Correlated Noise Mechanisms for Differentially Private Learning

A Tutorial on Mathematical Foundations and Practical Aspects

Krishna Pillutla
IIT Madras

Jalaj Upadhyay
Rutgers University

Christopher A. Choquette-Choo
Google DeepMind

Krishnamurthy Dvijotham
ServiceNow Research

Arun Ganesh
Google Research

Monika Henzinger
IST, Austria

Jonathan Katz
Google

Ryan McKenna
Google Research

H. Brendan McMahan
Google Research

Keith Rush
Google DeepMind

Thomas Steinke
Google DeepMind

Abhradeep Thakurta
Google DeepMind

Contents

Correlated Noise Mechanisms for Differentially Private Learning

Krishna Pillutla[1], Jalaj Upadhyay[2], Christopher A. Choquette-Choo[5], Krishnamurthy Dvijotham[3], Arun Ganesh[4], Monika Henzinger[6], Jonathan Katz[7], Ryan McKenna[4], H. Brendan McMahan[4], Keith Rush[5], Thomas Steinke[5] and Abhradeep Thakurta[5]

[1] *Indian Institute of Technology, Madras*
[2] *Rutgers University*
[3] *ServiceNow Research*
[4] *Google Research*
[5] *Google DeepMind*
[6] *Institute of Science and Technology Austria*
[7] *Google*

ABSTRACT

This monograph explores the design and analysis of correlated noise mechanisms for differential privacy (DP), focusing on their application to private training of AI and machine learning models via the core primitive of estimation of weighted prefix sums. While typical DP mechanisms inject independent noise into each step of a stochastic gradient (SGD) learning algorithm in order to protect the privacy of the training data, a growing body of recent research demonstrates that introducing (anti-)correlations in the noise can significantly improve privacy-utility trade-offs by carefully canceling out some of the noise added on earlier steps in subsequent steps. Such correlated noise mechanisms, known variously as matrix mechanisms, factorization mechanisms, and DP-Follow-the-Regularized-Leader (DP-FTRL) when applied to learning algorithms, have also been influential in practice, with industrial deployment at a global scale.

Unfortunately, the rapid development of this field and complex mathematical foundations pose a high barrier to entry for researchers and practitioners. This monograph provides a pedagogical tutorial of correlated noise mechanisms, with an emphasis on the the theoretical principles governing their design and derivation, and practical considerations such as scalability and runtime.

We start with private prefix sum estimation in the streaming setting, where each example is only processed on a single training step. We focus on mechanisms with structural constraints designed to balance high utility with low time and space complexity. Next, we allow each example to participate on multiple training steps, to accommodate practical private AI model training. While we focus on this example-level notion of privacy, we also discuss the straightforward generalization to uer-level privacy, where the DP guarantee extends to possibly multiple training examples all associated with the same user, as in practice user-level privacy is usually necessary. We also discuss how to numerically find the precise correlations (that is, the noise cancellation schedule) and offer implementation details and guidance for practitioners. Finally, we survey promising theoretical and applied open problems for researchers to contribute to this active and growing area.

Notation

We summarize the main notation in Tables 1 and 2.

Table 1: Notation summary (part 1). Matrices and vectors are denoted in boldface.

