of the variable node. In this case, the elements to be placed in the column $K - c_{e,b}$ will not be uniquely determined. Therefore, to have a unique shift, in this case, we need $l_e = 0$, which is condition 3 of Lemma 3.3.

3.11 Alternative Formulation of Almost Regular Interpolation Schemes

In this section, we discuss an alternative formulation of almost regular interpolation schemes based on the interpolation of $\mathbf{A}(x)\mathbf{B}(y)$ from its evaluations only, as done in B-PROC. In such an approach, for worker i, $\mathbf{A}(x)$ would be evaluated at the distinct evaluation points $\{x_{i,k} : k \in [0 : m_{A,i} - 1]\}$ and $\mathbf{B}(y)$ would be evaluated at the distinct evaluation points $\{y_{i,l} : l \in [0 : m_{B,i} - 1]\}$. In this case, computations assigned to worker i would be $\mathbf{A}(x_{i,k})\mathbf{B}(y_{i,l})$, where $k \in [0 : m_{A,i} - 1]$ and $l \in [0 : m_{B,i} - 1]$.

Remember that in all almost regular interpolation schemes, we have a priority score which determines the order in which the computations will be carried out by each worker. Each priority score is a function of the computation index, which is (k, l). For our alternative formulation, we can use the same priority scores defined for Hermite interpolation-based schemes such that (k, l) is the index for the computation $\mathbf{A}(x_{i,k})\mathbf{B}(y_{i,l})$. Then vertical, horizontal, N-zig-zag and Z-zig-zag order definitions follow. Thus, the two formulations are equivalent to each other, under the almost regularity condition, and Theorem 3.1 and Corollary 3.1 are also valid for this case.

Before proving this claim, we first present two useful lemmas. Note that we only provide proof for the N-zig-zag order. It can trivially extend to the Z-zig-zag order case.

Lemma 3.6. *From any worker i, assume all the responses obey N-zig-zag order. Consider all the nodes, i.e., evaluation points of the received responses at the master, of the form $(x_{i,k}, y_{i,1})$ and $(x_{i,k}, y_{i,2})$ for $k \in [1:p]$ where $1 \leq p \leq m_{A,i}$ depending on the number of responses received from worker i. Without losing generality, assume the first p rows of the interpolation matrix $\mathbf{M}$ depends on the evaluation points $(x_{i,k}, y_{i,2})$, $k \in [1:p]$. Let $\mathcal{U}_{(x_{i,k},y_{i,1}),\mathbf{M}} = \{(0,0),(0,1),\cdots,(0,l-1)\}$ for any l satisfying $1 \leq l \leq L-1$, and $\mathcal{U}_{(x_{i,k},y_{i,2}),\mathbf{M}} = \{(0,0)\}$. Consider $y_{i,1}$ as the pivot and $y_{i,2}$ as the variable. Then $(\alpha_1, \alpha_2) = (0, lp)$ is a quasi-unique shift and $|\mathcal{R}_{\mathbf{M}}(\alpha_1, \alpha_2)| = 1$. Hence,*

$$\frac{\partial^{lp}}{\partial y_{i,2}^{lp}} \det(\mathbf{M})\big|_{y_{i,2}=y_{i,1}} = C_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}(\mathbf{M}) \det(\nabla^{y}_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}\mathbf{M})\big|_{y_{i,2}=y_{i,1}} \tag{3.41}$$

by Definition 3.8, where $\tilde{\boldsymbol{k}} = \mathbf{0}$, $\tilde{\boldsymbol{l}}(k) = l, \forall k \in [1:p]$ and $\tilde{\boldsymbol{l}}(k) = 0, \forall k \in [p+1:KL]$. After such a coalescence, we obtain $\mathcal{U}_{(x_{i,k},y_{i,1}),\tilde{\mathbf{M}}} = \{(0,0),(0,1),\cdots,(0,l-1),(0,l)\}$, where $\forall k \in [1:p]$ and $\tilde{\mathbf{M}} = \nabla^{y}_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}\mathbf{M}\big|_{y_{i,2}=y_{i,1}}$.

Proof. For any $k \in [1:p]$, consider two nodes $(x_{i,k}, y_{i,1})$ and $(x_{i,k}, y_{i,2})$. While taking the derivative of $\det(\mathbf{M})$ with respect to $y_{i,2}$, the minimum derivative order to be applied to the row corresponding to the node $(x_{i,k}, y_{i,2})$ row is l since $\mathcal{U}_{(x_{i,k},y_{i,1}),\mathbf{M}} = \{(0,0),(0,1),\cdots,(0,l-1)\}$ has all the orders up to l. Otherwise, we would get two identical rows in $\nabla^{y}_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}\mathbf{M}$. Since we have p rows depending on $y_{i,2}$, and $\alpha_2 = lp$, we must have $\tilde{\boldsymbol{l}}(i) = l, \forall i \in [1:p]$ and $\tilde{\boldsymbol{l}}(k) = 0, \forall k \in [p+1:KL]$. This is the only possible $\tilde{\boldsymbol{l}}$ and proves the claim. □

Lemma 3.7. *From any worker i, assume all the responses obey the N-zig-zag order. Without losing generality, consider the nodes $(x_{i,1}, y_{i,1})$ and $z = (x_{i,2}, y_{i,1})$. Let $\mathcal{U}_{(x_{i,1},y_{i,1}),\mathbf{M}} = \{(i,j) : i \in [0:k-1], j \in [0:m-1]\}$, that is, it contains k columns with exactly $m \leq L$ elements, and $\mathcal{U}_{(x_{i,2},y_{i,1}),\mathbf{M}} = \{(0,0),(0,1),\cdots,(0,l-1)\}$, for any $l \in [0:m-1]$, i.e., one column with l elements. Consider $x_{i,2}$ as a variable and $x_{i,1}$ as the pivot. Then, $(\alpha_1, \alpha_2) = (lk, 0)$ is a quasi-unique shift with $|\mathcal{R}_{\mathbf{M}}(\alpha_1, \alpha_2)| = 1$. That is,*

$$\frac{\partial^{lk}}{\partial x_{i,2}^{lk}} \det(\mathbf{M})\big|_{x_{i,2}=x_{i,1}} = C_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}(\mathbf{M}) \det(\nabla^{x}_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}M)\big|_{x_{i,2}=x_{i,1}} \tag{3.42}$$

by Definition 3.8, where $\tilde{\boldsymbol{k}}(j) = k$, $\forall j \in [1:l]$, $\tilde{\boldsymbol{k}}(j) = 0$, $\forall j \in [l+1:KL]$ *and* $\tilde{\boldsymbol{l}} = \mathbf{0}$. *After such a coalescence, we obtain* $\mathcal{U}_{(x_{i,1},y_{i,1}),\tilde{\mathbf{M}}} = \{(i,j) : i \in [0:k-1], j \in [0:m-1]\} \cup \{(k,0),(k,1),\cdots,(k,l-1)\}$ *where* $\tilde{\mathbf{M}} = \nabla^{x}_{\tilde{\boldsymbol{k}},\tilde{\boldsymbol{l}}}\mathbf{M}\big|_{x_{i,2}=x_{i,1}}$.

Proof. While taking the kl-th order derivative of $\det(\mathbf{M})$ with respect to x, we need to allocate these kl shifts into those rows of $\mathbf{M}$ that depend on $x_{i,2}$, each of which corresponds to one element in the $\mathcal{U}_{(x_{i,2},y_{i,1}),\mathbf{M}}$. Recall that $\mathcal{U}_{(x_{i,1},y_{i,1}),\mathbf{M}} = \{(i,j) : i \in [0:k-1], j \in [0:m-1]\}$. After the shift with the order $(\alpha_1,\alpha_2) = (kl,0)$, there should not be any duplicate element in $\mathcal{U}_{(x_{i,1},y_{i,1}),\mathbf{M}}$. Thus, to each element of $\mathcal{U}_{(x_{i,2},y_{i,1}),\mathbf{M}}$, we will assign a derivative order of k, i.e., $\tilde{\boldsymbol{k}}(j) = k$, $\forall j \in [1:l]$ and $\tilde{\boldsymbol{k}}(j) = 0$, $\forall j \in [l+1:KL]$. All other allocations would generate duplicate elements in $\mathcal{U}_{(x_{i,1},y_{i,1}),\mathbf{M}}$ after the coalescence. □

Next, we state the equivalency between the alternative formulation we give in this section and the formulation in Section 3.7.

Lemma 3.8. *Assuming the alternative formula we have introduced is employed, if the set of evaluation points, or nodes, assigned to a worker has N-zig-zag order, then by a series of unique shifts, it can be reduced to a single node whose derivative set has also N-zig-zag order.*

Proof. Assume the master receive $\tilde{k}_i\tilde{l}_i$ evaluations of $\mathbf{A}(x)\mathbf{B}(y)$ from worker i. That is, the master receives the evaluations $\{\mathbf{A}(x_{i,k})\mathbf{B}(y_{i,l}) : k \in [1:\tilde{k}_i], l \in [1:\tilde{l}_i]\}$ where $\tilde{k}_i \leq m_{A,i}$ and $\tilde{l}_i \leq m_{B,i}$ such that they are in accordance with the constraints imposed by N-zig-zag order. Thus, we have derivative sets $\tilde{U}_{(x_{i,k},y_{i,l}),\mathbf{M}} = \{(0,0)\}$, $\forall k \in [1:\tilde{k}_i], \forall l \in [1:\tilde{l}_i]$, $\forall i \in [1:N]$ assuming each worker sent at least one response. Then, if we apply the coalescence procedure described in Section 3.7, and if we take y-directional derivatives first, according to Lemma 3.6, we can always find unique shifts in the coalescence procedure. Assume, without losing generality, during the y-directional derivatives, $y_{i,1}$ is taken as the pivot. After taking all y- directional derivatives, the derivative sets become $\mathcal{U}_{(x_{i,k},y_{i,1}),\mathbf{M}_2} = \{(0,l) : l \in [1:\tilde{l}_i]\}$, $\forall i \in [1:N]$, $\forall k \in [1:\tilde{k}_i]$. Then, according to Lemma 3.7, by only taking x-directional derivatives, we can find unique shifts in every coalescence step. Without losing generality, assuming $x_{i,1}$ is taken as pivot for the x-directional derivatives, we end up with $\mathcal{U}_{(x_{i,1},y_{i,1}),\mathbf{M}_3} = \{(k,l) : k \in [\tilde{k}_i], l \in [\tilde{l}_i]\}$, $\forall i \in [1:N]$, as claimed by the lemma. □

Observe that these derivative sets are equivalent to the derivative sets obtained as a result of Hermite interpolation-based bivariate polynomial codes. This proves the equivalency under the almost regularity condition.

3.12 Conclusion

In this chapter, we have proposed storage-efficient straggler exploitation techniques for distributed matrix multiplication with heterogeneous computation and storage capacities. Our proposed bivariate polynomial coding schemes efficiently utilize the workers' storage capacities. However, bivariate polynomial coding poses the problem of invertibility of an interpolation matrix, which is highly non-trivial, unlike univariate polynomial codes. To tackle this issue, we first proposed B-PROC based on the fact that the interpolation matrix of bivariate interpolation is always invertible if the evaluation points form a rectangular grid. However, in this scheme, some computations the master receives may not be useful since the information they provide is already obtained from previous responses. In order to avoid such redundant computations, we showed that as long as the workers follow a specific computation order, the interpolation matrix is invertible for almost every choice of the interpolation points. Based on this, we proposed BPC-VO and BPC-HO, which fully avoid the problem of redundant computations. However, although still much better than univariate schemes, the constraints imposed by the computation orders in BPC-VO and BPC-HO limit the efficiency of storage utilization and hence, may increase the average computation time when the storage capacities of the workers are limited. To overcome this, in BPC-NZO and BPC-ZZO, we relax these constraints by allowing a few redundant computations, which are still much less than those of B-PROC. We have shown that these proposed schemes' ability to exploit the workers' storage capacities is close to optimal. For different storage capacities, we numerically demonstrated that in terms of the average computation time, the proposed schemes in the chapter outperform existing schemes in the literature.

The almost regularity of bivariate polynomial coding schemes is itself a theoretically interesting result, which may guide proofs of other multivariate interpolation schemes for distributed matrix multiplication in more general situations. Additionally, we have identified the potential application of bivariate polynomial coding to secure matrix multiplication, which we have investigated in the next chapter.

Overall, our proposed techniques demonstrate close-to-optimal exploitation of workers' storage capacities and outperform existing schemes in the literature in terms of average computation time. These findings contribute significantly to the field of distributed matrix multiplication and have the potential to trigger future research in this area.

Chapter 4

Bivariate Polynomial Codes for Security

4.1 Abstract

In this chapter, we consider the problem of SDMM. Coded computation has been shown to be an effective solution in distributed matrix multiplication, both providing privacy against workers and boosting the computation speed by efficiently mitigating stragglers. We present a non-direct secure extension of the bivariate polynomial codes. In the previous chapter, we have shown that bivariate polynomial codes are able to further speed up distributed matrix multiplication by exploiting the partial work done by the stragglers rather than completely ignoring them while reducing the upload communication cost and/or the workers' storage's capacity needs. In this chapter, especially for upload communication or storage-constrained settings, the proposed approach reduces the average computation time of SDMM compared to its competitors in the literature.

4.2 Introduction

In this chapter, we aim to tackle two major challenges in distributed matrix multiplication, namely straggler mitigation and security.

Regarding the straggler mitigation problem, we have previously discussed various approaches based on univariate polynomial coding, including UPC, MatDot codes, and PolyDot codes in the Chapter 2 section. However, we have demonstrated that these approaches do not fully utilize the work done by straggling workers, resulting in suboptimal performance. While multi-message approaches such as those proposed in [56–58] partially address the problem by assigning multiple sub-products to each worker and communicating the result of each sub-product to the master as soon as it is completed, they suffer from increased upload communication costs and inefficient use of workers' storage.

To address these issues, we have introduced bivariate polynomial codes in the previous chapter, achieving a better trade-off between upload cost, storage efficiency, and average computation time. In this chapter, we aim to extend this solution to SDMM.

Hence, the second challenge we tackle in this chapter is ensuring the security of the multiplied matrices, which is of utmost importance in many practical applications.

In our earlier discussion in Chapter 2, we have reviewed two prominent approaches for SDMM: S-UPC [11] and GASP codes [13]. These approaches employ random matrix partitions during the encoding phase to mask the original matrix partitions, with the aim of keeping them hidden from the workers. While GASP codes are known to achieve a smaller recovery threshold than S-UPC, neither approach considers the joint problem of straggler mitigation and security. Moreover, they suffer from a lack of straggler exploitation and other issues associated with univariate polynomial codes. As a result, more efficient solutions are required to address these challenges.

The previous approach to address these challenges in [74] used rateless codes and a multi-message approach. In this work, computations are assigned adaptively in rounds, and in each round, workers are classified into clusters depending on their computation speeds. However, computations were not one-to-any replaceable, and results from a worker in a cluster were useful for decoding only if the results of all the sub-tasks assigned to that cluster and also to the fastest cluster were collected. Still, the proposed approach exhibited good average computation times by estimating and adapting to the computation speeds of the workers.

To overcome these limitations, in this work, for the multi-message, straggler-resistant, SDMM task, we propose a novel approach based on bivariate polynomial codes, which we call Secure Bivariate Polynomial (SBP) codes. We show that SBP codes outperform other schemes in the literature in terms of average computation time, when the upload cost budget is limited and when the number of fast workers is limited. Furthermore, we demonstrate that SBP codes exhibit low average computation times even in heterogeneous or homogeneous scenarios, where workers have significantly different or similar computation speeds, respectively.

In addition to SBP codes, we also propose an extension of GASP codes to the multi-message setting and evaluate its performance. Our results show that when the upload cost budget is sufficiently high, the proposed extension can considerably lower the average computation time of the SDMM task.

Overall, our proposed approaches address the challenges of straggler mitigation and security in the context of distributed matrix multiplication, and offer better performance than existing solutions in the literature.

4.3 Problem Setting

We study distributed matrix multiplication with strict security requirements. Unlike Chapter 3, and like the schemes in Section 2.1.4, in this chapter, the elements of our matrices are in a finite field $\mathbb{F}$, and we denote the size of the finite field by q. This is necessary due to our security requirements, which we will introduce and elaborate on shortly.

The master wants to multiply statistically independent matrices $\mathbf{A} \in \mathbb{F}^{r\times s}$ and $\mathbf{B} \in \mathbb{F}^{s\times t}$, $r, s, t \in \mathbb{Z}^+$, with the help of N dedicated workers, which possibly have heterogeneous computation speeds and storage capacities.

To offload the computation, the master divides the multiplication task into smaller sub-tasks, which are then assigned to workers. The master partitions $\mathbf{A}$ into K sub-matrices as $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1^T & \mathbf{A}_2^T & \cdots & \mathbf{A}_K^T\end{bmatrix}^T$, where $\mathbf{A}_i \in \mathbb{F}^{\frac{r}{K}\times s}$, $\forall i \in [1 : K]$, and $\mathbf{B}$ into L sub-matrices as $\mathbf{B} = \begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_L\end{bmatrix}$, where $\mathbf{B}_j \in \mathbb{F}^{s\times\frac{t}{L}}$, $\forall j \in [1 : L]$. The master sends coded versions of these partitions to the workers. In this section, unlike the setting discussed in Section 2.1.4, we allow multiple tasks assigned per worker, which is the multi-message setting considered in Chapter 3. We assume that there is an upload cost constraint per worker, denoted by u_i for worker i, which limits the maximum number of bits that can be transmitted from the master to each worker. This upload cost is a limiting factor on the number of coded partitions of $\mathbf{A}$, denoted by $m_{A,i}$, and of $\mathbf{B}$, denoted by $m_{B,i}$, that can be sent to each worker. Equivalently, this constraint can be seen as the storage constraints considered in Chapter 3. More specifically, for worker i, $m_{A,i}$ and $m_{B,i}$ must satisfy $(m_{A,i}rs/K + m_{B,i}st/L)\log_2(q) \leq u_i$. Provided that they comply with the upload cost constraint, $m_{A,i}$ and $m_{B,i}$ are chosen depending on the underlying coding scheme and the master sends coded partitions $\tilde{\mathbf{A}}_{i,k}$ and $\tilde{\mathbf{B}}_{i,l}$ to worker i, where $i \in [1 : N]$, $k \in [1 : m_{A,i}]$ and $l \in [1 : m_{B,i}]$. For simplicity, we describe a static setting, in which all the coded matrices are sent to the workers before they start computations. In a more dynamic scenario, matrix partitions can be delivered when they are needed, which would reduce the storage required by the workers. The workers multiply the received coded partitions of $\mathbf{A}$ and $\mathbf{B}$ as instructed by the underlying coding scheme and send the result of each computation to the master as soon as it is completed. Once the master receives a number of computations equal to the recovery threshold, R_{th}, it can decode the desired multiplication $\mathbf{AB}$.

We follow the same threat model discussed in Section 2.1.4, in which the workers are honest but curious. They follow the protocol, but they can use the received encoded matrices to obtain information about $\mathbf{A}$ and $\mathbf{B}$. We assume that up to T workers can collude. Hence, our security requirement is that no T workers are allowed to gain any information about the content of the multiplied matrices. That is,

$$I\left(\mathbf{A}, \mathbf{B}; \{\tilde{\mathbf{A}}_{i,k}, \tilde{\mathbf{B}}_{i,l} \mid k \in [m_{A,i}], l \in [m_{B,i}], i \in \mathcal{T}, \mathcal{T} \subset [1:N], |\mathcal{T}| = T\}\right) = 0, \tag{4.1}$$

where $\mathcal{T}$ is any subset with cardinality T of available worker indices.

As mentioned earlier, due to this strict security requirement, we use finite fields instead of real numbers, unlike Chapter 3. This is because sharing the matrix partitions in the case of real numbers would leak some information about them [75, 76] and it would not be possible to achieve zero mutual information between the coded partitions and the original matrices as required by Equation (4.1). Therefore, we choose to use finite fields.

Under this setting, the main problem we attempt to solve is minimizing the average computation time, which is defined as the time required for the master to collect sufficiently many computations to decode the desired computation $\mathbf{AB}$. We allow the workers' computation speeds to be homogeneous, i.e., the average speed of each available worker is close to each other, or heterogeneous, in which the average speeds of the workers vary. Workers can also straggle, i.e., become unresponsive temporarily.

4.4 Extension of Bivariate Polynomial Codes for SDMM

As a first attempt to improve the upload cost efficiency of SDMM, we provide the naive extension of bivariate polynomial codes proposed in Chapter 3 to the secure case. Consider the partitioning of matrices $\mathbf{A}$ and $\mathbf{B}$ as in Section 4.3. Recall from Equation (3.7) and Equation (3.8) that in bivariate polynomial codes, the encoding polynomials are constructed as

$$\mathbf{A}(x) = \mathbf{A}_1 + \mathbf{A}_2 x + \cdots + \mathbf{A}_K x^{K-1}, \tag{4.2}$$

$$\mathbf{B}(y) = \mathbf{B}_1 + \mathbf{B}_2 y + \cdots + \mathbf{B}_L y^{L-1}. \tag{4.3}$$

Therefore, at the master, the goal is to interpolate the following polynomial.

$$\mathbf{A}(x)\mathbf{B}(y) = \sum_{i=1}^{K}\sum_{j=1}^{L} \mathbf{A}_i\mathbf{B}_j x^{i-1}y^{j-1}. \tag{4.4}$$

Since worker $i \in [1 : N]$ can store $m_{A,i}$ partitions of $\mathbf{A}$ and $m_{B,i}$ partitions of $\mathbf{B}$, the master sends to worker i the first $m_{A,i}$ derivatives of $\mathbf{A}(x)$ and first $m_{B,i}$ derivatives of $\mathbf{B}(y)$, evaluated at distinct x_i and y_i, respectively. Each worker multiplies the received encoded partitions of $\mathbf{A}(x)$ and $\mathbf{B}(y)$ following either BPC-HO, BPC-VO, BPC-NZO or BPC-ZZO and sends the results of each computation as soon as it is finished. Then, once the master collects a sufficient number of computations from the workers, it instructs all the workers to stop and starts decoding $\mathbf{A}(x)\mathbf{B}(y)$.

In order to provide a simple direct extension of this scheme to SDMM in which up to T workers collude, for brevity, we limit the analysis to the case $m_{A,i} = m_A$ and $m_{B,i} = m_B$, $\forall i \in [1 : N]$. Thus, from a security point of view, each worker gets m_A and m_B coded partitions of $\mathbf{A}$ and $\mathbf{B}$, respectively. Since up to T workers collude, in total, m_AT coded partitions of $\mathbf{A}$ and m_BT coded partitions of $\mathbf{B}$ are leaked to the workers. To protect such a leakage, we need to add m_AT and m_BT random matrix partitions to $\mathbf{A}(x)$ and $\mathbf{B}(y)$, respectively. Thus, the encoding polynomials for this naive extension of bivariate polynomial codes to SDMM are constructed as

$$\mathbf{A}(x) = \mathbf{A}_1 + \mathbf{A}_2x + \cdots + \mathbf{A}_Kx^{K-1} + \sum_{i=1}^{m_AT} \mathbf{R}_i x^{K+i-1}, \tag{4.5}$$

$$\mathbf{B}(y) = \mathbf{B}_1 + \mathbf{B}_2y + \cdots + \mathbf{B}_Ly^{L-1} + \sum_{i=1}^{m_BT} \mathbf{S}_i x^{L+i-1}, \tag{4.6}$$

where $\mathbf{R}_i \in \mathbb{F}^{\frac{r}{K}\times s}$ and $\mathbf{S}_i \in \mathbb{F}^{s\times\frac{t}{L}}$ are matrix partitions whose elements are chosen uniformly at random from the elements of $\mathbb{F}_q$. Hence, the amount of randomness required for this scheme is $Ts(m_A\frac{r}{K} + m_B\frac{t}{L})$ elements of $\mathbb{F}_q$. Therefore, the polynomial to be interpolated at the master becomes

$$\begin{aligned}\mathbf{A}(x)\mathbf{B}(y) =& \sum_{i=1}^{K}\sum_{j=1}^{L} \mathbf{A}_i\mathbf{B}_j x^{i-1}y^{j-1} + \sum_{i=1}^{K}\sum_{j=L+1}^{m_BT} \mathbf{A}_i\mathbf{S}_j x^{i-1}y^{j-1}\\ &+ \sum_{i=K+1}^{m_AT}\sum_{j=1}^{L} \mathbf{R}_i\mathbf{B}_j x^{i-1}y^{j-1} + \sum_{i=K+1}^{m_AT}\sum_{j=L+1}^{m_BT} \mathbf{R}_i\mathbf{S}_j x^{i-1}y^{j-1}.\end{aligned} \tag{4.7}$$

Therefore, considering the number of monomials of $\mathbf{A}(x)\mathbf{B}(y)$ in Equation (4.7), $R_{th} = (K + m_AT)(L + m_BT)$ evaluations of $\mathbf{A}(x)\mathbf{B}(y)$ are needed to interpolate it, which is $\mathcal{O}(T^2)$ and hence, has a quadratic dependence on T.

Observe that in this naive extension, for a worker to provide $m = m_A m_B$ computations, uploading m_A coded partitions of $\mathbf{A}$ and m_B coded partitions of $\mathbf{B}$ are enough. This means that the upload cost of the scheme is on the order of $\sqrt{m}$. However, the price we pay for such a reduced upload cost is a quadratic dependence of the recovery threshold on the number of colluding workers. Such dependence on T may quickly become restrictive for typical T values and hence, the benefits of the naive extension of bivariate polynomial codes to SDMM may be outweighed by its drawbacks. Thus, we need more sophisticated schemes that can keep this low upload cost with a better scaling behaviour for the recovery threshold with respect to m and T. In the next section, we present our proposed solution for such a problem.

4.5 Secure Bivariate Polynomial (SBP) Codes

The objective of this section is to design a more sophisticated extension of bivariate polynomial codes for SDMM, which we call Secure Bivariate Polynomial (SBP) codes, in order to maintain the upload cost efficiency of bivariate polynomial codes while reducing the quadratic dependence of the recovery threshold on T.

4.5.1 Encoding Phase

In SBP coding scheme, the encoding polynomials are constructed as

$$\mathbf{A}(x) = \sum_{i=1}^{K} \mathbf{A}_i x^{i-1} + \sum_{i=1}^{T} \mathbf{R}_i x^{K+i-1}, \tag{4.8}$$

$$\mathbf{B}(x,y) = \sum_{i=1}^{L} \mathbf{B}_i y^{i-1} + \sum_{i=1}^{T}\sum_{j=1}^{m} \mathbf{S}_{i,j} x^{K+i-1} y^{j-1}, \tag{4.9}$$

where $m \leq L$ is the maximum number of sub-tasks any worker can complete. Matrices $\mathbf{R}_i \in \mathbb{F}_q^{\frac{r}{K}\times s}$ and $\mathbf{S}_{i,j} \in \mathbb{F}_q^{s\times\frac{t}{L}}$ are independent and uniform randomly generated from their corresponding domain for $i \in [1:T]$ and $j \in [1:m]$. This implies that the amount of randomness required for SBP scheme is $Ts(\frac{r}{K} + m\frac{t}{L})$ elements of $\mathbb{F}_q$.

For each worker i, the master evaluates $\mathbf{A}(x)$ at x_i and the derivatives of $\mathbf{B}(x,y)$ with respect to y up to the order $m-1$ at (x_i, y_i). We require that these evaluation points be distinct. Thus, to worker i, the master sends $\mathbf{A}(x_i)$ and $\mathcal{B}_i = \{\mathbf{B}(x_i,y_i), \partial_1\mathbf{B}(x_i,y_i), \ldots, \partial_{m-1}\mathbf{B}(x_i,y_i)\}$, where ∂_i denotes the i^{th} partial derivative with respect to y. Thus, in this coding scheme, we impose the storage or equivalently upload constraints that $m_{A,i} = 1$ and $m_{B,i} = m$.

In Equation (4.8) and Equation (4.9), the role of $\mathbf{R}_i$'s and $\mathbf{S}_{i,j}$'s is to mask the actual matrix partitions for security. The following theorem states that the coded matrix partitions sent to the workers do not leak any information about $\mathbf{A}$ and $\mathbf{B}$ as long as not more than T workers collude.

Theorem 4.1. *For the encoding scheme described above, we have*

$$I(\mathbf{A}, \mathbf{B}; \{\mathbf{A}(x_i), \mathcal{B}_i : i \in \mathcal{T}, \mathcal{T} \subset [1:N], |\mathcal{T}| = T\}) = 0. \tag{4.10}$$

Proof. Consider any $\mathcal{T}$ such that $\mathcal{T} \subset [1:N]$ and $|\mathcal{T}| = T$. Since $\mathbf{A}$ and $\mathbf{B}$ are independent, we have

$$I(\mathbf{A}, \mathbf{B}; \{\mathbf{A}(x_i), \mathcal{B}_i : i \in \mathcal{T}\}) = I(\mathbf{A}; \{\mathbf{A}(x_i) : i \in \mathcal{T}\}) + I(\mathbf{B}; \{\mathcal{B}_i : i \in \mathcal{T}\}). \tag{4.11}$$

Let us first bound $I(\mathbf{A}; \{\mathbf{A}(x_i) | i \in \mathcal{T}\})$ as follows.

$$\begin{aligned}
&I\left(\mathbf{A}; \{\mathbf{A}(x_i) : i \in \mathcal{T}\}\right) \\
&= H\left(\{\mathbf{A}(x_i) : i \in \mathcal{T}\}\right) - H\left(\{\mathbf{A}(x_i) : i \in \mathcal{T}\} | \mathbf{A}\right) && (4.12)\\
&= H\left(\{\mathbf{A}(x_i) : i \in \mathcal{T}\}\right) - H\left(\{\mathbf{R}_i : i \in [1:T]\} | \mathbf{A}\right) && (4.13)\\
&\overset{(a)}{=} H\left(\{\mathbf{A}(x_i) : i \in \mathcal{T}\}\right) - T\frac{rs}{K}\log(q) && (4.14)\\
&\overset{(b)}{\leq} \sum_{i=1}^{|\mathcal{T}|} H(\mathbf{A}(x_i)) - T\frac{rs}{K}\log(q) && (4.15)\\
&= T\frac{rs}{K}\log(q) - T\frac{rs}{K}\log(q) = 0, && (4.16)
\end{aligned}$$

where (a) follows from the fact that $\mathbf{R}_i$'s are independent of each other and of $\mathbf{A}$, (b) is due to the fact that the joint entropy of several random variables is upper bounded by the sum of the individual entropies of these random variables.

We can bound $I(\mathbf{B}; \{\mathcal{B}_i : i \in \mathcal{T}\})$ similarly as follows.

$$\begin{aligned}
&I\left(\mathbf{B}; \{\mathcal{B}_i : i \in \mathcal{T}\}\right) \\
&= H\left(\{\mathcal{B}_i : i \in \mathcal{T}\}\right) - H\left(\{\mathcal{B}_i : i \in \mathcal{T}\} | \mathbf{B}\right) && (4.17)\\
&= H\left(\{\mathcal{B}_i : i \in \mathcal{T}\}\right) - H(\{\mathbf{S}_{i,j} : i \in [1:T], j \in [1:m]\} | \mathbf{B}) && (4.18)\\
&= H\left(\{\mathcal{B}_i : i \in \mathcal{T}\}\right) - Tm\frac{st}{L}\log(q) && (4.19)\\
&\leq \sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{m} H(\mathbf{B}(x_i, y_j)) - Tm\frac{st}{L}\log(q) && (4.20)\\
&= Tm\frac{st}{L}\log(q) - Tm\frac{st}{L}\log(q) = 0. && (4.21)
\end{aligned}$$

The claim follows by substituting Equation (4.16) and Equation (4.21) into Equation (4.11). □

4.5.2 Computation Phase

After receiving $\mathbf{A}(x_i)$ and $\mathcal{B}_i$, with the increasing order of $j \in [1:m]$, worker i multiplies $\mathbf{A}(x_i)$ with $\partial_{j-1}\mathbf{B}(x_i, y_i)$. That is, the j^{th} completed computation is $\mathbf{A}(x_i)\partial_{j-1}\mathbf{B}(x_i, y_i)$. Then, as soon as each multiplication is completed, its result is communicated back to the master.

4.5.3 Decoding Phase

After collecting sufficiently many computations from the workers, the master aims to interpolate

$$\begin{aligned}\mathbf{A}(x)\mathbf{B}(x,y) = &\sum_{i\in[1:K]}\sum_{j\in[1:L]} \mathbf{A}_i\mathbf{B}_j x^{i-1}y^{j-1} + \sum_{i\in[1:K]}\sum_{t\in[1:T]}\sum_{j\in[1:m]} \mathbf{A}_i\mathbf{S}_{t,j}x^{K+i+t-2}y^{j-1} \\ &+ \sum_{i\in[1:T]}\sum_{j\in[1:L]} \mathbf{R}_i\mathbf{B}_j x^{K+i-1}y^{j-1} + \sum_{i\in[1:T]}\sum_{t\in[1:T]}\sum_{j\in[1:m]} \mathbf{R}_i\mathbf{S}_{t,j}x^{2K+i+t-2}y^{j-1}.\end{aligned} \tag{4.22}$$

Note that, in this scheme, every computation is equally useful; that is, the sub-tasks are one-to-any replaceable. In the following theorem, we give the recovery threshold expression, which specifies the minimum number of required computations and characterizes the probability that decoding fails, i.e., bivariate polynomial interpolation fails, due to the use of a finite field.

Theorem 4.2. *Assume the evaluation points (x_i, y_i) are chosen uniform randomly over the elements of $\mathbb{F}$. If the number of computations of sub-tasks received from the workers, which obey the computation order specified in Section 4.5.2 is greater than the recovery threshold $R_{th} \triangleq (K+T)L + m(K+T-1)$, then with probability at least $1 - d/q$, the master can interpolate the unique polynomial $\mathbf{A}(x)\mathbf{B}(x,y)$, where*

$$d \triangleq \frac{m}{2}\left(3(K+T)^2 + m(K+T) - 6K - 6T - m + 3\right) + \frac{(K+T)L}{2}(K+L+T-2). \tag{4.23}$$

We give the proof of Theorem 4.2 in Section 4.8. Theorem 4.2 states that we can make the probability of failure arbitrarily small by increasing the order q of the finite field.

Theorem 4.3. *The total upload cost of the SBP coding scheme is $N\,(rs/K + mst/L)\log_2(q)$ bits.*

Proof. The SBP coding scheme assigns m computations to each worker, by sending one coded partition of $\mathbf{A}$ and m coded partitions of $\mathbf{B}$. Remember that each coded partition of $\mathbf{A}$ is a matrix of size $\frac{r}{K} \times s$ and each coded partition of $\mathbf{B}$ is a matrix of size $s \times \frac{t}{L}$. Since there are N workers, the master uploads $N\,(rs/K + mst/L)$ elements of the field $\mathbb{F}$. Since, in total, there are q elements in $\mathbb{F}$, the total upload cost is $N\,(rs/K + mst/L)\log_2(q)$ bits. $\square$

Remark 4.1. The SBP scheme does not exploit any parameter of the underlying statistical model of the workers' speeds. Under a total upload cost constraint, if no prior information about the computation speeds of the workers is available, then assigning more computation load, m, to every worker is a favourable approach. Although this increases the recovery threshold, i.e., the term $m(K + T - 1)$, the faster workers do not run out of computations easily, avoiding the slowest workers dominating the computation time. The benefit of this prevails over the detriment due to the increase in the recovery threshold. Surely, if prior information about the computation speeds of the workers is available, we could exploit it by assigning more, but still less than L, computations to faster workers, which would result in fewer coded partitions leaked to the colluding workers. In this case, the recovery threshold would be lower, further increasing the protection against stragglers. However, the SBP scheme has been designed as agnostic to the delay model of the workers and specifically to maximize the number of sub-tasks delivered by a worker under an upload cost constraint. Thanks to the extra computations at workers, we show in our simulation results that a model-independent version of SBP is enough to beat model-dependent schemes such as the one in [74]. Thus, we expect the SBP scheme to work for large varieties of statistical models of the worker's speeds.

Remark 4.2. Similarly to Chapter 3, the decoding procedure of SBP scheme can be conducted by inverting the interpolation matrix and multiplying it with the vector of responses collected from the workers, which has a complexity of $\mathcal{O}(rc(KL)^2)$. Still, like the non-secure version of bivariate polynomial codes in Chapter 3, we conjecture that the decoding complexity of SBP can be reduced to almost linear by carefully choosing the evaluation points. Such an investigation is beyond the scope of this dissertation, and we leave it as an interesting future research direction.

4.6 Extension of GASP codes to multi-message setting

As one of the state-of-the-art schemes for SDMM, GASP codes are discussed in detail in Section 2.1.4. However, originally, GASP codes are designed for the single-message scenario, in which each worker is assigned a single computation task. In this section, we extend the GASP codes to the multi-message setting, which we call multi-message GASP (MM-GASP) scheme.

For the extension of GASP codes to the multi-message setting, i.e., MM-GASP, we assign $m > 1$ tasks to each worker. Thus, a worker can see m evaluations of $\mathbf{A}(x)$ and $\mathbf{B}(x)$, and any T colluding workers can see mT evaluations. Thus, to make the scheme secure against T colluding workers, we need to add mT random matrix partitions to each encoding polynomial, instead of T. Thus, for MM-GASP, we modify the encoding polynomials of GASP codes given in Equation (2.20) and Equation (2.21) such that

$$\mathbf{A}(x) = \sum_{i=1}^{K} \mathbf{A}_i x^{\alpha_i} + \sum_{i=1}^{mT} \mathbf{R}_i x^{\alpha_{K+i}}, \tag{4.24}$$

$$\mathbf{B}(x) = \sum_{i=1}^{L} \mathbf{B}_i x^{\beta_i} + \sum_{i=1}^{mT} \mathbf{S}_i x^{\beta_{L+i}}. \tag{4.25}$$

The elements of matrices $\mathbf{R}_i \in \mathbb{F}_q^{\frac{r}{K} \times s}$ and $\mathbf{S}_{i,j} \in \mathbb{F}_q^{s \times \frac{t}{L}}$ are independently and uniform randomly generated from $\mathbb{F}_q$. Therefore, the amount of randomness required for MM-GASP is $Tsm(\frac{r}{K} + \frac{t}{L})$ elements of $\mathbb{F}_q$.

Based on these encoding polynomials, the following theorem characterizes the recovery threshold of MM-GASP.

Theorem 4.4. *The recovery threshold of MM-GASP is given by*

$$R_{th}^{MM-GASP}(K, L, T) = \begin{cases} KL + K + L, & 1 = mT < L \leq K \\ KL + K + L + (mT)^2 + mT - 3, & 1 < mT < L \leq K \\ (K + mT)(L + 1) - 1, & L \leq mT < K \\ 2KL + 2mT - 1, & L \leq K \leq mT. \end{cases} \tag{4.26}$$

Proof. We can define $\tilde{T} = mT$ and write

$$\mathbf{A}(x) = \sum_{i=1}^{K} \mathbf{A}_i x^{\alpha_i} + \sum_{i=1}^{\tilde{T}} \mathbf{R}_i x^{\alpha_{K+i}}, \tag{4.27}$$

$$\mathbf{B}(x) = \sum_{i=1}^{L} \mathbf{B}_i x^{\beta_i} + \sum_{i=1}^{\tilde{T}} \mathbf{S}_i x^{\beta_{L+i}}. \tag{4.28}$$

Now, let us consider

$$\begin{aligned}\mathbf{A}(x)\mathbf{B}(x) = &\sum_{i=1}^{K}\sum_{j=1}^{L} \mathbf{A}_i\mathbf{B}_j x^{\alpha_i+\beta_i} + \sum_{i=1}^{K}\sum_{j=1}^{\tilde{T}} \mathbf{A}_i\mathbf{S}_j x^{\alpha_i+\beta_{L+i}} \\ &+ \sum_{i=1}^{\tilde{T}}\sum_{j=1}^{L} \mathbf{R}_i\mathbf{B}_j x^{\alpha_{K+i}+\beta_i} + \sum_{i=1}^{\tilde{T}}\sum_{j=1}^{\tilde{T}} \mathbf{R}_i\mathbf{S}_j x^{\alpha_{K+i}+\beta_{L+i}}.\end{aligned} \tag{4.29}$$

In the proof of Equation (2.30) in [13], α_i's and β_i's are chosen such that from the evaluations of $\mathbf{A}(x)\mathbf{B}(x)$, $\mathbf{A}_i\mathbf{B}_j$'s $\forall i \in [K], j \in [L]$ are decodable and the number of monomials in $\mathbf{A}(x)\mathbf{B}(x)$ whose coefficients are undesired terms is minimized. For this, we only need to consider the structure of $\mathbf{A}(x)\mathbf{B}(x)$ and m itself is not related other than determining the value of $\tilde{T}$. Therefore, the problem reduces to deriving the recovery threshold of a classical GASP coding scheme when $\tilde{T}$ workers collude, which is $R_{th}^{GASP}(K, L, \tilde{T})$ by Equation (2.30). If we substitute $\tilde{T} = mT$, then we obtain Equation (4.26).

□

Remark 4.3. In multi-message univariate polynomial coding schemes, such as in MM-GASP codes, if a worker is assigned m sub-tasks, then m coded partitions of both $\mathbf{A}$ and $\mathbf{B}$ are required. Thus, the total upload cost of MM-GASP is $Nm\,(rs/K + st/L)\log_2(q)$ bits, which is larger than that of SBP coding scheme.

The recovery thresholds of the MM-GASP codes and SBP codes can be compared as a function of the number of coded partitions m, by direct inspection of the recovery thresholds in Equation (4.26) and Theorem 4.2. Observe that SBP coding scheme's recovery threshold is smaller than that of the MM-GASP code if $L \leq mT < K$, and $K \leq TL+1$, which is satisfied as K and L become close to each other, or, if $L \leq K \leq mT$, and $(K - T)(L - m) \geq (1 - m)$ is satisfied. In Figure 4.1, we provide the recovery thresholds of the two schemes as a function of the number of computations allocated to each worker for $K = L = 100$ and $T = 30$.

We note that such a comparison may only be meaningful in the unlimited upload cost budget scenario. Otherwise, comparing the recovery thresholds for the same m might be misleading since, for a given upload cost constraint, each scheme provides a different

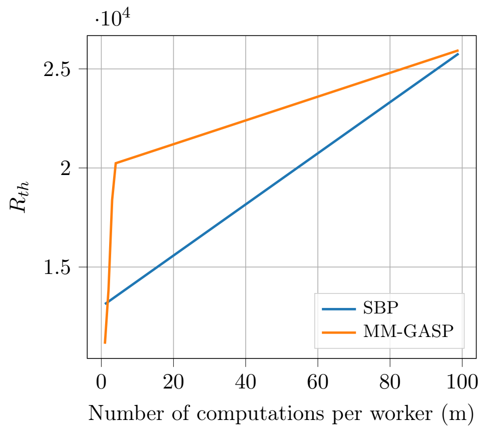


FIGURE 4.1: R_{th} vs. the number of computations assigned to each worker for SBP coding scheme and the MM-GASP scheme for $K = L = 100$ and $T = 30$.

number of sub-tasks, m, to workers, as detailed in Theorem 4.3 and Remark 4.3, for SBP and for MM-GASP, respectively. We provide further discussion on this issue in the next section, see Figure 4.3, where we show the recovery thresholds as a function of the total upload cost budget for a scenario with $K = L = 100$ and $T = 30$.

Similarly, for a fixed m, MM-GASP also requires more randomness compared to the bivariate schemes, which is $Tsm(\frac{r}{K}+\frac{t}{L})$ compared to $Ts(\frac{r}{K}+m\frac{t}{L})$ and $Ts(m_a\frac{r}{K}+m_B\frac{t}{L})$ for SBP and direct extension of bivariate codes, respectively.

4.7 Simulation Results and Discussion

In this section, we compare SBP codes with MM-GASP and the rateless coding scheme proposed in [74] in terms of the trade-off between the average computation time (ACT) and the total upload cost budget (UCB), under the scenarios with heterogeneous and homogeneous workers.

The comparison between the MM-GASP scheme and SBP coding scheme is direct, as both are based on the same set of assumptions. They achieve different recovery thresholds as a function of the L, M, T and m, but they both assume that the coded submatrices are uploaded only once before the computations start, no prior knowledge of the computation speeds of workers is needed or can be exploited, and the first R_{th} results received from any subset of the workers allow recovering the desired computation. However, the setting and the assumptions in [74] are slightly different. In the rateless coding scheme proposed in [74], computations are organized in rounds. If the speeds of the workers are not already known, in the first round, every worker is assigned one computation to estimate their speeds. Then, based on the known or estimated speeds, workers are grouped into c clusters, such that the workers with similar speeds are in the same cluster. We denote by n_u the number of workers belonging to cluster u, $u \in [c]$. In each round, for any computation within a cluster to be useful for decoding, we need all the workers in that cluster and also all workers in cluster 1, which is special, to finish their assigned tasks. Once all the workers in cluster u and cluster 1 finish their tasks, they provide d_u, and d_1 useful computations to the master, where $d_1 = \lfloor (n_1 - 2T + 1)/2 \rfloor$ and $d_u = \lfloor (n_u - T + 1)/2 \rfloor$ for $2 \leq u \leq c$. No further synchronization is needed among clusters, and a new task can be assigned to a worker as soon as it finishes its assigned task. Once $KL(1+\epsilon)$ useful computations are collected by the master from different clusters across multiple rounds, the decoding procedure can start. Here, ϵ is the overhead due to the Fountain codes used in [74], which is set to 0.05 in our simulations. The performance of this scheme depends critically on how well the distribution of the workers' speeds can be estimated. Observe that, if a worker in a fast cluster becomes a straggler, the finishing time of the overall cluster can be arbitrarily delayed. This is the main drawback of this scheme in comparison with SBP coding scheme and the MM-GASP, for which any computation at any worker is equally useful. However, the clear advantage of the rateless coding scheme is that the computation load m_i, i.e., the number of tasks assigned to worker i, does not need to be specified in advance, and tasks can be dynamically allocated to workers in each cluster across rounds. Moreover, the recovery threshold is not dominated by the maximum computation load $m = \max m_i$, as is the case for SBP coding and the MM-GASP schemes. Therefore, in order to allow the rateless coding scheme to benefit from this flexibility, in our simulations, we consider a total upload cost for [74], i.e., the computations are assigned to clusters until the total upload communication budget is met, while for MM-GASP and SBP we impose an upload cost constraint per worker. We emphasize that this is a relaxation of the problem formulation introduced in Section 4.3, and is only applied to the rateless coding scheme. Although SBP coding scheme and the MM-GASP code can also benefit from this relaxation when the computation statistics of the workers are known, such optimization is out of the scope of this dissertation and left as a potential future work.

In our simulations, following the literature [59,72], and similarly to Chapter 3, we assume that the time for a worker to finish one sub-task is distributed as a shifted exponential random variable with probability density function

$$f(t) = \begin{cases} \lambda e^{-\lambda(t-\nu)} & \text{if } t \geq \nu, \\ 0 & \text{otherwise,} \end{cases} \tag{4.30}$$

where the scale parameter λ controls the speed of the worker and the shift parameter ν is the minimum time duration for a task to be completed. Smaller λ implies slower workers and more tendency to straggle.

In each scenario considered in the following subsections, we run 1000 experiments independently with the given parameters and present the average computation time. We assume that the partitions of matrices $\mathbf{A}$ and $\mathbf{B}$ have the same size, i.e., $\frac{r}{K} = \frac{t}{L}$, in all of the scenarios considered. Given that each sub-task is a fraction $\frac{1}{KL}$ of the complete task, to facilitate the comparison between different configurations, we choose $\lambda \propto KL$, and $\nu \propto \frac{1}{KL}$, in all simulation setups.

4.7.1 Heterogeneous Workers

In this subsection, we assume that the computation speeds of the workers are heterogeneous. Specifically, we assume six *heterogeneity classes*, with scale parameters $\lambda_1 = 10^{-1} \times KL$, $\lambda_2 = 10^{-1.5} \times KL$, $\lambda_3 = 10^{-2} \times KL$, $\lambda_4 = 10^{-2.5} \times KL$, $\lambda_5 = 10^{-3} \times KL$ and $\lambda_6 = 10^{-3.5} \times KL$, and a common shift parameter of $\nu = 10/(KL)$ seconds. There are 75 workers for each class summing up to $N = 450$ workers in total, and assume that any subset of at most $T = N/15$ workers can collude. We divide both matrices $\mathbf{A}$ and $\mathbf{B}$ into $K = L = 100$ partitions. We evaluate the scheme in [74] for several numbers of clusters, c, to observe the effect of the mismatch between the actual number of heterogeneity classes and the chosen c value. While generating the clusters, we simply assign around N/c workers to each cluster, according to the estimated speeds in the first round. We do not change the parameter c across rounds.

First, we assume that workers' scale parameters do not deviate at all from the given parameters across the rounds. We call such workers as *stable workers.* In Figure 4.2, we plot the ACT of the compared schemes versus the total UCB by assuming stable workers. In Figure 4.3, we also show the actual recovery thresholds of SBP codes, MM-GASP codes, and the average recovery threshold of the rateless coding scheme for different c values. As the name suggests, the rateless scheme does not have a constant recovery threshold and its actual value depends on the computation speeds of the workers, the number of clusters, and the number of workers assigned to them. Therefore, we present the average recovery threshold for this scheme.

As observed in Figure 4.2, the SBP coding scheme is able to finish the overall task for much lower UCB values than the other two schemes. This is thanks to the fact that SBP is able to provide more computations, m, to workers for the same UCB, as highlighted in Remark 4.1. In SBP, increasing the total UCB leads to an increase in both m and R_{th}, as seen in Figure 4.3. This means that workers have to provide more computations to the master in order to attain R_{th}. However, the benefit of workers' ability to provide more computations outweighs the increase in R_{th}, and as a result, the ACT decreases when UCB increases. The reason for this is the heterogeneity of the workers' speeds.

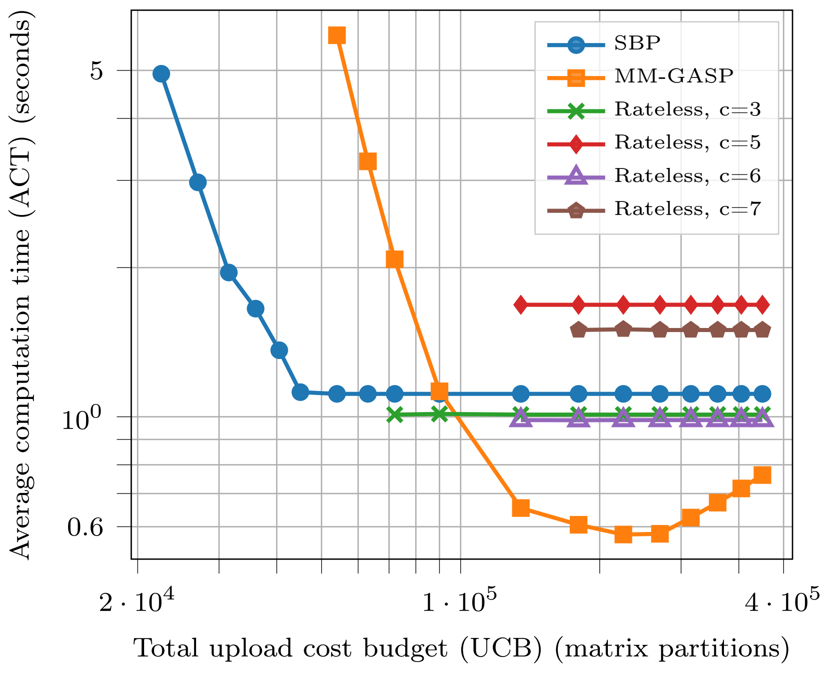


FIGURE 4.2: ACT vs. total UCB trade-off of the compared schemes with heterogeneous and stable workers.

That is, for a low total UCB, m is so small that the master cannot complete all R_{th} computations from only fast workers. When we increase m, the maximum number of computations the fast workers can provide also increases, and the benefit of this increase dominates over the increase in the R_{th}. For the SBP scheme, this is so until we reach a total UCB value corresponding to $m = L$ i.e., total UCB of $L \times N = 45000$. After this point, the ACT of SBP coding scheme stays constant. This is an inherent limitation of SBP coding scheme since the maximum value of m is L. Beyond that value, we cannot benefit from the additional UCB.

For MM-GASP codes, we observe that, although their recovery threshold is close to that of SBP coding scheme in the low UCB regime, as seen in Figure 4.3, the minimum total UCB for which MM-GASP codes are able to complete the overall task is larger than SBP coding scheme. That is because the MM-GASP scheme is a univariate scheme; and thus, for the same total UCB, the maximum number of computations a worker can provide is less than the one in SBP coding scheme. For the same reason, at intermediate total UCB availability, i.e., values less than 9×10^4 partitions, the ACT of the MM-GASP scheme is quite large compared to SBP coding scheme. However, for larger values of total UCB, we observe in Figure 4.2 that MM-GASP's ACT decreases rapidly, substantially outperforming the other two schemes. However, after a critical point, if the total UCB further increases, then the ACT starts to increase again. That is because, beyond that point, the increase in the recovery threshold is not compensated by the additional computations at workers. Unfortunately, operating at this point may not be always possible.

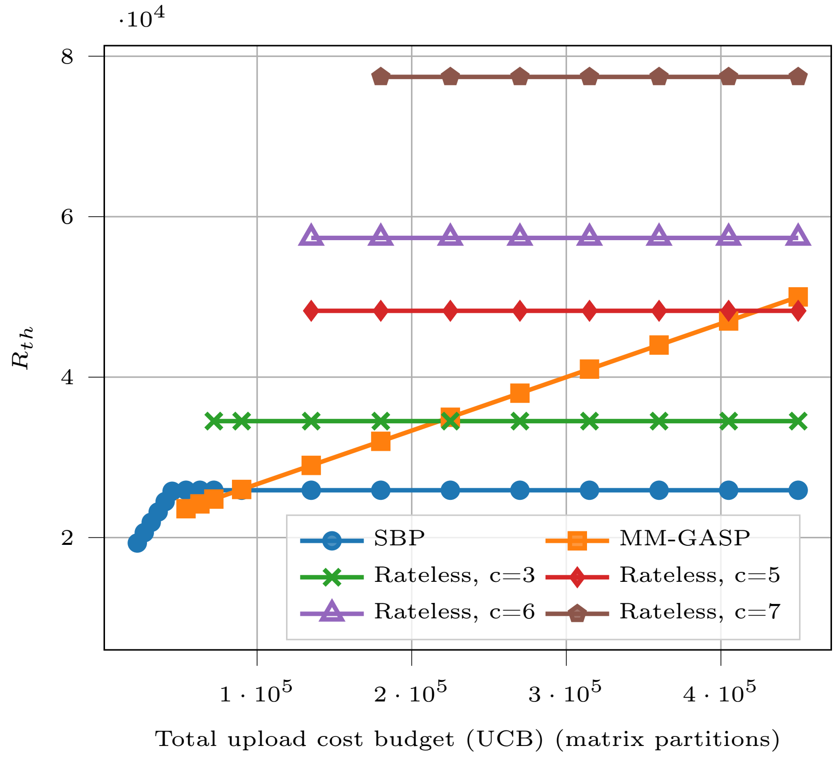


FIGURE 4.3: Average R_{th} of the compared schemes with heterogeneous and stable workers

Especially when we do not have any prior information about the statistics of the workers' speeds. Nevertheless, some heuristics can still be useful to approximate it and even if the optimal point cannot be found, a sufficiently close point can still be beneficial. Thus, we can conclude that, if a good heuristic can be found to identify a near-optimal m value, for large UCB values, MM-GASP codes can complete the overall task faster than SBP coding scheme as well as the rateless coding scheme. This makes MM-GASP codes a good alternative for the scenarios with high UCB availability.

Finally, similarly to MM-GASP codes, we observe that the rateless codes start being able to complete the overall task only at a relatively high total UCB value. That is because the rateless coding scheme assigns a new sub-task to a worker as soon as it finishes its task without waiting for the other clusters to finish. Thus, the UCB is greedily invested in the fastest cluster. However, despite its speed, in terms of the number of useful computations provided, the fastest cluster is less efficient than the other clusters. Recall that $d_1 = \lfloor (n_1 - 2T + 1)/2 \rfloor$, but $d_u = \lfloor (n_u - T + 1)/2 \rfloor$ for $2 \leq u \leq c$. Therefore, if the number of workers in the fastest cluster is limited, then for the low UCB values, the rateless scheme cannot complete the overall task since it runs out of the necessary upload resources before the master receives the minimum number of useful computations to decode $\mathbf{AB}$, which is $KL(1 + \epsilon)$.