n	Number of training steps. More generally, the number of steps on which private estimates are released.
m	Model dimension.
$\boldsymbol{A} \in \mathbb{R}^{n \times n}$	The *workload matrix*.
$\boldsymbol{A}_{\mathsf{pre}} \in \mathbb{R}^{n \times n}$	The lower-triangular matrix of ones, defining an *unweighted* prefix sum workload.
$\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$	Matrix factorization of $\boldsymbol{A}$, where $\boldsymbol{B}$ is called the *decoder matrix*, and $\boldsymbol{C}$ is the *strategy* or *encoder* matrix.
$\boldsymbol{C}^{-1}$	The *noise-correlating matrix*, the (pseudo)inverse of the strategy matrix $\boldsymbol{C}$. The matrix $\boldsymbol{C}^{-1}$ maps i.i.d. Gaussian noise to a non-i.i.d. distribution.
$\boldsymbol{G} \in \mathbb{R}^{n \times m}$	$= (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1})$ stacked row-wise, the sequence of private inputs processed by the DP mechanism (typically $\boldsymbol{g}_t = \boldsymbol{G}[t, :] \in \mathbb{R}^m$ is a gradient).
$\boldsymbol{S} \in \mathbb{R}^{n \times m}$	$= (\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{n-1})$ stacked row-wise, the (weighted) prefix sums to be computed, $\boldsymbol{S} = \boldsymbol{A}\boldsymbol{G}$, equivalently $\boldsymbol{s}_t = \sum_{\tau=0}^{t} \boldsymbol{A}[t, \tau]\boldsymbol{g}_\tau$.
$\widehat{\boldsymbol{G}} \in \mathbb{R}^{n \times m}$	$= (\widehat{\boldsymbol{g}}_0, \ldots, \widehat{\boldsymbol{g}}_{n-1})$ stacked row-wise, DP estimates of $\boldsymbol{G}$.
$\boldsymbol{Z} \in \mathbb{R}^{n \times m}$	$= (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{n-1})$ stacked row-wise, the source i.i.d. noise injected for DP; private estimates of $\boldsymbol{G}$ are given by $\widehat{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}$.
$\widetilde{\boldsymbol{Z}} \in \mathbb{R}^{n \times m}$	$= \boldsymbol{C}^{-1}\boldsymbol{Z}$, the correlated noise injected into the learning algorithm. The noise injected in step t is $\widetilde{\boldsymbol{z}} = \widetilde{\boldsymbol{Z}}[t, :]$.
$\hat{\boldsymbol{S}} \in \mathbb{R}^{n \times m}$	$= (\hat{\boldsymbol{s}}_0, \ldots \hat{\boldsymbol{s}}_{n-1})$ stacked row-wise, DP estimates of $\boldsymbol{S}$, computed as $\widehat{\boldsymbol{S}} = \boldsymbol{A}\widehat{\boldsymbol{G}}$.

Table 2: Notation summary (part 2). Matrices and vectors are denoted in boldface.

$[n]$	$= \{0, \dots, n-1\}$ for $n \geq 1$. Indexed quantities (matrices, vectors, sequences) are zero-indexed throughout.
$\mathbb{N}$	$= \{0, 1, 2, \dots\}$, the natural numbers including zero.
$\boldsymbol{M}^\top$	Transpose of a matrix $\boldsymbol{M}$.
$\boldsymbol{M}^\star$	A matrix $\boldsymbol{M}$ that is "optimal" in a context-dependent sense.
$\boldsymbol{M}^\dagger$	The Moore-Penrose pseudoinverse of matrix $\boldsymbol{M}$.
$\boldsymbol{M}[i, j]$	The $(i, j)^{\text{th}}$ entry of matrix $\boldsymbol{M}$, zero-indexed.
$\boldsymbol{M}[i, :]$, $\boldsymbol{M}[:, j]$	The i^{th} row and j^{th} column of $\boldsymbol{M}$, zero-indexed. The ith row of $\boldsymbol{M}$ is also often denoted $\boldsymbol{m}_i \coloneqq \boldsymbol{M}[i, :]$
$\|\boldsymbol{M}\|_{\text{F}}$	The Frobenius norm of a matrix $\boldsymbol{M}$.
$\|\boldsymbol{M}\|_{\mathsf{row}}$	$= \max_{t \in [n]} \|\boldsymbol{M}[t, :]\|_2$, the maximum L_2 norm of the rows of $\boldsymbol{M}$.
$\|\boldsymbol{M}\|_{\mathsf{col}}$	$= \max_{t \in [n]} \|\boldsymbol{M}[:, t]\|_2$, the maximum L_2 norm of the columns of $\boldsymbol{M}$.
ln	Natural logarithm, $\ln(2.718) \approx 1$.
$\mathbb{E}_{Z \sim P}[f(Z)]$	Expectation of the function f under the distribution P.
$\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(\mu, \nu^2)$	$\boldsymbol{Z}$ is a random $n \times m$ matrix whose entries are i.i.d. $\mathcal{N}(\mu, \nu^2)$.
d	Order of recurrence / degree / number of memory buffers.
$\boldsymbol{x} \in \mathcal{X}$	ML training example.
$\ell : \Theta \times \mathcal{X} \to \mathbb{R}$	Loss function.
$\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$	Parameters of AI model to be learned. Typically, $\boldsymbol{g}_t = \nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x})$.
B	The (mini)batch size for model training.
N	The total number of examples $\boldsymbol{x}$ in the training dataset D.
k	(Maximum) number of times an example/user participates in training.
b	Minimum separation between the participations of an example/user.