Moreover, we observe in Figure 4.3 that when the number of clusters is low, the recovery threshold is also lower, and the rateless scheme starts completing the overall task at a lower value of total UCB. That is because when c is low, since we assign N/c workers per cluster, there are more workers in the fastest cluster. However, as observed in Figure 4.2, a smaller recovery threshold does not always imply a lower ACT. In general, we expect that the rateless coding scheme performs well when c is equal to the number of heterogeneity classes, but, in this case, we also observe that it performs equally well for $c = 3$. That is because, for $c = 3$, there is no λ_i, $i \in [6]$ appearing in more than one cluster, i.e., workers in the same heterogeneity class are allocated to the same cluster. Therefore, for rateless codes, it is important to choose the design parameter c carefully. Although for some c values and for some large UCB availabilities, it may perform better than SBP and MM-GASP, in practice, we may not know the number of heterogeneity classes, and hence, such a clear grouping of computation statistics may not be possible. For such cases, SBP coding scheme or the MM-GASP may be preferable over the rateless coding scheme.

In addition to choosing c optimally, estimating the instantaneous speeds of the workers is another issue that needs to be addressed in rateless codes. In real-world scenarios, the speeds of the workers can occasionally change due to temporary failures, parallel job assignments, etc. To model this, we introduce another simulation scenario, in which workers' scale parameters can deviate from their original values with a very low probability ρ. We refer to such workers as *mostly-stable workers*. That is, in any round, a worker with λ_i sticks to λ_i with probability $1 - \rho$, but with a small probability ρ, it draws its scale parameter uniform randomly from $\{\lambda_j \mid j \in [1:6]\}$. We consider such a scenario to model the instantaneous changes in workers' speeds since the detection of such changes by the master and putting this worker to the correct cluster takes at least one round. Taking $\rho = 0.001$, we plot the ACT of the compared schemes in Figure 4.4.

We observe that even with such a small probability deviation from the estimated scale parameters, the performance of [74] degrades considerably. Thus, we can argue that, in addition to the substantial improvement in the low and the intermediate UCB values, SBP coding scheme can be advantageous over [74] even when high UCB is available depending on the statistics of the workers' speeds.

4.7.2 Homogeneous Workers

In this subsection, we consider the case in which the computation speeds of the workers are homogeneous, and we compare the ACTs of the considered schemes with respect to the available total UCB. That is, we have 450 workers as in Section 4.7.1, but this time,

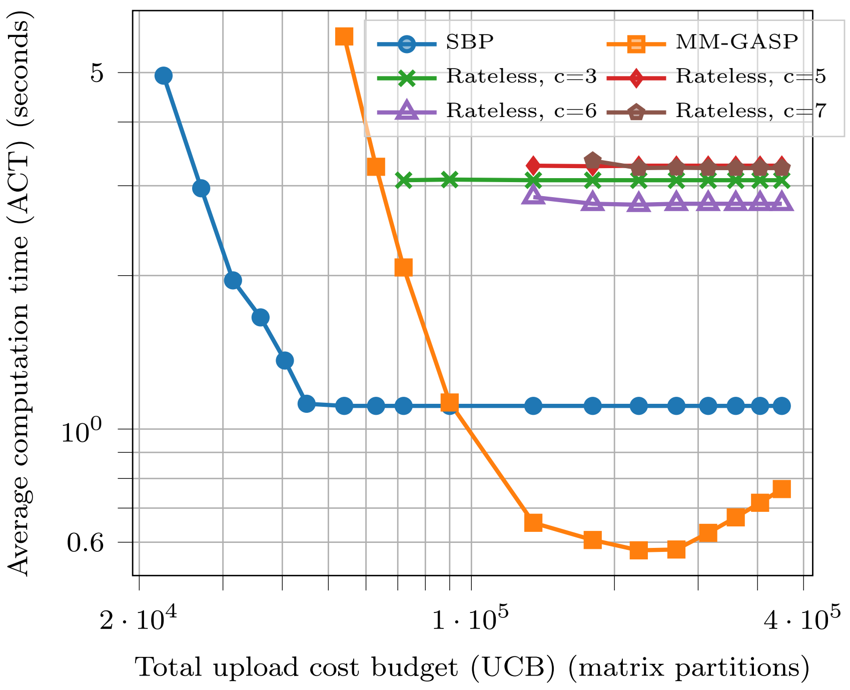


FIGURE 4.4: ACT vs. total UCB trade-off of the compared schemes with heterogeneous and mostly-stable workers.

all the workers follow the same statistics with $\lambda = 10^{-2} \times KL$ and $\nu = 10/(KL)$. We assume at most $T = N/15$ workers can collude, and we divide $\mathbf{A}$ and $\mathbf{B}$ into $K = L = 100$ partitions. Although homogeneous workers' speeds are considered, for the rateless scheme in [74], we consider different numbers of clusters $c \in [1:3]$ in order to analyse its effect. In Figure 4.5, we present the ACT versus UCB plot for this setting.

Similarly to the heterogeneous case discussed in Section 4.7.1, due to the upload cost efficiency of the bivariate polynomial codes, we observe that the minimum UCB for which SBP can complete the overall task is smaller than for the other schemes. Moreover, in this homogeneous case, we observe that the ACTs of SBP and MM-GASP only increase with the total UCB. That is because, due to the similarity in workers' speeds, there is no need for the faster workers to compensate for the slower ones. Therefore, rather than improving the ACT, increasing m beyond the minimum value, for which the schemes complete the overall task, results in a higher ACT since it also increases R_{th}. Therefore, we depict the best ACT for SBP and MM-GASP coding schemes in Figure 4.5 and Figure 4.6 by flat dashed lines.

On the other hand, for the rateless codes, we observe that, regardless of the number of clusters, c, considered, they perform significantly worse than SBP coding scheme for all UCB values. That is because, while the sub-tasks are one-to-any replaceable in SBP coding scheme, i.e, the result of any sub-task can compensate for the absence of any other sub-task, this is not the case in the rateless coding scheme. Since we consider the

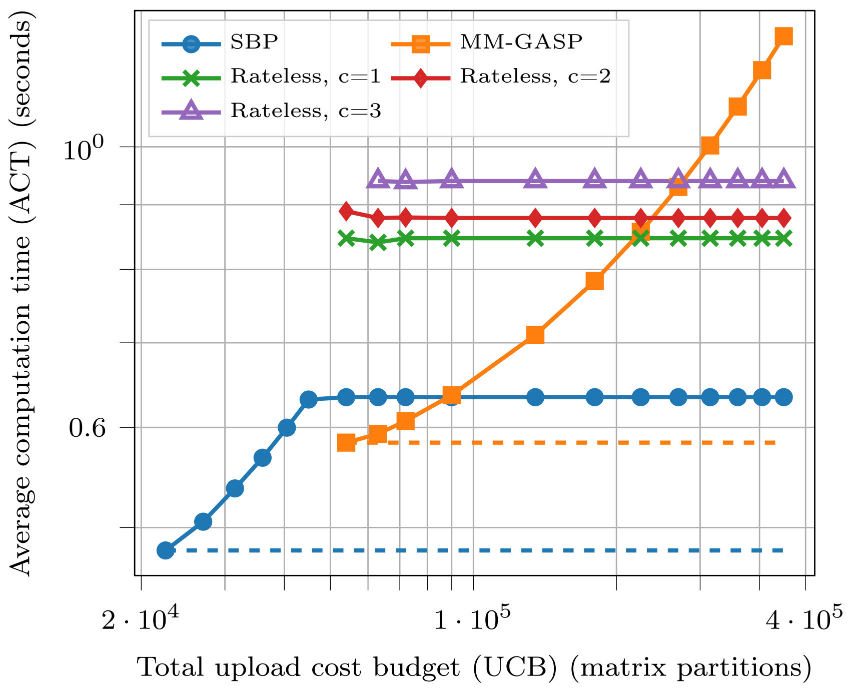


FIGURE 4.5: ACT vs. total UCB trade-off of the compared schemes with homogeneous and stable workers.

homogeneous workers in their speeds, there is not much difference between the clusters in the rateless coding scheme. Since to decode the sub-tasks in a cluster, all of the workers in that cluster must finish their sub-tasks, the ACT increases.

As we stated in Section 4.7.1, in a real-world scenario, the speeds of workers can occasionally change. To model this effect, in Figure 4.6, we provide the ACT versus UCB trade-off in the scenario in which the workers are mostly-stable with a transition probability $\rho = 0.001$. Since there is only one heterogeneity class in the homogeneous case, to simulate mostly-stable workers, we assume that a worker sticks to $\lambda = 10^{-2} \times KL$ with probability $1 - \rho$, but with probability ρ, its λ parameter is chosen uniformly between $\lambda = 10^{-3} \times KL$ and $\lambda = 10^{-4} \times KL$.

We observe that the effect of such a low probable deviation from the original parameters is considerable in the rateless codes since in order to utilize the computations in a cluster, all the workers in that cluster must complete their sub-tasks. If some of these workers straggle even only for one round, it can delay the overall computation significantly.

To conclude, we observe that, in the cases in which the workers' speeds are known to be close to each other, i.e., homogeneous, SBP coding scheme is preferable over both the rateless coding and the MM-GASP schemes.

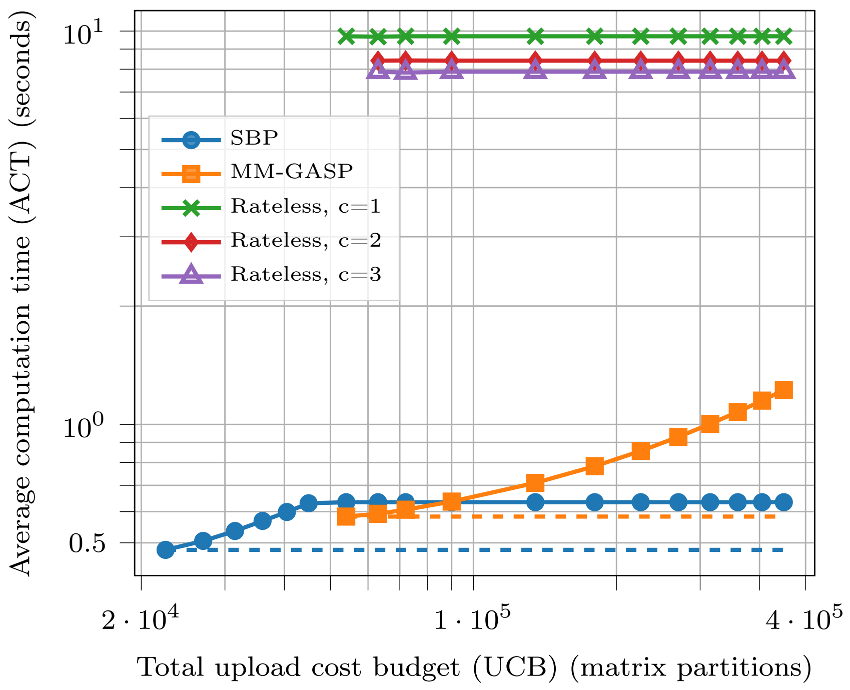


FIGURE 4.6: ACT vs. total UCB trade-off with homogeneous and mostly-stable workers.

4.8 Proof of Theorem 4.2

Firstly, we visually show that the degree of $\mathbf{A}(x)\mathbf{B}(x,y)$ is equal to the recovery threshold expression in Theorem 4.2. In Figure 4.7, we visualize the degrees of the monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$ in the $\deg(x)-\deg(y)$ plane. That is, each filled circle represents a monomial of $\mathbf{A}(x)\mathbf{B}(x,y)$. If we count them, we see that on the left rectangle, there are $(K+T)L$ elements and on the right rectangle, there are $m(K+T-1)$ elements. Hence, we conclude that the number of monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$ is $(K+T)L+m(K+T-1)$.

In the rest of the proof, we need to show that every possible combination of $(K+T)L+m(K+T-1)$ responses collected at the master interpolates a unique polynomial $\mathbf{A}(x)\mathbf{B}(x,y)$, implying $R_{th}=(K+T)L+m(K+T-1)$.

As done in Chapter 3, the bivariate polynomial interpolation problem can be formulated as solving a linear system of equations, whose unknowns are the coefficients of $\mathbf{A}(x)\mathbf{B}(x,y)$ and whose coefficient matrix, or interpolation matrix, consists of the monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$ and their derivatives with respect to y evaluated at the evaluation points of the workers, $(x_i,y_i), i\in[1:N]$. Different from Chapter 3, instead of KL, the dimension of the interpolation matrix in this case is $(K+T)L+m(K+T-1)$ since this is the number of monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$.

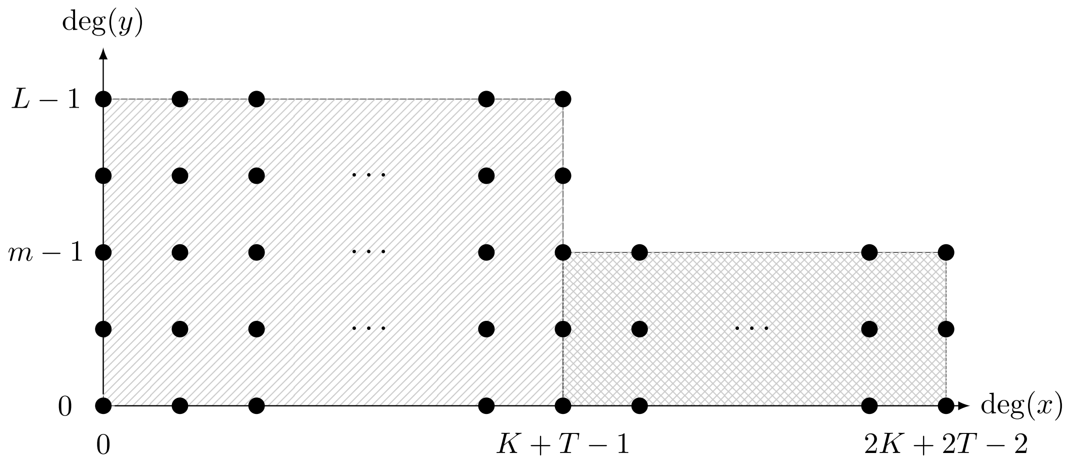


FIGURE 4.7: The visualization of the degrees of the monomials of $\mathbf{A}(x)\mathbf{B}(x, y)$ in the $\deg(x) - \deg(y)$ plane.

In Equation (4.31), we give an example interpolation matrix formed by any 5 workers, each of which provides $m = 2$ computations, when $K = L = 2$, $m = 2$ and $T = 1$. Observe that the first row represents the direct evaluation $\mathbf{A}(x_1)\mathbf{B}(x_1, y_1)$ from worker 1, and the second row represents $\mathbf{A}(x_1)\partial_1\mathbf{B}(x_1, y_1)$, again from worker 1. In general, any interpolation matrix formed by $R_{th} = 10$ computations received from any subset of workers is also valid, as long as the workers follow the computation order specified in Section Section 4.5.2.

$$\mathbf{M} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & y_1 & x_1y_1 & x_1^2y_1 & x_1^3y_1 & x_1^4y_1 \\ 0 & 0 & 0 & 0 & 0 & 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_5 & x_5^2 & x_5^3 & x_5^4 & y_5 & x_5y_5 & x_5^2y_5 & x_5^3y_5 & x_5^4y_5 \\ 0 & 0 & 0 & 0 & 0 & 1 & x_5 & x_5^2 & x_5^3 & x_5^4 \end{bmatrix}. \tag{4.31}$$

The problem of showing that any R_{th} responses from the workers interpolate to a unique polynomial is equivalent to showing that the corresponding interpolation matrix is non-singular. The theorem claims that this is the case with high probability. First, we need to show that there exist some evaluation points for which the determinant of the interpolation matrix is not zero. That is equivalent to showing that $\det(\mathbf{M})$ is not the zero polynomial of the evaluation points. In Chapter 3, such a result for the same type of interpolation matrices is shown for the real field $\mathbb{R}$. In this section, we extend this proof to $\mathbb{F}$. We show there exists some evaluation points such that $\det(\mathbf{M})$ is not zero by using the Taylor series expansion of $\det(\mathbf{M})$, in a similar manner done in Chapter 3, by using the fact that Taylor series expansion is also applicable in $\mathbb{F}$, as long as the degree of the polynomial $\mathbf{A}(x)\mathbf{B}(x, y)$ is smaller than the field size q. This can be guaranteed by choosing a large q. For further details on the applicability of Taylor series expansion in finite fields, we refer the readers to [77] and [78].

Without losing generality, let us assume first that n workers with $n \leq N$, provide, together, enough responses, i.e., as many as R_{th}, to interpolate $\mathbf{A}(x)\mathbf{B}(x, y)$. We further assume that (x_i, y_i) and (x_j, y_j) are two evaluation points at which the evaluations and derivatives of $\mathbf{A}(x)\mathbf{B}(x, y)$ are received by the master. We write the Taylor series expansion of $\det(\mathbf{M})$ by taking (x_i, y_i) as the pivot node and (x_j, y_j) as the variable node as

$$\det(\mathbf{M}) = \sum_{(\alpha_1,\alpha_2)\in\mathbb{N}^2} \frac{1}{\alpha_1!\alpha_2!}(x_j - x_i)^{\alpha_1}(y_j - y_i)^{\alpha_2} D_{\alpha_1,\alpha_2}(\tilde{Z}), \tag{4.32}$$

where $\tilde{Z} \triangleq \{(x_k, y_k) : k \in [1 : n]\} \setminus \{(x_j, y_j)\}$ and

$$D_{\alpha_1,\alpha_2}(\tilde{Z}) = \frac{\partial^{\alpha_1+\alpha_2}}{\partial x_j^{\alpha_1} \partial y_j^{\alpha_2}} \det(\mathbf{M})(x_j, y_j)\bigg|_{x_j=x_i, y_j=y_i}. \tag{4.33}$$

Following similar reasoning in Chapter 3, since the monomials $(x_j - x_i)^{\alpha_1}(y_j - y_i)^{\alpha_2}$ are linearly independent for different (α_1, α_2), if there is no relation between x and y coordinates of the evaluation points, i.e., x_i and x_j do not depend on y_i and y_j, $\det(\mathbf{M})$ is a zero polynomial of all evaluation points, if and only if $D_{\alpha_1,\alpha_2}(\tilde{Z}) = 0, \forall(\alpha_1, \alpha_2) \in \mathbb{N}^2$. Hence, in order to show that $\mathbf{M}$ is non-singular, it suffices to show that there exists at least one (α_1, α_2) making $D_{\alpha_1,\alpha_2}(\tilde{Z})$ nonzero.

Before showing that we define the notion of the unique shift in the following definition.

Definition 4.1. Recall from Lemma 3.1 that

$$\frac{\partial^{\alpha_1+\alpha_2}}{\partial x_j^{\alpha_1} \partial y_j^{\alpha_2}} \det(\mathbf{M})\bigg|_{x_j=x_i, y_j=y_i} = \sum_{(\boldsymbol{k},\boldsymbol{l})\in\mathcal{R}_M(\alpha_1,\alpha_2)} C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}) \det\left(\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}\right)\bigg|_{x_j=x_i, y_j=y_i}, \tag{4.34}$$

In this expression, if $\mathcal{R}_M(\alpha_1, \alpha_2)$ has only one element, i.e., there is only one $(\boldsymbol{k}, \boldsymbol{l})$ resulting in regular derivative paths, then (α_1, α_2) is called *unique shift order* and $(\boldsymbol{k}, \boldsymbol{l})$ is called a *unique shift.*

Note that the notion of unique shift is closely related to the notion of quasi-unique shift defined in Chapter 3 but more restrictive. While for an (α_1, α_2) pair to qualify as a unique shift order, it is required that there is only one possible $(\boldsymbol{k}, \boldsymbol{l})$ making $\det\left(\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}\right)\bigg|_{x_j=x_i,y_j=y_i} \neq 0$, it is sufficient that the derivative set of $\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j}\mathbf{M}\bigg|_{x_j=x_i,y_j=y_i}$ obeys N-zig-zag order to qualify the pair as a quasi-unique shift.

To show that there exists at least one (α_1, α_2) making $D_{\alpha_1,\alpha_2}(\tilde{Z})$ nonzero, first, observe that $D_{\alpha_1,\alpha_2}(\tilde{Z})$ in Equation (4.32) does not depend on (x_j, y_j) since after taking the derivatives with respect to (x_j, y_j), the resulting expression is evaluated at $x_j = x_i, y_j = y_i$. If (α_1, α_2) is a unique shift order, implying $|\mathcal{R}_M(\alpha_1, \alpha_2)| = 1$, then we can write Equation (4.34) as

$$\frac{\partial^{\alpha_1+\alpha_2}}{\partial x_j^{\alpha_1} \partial y_j^{\alpha_2}} \det(\mathbf{M}) \bigg|_{x_j=x_i, y_j=y_i} = C_{\boldsymbol{k},\boldsymbol{l}}(\mathbf{M}) \det \left(\nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j} \mathbf{M} \right) \bigg|_{x_j=x_i, y_j=y_i}, \tag{4.35}$$

such that $(\boldsymbol{k}, \boldsymbol{l})$ is the only element of $\mathcal{R}_M(\alpha_1, \alpha_2)$. If we define $\mathbf{M}_1 \triangleq \nabla_{\boldsymbol{k},\boldsymbol{l}}^{x_j,y_j} \mathbf{M} \;|_{x_j=x_i, y_j=y_i}$, then it is enough to show that $\det(\mathbf{M}_1)$ is not the zero polynomial. Notice that, $\mathbf{M}_1$ no longer depends on the evaluation point (x_j, y_j). We call such a procedure of transforming an interpolation matrix into another interpolation matrix via unique shifts as the coalescence of the variable node and the pivot node. After obtaining $\mathbf{M}_1$, we can employ the same idea to show $\mathbf{M}_1$ is non-singular. Namely, we can write the Taylor series expansion of $\det(\mathbf{M}_1)$ by choosing a new variable node and keeping the same pivot node (x_i, y_i). If there is a unique shift for this new coalescence, the resultant matrix $\mathbf{M}_2$ will not depend on neither the previous variable node (x_j, y_j) nor the current variable node. We can apply such coalescences successively as long as we can find a unique shift order (α_1, α_2) at each coalescence, until $\mathbf{M}_{\text{final}}$ depends only on one evaluation point, which is the pivot node, (x_i, y_i). In $\mathbf{M}_{\text{final}}$, the derivative set of (x_i, y_i) has all possible elements of the derivative order space of $\mathbf{A}(x)\mathbf{B}(x, y)$. Thus, $\mathbf{M}_{\text{final}}$ is a triangular matrix, implying its non-singularity.

To summarize, to prove that all possible interpolation matrices, $\mathbf{M}$, generated from our coding scheme and the corresponding computation order at the workers, are non-singular in general, we need to show that we can always find at least one unique shift for all the steps of the coalescence procedure. Our strategy to show that is based on the idea of keeping the derivative set of the pivot node to be a lower set. A lower set is defined as a set in which the presence of an element implies the presence of all possible elements smaller than this element. To decide if an element is smaller than any other element, we need to define an ordering rule. For our case, we define such an ordering as follows. Assume we denote our pivot node as $z_i = (x_i, y_i)$ and take two derivative orders $(a, b) \in \mathcal{U}_{z_i,M}$ and $(c, d) \in \mathcal{U}_{z_i,M}$, where a and c are the orders of the derivative with respect to x and b and d are the orders of the derivative with respect to y. We say $(a, b) < (c, d)$ if and only if $a < c$ or $a = c$ and $b < d$.

Before its formal statement, we illustrate our strategy to find a unique shift in all the coalescence steps in the following two examples.

Example 4.1. Assume $K = L = 5$, $T = 1$ and $m = 3$ and we are at the beginning of p-th coalescence step. Let us choose z_i as the pivot node and z_j as the variable node. Further, assume at the beginning we have $\mathcal{U}_{z_i,\mathbf{M}_{p-1}} = \{(a, b) : (a, b) \leq (1, 2)\}$

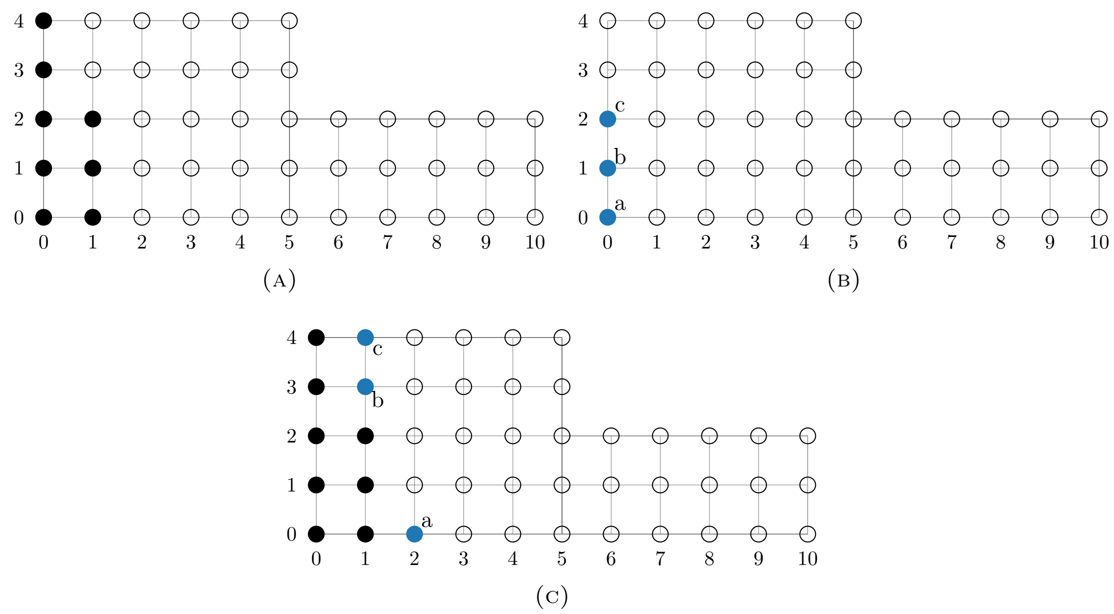


FIGURE 4.8: Depictions of the derivative sets in Example 4.1.

and $\mathcal{U}_{z_j,\mathbf{M}_{p-1}} = \{(a,b) : (a,b) \leq (0,2)\}$. We depict the derivative sets of $\mathcal{U}_{z_i,\mathbf{M}_{p-1}}$ and $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ in Figure 4.8a and Figure 4.8b. We will take smallest possible shift (α_1, α_2) such that the resultant $\mathcal{U}_{z_i,\mathbf{M}_p}$ after the coalescence is a lower set. Knowing the number of elements in $\mathcal{U}_{z_i,\mathbf{M}_p}$ after the coalescence, its shape is uniquely determined under the condition that it must be a lower set and we depict this set in Figure 4.8c. In Figure 4.8b and Figure 4.8c, we assign to each element of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ either the letter a, b or c so that we can track its location during and after the coalescence procedure. Recall that taking derivatives corresponds to shifting the elements of the derivative set in the derivative order space. Thus, in order to shift the elements of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ to their locations in the final shape in Figure 4.8c, we need to the total number of shifts in both x and y directions is 4, implying we need to choose $(\alpha_1, \alpha_2) = (4, 4)$. For this choice, we have $\boldsymbol{k}(i_a) = 2$, $\boldsymbol{k}(i_b) = 1$, $\boldsymbol{k}(i_c) = 1$, $\boldsymbol{l}(i_a) = 0$, $\boldsymbol{l}(i_b) = 2$ and $\boldsymbol{l}(i_c) = 2$ where i_a, i_b and i_c are the row-index of the elements a, b and c, respectively. Given this choice of (α_1, α_2), there is no other possible resulting shape for $\mathcal{U}_{z_i,\mathbf{M}_p}$ resulting a non-singular $\mathbf{M}_p$. To see this, observe that, if we write the derivative sets of $\mathcal{U}_{z_i,\mathbf{M}_p}$ after-the-coalescence for all possible $(\boldsymbol{k}, \boldsymbol{l})$ such that $\sum_i \boldsymbol{k}(i) = \alpha_1$ and $\sum_i \boldsymbol{l}(i) = \alpha_2$, then all, except the one depicted in Figure 4.8c will have overlapping elements making the corresponding interpolation matrix singular. Therefore, $(\alpha_1, \alpha_2) = (4, 4)$ is a unique shift order.