1

Introduction: Private SGD and Prefix Sums with Correlated Noise

Privacy-sensitive data is increasingly being collected and stored by both private entities and governments. Such data includes health records, business records, personal communications (including chats and emails), individuals' location history (enabled by smartphone GPS), internet content consumption records, and social media interactions. General trends and insights gleaned from such data hold the promise of great benefit: real-time estimates of traffic congestion, better user experiences across apps and websites, public health insights, and more. With the rise of powerful AI models, including large language models (LLMs), machine learning is increasingly becoming the principal technique for unlocking the value latent in private datasets, but this approach also entails real privacy risks: it is imperative to ensure that models do not leak sensitive information contained in their training data.

Recent work has shown that this is not just a theoretical risk. There is a large body of work showing that exact copies of (sometimes sensitive) training data can be extracted from production language models, even with just black-box API access to these models. Analyzing the privacy of sophisticated data processing mechanisms is challenging, as privacy leakage can be subtle and hard to detect.

Differential privacy (DP) is the gold standard for analyzing privacy-sensitive data with rigorous and mathematically justified privacy guarantees. It has been adapted to AI and machine learning model training applications in recent years and has garnered significant interest in the research community. Models trained with rigorous DP guarantees also have industrial deployments at a global scale by companies such as Google and Apple.

The workhorse of private machine learning is a differentially private version of stochastic gradient descent, known as DP-SGD. This algorithm adds isotropic Gaussian noise to the updates of a gradient descent-based learning algorithm. While most work on DP-SGD injects independent Gaussian noise in each step of the learning algorithm, recent work has shown that it can be beneficial to use noise that has non-zero correlations across training steps. In particular, correlated noise mechanisms power "the first production neural network trained directly on user data announced with a formal DP guarantee."[1]

This monograph serves as a pedagogical introduction to such correlated noise mechanisms, with a focus on theoretical principles behind their design and development and practical aspects such as scalability and run time. Due to the rapid development of this field, a fragmented prior literature, and complex mathematical foundations, researchers and practitioners face high barriers to entry into this topic. To bridge this gap, we present this tutorial aimed at a broad audience, ranging from early graduate students with basic machine learning knowledge to experts seeking a consolidated reference.

Outline We start this section with a brief introduction to differential privacy in Section 1.1 for readers not already familiar with this notion. Next, we describe the problem of private machine learning and the DP-SGD algorithm in Section 1.2. We also describe its connection to the problem of private weighted prefix sum estimation; this forms the basis for the correlated noise mechanisms introduced in Section 1.3 and later sections. Next, we develop some intuition as to why correlated noise

[1] https://research.google/blog/federated-learning-with-formal-differential-privacy-guarantees/

mechanisms can yield better privacy-utility tradeoffs in Section 1.4. We then discuss what considerations have to be made in designing correlated noise mechanisms in Section 1.5, ending the section with technical aspects of DP guarantees in Sections 1.6 and 1.7.

Sections and remarks marked with an asterisk (∗) go into technical details or other advanced material—they can safely be skipped on a first reading.

1.1 Introduction to Differential Privacy

Differential privacy is a notion of privacy that is defined on a collection of individual data records. In the machine learning context, a *data record* typically refers to an individual training example.[2] For the purposes of this monograph, we will make this assumption and deal with datasets D that are a collection of individual data records $x_1, \ldots, x_n$. We will assume that each x_i comes from a universe $\mathcal{X}$ and so $D \in \mathcal{X}^\star$ is an ordered sequence of elements of $\mathcal{X}$.

Differential privacy is defined for *mechanisms* that map input datasets to possible outcomes. In machine learning, the outcomes are typically the weights of a trained AI model. Formally, a mechanism $\mathcal{M} : \mathcal{X}^* \to \mathcal{Y}$ is a randomised function that takes as input a dataset $D \in \mathcal{X}^*$ and returns an output $\mathcal{M}(D) \in \mathcal{Y}$.

Differential privacy (DP) is a formal condition that ensures that the output $\mathcal{M}(D)$ of the mechanism does not leak "too much" information about any one "data unit" in the input dataset D. A data unit can refer to an example, or a set of examples corresponding to a *user*.

At a high level, DP is a stability condition on the mechanism $\mathcal{M}$, i.e., the output $\mathcal{M}(D')$ on a dataset D' obtained by changing D slightly should be *nearly indistinguishable* from $\mathcal{M}(D)$.

To make these ideas concrete, we use a notion of indistinguishability, and hence of DP, called *approximate differential privacy* and *Gaussian DP*. As a default, we take the adjacency relation (i.e., what "changing

[2] Other notions of what constitutes a data record are also sometimes used. For instance, *user-level* differential privacy defines a data record as all training examples containing data from one individual.

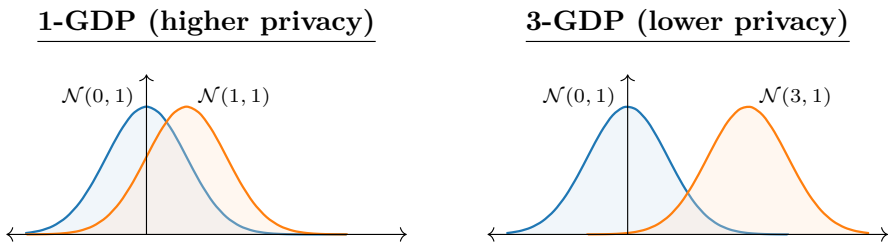


Figure 1.1: The notion of μ-GDP means that distinguishing between the outputs $\mathcal{M}(D)$ and $\mathcal{M}(D')$ of the mechanism for two adjacent datasets D and D' is as hard as distinguishing between $\mathcal{N}(0,1)$ and $\mathcal{N}(\mu,1)$. Smaller μ, as in the left subplot, means that an adversary will be less successful in distinguishing between the two distributions, corresponding to a higher level of privacy.

D slightly" means) to be the so-called "replace-one" relation at the example level.[3]

Definition 1.1 (Replace-One Example Adjacency)**.** We say that two datasets $D, D' \in \mathcal{X}^*$ are **adjacent** (denoted by $D \simeq D'$) in the replace-one notion at the example level if $|D \setminus D'| = |D' \setminus D| = 1$, i.e., D' is obtained from D by replacing one element with another.

Equipped with this notion of adjacency between datasets, we can now formally define one of the most widely used formulations of differential privacy.

Definition 1.2 ((ε, δ)-Differential Privacy)**.** Let $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ be a randomized algorithm, where $\mathcal{R}$ is the output domain. For fixed $\varepsilon > 0$ and $\delta \in [0, 1)$, we say that $\mathcal{M}$ satisfies (ε, δ)-differential privacy if for all measurable sets $S \subset \mathcal{R}$ and for all pairs of adjacent datasets $D, D' \in \mathcal{D}$, it holds that

$$\mathsf{Pr}[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \mathsf{Pr}[\mathcal{M}(D') \in S] + \delta.$$

A number of mechanisms are known to satisfy (ε, δ)-differential privacy, one of the most common being the *Gaussian mechanism* (see

[3]Other notions of adjacency and units of privacy are also of interest. For the most part, the key ideas can be applied directly to those other notions. To keep this monologue concentrated on providing the fundamentals, we defer this discussion to Section 1.7.

Definition 1.5). When analyzing this mechanism, it is often more intuitive to use an equivalent but analytically convenient alternative: *Gaussian differential privacy* (μ-GDP)[4].