Example 4.2. Let us consider the same setting in Example 4.1, except $\mathcal{U}_{z_i,\mathbf{M}_{p-1}} = \{(a,b) : (a,b) \leq (6,1)\}$. Since the maximum number of computations a worker can provide is $m = 3$, the cardinality of the derivative set of the variable node $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$, in this example, is at its maximum. Thus, we can directly follow the same procedure as in Example 4.1. Note that after the coalescence, $\mathcal{U}_{z_i\mathbf{M}_p}$ will have 34 elements, and the lower

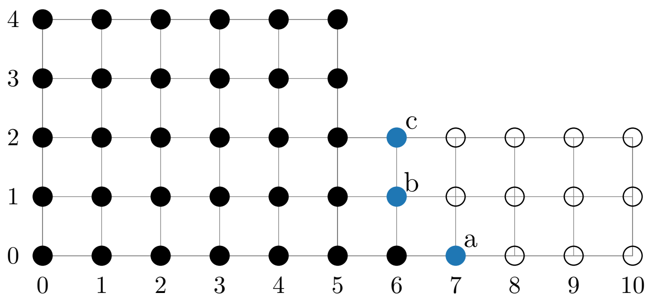


FIGURE 4.9: Depiction of $\mathcal{U}_{z_i,\mathbf{M}_p}$ after coalescence in Example 4.2

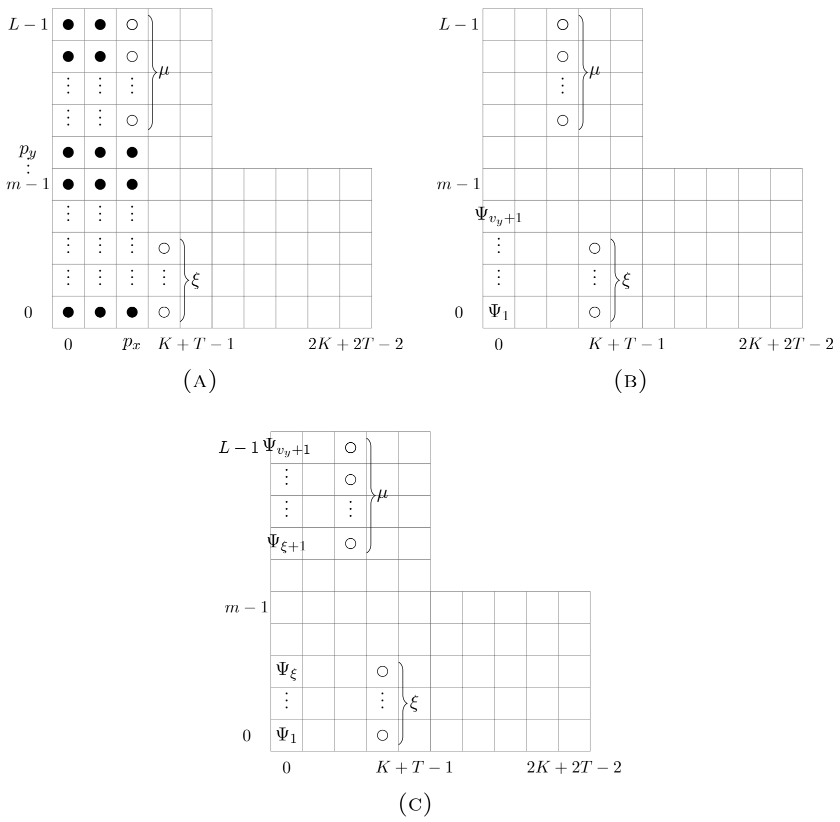


FIGURE 4.10: Visualization of the pivot node and the variable node during a coalescence

set having 34 elements is unique and well-defined. To obtain the shape in Figure 4.9, we need $(\alpha_1, \alpha_2) = (0, 19)$ with $\boldsymbol{k}(i_a) = 7$, $\boldsymbol{k}(i_b) = 6$ and $\boldsymbol{k}(i_c) = 6$, and it is a unique shift order since any other assignment of 19 shifts to a, b and c results in a non-singular $\mathbf{M}_p$.

Next, we formally state our strategy for an arbitrary coalescence step p. Since we choose one pivot node and use it for every coalescence step, we guarantee that the variable node's derivative set has always at most m elements. To generalize the procedure in Example 4.1 and Example 4.2, let us assume (p_x, p_y) is the largest element of the derivative set of the pivot node z_i, i.e., $\mathcal{U}_{z_i,\mathbf{M}_{p-1}} = \{(a, b) : (0, 0) \leq (a, b) \leq (p_x, p_y)\}$ and $(0, v_y)$ is the largest element of the derivative set of the variable node z_j, i.e., $\mathcal{U}_{z_j,\mathbf{M}_{p-1}} = \{(0, b) : 0 \leq b \leq v_y\}$

such that $v_y \leq m$. While calculating (α_1, α_2) pair, we first determine α_2, which is the total derivative order with respect to y_j, or equivalently the number of shifts towards y-direction in the derivative order set. This means that we first take the derivatives with respect to y_j, and then with respect to x_j. In Figure 4.10a, in the derivative order space, for $p_x \leq K + T - 1$, we depict the derivative set of the pivot node, i.e., $\mathcal{U}_{z_i,\mathbf{M}_{p-1}}$, by filled circles, and the locations to which the elements of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ will be placed after the coalescence by unfilled circles. Note that we determine these locations by inserting the elements of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ into $\mathcal{U}_{z_i,\mathbf{M}_{p-1}}$ such that the derivative set of the pivot after the coalescence, i.e., $\mathcal{U}_{z_i,\mathbf{M}_p}$, satisfies the lower set property. In Figure 4.10b, instead of the elements of $\mathcal{U}_{z_i,\mathbf{M}_{p-1}}$, we depict the elements of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ together with the locations they will be placed after the coalescence to facilitate visualization of the necessary shifts. To be able to keep track of the elements, we depict each one of them by Ψ_i, $i \in [1 : v_y + 1]$. We denote the number of elements in $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ to be shifted towards y-direction, by μ. We further define $\xi \triangleq v_y + 1 - \mu$. As shown in Figure 4.10a, when the number of empty spaces in the rightmost partially occupied column of $\mathcal{U}_{z_i,\mathbf{M}_{p-1}}$, which is $L - p_y - 1$ is smaller than $|\mathcal{U}_{z_j,\mathbf{M}_{p-1}}| = v_y + 1$, μ becomes $\mu = L - p_y - 1$, since these spaces must be filled. Otherwise, to fill as many as spaces possible, all elements of the derivative set of the pivot node are shifted towards y-direction and μ becomes $v_y + 1$. Thus, $\mu = \min\{L - p_y - 1, v_y + 1\}$ if $p_x \leq K + T - 1$ as this is the case considered in the figures. On the other hand, when $p_x > K + T - 1$, the same logic also applies but the maximum number of elements that can be placed in a column in Figure 4.10a would be m instead of L. Thus, the expression for μ is modified as $\mu = \min\{m - p_y - 1, v_y + 1\}$, which is obtained by replacing L with m.

Next, recall that only regular simple shifts are considered for unique shifts. Thus, while taking y-directional derivatives, i.e., shifts towards y-direction in Figure 4.10b, the sequence of the elements in the y-axis does not change. For instance, Ψ_{v_y+1} stays always on top of the elements denoted by $\Psi_i, i \in [1 : v_y]$. If, for example, as a result of some shifts, Ψ_{v_y} is placed on top of Ψ_{v_y+1}, then this would be possible only if the element Ψ_{v_y} is located in the same location as Ψ_{v_y+1} at some point, and this would contradict the assumption of regular simple shifts. Hence, there is only one resulting order after shifting the uppermost μ elements towards the y-direction. We show the elements of the variable node's derivative set after y-directional shifts in Figure 4.10c. All the remaining shifts, now, are the ones towards the x-direction so that the elements of $\mathcal{U}_{z_j,\mathbf{M}_{p-1}}$ are located in their intended locations, i.e., unfilled circles in Figure 4.10c. Notice that each Ψ_i is already aligned with its final location in the y-direction, and hence, each one of them will be shifted towards the x-direction by a sufficient amount. Therefore, these shifts also result in a unique shape. From these observations, we can conclude that whenever the derivative sets of the pivot node and the variable node are lower sets, there exists a unique shift for their coalescence.

From this discussion, we conclude that $\det(\mathbf{M})$ is not a zero polynomial for large enough q. Next, we need to find the upper bound on the probability $\det(\mathbf{M}) = 0$, when the evaluation points are sampled uniform randomly from $\mathbb{F}$.

Lemma 4.1. [79, Lemma 1] *Assume P is a non-zero, v-variate polynomial of variables $\alpha_i, i \in [1:v]$. Let d_1 be the degree of α_1 in $P(\alpha_1,\dots,\alpha_v)$, and $P_2(\alpha_2,\dots,\alpha_v)$ be the coefficient of $\alpha_1^{d_1}$in $P(\alpha_1,\dots,\alpha_v)$. Inductively, let d_j be the degree of α_j in $P_j(\alpha_j,\dots,\alpha_v)$ and $P_{j+1}(\alpha_{j+1},\dots,\alpha_v)$ be the coefficient of α_j in $P_j(\alpha_j,\dots,\alpha_v)$. Let $\mathbf{S}_j$ be a set of elements from a field $\mathbb{F}$, from which the coefficients of a_j's are chosen. Then, in the Cartesian product set $\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_v$, $P(\alpha_1,\dots,\alpha_v)$ has at most $|\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_v| \left(\frac{d_1}{|\mathbf{S}_1|} + \frac{d_2}{|\mathbf{S}_2|} + \cdots + \frac{d_v}{|\mathbf{S}_v|}\right)$ zeros.*

In our case, $\det(\mathbf{M})$ is a multivariate polynomial of the evaluation points (x_i, y_i) since the elements of $\mathbf{M}$ are the monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$ and their derivatives with respect to y, evaluated at some (x_i, y_i). Thus, in our case, v in Lemma 4.1 is the number of different evaluation points in $\mathbf{M}$. We choose the evaluation points from the whole field $\mathbb{F}$. Thus, $\mathbf{S}_j = \mathbb{F}$ and $|\mathbf{S}_j| = q, \forall j \in [1:v]$, and $|\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_v| = q^v$. Then, the number of zeros of $\det(\mathbf{M})$ is at most $q^{v-1}(d_1 + d_2 + \cdots + d_v)$. If we sample the evaluation points uniform randomly, then the probability that $\det(\mathbf{M}) = 0$ is $(d_1 + d_2 + \cdots + d_v)/q$, since we sample a v-tuple of evaluation points from $\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_v$. To find $d_1 + d_2 + \cdots + d_v$, we resort to the definition of determinant, that is $\det(\mathbf{M}) = \sum_{i=1}^{R_{th}} (-1)^{1+i} m_{1,i} \det(\mathbf{M}_{1,i})$, where $m_{1,i}$ is the element of $\mathbf{M}$ at row 1 and column i and $\mathbf{M}_{1,i}$ is the minor of $\mathbf{M}$ when row 1 and column i are removed [73, Corollary 7.22]. Thus, to identify the coefficients in Lemma 4.1, in the first row of $\mathbf{M}$, we start with the monomial with the largest degree. Assuming the monomials are placed in an increasing order of their degrees, the largest degree monomial is at column R_{th}. If that monomial is univariate, then d_1 is the degree of the monomial and the coefficient of $\alpha_1^{d_1}$ is $P_2(x_2,\dots,x_v) = \det(\mathbf{M}_{1,R_{th}})$. If the monomial is bivariate, then we take the degree of the corresponding evaluation of x, i.e., α_1, as d_1, and the degree of the corresponding evaluation of y, i.e., α_2, as d_2. In this case, the coefficient of α^{d_2} is $P_3(\alpha_3,\dots,\alpha_v) = \det(\mathbf{M}_{1,R_{th}})$. Next, we take $\mathbf{M}_{1,R_{th}}$, and repeat the same procedure. We do so until we reach a monomial of degree zero. In this procedure, since we visit all the monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$ evaluated at different evaluation points, i.e., α_i's, the sum $d_1 + d_2 + \cdots + d_v$ becomes the sum of degrees of all the monomials of $\mathbf{A}(x)\mathbf{B}(x,y)$. The next lemma helps us in computing this.

Lemma 4.2. *Consider the polynomial $P(x,y) = \sum_{i=0}^{a}\sum_{j=0}^{b} c_{ij}x^i y^j$, where $c_{i,j}$'s are scalars. The sum of the degrees of all the monomials of $P(x,y)$ is given by $\xi(a,b) \triangleq \frac{1}{2}(a+1)(b+1)(a+b)$.*

Proof. The sum of the degrees of all the monomials is given by

$$\begin{aligned}\sum_{i=0}^{a}\sum_{j=0}^{b}(i+j) = \sum_{i=0}^{a} i(b+1) + \sum_{i=0}^{a}\sum_{j=0}^{b} j &= \frac{a(a+1)}{2}(b+1) + \frac{b(b+1)}{2}(a+1) \\ &= \frac{1}{2}(a+1)(b+1)(a+b). \end{aligned} \tag{4.36}$$

□

To calculate the sum of degrees of $\mathbf{A}(x)\mathbf{B}(x,y)$, we use the depiction in Figure 4.7. By using Lemma 4.2, the sum of monomial degrees in the diagonally shaded rectangle in Figure 4.7 is

$$\xi(K+T-1, L-1) = \frac{1}{2}L(K+T)(K+L+T-2). \tag{4.37}$$

Moreover, the sum of monomial degrees in the rectangle shaded by crosshatches is given by

$$\begin{aligned}&\xi(2K+2T-2, m-1) - \xi(K+T-1, m-1) \\ &\quad = \frac{1}{2}m(2K+2T-1)(2K+2T+m-3) - \frac{1}{2}m(K+T)(K+T+m-2) \\ &\quad = \frac{m}{2}\left(3(K+T)^2 + m(K+T) - 6K - 6T - m + 3\right). \end{aligned} \tag{4.38}$$

By summing them we obtain $d_1 + d_2 + \cdots + d_v = \frac{m}{2}\left(3(K+T)^2 + m(K+T) - 6K - 6T - m + 3\right) + \frac{(K+T)L}{2}(K+L+T-2)$, which concludes the proof.

4.9 Conclusion

In this chapter, for straggler exploitation and security in distributed matrix multiplication, we have proposed a storage- and upload-cost-efficient bivariate polynomial coding scheme named SBP codes. Although the previous works usually assume the availability of at least as many workers as the recovery threshold, the multi-message approach allows the completion of the task even if the number of workers is less than the recovery threshold. Compared to univariate polynomial coding-based approaches including MM-GASP codes, SBP coding scheme has a lower upload cost and less storage requirement, making the assignment of several sub-tasks to each worker more resource efficient. Thanks to these properties, SBP codes considerably improve the average computation time for SDMM, especially when the number of workers, the upload cost budget, or the storage capacity is limited.

Chapter 5

Private Wireless Federated Learning with Anonymous OAC

5.1 Abstract

In conventional FL, DP guarantees can be obtained by injecting additional noise to local model updates before transmitting to the PS. In the wireless FL scenario, we show that the privacy of the system can be boosted by exploiting over-the-air computation (OAC) and anonymizing the clients. In OAC, clients transmit their model updates simultaneously and in an uncoded fashion, resulting in a much more efficient use of the available spectrum. We further exploit OAC to provide anonymity for the transmitting clients. The proposed approach improves the performance of private wireless FL by reducing the amount of noise that must be injected.

5.2 Introduction

FL [18] allows multiple clients, each with its own local dataset, to train a model collaboratively with the help of a PS without sharing their datasets. At each iteration of FL, the PS shares the most recent global model with the clients, which then use their local datasets to update the global model. Local updates are aggregated and averaged at the PS to update the global model. The fact that data never leaves the clients is considered to protect its privacy. However, many recent works including [37,38,40] show that model updates or gradients also leak a lot of information about the dataset. This calls for additional mechanisms to guarantee privacy. In this chapter, we consider DP, as it is a widely-adopted rigorous notion of privacy [53]. Given an algorithm which outputs some statistics about a dataset, if the change in the output probability distribution is tolerably small when the input database is changed with a very close neighbouring one, then the algorithm is deemed as differentially private. Many recent works exploit DP-based algorithms to provide rigorous privacy guarantees in machine learning [80–85].

We consider a FL setting with co-located clients communicating with the PS over a wireless multiple access channel (MAC). Recent work has shown that rather than conventional digital communication, the clients can transmit their local updates simultaneously in an uncoded fashion to enable OAC [86–88]. This results in a much more efficient use of the available resources, and significantly improves the learning performance. It is also remarked in [87] that the analog computation in a wireless domain makes privacy more easily attainable. A common method to provide DP guarantees is adding noise to the output with a variance proportional to the maximum change in the output under neighbouring datasets [53]. In the digital implementation of FL, where each client separately communicates with the PS, noise is separately added to each client's gradients in every local iteration [83] to ensure DP; whereas in analog computation, the PS only sees the sum of the updates together with the channel noise, which effectively protects all the updates. Thus, the same amount of protection can be achieved with less perturbation, improving the final accuracy.

Privacy in wireless FL through OAC has been recently studied in several works [89–92]. In [89], distributed SGD is studied with quantized gradients and privacy constraints. In [90], if the channel noise is not sufficient to satisfy the target DP guarantees, a subset of the clients add additional noise to their updates, benefiting all the clients. In [91] and [92], transmit power is adjusted for the same privacy guarantee. We note that, in [90–92], estimating channel state information (CSI) from the clients to the PS is for aligning their computations at the PS and ensuring the target privacy guarantee. However, in practice, clients estimate CSI from pilots transmitted by the PS. Hence, as an adversary, the PS can adjust the pilots to degrade the privacy level. Additionally, in [90], clients depend on others for privacy, which introduces additional point-of-failure.

Differently from the works cited above, our aim in this chapter is to achieve DP guarantees in wireless FL by exploiting the anonymity of the transmitters via OAC. Our main contributions can be summarized as follows.

- We study the effects of randomly sampling the clients and batching, on privacy. By forcing a constant receive power at the PS across iterations, we provide anonymity to the transmitting clients, and employ it for privacy.
- By distributing the noise generation process amongst the workers, we make the privacy guarantee resilient against the failure of transmitting nodes.

- We ensure that the proposed protocol is robust against pilot attacks.

5.3 System Model

We consider N wireless clients. Each client $i \in [1:N]$ hosts their local dataset $\mathcal{D}_i$, which are i.i.d. across the clients. A PS orchestrates these clients to learn a global model $\boldsymbol{w}$ by minimizing a global loss function $\mathcal{L} = \frac{1}{N}\sum_{i\in[1:N]}\frac{1}{|\mathcal{D}_i|}\sum_{d\in\mathcal{D}_i}\ell(\boldsymbol{w},d)$, where $\ell(\boldsymbol{w},d)$ is an application-specific empirical loss function. We employ distributed SGD to iteratively minimize $\mathcal{L}$ across the clients. At the beginning of iteration t, the PS broadcasts the global model $\boldsymbol{w}_t$ to all the clients. Then, the clients participating in the learning process in that iteration transmit their gradient estimates over a Gaussian MAC. In round t, the PS receives the superposition of the signals transmitted by the participating clients as

$$\boldsymbol{y}[t] = \sum_{i\in\mathcal{A}_t} c_{i,t}\boldsymbol{x}_i[t] + \boldsymbol{z}[t],$$

where $\boldsymbol{z}[t] \sim \mathcal{N}(0, N_0\boldsymbol{I})$ is the channel noise, $\mathcal{A}_t$ is the set of participating clients, $\boldsymbol{x}_i[t]$ is the signal transmitted by client i, and $c_{i,t}$ is the channel coefficient from client i to the PS. We assume that the transmitters perfectly know and correct the phase shift in their channels. Therefore, for simplicity, we assume real channel gains in the rest of the chapter, i.e., $c_{i,t} \in \mathbb{R}^+$.

We assume that $c_{i,t}$ is independent across the users and rounds, but remains constant within one round. We further assume that only the agents with sufficiently good channel coefficients participate in each round to increase power efficiency. We assume this happens with probability $p \in (0,1]$ for each client, independently of other clients. We leave the analysis of this probability for different fading distributions as future work. If the client i participates in a round, then it samples uniformly from its own local dataset such that every data point is sampled independently from the other samples with probability $q \in (0,1]$. Let $\mathcal{B}_{i,t}$ denote the set of samples in the batch used by the i^{th} client if it is participating in round t. Let $a_t \triangleq |\mathcal{A}_t|$ and $b_{i,t} \triangleq |\mathcal{B}_{i,t}|$. Both a_t and $b_{i,t}$ are binomial random variables with parameters p and q.

Each participating client computes a gradient estimate based on the random samples in its batch. We clip gradients such that L_2-norm of every *per-sample* gradient does not exceed a threshold C. That is, as done in [83], for a sample d, the gradient for that sample is calculated as

$$\boldsymbol{g}_t(d) \triangleq \nabla\ell(\boldsymbol{w}_t,d) \times \max\left\{1, \frac{C}{\|\nabla\ell(\boldsymbol{w}_t,d)\|_2}\right\}. \tag{5.1}$$

After gradients are calculated and clipped, client i transmits

$$\boldsymbol{x}_i[t] = h_{i,t}\left(\xi_{i,t}\sum_{d\in\mathcal{B}_{i,t}}\boldsymbol{g}_t(d)+\beta_{i,t}\boldsymbol{n}_{i,t}\right), \tag{5.2}$$

where $\boldsymbol{n}_{i,t}\sim\mathcal{N}(0,\sigma_{i,t}^2\boldsymbol{I})$, $\xi_{i,t}$ and $\beta_{i,t}$ are scaling factors, and $h_{i,t}\triangleq(\tilde{c}_{i,t})^{-1}$. $\tilde{c}_{i,t}$ is the CSI which can be manipulated by the PS by a multiplicative factor of $k\in(0,1]$, i.e., $\tilde{c}_{i,t}=k\cdot c_{i,t}$.

For brevity, throughout this chapter, we assume that the local datasets are of the same size, although our analysis can be easily extended by adding another scaling factor to the gradient if this is not the case.

After receiving the signals simultaneously transmitted by the clients, using the received signal $\boldsymbol{y}[t]$, the PS updates the model as $\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\eta\boldsymbol{y}[t]$, where η is the learning rate. We assume that the number of participating clients a_t and the batch sizes $b_{i,t}$ are known to the clients but not to the PS. This can be achieved by keeping a common random number generator seed across the clients, or alternatively by encrypted communication between the clients. Since this is only sharing two real numbers, the communication overhead is negligible.

5.3.1 Threat Model

We assume that the PS is semi-honest and curious. It is honest in the sense that it wants to correctly acquire the final trained model, and thus, follows the model averaging step accurately, but it also would like to acquire all possible sensitive information about single individual data points. Therefore, it can attack the CSI estimation process to encourage the clients to increase their transmit power or to add less noise to their transmissions by suggesting that their channel quality is worse than it is in reality. We note that, if a common pilot signal is used to learn the CSI by all the clients, the CSI values can only be scaled by the same parameter across the clients.

We assume that the clients are honest and trusted. They do not intentionally attack the learning process; however, we also do not depend on a few clients for privacy as the clients may fail to transmit unintentionally. We further assume that the PS has only one receiving antenna, due to which it is unable to determine the spatial directions of the incoming signals. Consequently, the PS cannot differentiate between the sources of the received signals based on the transmission direction. In case the PS server is unable to determine whether a client is active or not during a specific time, the client is considered *anonymous* to the PS for that corresponding round.

5.4 Main Results

As we discussed in the previous section, we sample the participating clients and the local datasets to be used in each iteration. It is well known that the privacy of randomized mechanisms can be amplified via subsampling [52, 93, 94]; however, these results are for the centralized setting. To take advantage of sampling in the distributed setting, we make sure that all of the data samples are chosen independently and uniformly with probability pq. However, the challenge is that since the local datasets of different clients are distinct, the conditional probability of a data point being chosen given some other data point is already sampled may not be the same as the marginal probability of the same event. For example, if two data points are hosted by the same client, then the conditional probability of the second data point being chosen given the first point is already chosen is q, instead of pq. One way of overcoming this problem is shuffling the whole dataset across the clients after each iteration to cancel the effect of locality. However, this would incur a large communication cost. A better way is to exploit the wireless nature of the protocol as we explain next.

5.4.1 Ensuring the anonymity of clients

Since the PS can only see the aggregated signal, it lacks the information on which clients have transmitted. Still, if the PS knows the number of participating clients, then it can collect some information across the iterations and infer some dependency between the samples, such as two samples being hosted by the same client. In such cases, the i.i.d. assumption on the sampling probability of data points does not hold. Next, we show how we mask the number of participating clients.

Lemma 5.1. *If $\xi_{i,t} = 1/b_t, \forall i \in \mathcal{A}_t$, where $b_t \triangleq \sum_{i\in\mathcal{A}_t} b_{i,t}$, the PS cannot infer the number of clients actively transmitting in round t.*

Proof sketch: Since the clients scale their local gradients by b_t before transmitting, and the PS can only see the average gradient at each round, the received power level at the PS is independent of the number of the transmitting clients or their identities. Thus, by utilizing the information about the power level, it is not possible to infer any information about the number of transmitting clients.

Due to Lemma 5.1, knowing that any set of data is already sampled does not convey any information to the PS about the identities of the remaining sampled data points unless it learns the identities of the transmitting clients. Therefore, batching local samples independently with probability q and similarly sampling the transmitting clients independently with probability p is equivalent to subsampling in the centralized setting with probability pq.

5.4.2 Robustness to transmission failures

As we have stated, we want our scheme to be robust against transmission failures of clients that are scheduled to transmit at a certain iteration but failed to do so for some reason. This is achieved by distributing the noise generation across the clients.

Lemma 5.2. *If we choose $\beta_{i,t} = 1/\sqrt{a_t}$ and $\sigma_{i,t} = \hat{\sigma}_t, \forall i \in \mathcal{A}_t$, then the received signal at the PS becomes*

$$\boldsymbol{y}[t] = \sum_{i \in \mathcal{A}_t} \frac{1}{b_t} \sum_{j \in \mathcal{B}_{i,t}} \boldsymbol{g}_t(j) + \boldsymbol{n}[t] + \boldsymbol{z}[t], \tag{5.3}$$

where $\boldsymbol{n}[t] \sim \mathcal{N}(0, \sigma_t^2 \boldsymbol{I})$, and $\sigma_t = \hat{\sigma}_t$. When $k < a_t$ clients in $\mathcal{A}_t$ fail to transmit, the noise variance degrades to $\sigma_t = \hat{\sigma}_t \sqrt{(a_t - k)/a_t}$.

Proof sketch: If k clients fail, there are $a_t - k$ remaining clients transmitting. Since the noise added by the clients is independent and each has the variance $\hat{\sigma}_t^2/a_t$, we have $\sigma_t^2 = \hat{\sigma}_t^2(a_t - k)/a_t$. If there is no failure, $k = 0$, and we obtain $\sigma_t = \hat{\sigma}_t$.

As we see in Lemma 5.2, in case of transmission failures in $\mathcal{A}_t$, the variance of the total noise degrades gracefully, and so does the privacy. Hence, such transmission failures are not catastrophic.

5.4.3 Robustness to manipulated CSI values

In practice, given a channel noise variance N_0, we need to tune σ so that the total noise variance $N_0 + \sigma^2$ is sufficient to meet the desired privacy level, i.,e, (ε, δ)-DP. However, if we rely on the channel noise and our channel is better than we think, i.e., less N_0 than anticipated, then, the total noise variance may be less than it should be to guarantee the desired privacy levels. This results in larger ε and δ values than their target values, resulting in a possible privacy breach. To avoid this, while calculating the privacy guarantees, we simply ignore the channel noise since CSI values are prone to attacks by the PS. Then, the ε value we get is the upper bound of the real ε value in the presence of channel noise.

5.4.4 Privacy analysis

In the next theorem, we present a result which shows the boosting effect of sampling on the overall privacy.

Theorem 5.1. *Each round t in our FL scheme with OAC is $(\alpha, \frac{2p^2q^2\alpha}{\tilde{\sigma}_t^2})$-RDP, where $\tilde{\sigma}_t \triangleq \frac{\sigma_t b_t}{2C}$, if $pq \leq 1/5$, $\tilde{\sigma}_t \geq 4$ and the following inequalities are satisfied*

$$1 < \alpha \leq \frac{1}{2}\tilde{\sigma}_t^2 \log\left(1 + \frac{1}{pq(\alpha-1)}\right) - 2\log(\tilde{\sigma}_t), \tag{5.4}$$

and

$$\alpha \leq \frac{\frac{1}{2}\tilde{\sigma}_t^2 \log^2(1 + \frac{1}{pq(\alpha-1)}) - \log 5 - 2\log(\tilde{\sigma}_t)}{\log\left(1 + \frac{1}{pq(\alpha-1)}\right) + \log(pq\alpha) + \frac{1}{2\tilde{\sigma}_t^2}}. \tag{5.5}$$

Proof. Consider Equation (5.3) and define $\boldsymbol{f}[t] \triangleq \boldsymbol{y}[t] - \boldsymbol{n}[t] - \boldsymbol{z}[t] = \frac{1}{b_t}\sum_{i\in\mathcal{A}_t}\sum_{j\in\mathcal{B}_{i,t}} \boldsymbol{g}_t(j)$. Let us define $\Delta f \triangleq \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$, where $\mathcal{D}$ and $\mathcal{D}'$ are two datasets with the same size, differing only in one data point. For the same realization of random batching and client selection processes, the batch sizes $\mathcal{B}_{i,t}(\mathcal{D})$ and $\mathcal{B}_{i,t}(\mathcal{D}')$ also differ at most in one element. Since $\|\boldsymbol{g}_t(j)\|_2 \leq C$, we find $\Delta f = \frac{2C}{b_t}$. With Gaussian Mechanism (GM), according to Theorem 2.3, our mechanism is $(\alpha, \alpha(\Delta f)^2/(2\sigma_t^2))$-RDP without sampling. Moreover, by Lemma 5.1, the sampling probability of each sample is pq. The remaining of the lemma follows from the direct application of [94, Theorem 11]. □

It is worth noting that if the conditions in Theorem 5.1 are not satisfied, the RDP of the sampled Gaussian mechanism can still be computed numerically using the procedure given in [94, Section 3.3], and improvement due to sampling is still valid.

Process to compute (ε, δ)-DP after T iterations: According to Lemma 2.1, if the received message at round t is (α, ϵ_t)-RDP, then after T iterations, the mechanism is $\left(\alpha, \sum_{t\in[T]} \epsilon_t\right)$-RDP. According to Lemma 2.2, it is $(\sum_{t\in[T]} \epsilon_t + \log(1/\delta)/(\alpha-1), \delta)$-DP. We observe that although ϵ_t depends on the parameters in our mechanism, α does not. It is a parameter we choose to minimize $(\sum_{t\in[T]} \epsilon_t + \log(1/\delta)/(\alpha-1), \delta)$, which we compute for several α values and take the best α among them. Since both the analytical (if it exists) and the numerical computations of the composed (α, ε)-RDP is computationally cheap, this is feasible.

We note that employing RDP-based composition, we obtain a much tighter bound on the final ε value compared to the advanced composition theorem [53], which is used in [90, 92].

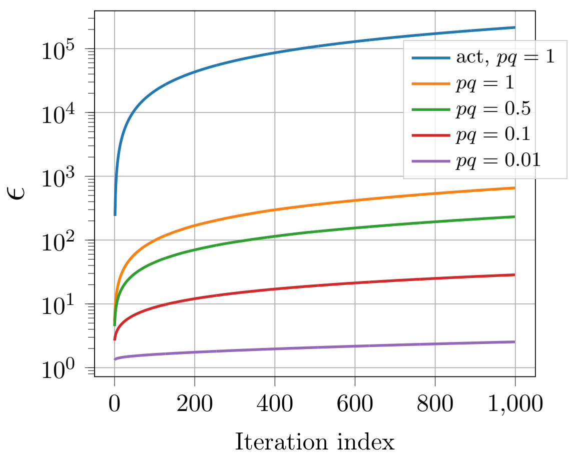


FIGURE 5.1: Total privacy measured by ε for different sampling rates across iterations.

5.5 Numerical Results

In this section, we numerically calculate the DP guarantees, i.e., ε values of a learning task for $\delta = 10^{-5}$, for different sampling rates given by pq. Note that $pq = 1$ means there is no sampling, implying all the clients participate and they use all their local datasets. We assume $\Delta f = 1$ and $\sigma_t = 1, \forall t \in [T]$. For $pq = 1$, we compute the total privacy spent, i.e., the composite ε by using both the advanced composition theorem [53, Theorem 3.20], which is denoted by 'act' in the figure, and the RDP approach given in Lemma 2.1. Since we numerically verify that RDP accounting performs much better for $pq = 1$, we use it to calculate the composite ε for all the other values of pq. For numerically computing the composition with RDP, we used Opacus library [95]. We tested α values from 1 to 64, and picked the one minimizing the composite ε value.

We observe in Figure 5.1 that for $pq = 1$ and $pq = 0.5$, the resultant ε values are very high, namely more than 100, which is quite a weak privacy level. Roughly speaking, it means that the output distribution of the final learned model may change up to a factor of e^{100} when only one element is changed in the dataset. However, when we have smaller sampling rates, the privacy guarantee comes into an acceptable range. In an edge network setting, we expect many clients to participate in learning, with substantial local datasets. Therefore, pq values on the order of 0.01 are not difficult to achieve.

5.6 Conclusion

In this chapter, we proposed a framework which utilizes anonymity in wireless transmissions to enable private wireless FL across edge clients. This is achieved by employing OAC instead of orthogonal transmissions, which would reveal the identity of the clients.

In particular, we exploit random subsampling of both the clients and the local data samples to achieve reasonable DP guarantees. As opposed to recent works on the topic, we do not depend on the channel gain or the noise at the PS for privacy guarantees as these values are prone to attacks; although our method is orthogonal to these techniques and can be combined with them. Our results demonstrate yet another favourable property of OAC in wireless edge learning.

Chapter 6

Privacy Amplification via Random Participation in Federated Learning

6.1 Abstract

Running a randomized algorithm on a subsampled dataset instead of the entire dataset amplifies differential privacy guarantees. In this work, in a federated setting, we consider the random participation of the clients in addition to subsampling their local datasets. Since such random participation of the clients creates a correlation among the samples of the same client in their subsampling, we analyze the corresponding privacy amplification via non-uniform subsampling. We show that when the size of the local datasets is small, the privacy guarantees via random participation are close to those of the centralized setting, in which the entire dataset is located in a single host and subsampled. On the other hand, when the local datasets are large, observing the output of the algorithm may disclose the identities of the sampled clients with high confidence. Our analysis reveals that, even in this case, privacy guarantees via random participation outperform those via only local subsampling.

6.2 Introduction

FL framework allows several clients to collaboratively and iteratively learn from each other's data with the coordination of PS [18]. In a typical scenario, the PS averages the model updates received from clients based on their local data. Then, it updates the global model and broadcasts the updated global model back to clients for the next iteration. In this learning model, since the data itself never leaves the participating clients, in terms of privacy, the FL framework is usually perceived as superior to centralized training, in which entire data is offloaded to a central server. Although this is true to some extent, as discussed in Chapter 2 and Chapter 5, many recent works [37–41] have shown that the model updates from the clients, as well as the final deployed model, can leak many features of the clients' local datasets, including reconstruction of some data samples used for training. Therefore, additional mechanisms with formal and quantifiable privacy guarantees need to be employed in FL.

Hence, as the gold standard quantifying the privacy leakage in privacy-preserving data analysis tasks, in this chapter, we consider the notion of DP to quantify privacy guarantees. A deterministic algorithm can be made differentially private by randomizing its output as long as the introduced randomness is independent of the input and secret to the adversary. Output perturbation via an additive noise such as Gaussian or Laplace noise [47] is a common example of such randomization techniques. However, there exist other sources of randomness that do not guarantee indistinguishability alone, but they can amplify the final DP guarantees when they are cascaded with techniques already guaranteeing DP. *Subsampling* [52, 93, 94, 96] and *shuffling* [97, 98] are among the most prominent of such tools. In subsampling, a random subset from the original dataset is sampled to be used as the input to the privacy-preserving mechanism. In shuffling, on the other hand, the outputs resulting from different inputs are randomly shuffled so that mapping between the inputs and the outputs is masked, resulting in anonymized outputs. Such privacy amplification techniques further confuse an adversary when used in conjunction with output perturbation techniques, and the amplified privacy guarantees are achieved without sacrificing utility.

In this chapter, we consider a FL setting with distributed SGD using a trusted PS, in which formal DP guarantees are achieved for all intermediate models including the final deployed one. For this, we update the global model by the average of the gradients collected from participating clients, which is perturbed via an additive Gaussian noise, following the seminal work of Abadi et al. [83].

To further amplify the privacy guarantee, we consider two types of sampling, which are client sampling and local dataset sampling. In each iteration, first, available clients during that iteration randomly decide whether to participate or not, independently from the decisions of other clients. Then, each client that decides to participate samples a subset from its local dataset such that each element is sampled independently from the other elements in the local dataset. Such a sample technique, in which every element is sampled independently and with identical probability, is called *Poisson sampling.* Hence, while sampling both the participating clients and their local datasets, we employ Poisson sampling. Analysing the privacy amplification guarantee of this sampling technique in a federated setting is challenging. More specifically, the random participation of the clients introduces a correlation between the elements located in a single client in their sampling. For example, given an element is sampled from a specific client, the probability of another element being sampled from the same client is larger than the probability of another element from another client being sampled. Therefore, employing random participation of clients and local dataset sampling together poses a non-uniform sampling problem, and in this work, we analyse the central DP guarantees of such a setting.

Note that this problem is closely related to the one discussed in Chapter 5, where we assume that when the PS does not know the number of sampled clients, it is not able to infer the correlation between the sampled examples. This is due to the anonymity provided by OAC. However, such an assumption may be unrealistic for some settings considering OAC may require special hardware to realize the analog and simultaneous transmission of gradients. Moreover, note that the setting in Chapter 5, we do not trust the PS, while in the setting in this chapter, we assume a trusted PS.

6.2.1 Related Work and Motivation

For FL scenarios with many participants, sampling the participating clients is essential for efficient use of the power and the communication resources even when privacy is not a constraint. Consider a learning problem with millions of mobile devices. Thousands of iterations are carried out in a typical learning task, and if each mobile device participates in every iteration, the power spent on the computations and the extra communication to transmit the results to the PS will result in a very large overall energy cost. In addition, the communication overhead of such a dense participation may cause congestion in the communication networks; and thus, is not scalable. Moreover, for the learning tasks employing mini-batch optimization techniques, a limited number of samples per iteration is preferable. Hence, client sampling can help to reach the targeted batch size in addition to the local dataset sampling. All these aforementioned benefits make client sampling one of the indispensable components of FL, whose effect on the convergence and final performance of FL algorithms have been widely studied [99–101]. In addition to these benefits, since it brings additional randomness to the training procedure, it is expected to bring some additional amplification to the DP guarantees. Hence, rather than proposing the inclusion of a new mechanism to amplify DP guarantees, in this chapter, our goal is to utilize an already-employed mechanism for efficiency purposes for also improving the privacy guarantees of the task.

Differentially private deep learning in a centralized setting is studied in [83]. The authors employ Poisson subsampling of the dataset and the gradient of every sample is clipped and Gaussian noise is added to it. They further introduce the moments' accountant technique to calculate the composition of the total privacy leakage throughout the iterations. Similar techniques to those in [83] are extended to the federated setting in [84, 85, 90]. In these works, similarly to our approach, client sampling is considered, but the achieved DP guarantees are at the client-level, which means the participation of a client is indistinguishable from its non-participation. Although client-level DP guarantees are meaningful when the local dataset is composed of elements from the same source, e.g., personal photos stored in a mobile phone, in some cases, it might be too conservative. To achieve DP guarantees protecting all the elements in a client, a larger amount of noise must be introduced compared to sample-level guarantees, and hence, client-level guarantees result in a larger loss in utility. Moreover, in some cases, the datasets stored by the clients may be comprised of samples from different individuals, and hence, it may not be necessary to protect client-level privacy. For instance, while learning from medical data, in a cross-silo setting, each client may represent a hospital and sample-level DP guarantees would suffice to protect the privacy of individual patients similar to the setting considered in [23].

Sample-level DP in the federated setting with client sampling is studied in [1, 102, 103] together with shuffling. In the analysis carried out in [1], each client is assumed to store only one sample, in which case the client-level guarantees are equivalent to sample-level guarantees. While each client is allowed to store more than one sample in [102, 103], the number of sampled elements is determined in advance. In [103], only one element is sampled from the local dataset at each iteration, while more than one but a constant number of elements are sampled in [102]. It is also worth noting that different from [1] and the setting of this chapter, in [102,103], communication efficiency is studied together with privacy, and a random compression mechanism is used as the randomizer. In all of these works [1, 102, 103], the clients are assumed to employ a local randomizer that satisfies pure DP, and the privacy analysis is based on shuffling the local responses. Such an assumption of pure DP for the local randomizers turns out to be useful while jointly analysing client sampling and shuffling. However, if the local randomizers provide approximate DP guarantees instead of pure DP, which is the case when Gaussian noise is used as the randomizer, central privacy guarantees may be substantially degraded. Therefore, novel methods to jointly analyze privacy amplification via client sampling and local dataset sampling that does not rely on shuffling are needed, and this constitutes the main motivation and contribution of our work.

6.2.2 Our Contributions

The major contributions of our work presented in this chapter and its novelty with respect to the current literature can be summarized as follows:

- We propose a client sampling algorithm for privacy amplification in FL. Each available client randomly decides to participate or not at each iteration. Having decided to participate, each client then samples its local dataset randomly.
- Since each client decides to participate or not independently from any other external factor, it can account for its own privacy loss after each iteration without requiring any other iteration-specific information about the system, such as the number of available clients, the number of sampled clients, or the number of sampled elements by the participating clients. This local sampling and privacy accounting mechanism makes the implementation of our algorithm feasible in practice.
- We provide a theoretical analysis of the central privacy guarantees of the proposed algorithm. Since random client participation leads to a non-uniform sampling among the data points stored at different clients, the analysis is non-trivial and cannot be directly obtained from the standard privacy amplification results via subsampling. Unlike the literature, our analysis does not rely on shuffling, and we do not need local randomizers with pure DP guarantees. This is especially useful when Gaussian noise is used as a randomizer.

6.3 Problem Setting

In our setting, we consider a FL scenario with N clients and a PS. Each client i, $i \in [1 : N]$, holds a dataset $\mathcal{D}_i$ such that they form $\mathcal{D} = \cup_{i\in[1:N]}\mathcal{D}_i$. For simplicity, we assume that each client holds the same number of data samples, i.e., $|\mathcal{D}_i| = d$, $\forall i \in [1 : N]$, but our analysis applies to different $|\mathcal{D}_i|$ values as well. However, we allow one of the clients to violate this assumption to satisfy the remove-add neighbouring relation, which is the neighbouring relation assumed throughout this chapter. That is, at most one of the clients can store a dataset of size $d + 1$. During training, we consider differentially private distributed stochastic gradient descent (DP-DSGD). Namely, the clients and the PS collectively learn a model $\boldsymbol{w} \in \mathbb{R}^m$ through $T \in \mathbb{Z}^+$ iterations by minimizing the loss function

$$\hat{\ell}(\boldsymbol{w}, \mathcal{D}) = \frac{1}{N} \sum_{i\in[1:N]} \frac{1}{|\mathcal{D}_i|} \sum_{s\in\mathcal{D}_i} \ell(\boldsymbol{w}, s), \tag{6.1}$$

where ℓ is a general loss function depending on the nature of the learning problem. In general, we allow it to be non-convex; hence, our results are applicable to deep neural networks. Since clients can go offline for several reasons in FL, in each iteration t, we assume $N_t \leq N$ clients are available and N_t is known to the PS but not necessarily to the clients.

6.3.1 Threat Model

Following [1, 102, 103], we assume a trusted PS and honest but curious clients, i.e., they do not deviate from the protocol but they can try to learn about the local datasets of other clients. We further assume the presence of a secure communication channel between each client and the PS such that neither the presence of communication nor the content of the transmitted messages are disclosed to third parties. Therefore, in the case of random participation, for an adversary, it is not possible to infer the participating clients by tapping into the channels.

Trusted PS assumption may sound strong, but due to the nature of the client sampling problem, it is essential. Keeping the identities of the participating clients secret from the PS is difficult without employing a trusted third party. Hence, as done in similar works [1, 102, 103], we make this assumption. This assumption is valid for the scenarios in which the training is done with the help of a trusted PS, and the model is publicly deployed after training. If the PS is not trusted, client sampling can still be employed for a price of increased communication cost, and our analysis still holds. We elaborate on this in Section 6.5.5.

6.4 Main Results

In this section, we first present the details of the proposed algorithm and then we state its privacy guarantees.

6.4.1 Algorithm Description

At the beginning of each iteration $t \in [1 : T]$, independently from other clients, each client randomly decides whether to participate or not in that iteration, with probability p. If a client decides to participate, then it further samples a subset from its local dataset such that each element is sampled i.i.d. with probability q. Let the set of sampled clients in iteration t be $\mathcal{P}_t$ and the set of sampled elements in client $i \in \mathcal{P}_t$ be $\mathcal{S}_{i,t}$. After sampling, client $i \in \mathcal{P}_t$ computes the gradients $\nabla \ell(\boldsymbol{w}, s)$ for all $s \in \mathcal{S}_{i,t}$. If the L_2 norm of the gradient of any sample is greater than a predetermined constant C, then it is scaled down to C to guarantee $||\nabla \ell(\boldsymbol{w}, s)||_2 \leq C$. Then, the client i aggregates all the sample gradients, and sends the sum to the PS. The PS further aggregates all the summations from the participating clients and adds a Gaussian noise $\mathcal{N}(0, \sigma^2 \mathbf{I}_m)$ to the sum, where $\mathbf{I}_m$ is the identity matrix with dimension m. Then, it scales the noisy sum by $\frac{1}{pN_t qd}$ to have an unbiased estimate of the gradient. The final expression after these operations at the PS is

$$\hat{\boldsymbol{g}}_t = \frac{1}{pN_t qd} \left(\sum_{i \in \mathcal{P}_t} \sum_{s \in \mathcal{S}_{i,t}} \nabla \ell(\boldsymbol{w}, s) + \mathcal{N}(0, \sigma^2 \mathbf{I}_m) \right). \tag{6.2}$$

Finally, the PS updates the model as $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \hat{\boldsymbol{g}}_t$, where η_t is the learning rate for the iteration t. The pseudo-code for our procedure is given in Algorithm 1.

Algorithm 1 DP-DSGD with random participation

```
Protocol in client $i$:
  for $t \in [1 : T]$ do
    Sample $B \sim Bern(p)$
    if $B = 1$ then
      Inform the PS that $B = 1$
      Receive $\boldsymbol{w}_t$ from the PS
      Initialize $\boldsymbol{g}_i = 0$
      Sample $\mathcal{S}_i$ from $\mathcal{D}_i$, w.p. $q$, i.i.d. for each sample
      for $s \in \mathcal{S}_i$ do
        Increment $\boldsymbol{g}_i$ by $\nabla \ell(\boldsymbol{w}_t, s) / \max\left\{1, \frac{||\nabla \ell(\boldsymbol{w}_t, s)||_2}{C}\right\}$
      end for
      Send $\boldsymbol{g}_i$ to PS
    end if
  end for

Protocol in the PS:
  for $t \in [1 : T]$ do
    Learn $\mathcal{P}_t$ from the clients
    Broadcast $\boldsymbol{w}_t$ to $\forall i \in \mathcal{P}_t$
    Receive $\boldsymbol{g}_i$ from $\forall i \in \mathcal{P}_t$
    Aggregate, randomize and scale:
    $\hat{\boldsymbol{g}}_t = \frac{1}{pN_t qd} \left( \sum_{i \in P_i} g_i + \mathcal{N}(0, \sigma^2 \mathbf{I}_m) \right)$
    Update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \hat{\boldsymbol{g}}_t$
  end for
```

6.4.2 Privacy Guarantees

First, we note that the exact analysis of DP guarantees of the proposed scheme due to non-uniform sampling is challenging. Instead, in the following, we provide upper and lower bounds, which provide useful insights into the behaviours of the privacy guarantees. In the following theorem, we present an upper bound of DP guarantees of DP-DSGD with random participation.

Theorem 6.1. *Each iteration of DP-DSGD with random participation algorithm is (ε, δ)-DP, where for any $\varepsilon > 0$,*

$$\delta < \sum_{i=0}^{d} \binom{d}{i} q^i (1-q)^{d-i} \Big((1-p)(1-e^{\varepsilon}) \bar{\Phi}_{0,\sigma}(z_i) \\ + p(1-q-e^{\varepsilon}) \bar{\Phi}_{iC,\sigma}(z_i) + pq \bar{\Phi}_{(i+1)C,\sigma}(z_i) \Big), \quad (6.3)$$

where $\bar{\Phi}_{\mu,\sigma}(z)$ is the complementary cumulative distribution function (C-CDF) of the Gaussian distribution with mean μ and standard deviation σ at z, and z_i, $i \in [0:d]$, is the solution of the following equation for variable z:

$$\big((1-p)(1-e^{\varepsilon}) \mathcal{N}(z, 0, \sigma^2) + p(1-q-e^{\varepsilon}) \mathcal{N}(z, iC, \sigma^2) + pq \mathcal{N}(z, (i+1)C, \sigma^2) \big) = 0. \quad (6.4)$$

Next, we present also the lower bound.

Theorem 6.2. *Each iteration of DP-DSGD with random participation algorithm is (ε, δ)-DP, where for any $\varepsilon > 0$,*

$$\delta > \max_{-C \le \mu \le C} (1-p)(1-e^{\varepsilon}) \bar{\Phi}_{0,\sigma}(z^*) \\ + \sum_{i=0}^{d} \binom{d}{i} q^i (1-q)^{d-i} p(1-q-e^{\varepsilon}) \bar{\Phi}_{i\mu,\sigma}(z^*) + pq \bar{\Phi}_{i\mu+C,\sigma}(z^*), \quad (6.5)$$

where z^* is the smallest solution of the following equations for variable z:

$$(1-p)(1-e^{\varepsilon}) \mathcal{N}(z, 0, \sigma) + \sum_{i=0}^{d} \binom{d}{i} q^i (1-q)^{d-i} p(1-q-e^{\varepsilon}) \mathcal{N}(z, i\mu, \sigma) + pq \mathcal{N}(z, i\mu+C, \sigma) = 0. \quad (6.6)$$

Note that although the analytical solutions of Equation (6.4) and Equation (6.6) are not tractable, they can be efficiently solved numerically. We provide a more detailed discussion on this and the proofs of Theorems 6.1 and 6.2 in Section 6.7.1 and Section 6.7.2, respectively.

6.5 Discussion and Numerical Results

Firstly, we emphasize that the number of available clients N_t are needed only at the PS to correctly scale the average of the gradients. The clients do not need N_t or any other parameter neither to execute their algorithms nor to calculate their own privacy losses. Moreover, the noise is added by the PS centrally, and its variance is publicly known and does not depend on any iteration-specific information such as the number of sampled clients. Since the parameters in Equations (6.3) and (6.5) are all available locally, each client can keep track of its own privacy loss without needing any iteration-specific information about the system, which eliminates the need for central coordination.

Central noise addition might be seen as a disadvantage of our scheme since there are no local randomizers at the clients, but in the case of a trusted PS, our privacy guarantees are central and this does not constitute a limitation.

In the following, we introduce three baselines and compare their privacy guarantees with those of the proposed upper and lower bounds. Since the previous works on random client participation provide client-level DP guarantees, a comparison of these schemes with our work would not be fair since client-level privacy guarantees are much more restrictive. Nevertheless, in Section 6.5.3, we provide some comparisons to emphasize what improvement our scheme brings upon previous work.

6.5.1 Baselines

6.5.1.1 Only Local Sampling (OLS)

The first setting we consider is the one in which there is no randomness in client participation, i.e., $p = 1$, and each client samples from its local dataset such that each element is sampled with probability q in an i.i.d. fashion. We consider Only Local Sampling (OLS) to demonstrate the benefits of random participation. The privacy guarantees of local subsampling in OLS is given by the following theorem.

Theorem 6.3. *Each iteration of OLS scheme is (ε, δ)-DP, where, for any $\varepsilon' > 0$,*

$$\varepsilon = \log\left(1 + q\left(e^{\varepsilon' - 1}\right)\right), \tag{6.7}$$

$$\delta = q\left(\Phi\left(\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right) - e^{\varepsilon'}\Phi\left(-\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right)\right). \tag{6.8}$$

The proof of Theorem 6.3 is given in Section 6.7.3.

6.5.1.2 Weak Client Sampling (WCS)

In this setting, our sampling procedure is the same as in Algorithm 1, i.e., we both employ client and local dataset sampling. However, we assume that when the clients randomly decide whether to participate or not, the identities of the participating clients are disclosed. Hence, in Weak Client Sampling (WCS), random participation of clients helps amplify DP guarantees only since when some clients are not sampled, no information is leaked from them. This results in weaker privacy guarantees compared to the case where the identities of participating clients are hidden, and we consider this setting as another upper bound. Note that unlike Theorem 6.1, this upper bound does not depend on d, the number of elements each client hosts. To elaborate WCS, let us assume $\mathcal{D}'$ has the same elements as $\mathcal{D}$ except for an additional element x'. If the client storing x' is sampled, then the privacy leakage is the same as OLS. On the other hand, when this specific client is not sampled, no information is leaked at all about the existence of x'. This implies that the privacy guarantees of WCS should still be better than OLS. In the following theorem, we present the formal privacy guarantees of WCS.

Theorem 6.4. *Each iteration of WCS scheme is* (ε, δ)*-DP, with, for any* $\varepsilon' > 0$

$$\varepsilon = \log\left(1 + q\left(e^{\varepsilon' - 1}\right)\right), \tag{6.9}$$

$$\delta = pq\left(\Phi\left(\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right) - e^{\varepsilon'}\Phi\left(-\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right)\right). \tag{6.10}$$

The proof of Theorem 6.4 is given in Section 6.7.4.

6.5.1.3 Centralized Shuffling (CS)

In Centralized Shuffling (CS), we have the same client sampling and dataset sampling procedure with probabilities p and q, respectively, but at the beginning of each iteration, the elements stored in all the clients are shuffled centrally and uniformly. This is too costly and far from practice, but we consider this case as a lower bound to measure the performance of our scheme. The privacy guarantees of CS are derived in the same way as in Theorem 6.3. Notice that the case in Theorem 6.3 and CS are the same except for the probability of an element being sampled, which is pq in CS and q in OLS. Hence, the next corollary follows.

Corollary 6.1. *Each iteration of the CS scheme is* (ε, δ)*-DP where, for any* $\varepsilon' > 0$*,*

$$\varepsilon = \log\left(1 + pq\left(e^{\varepsilon' - 1}\right)\right), \tag{6.11}$$

$$\delta = pq\left(\Phi\left(\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right) - e^{\varepsilon'}\Phi\left(-\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right)\right). \tag{6.12}$$

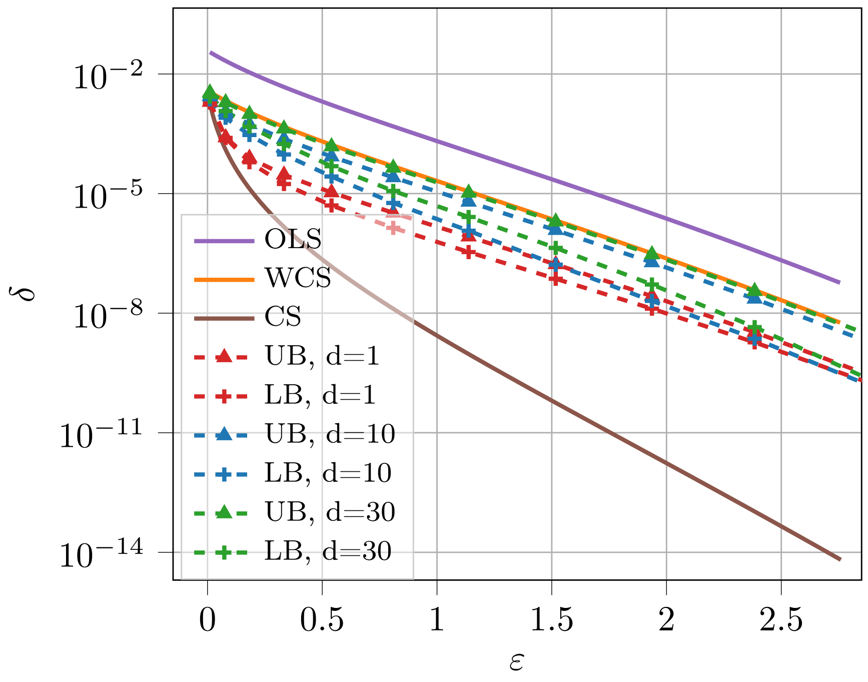


FIGURE 6.1: δ vs ε trade-off for the considered schemes.

6.5.2 Comparison to Baselines

For a single iteration of the learning task, we present the trade-offs ε vs. δ, ε vs. σ and δ vs. σ, in Figures 6.1 to 6.3, respectively, for the proposed analysis technique and the baselines we consider. We denote the upper bound from Theorem 6.1 by UB and the lower bound from Theorem 6.2 by LB in the figures. While generating the plots, we use $C = 1$ and $p = q = 0.1$. For ε vs. δ trade-off, we take $\sigma = 1$, for ε vs. σ trade-off, we take $\delta = 10^{-6}$ and for δ vs. σ trade-off, we take $\varepsilon = 1$.