Gaussian differential privacy Informally, a mechanism $\mathcal{M}$ is *μ-Gaussian differentially private* (μ-GDP) if for any pair of adjacent datasets $D \simeq D'$, distinguishing between the distribution of $\mathcal{M}(D)$ and $\mathcal{M}(D')$ is *no easier than* distinguishing between the Gaussian distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(\mu,1)$ based on a single sample from each distribution. Thus, μ-GDP is a stronger privacy guarantee for smaller μ, as illustrated in Fig. 1.1. Note that $\mu = 0$ gives perfect indistinguishability (i.e., perfect privacy), $\mu = 1$ gives reasonably good indistinguishability, and $\mu = \infty$ gives no indistinguishability (i.e., no privacy). This intuition is captured formally by the following definition:

Definition 1.3 (Gaussian Differential Privacy)**.** A mechanism $\mathcal{M}$ is *μ-Gaussian differentially private* (μ-GDP) if, for any adjacent datasets $D \simeq D'$, there is a (possibly randomized) function $g : \mathbb{R} \to \mathcal{Y}$ with:[5]

$$g(Z) \stackrel{\mathrm{d}}{=} \mathcal{M}(D) \quad \text{for} \quad Z \sim \mathcal{N}(0,1), \quad \text{and}$$
$$g(Z') \stackrel{\mathrm{d}}{=} \mathcal{M}(D') \quad \text{for} \quad Z' \sim \mathcal{N}(\mu,1),$$

where $X \stackrel{\mathrm{d}}{=} Y$ means random variables X and Y are identically distributed.

Gaussian Mechanism As mentioned above, the Gaussian mechanism is a canonical GDP mechanism: it creates a differentially private version of any deterministic function f by adding zero mean Gaussian noise (with the appropriate variance) to the output of f. This noise is scaled

[4]Throughout this monograph, we use the most convenient formalism depending on context. For instance, μ-GDP is used in Section 1 and Section 2, while (ε,δ)-DP is used in Section 3. Most of our results do not depend on the specific notion of differential privacy used.

[5]$\mathcal{N}(\mu,\nu^2)$ denotes the univariate Gaussian distribution with mean $\mu \in \mathbb{R}$ and variance $\nu^2 > 0$. Its probability density function is $p(z) = \frac{1}{\sqrt{2\pi\nu^2}} e^{-(z-\mu)^2/(2\nu^2)}$.

according to the *sensitivity* of f, which is a measure of how much the output of f is affected by altering a single record in the input dataset:

Definition 1.4. The ℓ_2**-sensitivity** of a vector-valued function $f : \mathcal{X}^* \to \mathbb{R}^m$ is defined as $\mathsf{sens}(f) := \max_{D \simeq D'} \|f(D) - f(D')\|_2$.

The ℓ_2-sensitivity determines how much noise we need to add to $f(D)$ to ensure privacy. This is made formal by the Gaussian mechanism.

Definition 1.5. Given a function $f : \mathcal{X}^* \to \mathbb{R}^m$ and noise multiplier σ, set $\nu = \sigma \cdot \mathsf{sens}(f)$. Then, the **Gaussian mechanism** on input D outputs $f(D) + \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}_m(0, \nu^2)$ is a random vector in $\mathbb{R}^m$ with i.i.d. entries drawn from $\mathcal{N}(0, \nu^2)$.

This recipe remains unchanged if the function f outputs a matrix or a higher order tensor: we conceptually flatten it into a vector, compute its sensitivity, add entry-wise i.i.d. Gaussian noise, and reshape this vector into its original shape. To determine the privacy guarantees of the Gaussian mechanism, we only need to know the *noise multiplier* σ.

Lemma 1.6. For any function $f : \mathcal{X}^* \to \mathbb{R}^m$, the Gaussian mechanism with noise multiplier σ is $1/\sigma$-GDP.

Proof Sketch. In the proof sketch, we only consider a scalar-valued function, f, to highlight the main idea behind the proof. The same proof idea extends to $m > 1$ dimensions by utilizing the rotational invariance of the multivariate standard Gaussian distribution. We cover the full proof in the starred Section 1.9.