We observe that the privacy guarantees of DP-DSGD with client sampling lie in between CS and WCS. Inferring whether a client is sampled or not by observing the corresponding model update is easier if the participation of one client changes the model update significantly, i.e., large sensitivity to the addition or removal of one client. From all figures, we see that as d increases, the privacy amplification due to client sampling becomes weaker. This applies to both the upper bound in Theorem 6.1 and the lower bound in Theorem 6.2. This can be explained by the fact that the sensitivity of the gradient sum received by the PS with respect to one element is bounded by C and the number of sampled elements from a client i, i.e., $|\mathcal{S}_{i,t}|$, determines how easy it is to make an inference about the participation of that client. That is, more sampled elements from a client may generate more opportunities for the adversaries to determine if this client is sampled or not, hence weakening the privacy amplification due to random client participation. Since smaller $d \cdot q$ values imply smaller expected values of $|\mathcal{S}_{i,t}|$, in the figures, we observe that smaller $d \cdot q$ values result in stronger privacy guarantees, and our scheme performs quite

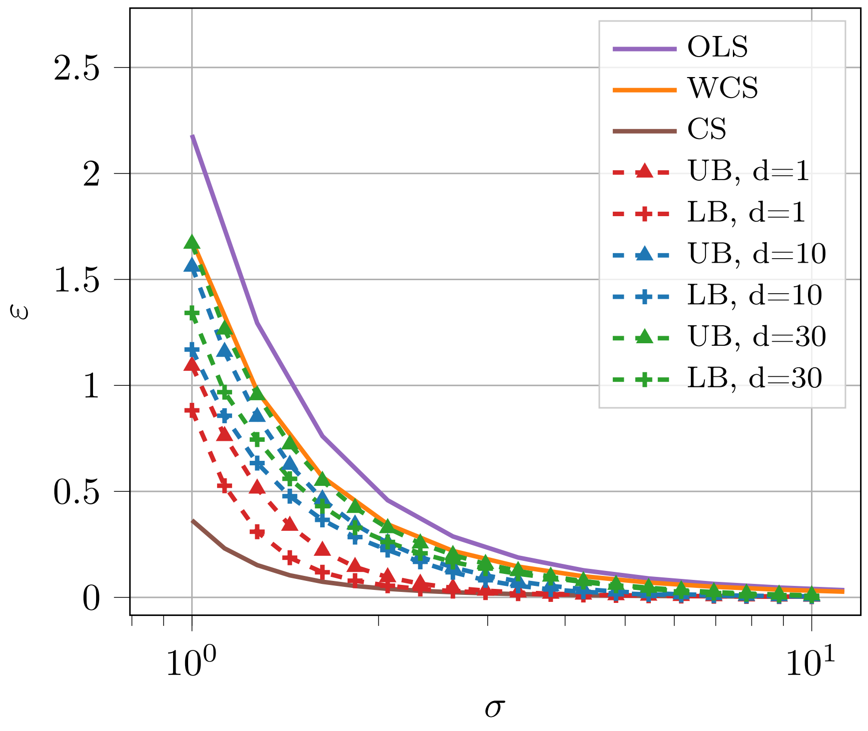


FIGURE 6.2: ε vs σ trade-off for the considered schemes when $\delta = 10^{-6}$.

close to the CS when $d \cdot q$ is small. For example, for $d = 1$, CS and our scheme are almost identical since it is quite hard to distinguish if the sole element of each client is sampled or not and hence, it is hard to make inferences about the participation of any client. Increasing d gives more information to an adversary about the identities of the sampled clients, but for $d = 10$ the privacy amplification via client sampling may be still considerably effective and results in better privacy guarantees than OLS and WCS. As the value of $d \cdot q$ increases, so does the privacy leakage of our scheme as we observe in all three figures, and the privacy guarantees of the proposed scheme converge to those of WCS as $d \to \infty$. For the considered set of parameters, since there are no significant differences between the privacy guarantees of WCS and the proposed scheme for values $d \cdot q > 3$, for clarity, we do not show the results beyond $d = 30$, which is almost the same curve as WCS. Such a convergence of our scheme to WCS is intuitive since when d is large, an adversary can gain a significant amount of information about the number of sampled clients and their identities. Since the assumption in WCS is that the identities of the sampled clients are known, our scheme becomes very close to the WCS, when d is large.

Note that in many practical cases, the dataset sizes may be much larger than those considered in this section. This does not necessarily mean that for the cases $d > 30$, the proposed scheme will be always equivalent to WCS. If the local dataset sizes of the clients are large, then much smaller q values can be employed. In fact, smaller $d \cdot q$ values result

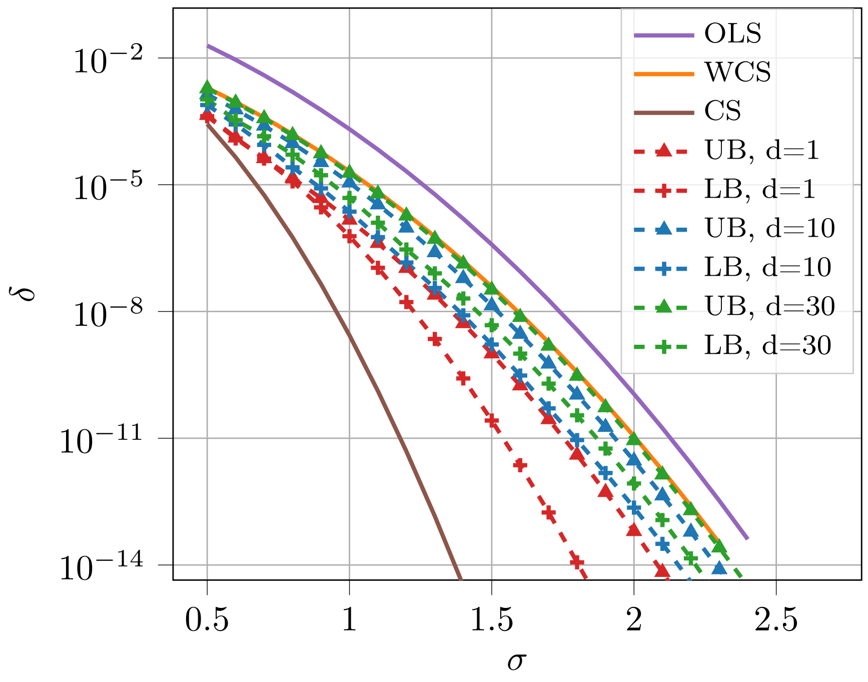


FIGURE 6.3: δ vs σ trade-off for the considered schemes when $\varepsilon = 1$.

in better privacy guarantees in general. This is the essence of what we have observed in this section. We consider an example with larger datasets in Section 6.6.2.

6.5.3 Comparison to Prior Work

Previous works on random participation provide client-level guarantees and assume local randomized with pure ε-DP guarantees. Hence, the comparison of these schemes with our work would not be fair for these works since the conversion from ε-DP guarantees to (ε, δ)-DP considerably degrades the privacy guarantees. Further, these guarantees are client-level, which is a more powerful privacy guarantee, so they may not be directly comparable with our results. Nevertheless, here we provide the comparison of our results with those of [1] in Figure 6.4 for a region where we are able to get meaningful privacy guarantees for the scheme in [1]. We used $p = q = 0.1$, $C = 1.0$, and $\sigma = 3.0$. We observe that in terms of approximate DP guarantees, the proposed analysis achieves a much better ε vs. δ trade-off than [1].

6.5.4 Local Sampling vs. Client Sampling

Both Theorems 6.1 and 6.2 imply that smaller p and q values improve the DP guarantees of DP-DSGD with client sampling. However, we may have some constraints that limit the minimum values p and q can take. For example, the batch size is a critical hyperparameter

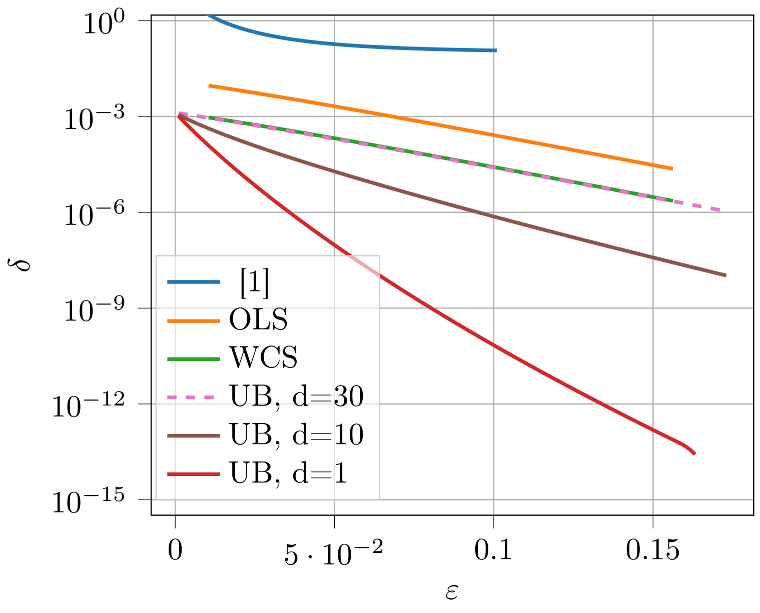


FIGURE 6.4: Comparison between [1] and our baselines in terms of ε vs. δ trade-off.

that significantly impacts the convergence of learning algorithms. While smaller values can lead to noisy updates, larger values may hinder the model's ability to learn details from the dataset. Hence, using a fixed batch size, which is determined by the value of pq and the size of the whole dataset, imposes a constraint on pq. In this section, we investigate how the DP guarantees of DP-DSGD with client sampling and our baselines are affected by the choice of p and q values while pq is kept fixed.

In Figure 6.5, given $pq = 10^{-4}$, we plot the ε vs. q trade-off for $\delta = 10^{-6}$, when $C = 1$ and $\sigma = 1$. We observe that, while we increase q, which corresponds to decreasing p at the same rate, the DP guarantees of all the schemes degrade. When all clients participate, i.e., $p = 1.0$, all the schemes become equivalent and ε is at its best value. Although this implies that we should rely only on local sampling to meet our target batch size, in most real-world scenarios, this is not possible for the reasons we have discussed in Section 6.2.1. Hence, the DP guarantees in the presence of client sampling are sandwiched between the bounds dictated by Theorems 6.1 and 6.2.

In accordance with the observations in Section 6.5.2, we further observe that as q increases, smaller dataset sizes at the clients, i.e., smaller d, become more favourable since they result in smaller ε values.

6.5.5 Modification of the proposed scheme when the PS is not trusted

When the PS is not trusted, client sampling can still be applied, and our theoretical analysis is still valid. In Algorithm 1, since Gaussian noise is added by the PS, the same

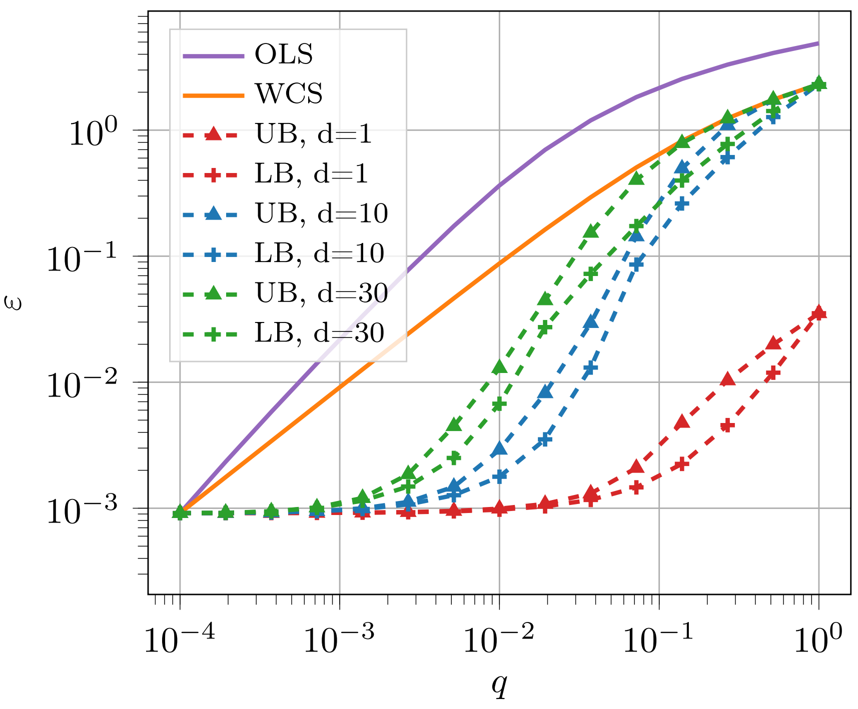


FIGURE 6.5: Trade-off between q and ε when $pq = 10^{-4}$ and $\delta = 10^{-6}$.

noise protects all the clients' updates at the same time. In the case of an untrusted PS, we cannot rely on the noise addition by the PS; and hence, each client must add Gaussian noise to protect its own privacy. This causes a higher variance of the effective noise added to the total update received by the PS; and hence, reduces the accuracy compared to the trusted PS case.

On the other hand, the sampling can still be done in the following manner. First, each client decides to participate or not independently with probability p, and if a client decides not to participate, it sends pure noise to the PS without processing its local dataset since we do not want the PS to learn if a client is sampled or not. If it decides to participate, then it samples its local dataset, calculates the gradients of the sampled elements, and sends the average of the gradients by adding Gaussian noise. This way, the PS will always receive a message from all the available clients, and our analysis on client sampling is still valid. One disadvantage of this modified protocol is that an available client sends a message to the PS at each iteration, even when it is not participating, and hence, the communication efficiency of the scheme degrades, which is one of the main motivations of the client sampling. Note that the reduction in the computational costs due to client sampling is still valid.

If the communication costs are scarce in a setting, alternatively, we may prefer a client communicates via the PS only if it decides to participate in a round. In this case, the PS would clearly learn if a client is participating or not, and our privacy guarantees are reduced to the one provided by WCS. Although it provides weaker guarantees than the proposed analysis method, we have seen in Section 6.5 and Section 6.6 that WCS considerably improves upon OLS, and can still provide meaningful privacy amplification due to client sampling.

6.6 Empirical Evaluation

In this section, we consider two different experimental settings to empirically show the improvements in the accuracy when the variance of the Gaussian noise added for DP guarantees is determined according to Theorems 6.1 and 6.2. In both experimental settings, we use EMNIST dataset [104], which is an extension of the MNIST dataset [105], where, in addition to the handwritten digits, uppercase and lowercase letters are also included. Although there are different possible splits in this dataset, we use the split based on digit and letter classification, referred to as *ByClass* in [104]. Hence, we have 62 possible classes. This dataset includes 814255 data samples in total, and we use 697932 of them as the training set and the rest as the test set, which is the standard partition employed in PyTorch [106]. In both scenarios, we use a simple deep neural network called LeNet [105], which is composed of two layers of convolutional neural networks (CNNs) followed by three fully connected layers. In Figure 6.6, we give the visualization of the network with all necessary details.

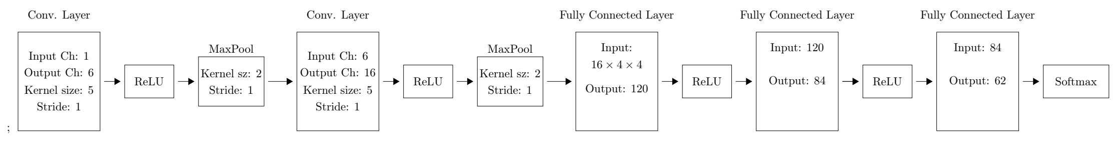


FIGURE 6.6: Network Architecture Used in the Experiments

6.6.1 Many Clients and Small Datasets Scenario

In the first scenario, we assume there are a large number of users while each user has only a few number of samples. Motivated by the findings in Figure 6.1, we choose that each client has $d = 10$ samples and we equally split the training set across 69793 clients. Since we have a relatively large number of clients, we choose the client sampling probability as $p = 0.001$, and each sampled client subsamples its local dataset with probability $q = 0.1$. These sampling rates correspond to an average batch size of 70, which we verify to provide a good accuracy via hyperparameter search. For each iteration of Algorithm 1, we aim to achieve a fixed privacy guarantee such that $\varepsilon = 0.015$ and $\delta = 10^{-6}$. These values may seem too conservative but note that training deep learning models is an iterative procedure, and at the end of the training, the total privacy leakage should be reasonably

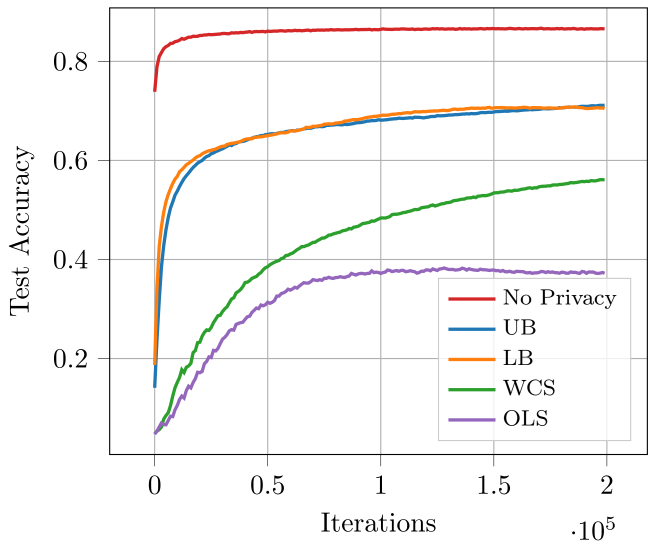


FIGURE 6.7: Many clients and small datasets scenario.

low. To match these guarantees to the upper bound in Theorem 6.1, the server should add Gaussian noise with a standard deviation of $\sigma = 1.52$. On the other hand, we use $\sigma = 1.395$ to match these guarantees to the lower bound in Theorem 6.2. Similarly, for WCS, according to Theorem 6.4, we should have $\sigma = 7.65$, and for OLS, according Theorem 6.3, we should have $\sigma = 22.4$. Moreover, to provide a baseline, we also train our model ignoring privacy constraints, i.e., $\sigma = 0$. For the proposed method, WCS and OLS, we use a gradient clipping value of $C = 1$, while no such restriction is imposed in the non-private case.

For training, we use SGD optimizer with a momentum of 0.9. We use the learning rate of 0.002 for the non-private case and for the DP-DSGD. For WCS and OLS, we observed the learning rate should be smaller due to a higher noise addition. Hence, the learning rate of 0.0001 is used for these schemes.

In Figure 6.7, we present the test accuracies of the proposed scheme and all the baselines with respect to the number of iterations. We observe that by using our analysis of privacy amplification via client sampling, we can attain a much higher accuracy than the other analysis methods. We further observe that, although it constitutes an upper bound on (ε, δ), WCS still brings a non-trivial improvement over OLS. In fact, this shows that even if the identities of the sampled clients can be inferred with high confidence from the gradients, the analysis provided in Theorem 6.4 for client sampling still provides useful privacy guarantees. Finally, we observe that the analysis based on only local sampling, i.e., OLS, has quite a poor performance due to the very high noise addition in this scheme, in order to attain $\varepsilon = 0.015$.

6.6.2 Few Clients and Large Datasets Scenario

Next, we consider the scenario with fewer clients, but each client has a larger dataset. We assume each client stores $d = 1000$ data points and there are 697 clients. Each client is independently sampled with a probability of $p = 0.1$ and each participating client samples its data points independently each with a probability of $q = 0.001$. Similar to the scenario in Section 6.6.1, we have an average batch size of 70. Again, we fix per-iteration privacy guarantees as $\varepsilon = 0.015$ and $\delta = 10^{-6}$. To match them to the upper bound in Theorem 6.1, the server should add Gaussian noise with $\sigma = 0.815$, and to match them to the lower bound in Theorem 6.2, the server should add Gaussian noise with $\sigma = 0.697$. For WCS and OLS, we have $\sigma = 0.873$ and $\sigma = 1.103$, according to Theorem 6.4 and Theorem 6.3, respectively. While we use $C = 1$ for the gradient clipping parameter in the private cases, we do not clip the gradients or add any noise in the non-private case.

During the training, we used SGD optimizer with a momentum of 0.9. The learning rate is 0.002, for all schemes.

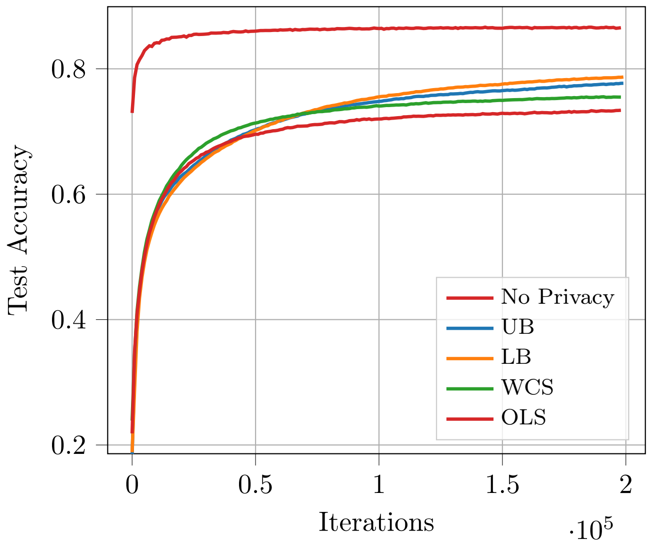


FIGURE 6.8: Few clients and large datasets scenario.

In Figure 6.8, we present the results of our experiments. Again, we observe that the accuracy of each scheme is inversely proportional to the amount of added noise; and hence, the proposed analysis scheme results in the best accuracies. However, compared to the setting in Section 6.6.1, in this setting, the performances of the schemes are closer to each other. As discussed in Section 6.5.4, that is because we have a smaller value of q although pq value is the same. Hence, a smaller amount of noise is enough to attain $\varepsilon = 0.015$.

6.7 Proofs

6.7.1 Proof of Theorem 6.1

Without loss of generality, we assume that $\mathcal{D}'$ has one more element than $\mathcal{D}$ and this extra element is stored by the first client, i.e., $\mathcal{D}_1' = \mathcal{D}_1 \cup \{x'\}$. Thus, the distribution of the model update in a single iteration when user 1 stores $\mathcal{D}_1'$ is

$$\xi(z) = (1-p)\,\xi_0(z) + p\left((1-q)\,\xi_1(z) + q\xi_2(z)\right) \tag{6.13}$$

and the distribution when user 1 stores $\mathcal{D}_1$ is

$$\xi'(z) = (1-p)\,\xi_0(z) + p\xi_1(z) \tag{6.14}$$

where ξ_0 is the distribution when user 1 is not sampled, ξ_1 is the distribution when user 1 is sampled but x' is not sampled, ξ_2 is the distribution when both user 1 and x' are sampled and z is the observed model update, and hence, $z \in \mathbb{R}^m$. In the proposed algorithm, all these distributions are multivariate Gaussian mixture distributions of dimension m. For clearer notation, we omit z in the rest of the section.

Remember from Definition 2.3 that the hockey-stick divergence between two distributions is written as

$$D_\alpha(\xi||\xi') \triangleq \int_Z \left[\xi(z) - \alpha\xi'(z)\right]_+ dz. \tag{6.15}$$

Hence,

$$\begin{aligned}\delta = D_{e^\varepsilon}(\xi||\xi') = \int_Z \Big[&(1-p)\xi_0(z) + p(1-q)\xi_1(z) \\ &+ pq\xi_2(z) - e^\varepsilon\left((1-p)\xi_0(z) + p\xi_1(z)\right)\Big]_+ dz \\ = \int_Z &\left[(1-p)(1-e^\varepsilon)\xi_0(z) + p(1-q-e^\varepsilon)\xi_1(z) + pq\xi_2(z)\right]_+ dz. \end{aligned} \tag{6.16}$$

Observe that there is a coupling between ξ_0, ξ_1 and ξ_2 such that except sampling of user 1 and x', the other sampled users and the sampled elements are the same. Therefore, ξ_1 has a set of elements sampled from user 1 except x' and the same sampled users and elements sampled in ξ_0. Similarly, ξ_2 has x' sampled as well as all the users and the elements sampled in ξ_1. As we stated previously, ξ_0, ξ_1 and ξ_2 are multivariate Gaussian mixture distributions. Next, let us define $\mathbb{P}$ be the set of all possible samplings of clients given client 1 is not sampled. Similarly, let us define $\mathbb{S}_\mathcal{I}$ as the set of all possible samplings of elements from the clients in the set $\mathcal{I}$ given these clients are sampled. Moreover, let

$\mathbb{S}_1$ be the set of all possible samplings of the elements stored in client 1 given x' is not sampled.

Thus, given $\mathcal{P} \in \mathbb{P}$, $\mathcal{S} \in \mathbb{S}_{\mathcal{P}}$ and $\mathcal{S}_1 \in \mathbb{S}_1$ are actual samplings of clients and the elements, respectively, we can write $\xi_1(\mathcal{P}, \mathcal{S}, \mathcal{S}_1) \sim \mathcal{N}(\boldsymbol{\mu}(\mathcal{P} \cup \{1\}, \mathcal{S} \cup \mathcal{S}_1), \sigma^2\mathbf{I}_m)$, $\xi_2(\mathcal{P}, \mathcal{S}, \mathcal{S}_1) \sim \mathcal{N}(\boldsymbol{\mu}(\mathcal{P} \cup \{1\}, \mathcal{S} \cup \mathcal{S}_1 \cup \{x'\}), \sigma^2\mathbf{I}_m)$ and $\xi_0(\mathcal{P}, \mathcal{S}) \sim \mathcal{N}(\boldsymbol{\mu}(\mathcal{P}, \mathcal{S}), \sigma^2\mathbf{I}_m)$ where $\boldsymbol{\mu}(\mathcal{P}, \mathcal{S})$ is the observed model update when the set of sampled clients is $\mathcal{P}$ and the set of sampled elements is $\mathcal{S}$.

As a result, we can write Equation (6.15) as

$$\begin{aligned} D_{e^\varepsilon}\left(\xi || \xi'\right) = \int_Z \sum_{\mathcal{P}\in\mathbb{P}} \sum_{\mathcal{S}\in\mathbb{S}_{\mathcal{P}}} \Pr(\mathcal{P}, \mathcal{S}) \Big[(1-p)(1-e^\varepsilon) \sum_{\mathcal{P}\in\mathbb{P}} \sum_{\mathcal{S}\in\mathbb{S}_{\mathcal{P}}} \Pr(\mathcal{P}, \mathcal{S}) \mathcal{N}(z, \boldsymbol{\mu}(\mathcal{P}, \mathcal{S}), \sigma^2\mathbf{I}_m) \\ + p(1-q-e^\varepsilon) \sum_{\mathcal{P}\in\mathbb{P}} \sum_{\mathcal{S}\in\mathbb{S}_{\mathcal{P}}} \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{P}, \mathcal{S}) \Pr(\mathcal{S}_1) \mathcal{N}(z, \boldsymbol{\mu}(\mathcal{P} \cup \{1\}, \mathcal{S} \cup \mathcal{S}_1), \sigma^2\mathbf{I}_m) \\ + pq \sum_{\mathcal{P}\in\mathbb{P}} \sum_{\mathcal{S}\in\mathbb{S}_{\mathcal{P}}} \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{P}, \mathcal{S}) \Pr(\mathcal{S}_1) \mathcal{N}(z, \boldsymbol{\mu}(\mathcal{P} \cup \{1\}, \mathcal{S} \cup \mathcal{S}_1 \cup \{x'\}), \sigma^2\mathbf{I}_m) \Big]_+ dz, \quad (6.17) \end{aligned}$$

Since $\mathcal{P}$ and $\mathcal{S}$ are common client and local dataset samplings to all ξ_0, ξ_1 and ξ_2, we take ξ_0 as the reference point, i.e., $\mathcal{N}(z, \boldsymbol{\mu}(\mathcal{P}, \mathcal{S}), \sigma^2\mathbf{I}_m) = 0$, and get rid of the dependence on $\mathcal{P}$ and $\mathcal{S}$. That is,

$$\begin{aligned} D_{e^\varepsilon}\left(\xi || \xi'\right) = \int_Z \Big[(1-p)(1-e^\varepsilon) \mathcal{N}(z, 0, \sigma^2\mathbf{I}_m) \\ + p(1-q-e^\varepsilon) \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{S}_1) \mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1), \sigma^2\mathbf{I}_m) \\ + pq \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{S}_1) \mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\}), \sigma^2\mathbf{I}_m) \Big]_+ dz. \quad (6.18) \end{aligned}$$

Notice that now, we use $\bar{\boldsymbol{\mu}}$ as the center of the normal distributions relative to $\mathcal{N}(z, \boldsymbol{\mu}(\mathcal{P}, \mathcal{S}), \sigma^2\boldsymbol{I}_m)$. Since first term does not depend on $\mathcal{S}_1$, we can write

$$\begin{aligned} D_{e^\varepsilon}\left(\xi || \xi'\right) = \int_Z \Big[(1-p)(1-e^\varepsilon) \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{S}_1) \mathcal{N}(z, 0, \sigma^2\mathbf{I}_m) \\ + p(1-q-e^\varepsilon) \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{S}_1) \mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1), \sigma^2\mathbf{I}_m) \\ + pq \sum_{\mathcal{S}_1\in\mathbb{S}_1} \Pr(\mathcal{S}_1) \mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\}), \sigma^2\mathbf{I}_m) \Big]_+ dz. \quad (6.19) \end{aligned}$$

Due to the joint convexity of hockey-stick divergence, we can use Equation (6.19) to upper bound $D_{e^\varepsilon}(\xi||\xi')$ as

$$D_{e^\varepsilon}\left(\xi||\xi'\right) \leq \sum_{\mathcal{S}_1 \in \mathbb{S}_1} \Pr(\mathcal{S}_1) \int_Z \Big[(1-p)(1-e^\varepsilon)\mathcal{N}(z, 0, \sigma^2\mathbf{I}_m) + p(1-q-e^\varepsilon)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1), \sigma^2\mathbf{I}_m) + pq\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\}), \sigma^2\mathbf{I}_m)\Big]_+ dz. \quad (6.20)$$

In general, $\bar{\boldsymbol{\mu}}(1, \mathcal{S}_1)$ and $\bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\})$ depend on the sample gradients generated from the sampled elements in $\mathcal{S}_1$ and x'. However, since we are interested in the worst-case analysis, we want to maximize the distance between the centers of $\mathcal{N}(0, \sigma^2\boldsymbol{I}_m)$ and $\mathcal{N}(\bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\}), \sigma^2\boldsymbol{I}_m)$. For this, we assume that each sampled element changes the mean $\bar{\boldsymbol{\mu}}$ exactly by the maximum possible amount, i.e., C, at the same direction with all other elements. Hence, if $|\mathcal{S}_1| = i$, $0 \leq i \leq d$, then we can write Equation (6.20), in terms of univariate normal distributions, as

$$D_{e^\varepsilon}\left(\xi||\xi'\right) \leq \sum_{i=0}^{d} \binom{d}{i} q^i (1-q)^{d-i} \int_Z \Big[(1-p)(1-e^\varepsilon)\mathcal{N}(z, 0, \sigma^2) + p(1-q-e^\varepsilon)\mathcal{N}(z, iC, \sigma^2) + pq\mathcal{N}(z, (i+1)C, \sigma^2)\Big]_+ dz. \quad (6.21)$$

For a specific i, in Equation (6.21), the expression inside $[\cdot]_+$ goes to zero when $z \to \infty$ and $z \to -\infty$. Other than these, the expression has only one zero-crossing z_i and for all finite $z > z_i$, it is positive. Therefore, taking the integral $z \geq z_i$ is enough. Moreover, since there is only one zero-crossing for the expression, it is easy to numerically solve it, i.e.,

$$(1-p)(1-e^\varepsilon)\mathcal{N}(z, 0, \sigma^2) + p(1-q-e^\varepsilon)\mathcal{N}(z, iC, \sigma^2) + pq\mathcal{N}(z, (i+1)C, \sigma^2) = 0 \quad (6.22)$$

for z, where z_i is the solution. If we put the solutions z_i, $i \in [0:d]$ in place, we can write Equation (6.21) as

$$D_{e^\varepsilon}\left(\xi||\xi'\right) \leq \sum_{i=0}^{d} \binom{d}{i} q^i (1-q)^{d-i} \Big((1-p)(1-e^\varepsilon)\bar{\Phi}_{0,\sigma}(z_i) + p(1-q-e^\varepsilon)\bar{\Phi}_{iC,\sigma}(z_i) + pq\bar{\Phi}_{(i+1)C,\sigma}(z_i)\Big), \quad (6.23)$$

which proves the claim of Theorem 6.1.

6.7.2 Proof of Theorem 6.2

Since Theorem 6.2 claims a lower bound on $D_{e^\varepsilon}(\xi||\xi')$, we start from Equation (6.19). To evaluate this expression, we need $\bar{\boldsymbol{\mu}}(1, \mathcal{S}_1)$ and $\bar{\boldsymbol{\mu}}(1, \mathcal{S}_1 \cup \{x'\})$, which depend on the sample gradients generated from the sampled elements in $\mathcal{S}_1$ and x'. To obtain the worst case divergence, for each element indexed by i stored at client 1, we should consider a sensitivity vector variable, $\boldsymbol{\mu}_i$ such that the gradient of sample i affects the total gradient by that amount. Then, $D_{e^\varepsilon}(\xi||\xi')$ should be maximized over these variables $\boldsymbol{\mu}_i$. Unfortunately, this is a difficult optimization problem to solve making also difficult to estimate the worst-case achievable value of $D_{e^\varepsilon}(\xi||\xi')$.

Instead, we employ a proxy by restricting the values of $\boldsymbol{\mu}_i$ to be the same and pointing to the same direction for all elements of client 1, except x'. Since they point to the same direction, without losing generality, we assume only the first coordinate of the vector $\boldsymbol{\mu}_i$ is non-zero and denote it by the scalar μ. For the effect of x' on the total gradient, we take it as at the maximum possible value C but still keeping the same direction of the others. Hence, from Equation (6.19), for any $-C \leq \mu \leq C$, $D_{e^\varepsilon}(\xi||\xi')$ can be lower bounded as

$$D_{e^\varepsilon}(\xi||\xi') \geq \int_Z \Big[(1-p)(1-e^\varepsilon)\mathcal{N}(z,0,\sigma^2)+\sum_{i=0}^{d}\binom{d}{i}q^i(1-q)^{d-i}\Big(p(1-q-e^\varepsilon)\mathcal{N}(z,i\mu,\sigma^2) + pq\mathcal{N}(z,i\mu+C,\sigma^2)\Big)\Big]_+ dz. \quad (6.24)$$

This is a lower bound since we restrict $\boldsymbol{\mu}_i$'s to be the same for all elements of client 1. Still, we can have a tighter lower bound on $D_{e^\varepsilon}(\xi||\xi')$ by optimizing the expression Equation (6.24) over μ. Hence, we have

$$D_{e^\varepsilon}(\xi||\xi') \geq \max_{-C\leq\mu\leq C} \int_Z \Big[(1-p)(1-e^\varepsilon)\mathcal{N}(z,0,\sigma^2)+ \sum_{i=0}^{d}\binom{d}{i}q^i(1-q)^{d-i}\Big(p(1-q-e^\varepsilon)\mathcal{N}(z,i\mu,\sigma^2) + pq\mathcal{N}(z,i\mu+C,\sigma^2)\Big)\Big]_+ dz. \quad (6.25)$$

This time, although the expression inside $[]_+$ may have more than one zero crossings, since the lower bound is already an underestimation of $D_{e^\varepsilon}(\xi||\xi')$, we can take its smallest zero crossing and integrate the expression inside $[]_+$ from that point to the infinity to have a lower bound on hockey-stick divergence. Hence, for a given μ, if z^* is the smallest solution of

$$(1-p)(1-e^{\varepsilon})\mathcal{N}(z,0,\sigma^2) \\ +\sum_{i=0}^{d}\binom{d}{i}q^i(1-q)^{d-i}\Big(p(1-q-e^{\varepsilon})\mathcal{N}(z,i\mu,\sigma^2)+pq\mathcal{N}(z,i\mu+C,\sigma^2)\Big)=0, \quad (6.26)$$

we have

$$D_{e^{\varepsilon}}\left(\xi||\xi'\right) \geq \max_{-C\leq\mu\leq C}(1-p)(1-e^{\varepsilon})\bar{\Phi}_{0,\sigma}(z^*) \\ +\sum_{i=0}^{d}\binom{d}{i}q^i(1-q)^{d-i}\Big(p(1-q-e^{\varepsilon})\bar{\Phi}_{i\mu,\sigma}(z^*)+pq\bar{\Phi}_{i\mu+C,\sigma}(z^*)\Big), \quad (6.27)$$

which proves Theorem 6.2.

6.7.3 Proof of Theorem 6.3

Before the proof, we present the following theorem from [52], which will be useful in our proof.

Theorem 6.5 (Theorem 8 in [52]). *Let M be a randomized mechanism satisfying $(\varepsilon, \delta_M(\varepsilon))$-DP. We define M' as the randomized mechanism when the input is first sampled from a larger dataset with Poisson sampling with probability q, then M is applied on the sampled subset. Then, we have $\delta_{M'}(\varepsilon') \leq q\delta_M(\varepsilon)$, where $\varepsilon' = \log\left(1+q(e^{\varepsilon}-1)\right)$.*

According to Theorem 6.5, if a mechanism M satisfying (ε, δ)-DP takes as the input a subset sampled from the entire dataset with Poisson sampling, we have $\delta_{M'}(\varepsilon') \leq q\delta_M(\varepsilon)$ and $\varepsilon' = \log\left(1+q(e^{\varepsilon}-1)\right)$, where M' is the mechanism M cascaded with sampling. Moreover, from Theorem 2.2, we know ε vs δ trade-off for Gaussian mechanism. The claim follows from combining Theorem 6.5 and Theorem 2.2. □

6.7.4 Proof of Theorem 6.4

The proof can be obtained by following the same steps as in the proof of Theorem 6.1 until Equation (6.19). In Equation (6.19) if we ignore the first term, i.e. $(1-p)(1-e^{\varepsilon})\sum_{\mathcal{S}_1\in\mathbb{S}_1}\Pr(\mathcal{S}_1)\mathcal{N}(0,\sigma^2)$, then we obtain a looser upper bound with larger resulting δ. Moreover, ignoring this expression is equivalent to ignoring the contribution of the client sampling when client 1 is sampled. This results in

$$\begin{aligned}
D_{e^\varepsilon}\left(\xi || \xi'\right) = \int_Z \Big[& p(1-q-e^\varepsilon) \sum_{\mathcal{S}_1 \in \mathbb{S}_1} \Pr(\mathcal{S}_1)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1), \sigma^2 \mathbf{I}_m) \\
& + pq \sum_{\mathcal{S}_1 \in \mathbb{S}_1} \Pr(\mathcal{S}_1)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1 \cup \{x'\}), \sigma^2 \mathbf{I}_m) \Big]_+ dz \\
= p \int_Z \Big[& (1-q-e^\varepsilon) \sum_{\mathcal{S}_1 \in \mathbb{S}_1} \Pr(\mathcal{S}_1)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1), \sigma^2 \mathbf{I}_m) \\
& + q \sum_{\mathcal{S}_1 \in \mathbb{S}_1} \Pr(\mathcal{S}_1)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1 \cup \{x'\}), \sigma^2 \mathbf{I}_m) \Big]_+ dz \\
\leq p \max_{\mathcal{S}_1 \in \mathbb{S}_1} \int_Z \Big[& (1-q-e^\varepsilon)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1), \sigma^2 \mathbf{I}_m) \\
& + q\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1 \cup \{x'\}), \sigma^2 \mathbf{I}_m) \Big]_+ dz. \quad (6.28)
\end{aligned}$$

Note that $\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1), \sigma^2 \mathbf{I}_m)$ and $\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1 \cup \{x'\}), \sigma^2 \mathbf{I}_m)$ are two normal distributions whose mean differs at most by C. Hence, in Equation (6.28), without p, the expression represents OLS. That is, by Theorem 6.3,

$$\begin{aligned}
\max_{\mathcal{S}_1 \in \mathbb{S}_1} \int_Z \Big[& (1-q-e^\varepsilon)\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1), \sigma^2 \mathbf{I}_m) \\
& + q\mathcal{N}(z, \bar{\boldsymbol{\mu}}(1,\mathcal{S}_1 \cup \{x'\}), \sigma^2 \mathbf{I}_m) \Big]_+ dz \\
& = q\left(\Phi\left(\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right) - e^{\varepsilon'}\Phi\left(-\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right)\right), \quad (6.29)
\end{aligned}$$

where

$$\varepsilon = \log\left(1 + q\left(e^{\varepsilon' - 1}\right)\right). \quad (6.30)$$

Hence, we obtain,

$$D_{e^\varepsilon} \leq pq\left(\Phi\left(\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right) - e^{\varepsilon'}\Phi\left(-\frac{C}{2\sigma} - \frac{\sigma\varepsilon'}{C}\right)\right), \quad (6.31)$$

which proves the claim of Theorem 6.4. □

6.8 Conclusion

Client sampling is widely accepted as a core aspect of FL to reduce the energy consumption of clients and to limit the communication load. In this chapter, we have analyzed the privacy amplification provided by client sampling in FL to amplify sample-level DP guarantees. While the privacy amplification effect of local data sampling has been well studied, to the best of our knowledge, the work presented in this chapter is the first to study the approximate DP implications of client sampling in FL. Previous works employing client sampling either provide user-level DP guarantees, or their analyses rely on shuffling. Although shuffling provides strong central privacy guarantees when local randomizers with pure DP guarantees are available, when the local randomizers satisfy approximate DP, the central privacy guarantees degrade considerably. This poses a challenge when Gaussian noise is used as a randomizer in FL settings. Instead, we have provided an analysis that does not rely on shuffling and allows the use of Gaussian noise without degrading the privacy guarantees. We have also shown that when the number of samples available at each client is small, we obtain privacy guarantees close to those of centralized training, in which uniform sampling is possible. Moreover, even if the local dataset sizes are very large, we have shown that client sampling is still quite beneficial compared to only local dataset sampling. Moreover, since available clients locally decide to participate or not, independently from each other and from any other system parameter, they can keep track of their own privacy losses; and therefore, our scheme trivially scales to large systems, and is feasible to apply in practice.

Chapter 7

Communication Efficient Private Federated Learning Using Dithering

7.1 Abstract

The task of preserving privacy while ensuring efficient communication is a fundamental challenge in federated learning. In this work, we tackle this challenge in the trusted aggregator model, and propose a solution that achieves both objectives simultaneously. We show that employing a quantization scheme based on subtractive dithering at the clients can effectively replicate the normal noise addition process at the aggregator. This implies that we can guarantee the same level of differential privacy against other clients while substantially reducing the amount of communication required, as opposed to transmitting full precision gradients and using central noise addition. We also experimentally demonstrate that the accuracy of our proposed approach matches that of the full precision gradient method.

7.2 Introduction

FL framework allows multiple clients to collaboratively learn a model with the help of a PS, without sharing their local datasets [18]. Instead, clients share their local updates with the PS after the local training round and the PS broadcasts the aggregated model back to the clients. Given the size of modern neural network architectures, this brings a massive amount of communication overhead. Hence, one core challenge in FL is the communication cost.

Privacy is a significant concern when it comes to ML since the solutions depend on data that can reveal sensitive information about its owner. Hence, while training ML models, it is vital to prevent any sensitive features from the training set from leaking. Although privacy is one of the core promises of FL since data never leaves clients and only the local model updates are shared, as discussed in the previous chapters, it has been shown that such updates, and even the final trained model are enough to reveal sensitive information about the training set [36, 39, 40, 43]. This calls for the necessity of additional privacy-preserving mechanisms in FL.

In this chapter, we jointly address privacy and communication efficiency in the trusted aggregator model. Privacy-wise, our aim is to provide central DP guarantees; that is, we want the average of the client updates to satisfy the DP guarantees. The trusted aggregator model covers the scenarios in which the PS is trusted by the clients. This can be the case when the clients give their data to a trusted organization, such as a government agency, an independent regulator, or a research institution while they do not want to reveal their data to other clients via model updates, or to third parties via the final deployed model. However, a trusted aggregator model may not always require a trusted PS. For example, trusted execution environments (TEE) can be employed at the PS and data encryption between TEE and the clients and the code run at the TEE can be formally verified [107, 108]. Another application area of our scheme is distributed learning, such as the settings in [29]. In such settings, a massive amount of data belonging to the same entity is used to train a model. So, the training task is distributed across many GPUs or servers, called worker terminals owned by the same entity. In order to avoid privacy leakage from the final deployed model, which can be accessed by third parties either via white-box or black-box access, the training procedure must satisfy DP. In such cases, our scheme significantly reduces the communication cost from GPUs or the worker terminals to the PS. Hence, the trusted aggregator model we consider is only an abstraction but not a stringent system requirement, and applies to many scenarios encountered in practice.

Communication-efficient FL has been an active research area [26, 27, 29]. However, when it comes to DP guarantees, directly extending these techniques is suboptimal since both compression for communication efficiency and privacy introduce separate errors. Works such as [109–115] consider tackling these two problems jointly. However, they mostly aim at guaranteeing local DP, in which each update from clients is separately protected. Unfortunately, such a stringent requirement considerably hurts the final model's performance. Moreover, due to the use of specially designed mechanisms to satisfy local DP, the aforementioned methods are not directly extendable to central DP.

The main idea of our proposed method is that the randomization required for privacy hurts the accuracy, and hence, it may not be necessary for clients to send full precision updates since they will be already destroyed by the PS to some extent after adding noise. Instead, we propose using subtractive dithering quantization on the client updates prior to sending them to the PS. This reduces the communication cost while keeping the same accuracy. We show that if the quantization step size is randomly generated following a particular distribution, with the help of shared randomness between each client and the PS, the noise in the reconstructed update at the PS follows a normal distribution for any third party that does not have access to the common randomness. By employing dithered quantization, we simulate the normal noise addition process to ensure DP and avoid adding noise twice for quantization and DP, while significantly reducing the communication overhead.

7.3 Problem Setting

We consider FL with N clients and a PS. Each client i has its own dataset, $\mathcal{D}_i$ and they collaboratively learn a model $f_{\mathbf{w}}$ with parameter $\mathbf{w} \in \mathbb{R}^m$ by minimizing a cumulative loss function,

$$\frac{1}{N} \sum_{i \in [1:N]} \frac{1}{|\mathcal{D}_i|} \sum_{d \in \mathcal{D}_i} \ell \left(f_{\mathbf{w}} \left(d_f \right), d_l \right), \tag{7.1}$$

where ℓ is the local loss function, d_f and d_l are the features and the label of the data point d, i.e, $d = (d_f, d_l)$. In the sequel, for brevity, we write $\ell(\mathbf{w}, d) \triangleq \ell \left(f_{\mathbf{w}} \left(d_f \right), d_l \right)$.

For training, we consider distributed SGD optimization. That is, at each iteration t, each available client i samples a small batch, $\mathcal{B}_{i,t} \in \mathcal{D}_i$, of average size B, and for each sample in the batch, computes the gradients of the loss function with respect to the current model parameters, i.e., $\nabla \ell(\mathbf{w}_t, d)$, $d \in \mathcal{B}_{i,t}$. Then, each client sends the average of the sample gradients, i.e., $\nabla \ell_i^t \triangleq \frac{1}{B} \sum_{d \in \mathcal{B}_{i,t}} \nabla \ell(\mathbf{w}_t, d)$, to the PS, where these gradients are further averaged to obtain the global average, denoted by $\mathbf{g}$. Finally, the new global model, which is updated by $\mathbf{g}$, is broadcast to all available clients for the consecutive update round.

Threat Model: In this chapter, we stick to the trusted aggregator model, i.e., the PS is trusted. Moreover, for compression, we assume that there are separate sources of common randomness shared between each client and the PS. Clients are assumed to be honest but curious. That is, they adhere to the protocol but may try to infer sensitive client information from average updates received from the PS. Hence, one of our goals is to protect the privacy of each client's local dataset from other clients since the updated model across rounds may reveal important sensitive information. Besides, once the training is completed, the final deployed model may leak sensitive information as well. Hence, we also aim to protect privacy leakage from the final deployed model, which makes our model and solution relevant even when the clients are trusted, as in distributed learning.

7.4 Proposed Method

In our proposed solution, we reduce the communication cost of each client update to the PS by quantizing them using subtractive dithering. We choose the step size of the quantization and the dithering parameters based on a gamma random variable. Such a trick achieves a quantization error that follows a Gaussian distribution, and hence, results in (ε, δ)-DP guarantees, as shown in Theorem 7.4. Our solution uses the following fact about the scale mixture of uniform distributions, which appears in [116].

Lemma 7.1. *If* $(X|V = v) \sim \mathrm{Unif}(\mu - \sigma\sqrt{v}, \mu + \sigma\sqrt{v})$*, and* $V \sim \Gamma[3/2, 1/2]$*, then* $X \sim \mathcal{N}(\mu, \sigma^2)$*, where* $\Gamma[3/2, 1/2]$ *is the gamma distribution with shape and rate parameters 3/2 and 1/2, respectively.*

This lemma states that if the realization of V, which follows a gamma distribution $\Gamma[3/2, 1/2]$, is not known, then the distribution of X becomes a normal distribution.

Our solution also utilizes the following fact about subtractive dithering [117, 118].

Lemma 7.2. *Let* Y *be the scalar to be quantized and* $\hat{Y} = Q(Y + U) - U$*, where* $U \sim \mathrm{Unif}\left(-\frac{\Delta}{2}, \frac{\Delta}{2}\right)$ *and* Q *is the quantization function with step size* Δ*. Then,* $\hat{Y} = Y + U'$*, where* $U' \sim \mathrm{Unif}\left(-\frac{\Delta}{2}, \frac{\Delta}{2}\right)$ *and independent from* U*.*

We summarize our proposed method in Algorithm 2. To generate batches $\mathcal{B}_{i,t}$, each client i employs *Poisson sampling*, that is, each sample in the local dataset $\mathcal{D}_i$ is sampled independently with probability p. Hence, $B = p|\mathcal{D}_i|$. To achieve formal privacy guarantees, at each client, we clip each sample gradient in the batch so that its L_2-norm is bounded by a parameter C. Then, the client computes the average of the sample gradients, $\nabla \ell_i^t$. Since $||\nabla \ell(\mathbf{w}_t, d)||_2 \leq C$, each element of the vector $\nabla \ell_i^t$ also lies within the range $[-C, C]$.

The client quantizes $\nabla \ell_i^t$ prior to sending it to the PS. Before quantization, for each element $(\nabla \ell_i^t)_j$, $j \in [m]$, the client samples $V_{i,j} = v_{i,j}$ from the gamma distribution $\Gamma[3/2, 1/2]$, and set the quantization step size $\Delta_{i,j} = 2\sigma\sqrt{v_{i.j}}$. Hence, the client uses a

Algorithm 2 Proposed Algorithm

```
Protocol in client $i$:
for $t \in [1:T]$ do
  Receive $\mathbf{w}_{t-1}$ from the PS
  Sample $\mathcal{B}_{i,t}$ from $\mathcal{D}_i$
  for $d = (d_f, d_l) \in \mathcal{B}_{i,t}$ do
    Calculate $\nabla\ell(\mathbf{w}_t, d)$
    Clip: $\nabla\ell(\mathbf{w}_t, d) = \nabla\ell(\mathbf{w}_t, d)/\max\left\{1, \frac{||\nabla\ell(\mathbf{w}_t,d)||_2}{C}\right\}$
  end for
  Calculate average gradient $\nabla\ell_i^t = \frac{1}{B}\sum_{d\in\mathcal{B}_{i,t}} \nabla\ell(\mathbf{w}_t, d)$
  for $j \in [m]$ do
    Sample $v_{i,j} \sim \Gamma[3/2, 1/2]$
    Calculate $\Delta_{i,j} = 2\sigma\sqrt{v_{i,j}}$
    Sample $U_{i,j} \sim \text{Unif}\left(-\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}\right)$
    Quantize: $\mathbf{m}_{i,j} = Q\left((\nabla\ell_i^t)_j + U_{i,j}\right)$ mapping it to the closest value in
$\{\cdots, -\frac{3\Delta_{i,j}}{2}, -\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}, \frac{3\Delta_{i,j}}{2}, \cdots\}$.
    Send the $\mathbf{m}_{i,j}$ by using $2C/\Delta_{i,j}$ bits to the PS.
  end for
end for

Protocol in the PS:
for $t \in [1:T]$ do
  for $i \in [1:N]$ do
    for $j \in [m]$ do
      Sample $v_{i,j} \sim \Gamma[3/2, 1/2]$ and $U_j \sim \text{Unif}\left(-\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}\right)$ using the shared randomness
with client $i$.
      Receive $\mathbf{m}_{i,j}$ and decode as $Q\left((\nabla\ell_i^t)_j + U_{i,j}\right)$
      Estimate $(\hat{\nabla}\ell_i^t)_j = Q\left((\nabla\ell_i^t)_j + U_{i,j}\right) - U_j$
    end for
  end for
  Average gradients $\mathbf{g}_t = \frac{1}{N}\sum_{i\in[1:N]}(\nabla\ell_i^t)_j$
  Update the model $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{g}_t$
  Broadcast $\mathbf{w}_{t+1}$ to all clients
end for
```

separate step size parameter for every element of $\nabla\ell_i^t$. Accordingly, the representative quantization points are set as $\{\cdots, -\frac{3\Delta_{i,j}}{2}, -\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}, \frac{3\Delta_{i,j}}{2}, \cdots\}$. Then, for each element of $(\nabla\ell_i^t)_j$, $j \in [m]$, it samples $U_{i,j}$ from $\text{Unif}\left(-\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}\right)$ and quantizes $(\nabla\ell_i^t)_j + U_{i,j}$. To be precise, we use the quantization function $Q(x) \triangleq \left\lceil \frac{x-\Delta/2}{\Delta} \right\rfloor \Delta + \frac{\Delta}{2}$, where $\lceil\cdot\rfloor$ is the function rounding its argument to the nearest integer.

To transmit the quantized gradients to the PS, client i encodes each element $(\nabla\ell_i^t)_j$ using $b_{i,j} \triangleq \left\lceil \log_2\left(2 \cdot \left\lceil \frac{C}{\Delta_{i,j}} + 1 \right\rfloor\right) \right\rceil$ bits since $|(\nabla\ell_i^t)_j| \le C$, resulting in $\sum_{j\in[m]} b_{i,j}$ bits of communication from client i to the PS in round t. We denote the message sent by client i in round t by $\mathbf{m}_{i,j}$.

Using the common randomness shared between client i and the PS, the same realizations of $V_{i,j}$'s and $U_{i,j}$'s generated by client i can also be obtained by the PS. From $\mathbf{m}_{i,j}$, the

PS can decode the value of $Q\left((\nabla\ell_i^t)_j + U_{i,j}\right)$, $\forall i \in [1:N]$ and $\forall j \in [m]$. Via subtractive dithering, the PS estimates $(\nabla\ell_i^t)_j$ as $(\hat{\nabla}\ell_i^t)_j \triangleq Q\left((\nabla\ell_i^t)_j + U_{i,j}\right) - U_{i,j} = (\nabla\ell_i^t)_j + U'_{i,j}$, where $U'_{i,j} \sim \text{Unif}\left(-\frac{\Delta_{i,j}}{2}, \frac{\Delta_{i,j}}{2}\right)$. Finally, from $(\hat{\nabla}\ell_i^t)_j$, $\forall i \in [1:N]$, the PS calculates the global gradient average $\mathbf{g}$, and updates the global model accordingly. Then it broadcasts the new global model to the clients for the next round. The global gradient average satisfies central DP requirement as stated by the following theorem.

Theorem 7.3. *Global gradient average* $\mathbf{g}$ *is a noisy estimate of the averages of the local gradients such that*

$$\mathbf{g} = \frac{1}{N}\sum_{i\in[1:N]} (\nabla\ell_i^t)_i + \mathcal{N}\left(0, \frac{\sigma^2}{N}\mathbf{I}_m\right). \tag{7.2}$$

Hence, $\forall\varepsilon' > 0$, $\mathbf{g}$ *satisfies* (ε,δ)*-DP in sample-level against clients for* $\varepsilon = \log\left(1 + p(e^{\varepsilon'} - 1)\right)$ *and*

$$\delta = p\cdot\Phi\left(\frac{C}{\sigma B\sqrt{N}} - \frac{\varepsilon'\sigma B\sqrt{N}}{2C}\right) - p\cdot e^{\varepsilon'}\Phi\left(-\frac{C}{\sigma B\sqrt{N}} - \frac{\varepsilon'\sigma B\sqrt{N}}{2C}\right), \tag{7.3}$$

where Φ *denotes the CDF of the standard normal distribution.*

Proof. For j^{th} element of $(\nabla\ell_i^t)_i$, $i \in [1:N]$, since $\Delta_{i,j} = 2\sigma\sqrt{v_{i,j}}$, $U'_{i,j}$ is distributed as $\text{Unif}(-\sigma\sqrt{v_{i,j}}, \sigma\sqrt{v_{i,j}})$. Since $v_{i,j}$ is a sample from $\Gamma[3/2, 1/2]$, according to Theorem 7.1, $U'_{i,j}$ is distributed as $\mathcal{N}(0,\sigma^2)$ when the realization of $v_{i,j}$ is unknown. Since this is the case for any client $k \neq i$, the quantization noise of one client is normally distributed from the perspective of any other client. When we consider the averaging operation at the CS, we obtain Equation (7.2). For the (ε,δ)-DP guarantee, we use the following lemma, which is the joint statement of Theorem 8 in [36] and Theorem 8 in [52].