Consider the special case of a scalar-valued function f with $m = 1$. Consider a pair of worst-case adjacent datasets $D \simeq D'$ such that $f(D') - f(D) = \mathsf{sens}(f)$. If $f(D)$ and $f(D')$ are closer, it only makes the mechanism outputs $\mathcal{M}(D), \mathcal{M}(D')$ more indistinguishable. We exhibit a function $g : \mathbb{R} \to \mathbb{R}$ such that $\mathcal{M}(D) \overset{\mathrm{d}}{=} g(Z)$ and $\mathcal{M}(D') \overset{\mathrm{d}}{=} g(Z + \mu)$ for $Z \sim \mathcal{N}(0, 1)$, as required by Definition 1.3:

$$g(s) := f(D) \; + \; \sigma \cdot \mathsf{sens}(f) \cdot s \, .$$

Plugging in $s = Z \sim \mathcal{N}(0,1)$, we can verify that $g(Z) \overset{\mathrm{d}}{=} \mathcal{M}(D)$. Instead, with $s = Z + \mu$ for $\mu = 1/\sigma$, we get

$$\begin{aligned} g(Z+\mu) &= f(D) + \sigma \cdot \mathsf{sens}(f)\ (Z+\mu) \\ &= f(D') + \sigma \cdot \mathsf{sens}(f) \cdot Z \overset{\mathrm{d}}{=} \mathcal{M}(D')\,, \end{aligned}$$

since $\mu = 1/\sigma$ and $\mathsf{sens}(f) = f(D') - f(D)$. □

Post-processing One of the important properties of differential privacy is that it is preserved under arbitrary post-processing as long as the post-processing function does not use any part of the confidential data:

> **Lemma 1.7.** Let $\mathcal{M}$ be a randomized algorithm that is μ-GDP. Let g be an arbitrary randomized mapping. Then $g \circ \mathcal{M}(D) := g(\mathcal{M}(D))$ is also μ-GDP.

Post-processing is a fundamental property of DP. It can be used to improve the utility or applicability of differentially private algorithms, such as reducing noise or enforcing domain constraints, without affecting the privacy guarantees.

Components of a DP Guarantee In real-world applications, we should specify three orthogonal components when providing a DP guarantee:[6]

- **The privacy unit** defines what entity's privacy is being protected by fixing the semantics of a one-unit change between D and D'. For example, we could protect the privacy of individual examples in the dataset, or allow a one-unit change to modify all examples derived from a single user or entity.
- **The adjacency relation** formalizes what it means to obtain an adjacent dataset, in the context of a particular unit of privacy. We could replace one unit (as in Definition 1.1), allow the addition or removal of one unit (*add-or-remove* DP), or zero-out the contributions of a unit (Section 1.7).

[6] In practical AI model training, a complete report of the DP guarantee should include additional details of the data access assumed and how the DP guarantee was computed.

- **The indistinguishability notion** is the mathematical formulation used to quantify indistinguishability. Common choices include Gaussian DP, (ε, δ)-DP, and zero-concentrated DP. Tight conversions exist between different indistinguishability notions (for a given privacy unit and adjacency; see the appendix for a brief review).

The fundamental principles discussed in this monograph are broadly applicable across various choices for each of these three components. Unless explicitly stated otherwise, we default to examples as the privacy unit, replace-one as the adjacency unit, and Gaussian DP as the notion of indistinguishability.

1.2 Private Learning via Weighted Prefix Sums

In this monograph, our key primitive will be the DP estimation of weighted prefix sums of a sequence of input vectors. In AI and machine learning, this arises most commonly in the privatization of stochastic gradient descent, where each vector is a single SGD model update.

Stochastic Gradient Descent (SGD) Suppose we wish to find a model $\boldsymbol{\theta} \in \Theta$ from a *parameter space* $\Theta \subset \mathbb{R}^m$ that minimizes the objective, more commonly known as *stochastic optimization*, defined as

$$\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathsf{data}}}[\ell(\boldsymbol{\theta}, \boldsymbol{x})]. \tag{1.1}$$

Here, $\ell(\boldsymbol{\theta}, \boldsymbol{x})$ is the *loss* of making a prediction with model parameters $\boldsymbol{\theta}$ on a datapoint $\boldsymbol{x}$, which is in turn drawn i.i.d. from a data distribution $\mathbb{P}_{\mathsf{data}}$. In practice, we will have access only to a finite sample of datapoints $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{n-1}$ sampled i.i.d. from $\mathbb{P}_{\mathsf{data}}$, leading to an *empirical risk minimization* problem:

$$\min_{\boldsymbol{\theta} \in \Theta} \frac{1}{n} \sum_{i \in [n]} \ell(\boldsymbol{\theta}, \boldsymbol{x}_i)\,, \tag{1.2}$$

where we use shorthand $[n] \coloneqq \{0, \ldots, n-1\}$.