Lemma 7.4. *Let* f *be a function satisfying* $||f(X) - f(X')||_2 \leq L$ *and* g *be the Poisson sampling function with propability* p*. Then, for any* $\varepsilon \geq 0$ *and* $\delta \in [0,1]$*, the subsampled Gaussian mechanism* $M(X) = f\circ g(X) + \mathcal{N}(0,\sigma'^2)$ *is* (ε,δ)*-DP if and only if,* $\forall\varepsilon' > 0$, $\varepsilon = \log\left(1 + p(e^{\varepsilon'} - 1)\right)$ *and*

$$p\cdot\Phi\left(\frac{L}{2\sigma'} - \frac{\varepsilon'\sigma'}{L}\right) - p\cdot e^{\varepsilon'}\Phi\left(-\frac{L}{2\sigma'} - \frac{\varepsilon'\sigma'}{L}\right) \leq \delta. \tag{7.4}$$

Since every client divides the sum of its sample gradients by B, and the PS also averages them over N clients, the effect of the gradient of a single sample is at most $C/(BN)$. This makes the L_2 sensitivity of the signal $\frac{1}{N}\sum_{i\in[1:N]}(\nabla\ell_i^t)_i$ to be $2C/(BN)$. If we put this quantity in place of L in Equation (7.4), and set $\sigma' = \sigma/\sqrt{N}$ from Equation (7.2), we obtain Equation (7.3). □

Dataset	Baseline	Final ε	Accuracy	Epochs	$\sigma/\sqrt{N}$	C	BN
MNIST	Proposed Scheme	1.45	91.39±1.04	10	0.05	2.0	32
	Uncomp-DP	1.45	91.97±0.76	10	0.05	2.0	
	Non-private	∞	98.90±0.09	10	0	∞	
EMNIST	Proposed Scheme	0.95	70.17±0.26	10	0.05	2.0	32
	Uncomp-DP	0.95	70.02±0.47	10	0.05	2.0	
	Non-private	∞	85.11±0.13	10	0	∞	
CIFAR-10	Proposed Scheme	7.03	51.66±0.03	100	0.01	1.0	64
	Uncomp-DP	7.03	50.92±1.35	100	0.01	1.0	
	Non-private	∞	80.95±0.64	50	0	∞	

TABLE 7.1: Experimental Results

7.5 Experiments

In this section, we conduct numerical experiments using the proposed method and two other baselines, namely *Uncomp-DP* and *non-private*. In both baselines, we use an uncompressed transmission from the clients to the PS using double precision. In Uncomp-DP, the PS adds the required amount of noise to the sum of received client updates to achieve the target (ε, δ)-DP guarantees. On the other hand, in the non-private case, we do not impose any privacy requirements and the PS only averages the client updates.

We evaluate the proposed scheme and the baselines on the MNIST, EMNIST (*ByClass* partition) and CIFAR-10 datasets. We employ LeNet architecture [105] for MNIST and EMNIST, and ResNet-18 [119] for CIFAR-10. For DP training, we use the Opacus library [120] and employ the privacy accounting techniques based on Renyi-DP [83, 94]. To simulate FL, we evenly distribute the dataset among N clients, i.e., same $|\mathcal{D}_i|$'s $\forall i \in [1 : N]$. We determine the Poisson sampling probability p by simply dividing the expected total batch size at the PS, i.e., BN, by the number of data points in the whole dataset, $|\mathcal{D}_i|N$. We keep BN constant across different N's and we consider it as a hyperparameter to tune independent of N. Similarly, we tune the value of σ so that $\sigma/\sqrt{N}$ remains constant so that the same DP guarantees hold regardless of the number of clients. While, for the non-private cases, we train until convergence, for the private cases, we determine the number of epochs to reach reasonable privacy and accuracy levels.

We repeat each experiment 10 times and report the average accuracies in Table 7.1 along with the final ε value after composition for $\delta = 10^{-6}$, length of the gradient vector per round, communication cost per round and some related training hyperparameters. In general, in the experiments with all three datasets, we observe that the accuracies of the proposed method and the Uncomp-DP match. This verifies our theory claiming that subtractive dithering quantization is equivalent to adding Gaussian noise at the PS. Hence, the privacy accounting of these methods also matches.

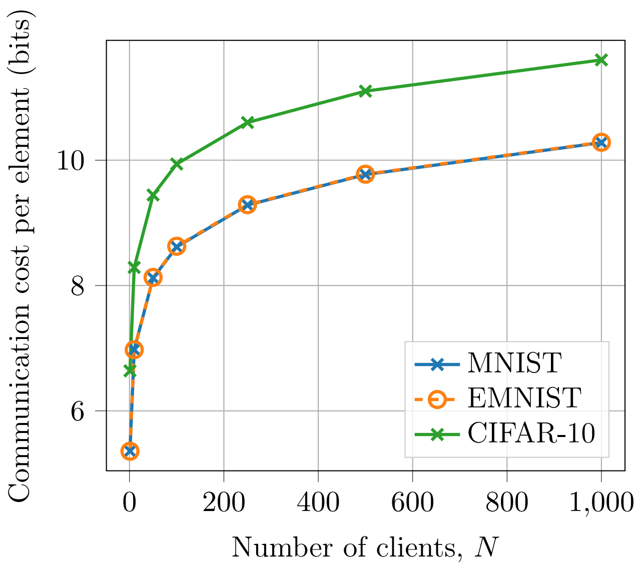


FIGURE 7.1: Communication cost vs. number of clients for the proposed scheme.

Since we tune B and σ so that $\sigma/\sqrt{N}$ and BN remain constant with N, the accuracies of our experiments does not depend on the number of clients involved, which means that the same accuracies in Table 7.1 applies to different N values. However, since now σ depends on N, the communication cost per client increases with the number of clients involved. Fortunately, we observe that the communication cost scales logarithmically with the number of clients; and hence, even with a very large number of clients, our scheme still uses significantly less communication. We plot the average communication cost per gradient element via numerical simulations in Figure 7.1. We observe that in MNIST and EMNIST experiments per-element costs match since we use the same C and σ parameters. For CIFAR-10, to have good accuracy, we tune the hyperparameters differently, and hence, we end up with a slightly larger communication cost. In all the cases, however, compared to double precision, which uses 64 bits per element, we use 12 to 5.5 times less communication depending on the number of clients without sacrificing accuracy. This observation shows that the proposed scheme saves a significant amount of communication for free in DP training.

7.6 Conclusion

In this chapter, through both theoretical analysis and experimental demonstrations, we have shown that using subtractive dithering quantization in the trusted aggregator model of FL can produce the same level of DP and accuracy guarantees as Gaussian noise addition, while utilizing fewer communication resources. This technique may prove useful in speeding up privacy-sensitive learning in communication-scarce scenarios such as edge training or time-critical industrial applications. Although the trusted aggregator model has many real-world applications, one possible area of future exploration is extending our methods to situations where trust in the PS is difficult to achieve. Additionally, exploring the possibility of extending the proposed technique to joint quantization would be an interesting future research direction.

Chapter 8

Conclusion

In this dissertation, we studied several research problems in the field of distributed computing. Common to all the problems studied, the ultimate goal was to make distributed computing and learning more accessible to all parties requiring it.

Our first objective was to expedite large-scale computations, even in the presence of unreliable computing nodes. For organizations with limited computational resources and financial constraints, accessing high-quality computing nodes can pose significant challenges. Therefore, we aimed to ensure that even with such constraints, entities can leverage their resources effectively and complete their tasks within a reasonable timeframe. However, in doing so, data sharing is inevitable, which raises concerns about data privacy. Our secondary objective was to ensure that privacy breaches are avoided even when organizations with limited resources need to outsource their computations to external nodes. Thus, we extended our methods to provide a framework for reliable and fast computations while guaranteeing data privacy. Our research in Chapters 3 and 4 has tackled these concerns via coded computation techniques.

Another primary objective of our research was to ensure that clients who participate in FL are not adversely affected in terms of their privacy and resources. This is crucial because clients should not have to choose between the benefits of modern machine learning and their privacy. By providing opportunities for clients to participate in FL without compromising their privacy, we can ensure that machine learning techniques benefit society as a whole, rather than just economically powerful entities. Our research, as presented in Chapters 5 to 7, aligns with these goals and aims to contribute to the development of privacy-preserving FL techniques.

In the subsequent sections, we provide a more detailed account of the challenges we encountered, the achievements we made, and the future directions of the research problems we focused on. Through this discussion, we hope to contribute to the ongoing efforts to address the challenges associated with our research area and to inspire further progress towards our shared goals.

8.1 Challenges, Achievements and Future Directions

8.1.1 Coded, Secure Distributed Matrix Multiplication

More than half of this dissertation is dedicated to addressing the problem of secure, storage- and upload-cost-efficient straggler exploitation for distributed matrix multiplication, as discussed in Chapters 3 and 4. At the outset of our research, we identified the issue of inefficient straggler exploitation, whereby widely adopted univariate polynomial codes were limited to using only the same-point evaluations of the encoding polynomials. As a result, the number of computations that workers can provide was linear with their storage capacity, despite the potential for a quadratic number of computations due to the multiplication of different matrices' partitions.

To address this issue, we proposed a simple solution: using bivariate polynomials to ensure that the encoding polynomials' evaluations are independent. However, this approach posed a new challenge in the form of the invertibility of the interpolation matrix, which was highly non-trivial compared to univariate polynomial codes. Furthermore, this problem had not been studied in the context of coding theory, necessitating a deep dive into polynomial interpolation theory and proof techniques in this domain.

To build the necessary theoretical foundation, we consulted the book [65], which provided valuable insights despite its challenging nature. With this background knowledge, we were able to obtain several invertibility results and develop a range of schemes, including B-PROC, BPC-VO, BPC-HO, BPC-NZO, and BPC-ZZO, for which the invertibility proof holds. We believe that the proof technique we developed is itself a theoretically interesting result that may guide the proofs of other multivariate interpolation schemes for distributed matrix multiplication in more general situations.

Moreover, as demonstrated in Chapter 3, the proposed schemes can exploit the workers' storage capacities in a way close to optimal, and for different storage capacities, they outperform existing schemes in the literature in terms of the average computation time.

Subsequently, we shifted our focus towards extending the proposed bivariate polynomial coding schemes to ensure the security of the multiplied matrices. To achieve this, we adopted techniques similar to those used in existing univariate polynomial coding-based schemes from the literature. Specifically, we extended the coding to be defined over a finite field, introduced random terms to the encoding polynomials to mask the input matrices, and employed techniques to minimize the number of monomials whose coefficients include undesired multiplications among random terms in the final decoding polynomial. The primary challenge in this endeavour was to extend the invertibility proof such that it is valid in finite fields. This required some literature research on the Taylor series expansion on finite fields. However, compared to the initial challenge of developing the proof technique, this extension was relatively less demanding.

We note that our proposed secure scheme, as demonstrated in Chapter 4, achieves a significant improvement in the average computation time over the rival schemes in the literature.

Despite the significant progress made in the development of bivariate polynomial coding for secure distributed matrix multiplication, there remain several open problems that require further investigation. Notably, the secure version of the scheme does not utilize the storage or upload cost budget in a near-optimal manner, unlike the initial construction without security guarantees. However, it is still considerably more efficient than the secure univariate counterparts. Addressing this limitation represents a promising direction for future research, with the potential to further enhance the performance of secure distributed matrix multiplication.

In conclusion, while coding theoretical approaches hold great potential for accelerating large-scale computations, their widespread adoption may be hindered by the encoding and decoding complexity, as well as the development efforts required. Although the encoding and decoding complexity does not necessarily outweigh the speedup benefits, the improvements may be incremental. However, in the context of security concerns, coded computation is likely to emerge as one of the popular solutions in the future. This is because coded computation can accelerate computations while incurring less overhead compared to secure multi-party computation-based solutions. As such, further research and implementation are needed to explore the potential of coded computation in practical frameworks addressing their data security and dependability concerns.

8.1.2 Privacy-Preserving Federated Learning

With the goal of preserving the privacy of clients participating in training in FL, we initially focused on the problem of ensuring privacy in a wireless medium. This setting is motivated by the need to enable edge devices, which have limited power, computation, and communication resources, to participate efficiently in the training process. In Chapter 5, we demonstrate that the scarcity of communication resources can be leveraged to advantage by applying the concept of OAC. OAC utilizes simultaneous transmissions to efficiently use bandwidth, and we show that this approach can also provide anonymity to the transmitting clients. This allows us to apply previous results on privacy amplification via sampling to client sampling, as the receiver does not know the identities of the transmitting clients by nature. Therefore, we demonstrate another useful property of OAC, namely anonymity, in addition to its communication benefits.

However, it is important to note that the assumptions made in this scheme may be too strong. Firstly, the scheme requires specialized hardware that enables clients to transmit their updates simultaneously in an analog fashion. While this is technically feasible, the current infrastructure is designed and optimized for digital communications, making it difficult to apply this scheme in practice. Therefore, more practical and system-oriented research is required in OAC to make such schemes applicable.

Furthermore, in terms of privacy, the assumption that the PS cannot learn any information about the sampled clients due to the lack of knowledge of the number of participating clients is also too strong. Although it may be challenging in practice, a powerful adversary with knowledge of all the sampled devices except one in all training rounds may eventually learn some information about the participation of this secret client. Assuming otherwise is somewhat against the strong adversary assumption of differential privacy. Therefore, it is necessary to consider the possibility of such an adversary and develop more robust privacy-preserving mechanisms.

Motivated by this, we studied the leakage of the sampled devices directly from the aggregated client updates. To this end, we conducted a theoretical investigation of the joint privacy amplification effect of client sampling and local dataset sampling in each participating client, as detailed in Chapter 6. Unlike the setting described in Chapter 5, we considered a conventional FL setting with a trusted PS.

Our analysis focused on the correlation generated among the samples of the same client in their subsampling due to the joint sampling of clients and local datasets. We provided a theoretical analysis of such non-uniform subsampling, which revealed that when the size of the local datasets is small, the privacy guarantees via random participation are close to those of the centralized setting. However, when the local datasets are large, observing the output of the algorithm may disclose the identities of the sampled clients with high confidence. This finding is in accordance with the intuition, as clients with larger datasets provide more sample gradients when sampled than clients with smaller datasets, making them easier to detect.

Our work provides a rigorous analysis of the proposed sampling method, as detailed in Theorem 6.1 and Theorem 6.2. However, in terms of theoretical elegance, the corresponding expressions for ε and δ are somewhat complex and not in closed form. To find values of ε and δ, numerical methods are required. Fortunately, these methods are relatively easy and numerically efficient to implement.

One limitation of our current analysis is the lack of composition results, which are crucial in iterative algorithms that require multiple accesses to the datasets, such as FL. While we could use the advanced composition theorem to provide a result, the composition analysis provided by this theorem is usually not tight. Since the composition analysis of the plain Gaussian mechanism without sampling is quite tight, the analysis via the advanced composition theorem would fail to demonstrate the superiority of the proposed sampling method.

Therefore, a tight analysis of the proposed sampling method is still an important open problem that requires further investigation. In our continued research on the topic, we will aim to address this issue and provide a more comprehensive analysis of the proposed sampling method.

Finally, in Chapter 7, we shift our focus to the important topic of communication efficiency in private FL, which is currently a very active research area. Our contribution to this area is the suggestion that, via an existing channel simulation technique based on subtractive dithering, we can use much fewer communication resources. This is made possible by random quantization, where the quantization levels are determined by a gamma-distributed random variable. Remarkably, the marginal distribution of the final distortion on the original signal becomes a Gaussian random variable, which is a perfect source for differential privacy guarantees.

However, it is important to note that the random quantization level must be known by the PS to decode the noisy client update. Therefore, the scheme does not provide privacy against the PS. Nevertheless, the clients' data are still kept private against other clients and all adversaries who can only see the final deployed model. Although the trusted PS may be criticized by the privacy community, such scenarios are already applied in many real-world applications, and our scheme can be very beneficial in these scenarios. We provide a detailed discussion of these settings in Chapter 7.

Despite our proposed solution, there is still a fundamental open question in the area of communication-efficient FL: "What is the minimum required distortion to achieve (ε, δ)-DP in both local or central sense, and what is the compression technique to achieve this distortion?". This question remains a challenge for researchers in the field, and fully or partially answering it will be a future endeavour for us and others working in this area.

Bibliography

[1] B. Balle, P. Kairouz, B. McMahan, O. D. Thakkar, and A. Thakurta, “Privacy amplification via random check-ins,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[2] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Polynomial codes: An optimal design for high-dimensional coded matrix multiplication,” in *Proc. Int’l Conf. on Neural Information Processing Systems*, 2017, pp. 4406–4416.

[3] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[5] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, “A unified coded deep neural network training strategy based on generalized polydot codes for matrix multiplication,” *arXiv preprint arXiv:1811.10751*, 2019.

[6] Z. Jia and S. A. Jafar, “Cross subspace alignment codes for coded distributed batch computation,” *arXiv preprint arXiv:1909.13873*, 2019.

[7] M. Fahim and V. R. Cadambe, “Numerically stable polynomially coded computing,” *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2758–2785, 2021.

[8] A. Ramamoorthy and L. Tang, “Numerically stable coded matrix computations via circulant and rotation matrix embeddings,” *arXiv preprint arXiv:1910.06515*, 2019.

[9] A. M. Subramaniam, A. Heidarzadeh, and K. 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). IEEE, 2019, pp. 253–259.

[10] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 6266–6282, 2021.

[11] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2018, pp. 1–6.

[12] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *arXiv preprint arXiv:1810.13006*, 2018.

[13] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[14] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

[15] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[16] J. Kakar, A. Khristoforov, S. Ebadifar, and A. Sezgin, "Uplink cost adjustable schemes in secure distributed matrix multiplication," *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 1124–1129, 2020.

[17] N. Mital, C. Ling, and D. Gunduz, "Secure distributed matrix computation with discrete Fourier transform," *arXiv preprint arXiv:2007.03972*, 2020.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics.* PMLR, 2017, pp. 1273–1282.

[19] C. Dupuy, T. G. Roosta, L. Long, C. Chung, R. Gupta, and S. Avestimehr, "Learnings from federated learning in the real world," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2022, pp. 8767–8771.

[20] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[21] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2019.

[22] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[23] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gündüz, "Dopamine: Differentially private federated learning on medical data," *arXiv preprint arXiv:2101.11693*, 2021.

[24] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth annual conference of the international speech communication association*, 2014.

[25] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[26] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in neural information processing systems*, vol. 30, 2017.

[27] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Int'l Conf. on Machine Learning*, 2017, pp. 3329–3337.

[28] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," *Advances in neural information processing systems*, vol. 31, 2018.

[29] S. Horváth, C.-Y. Ho, L. Horváth, A. N. Sahu, M. Canini, and P. Richtárik, "Natural compression for distributed deep learning," *Proceedings of Machine Learning Research vol*, vol. 145, pp. 1–40, 2022.

[30] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," *Advances in neural information processing systems*, vol. 24, 2011.

[31] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[32] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3368–3376.

[33] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *International Conference on Machine Learning*. PMLR, 2018, pp. 903–912.

[34] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[35] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[36] B. Balle and Y.-X. Wang, "Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *International Conference on Machine Learning*. PMLR, 2018, pp. 394–403.

[37] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[38] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 774–14 784.

[39] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 267–284.

[40] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 16 937–16 947. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf

[41] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data set inference and reconstruction attacks in online learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1291–1308.

[42] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, “Extracting training data from large language models,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[43] N. Haim, G. Vardi, G. Yehudai, O. Shamir, and M. Irani, “Reconstructing training data from trained neural networks,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 911–22 924, 2022.

[44] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[45] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “{BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning,” in *2020 USENIX annual technical conference (USENIX ATC 20)*, 2020, pp. 493–506.

[46] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, “Homomorphic encryption-based privacy-preserving federated learning in iot-enabled healthcare system,” *IEEE Transactions on Network Science and Engineering*, 2022.

[47] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of cryptography conference.* Springer, 2006, pp. 265–284.

[48] S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[49] Q. Geng and P. Viswanath, “The optimal noise-adding mechanism in differential privacy,” *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 925–951, 2015.

[50] I. Sason and S. Verdú, “f-divergence inequalities,” *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 5973–6006, 2016.

[51] G. Barthe and F. Olmedo, “Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs,” in *International Colloquium on Automata, Languages, and Programming.* Springer, 2013, pp. 49–60.

[52] B. Balle, G. Barthe, and M. Gaboardi, “Privacy amplification by subsampling: Tight analyses via couplings and divergences,” in *Advances in Neural Information Processing Systems*, 2018, pp. 6277–6287.

[53] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[54] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[55] S. Asoodeh, J. Liao, F. P. Calmon, O. Kosut, and L. Sankar, "A better bound gives a hundred rounds: Enhanced privacy guarantees via f-divergences," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 920–925.

[56] M. M. Amiri and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," *IEEE Transactions on Signal Processing*, vol. 67, no. 24, pp. 6270–6284, 2019.

[57] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1988–1992.

[58] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware distributed learning: Communication–computation latency trade-off," *Entropy*, vol. 22, no. 5, p. 544, 2020.

[59] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[60] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1620–1624.

[61] S. Kiani, N. Ferdinand, and S. C. Draper, "Hierarchical coded matrix multiplication," in *2019 16th Canadian Workshop on Information Theory (CWIT)*. IEEE, 2019, pp. 1–6.

[62] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5152–5160.

[63] A. Mallick, M. Chaudhari, and G. Joshi, "Fast and efficient distributed matrix-vector multiplication using rateless fountain codes," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8192–8196.

[64] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, “Factored lt and factored raptor codes for large-scale distributed matrix multiplication,” *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 893–906, 2021.

[65] R. A. Lorentz, *Multivariate Birkhoff Interpolation.* Springer, 2006.

[66] K. E. Atkinson, *An Introduction to Numerical Analysis, Second Edition.* John Wiley & Sons, 1988.

[67] T. Sauer and Y. Xu, “On multivariate hermite interpolation,” *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 207–259, 1995.

[68] T. Sauer, “Computational aspects of multivariate polynomial interpolation,” *Advances in Computational Mathematics*, vol. 3, no. 3, pp. 219–237, 1995.

[69] J. Von Zur Gathen and J. Gerhard, *Modern computer algebra.* Cambridge university press, 2013.

[70] D. E. Muller, “Application of boolean algebra to switching circuit design and to error detection,” *Transactions of the IRE professional group on electronic computers*, no. 3, pp. 6–12, 1954.

[71] I. S. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” *IEEE Transactions on Information Theory*, vol. 4, no. 4, pp. 38–49, 1954.

[72] G. Liang and U. C. Kozat, “Tofec: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications.* IEEE, 2014, pp. 826–834.

[73] J. Liesen and V. Mehrmann, *Linear Algebra.* Springer Undergraduate Mathematics Series, 2015.

[74] R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, “Fountain codes for private distributed matrix-matrix multiplication,” in *2020 International Symposium on Information Theory and Its Applications (ISITA).* IEEE, 2020, pp. 480–484.

[75] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, “Analog lagrange coded computing,” *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[76] O. Makkonen and C. Hollanti, “Analog secure distributed matrix multiplication over complex numbers,” in *2022 IEEE International Symposium on Information Theory (ISIT).* IEEE, 2022, pp. 1211–1216.

[77] K. Hoffman and R. Kunze, “Linear algebra,” *Englewood Cliffs, New Jersey*, 1971.

[78] F. Fontein, “The Hasse derivative,” Aug 2009. [Online]. Available: https://math.fontein.de/2009/08/12/the-hasse-derivative/

[79] J. T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities,” *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.

[80] Ú. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 1054–1067.

[81] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, “Collecting and analyzing multidimensional data with local differential privacy,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 638–649.

[82] S. Wang, L. Huang, Y. Nie, X. Zhang, P. Wang, H. Xu, and W. Yang, “Local differential private data aggregation for discrete distribution estimation,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 2046–2059, 2019.

[83] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[84] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” *arXiv preprint arXiv:1710.06963*, 2017.

[85] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning: A client level perspective,” *arXiv preprint arXiv:1712.07557*, 2017.

[86] M. M. Amiri and D. Gündüz, “Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[87] ——, “Federated learning over wireless fading channels,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, 2020.

[88] G. Zhu, Y. Wang, and K. Huang, “Broadband analog aggregation for low-latency federated edge learning,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, 2019.

[89] A. Sonee and S. Rini, “Efficient federated learning over multiple access channel with differential privacy constraints,” *arXiv preprint arXiv:2005.07776*, 2020.

[90] M. Seif, W.-T. Chang, and R. Tandon, "Privacy amplification for federated learning via user sampling and wireless aggregation," *arXiv preprint arXiv:2103.01953*, 2021.

[91] Y. Koda, K. Yamamoto, T. Nishio, and M. Morikura, "Differentially private aircomp federated learning with power adaptation harnessing receiver noise," *arXiv preprint arXiv:2004.06337*, 2020.

[92] D. Liu and O. Simeone, "Privacy for free: Wireless federated learning via uncoded transmission with adaptive power control," *arXiv preprint arXiv:2006.05459*, 2020.

[93] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *The 22nd International Conference on Artificial Intelligence and Statistics.* PMLR, 2019, pp. 1226–1235.

[94] I. Mironov, K. Talwar, and L. Zhang, "Renyi differential privacy of the sampled Gaussian mechanism," *arXiv preprint arXiv:1908.10530*, 2019.

[95] F. AI, "Opacus," https://github.com/pytorch/opacus.

[96] K. Chaudhuri and N. Mishra, "When random sampling preserves privacy," in *Annual International Cryptology Conference.* Springer, 2006, pp. 198–213.

[97] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms.* SIAM, 2019, pp. 2468–2479.

[98] V. Feldman, A. McMillan, and K. Talwar, "Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling," *arXiv preprint arXiv:2012.12803*, 2020.

[99] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *arXiv preprint arXiv:2010.13723*, 2020.

[100] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2021, pp. 3403–3411.

[101] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[102] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of differential privacy in federated learning," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2021, pp. 2521–2529.

[103] A. M. Girgis, D. Data, and S. Diggavi, "Differentially private federated learning with shuffling and client self-sampling," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 338–343.

[104] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

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

[106] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[107] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: privacy-preserving federated learning with trusted execution environments," in *Proc. of Annual Int'l Conf. on mobile Sys., Apps., and Services*, 2021, pp. 94–108.

[108] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *Int'l Conf. on Artificial Int. and Stats.*, 2022, pp. 3581–3607.

[109] S. Amiri, A. Belloum, S. Klous, and L. Gommans, "Compressive differentially private federated learning through universal vector quantization," in *AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2021, pp. 2–9.

[110] A. Triastcyn, M. Reisser, and C. Louizos, "Dp-rec: Private & communication-efficient federated learning," *arXiv preprint arXiv:2111.05454*, 2021.

[111] K. Chaudhuri, C. Guo, and M. Rabbat, "Privacy-aware compression for federated data analysis," in *Uncertainty in Artificial Intelligence*, 2022, pp. 296–306.

[112] A. Shah, W.-N. Chen, J. Balle, P. Kairouz, and L. Theis, "Optimal compression of locally differentially private mechanisms," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 7680–7723.

[113] H. Asi, V. Feldman, J. Nelson, H. L. Nguyen, and K. Talwar, “Fast optimal locally private mean estimation via random projections,” *arXiv:2306.04444*, 2023.

[114] B. Isik, W.-N. Chen, A. Ozgur, T. Weissman, and A. No, “Exact optimality of communication-privacy-utility tradeoffs in distributed mean estimation,” *arXiv:2306.04924*, 2023.

[115] N. Lang, E. Sofer, T. Shaked, and N. Shlezinger, “Joint privacy enhancement and quantization in federated learning,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 295–310, 2023.

[116] S. Walker, “The uniform power distribution,” *Journal of Applied Statistics*, vol. 26, no. 4, pp. 509–517, 1999.

[117] S. P. Lipshitz, R. A. Wannamaker, and J. Vanderkooy, “Quantization and dither: A theoretical survey,” *Journal of the audio engineering society*, vol. 40, no. 5, pp. 355–375, 1992.

[118] L. Roberts, “Picture coding using pseudo-random noise,” *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 145–154, 1962.

[119] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. on Comp. Vision and Pattern Recog.*, 2016, pp. 770–778.

[120] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao *et al.*, “Opacus: User-friendly differential privacy library in PyTorch,” *arXiv:2109.12298*, 2021.