The standard workhorse algorithm used for solving the empirical risk minimization problem is *stochastic gradient descent* (SGD). It is presented in Algorithm 1.1.

Algorithm 1.1 SGD

Inputs: Dataset D, number of steps n, learning rate η

Pick an initial model $\boldsymbol{\theta}_0 \in \Theta$
for $t = 0, \ldots, n-1$ **do**
  Receive a fresh data point $\boldsymbol{x}_t \in D$
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)$
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\, \boldsymbol{g}_t$
Return $\boldsymbol{\theta}_n$

Algorithm 1.2 DP-SGD

Inputs: Inputs D, n, η to SGD, clip norm ζ, noise variance ν^2

Pick an initial model $\boldsymbol{\theta}_0 \in \Theta$
for $t = 0, \ldots, n-1$ **do**
  Receive fresh $\boldsymbol{x}_t \in D$
  $\boldsymbol{g}_t \leftarrow \mathsf{clip}_\zeta\big(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)\big)$
  $\widehat{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t + \mathcal{N}_m\big(0, \nu^2\big)$
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\, \widehat{\boldsymbol{g}}_t$
Return $\boldsymbol{\theta}_n$

Figure 1.2: Stochastic Gradient Descent (SGD) and differentially private stochastic gradient descent (DP-SGD), both with a batch size of 1. Note that DP-SGD clips its gradients to norm at most ζ using the function $\mathsf{clip}_\zeta(\boldsymbol{v}) := \boldsymbol{v} \cdot \min\{1, \zeta / \|\boldsymbol{v}\|_2\}$

> **Remark 1.8.** In AI and machine learning applications, stochastic gradient optimization typically uses gradients computed on mini-batches of examples, and might use a different update rule than the fixed learning rate given in Algorithm 1.1 (for example, momentum or adaptively chosen learning rates). Furthermore, we typically make multiple passes through the data, in contrast to the streaming assumption that each datapoint is processed only once. We will consider these extensions later, in Section 3. The discussions in this chapter directly generalize to larger batches and other first-order optimizers, while the lifting the streaming assumption requires significant extensions, and is the subject of Section 3.

Differentially Private SGD (DP-SGD) Suppose we wish to solve the learning problem in Eq. (1.2) with a differential privacy constraint such as μ-GDP. This is achieved by a differentially private version of SGD, known as DP-SGD, and is contrasted with the non-private version of SGD in Algorithm 1.2.

DP-SGD makes two key modifications to SGD:

- **Gradient Clipping**: In order to restrict the ℓ_2 sensitivity of a data point $\boldsymbol{x}_t$, each gradient $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)$ is clipped so that its ℓ_2

norm is at most some constant $\zeta > 0$. That is, if the gradient has a larger norm, we multiply it by the largest constant $\alpha > 0$ such that $\boldsymbol{g}_t = \alpha \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)$ satisfies $\|\boldsymbol{g}_t\|_2 \leq \zeta$. This is performed by Line 4 of Fig. 1.2 (right) using the following clipping function: for a vector $\boldsymbol{v}$ and $\zeta > 0$

$$\mathsf{clip}_\zeta(\boldsymbol{v}) := \boldsymbol{v} \cdot \min\left\{1, \frac{\zeta}{\|\boldsymbol{v}\|_2}\right\} \tag{1.3}$$

- **Noise Addition**: DP-SGD adds independent zero-mean Gaussian noise $\boldsymbol{z}_t \sim \mathcal{N}_m(0, \nu^2)$ to the (clipped) gradients $\boldsymbol{g}_t$ in Line 5 of Fig. 1.2 (right). Here, the noise variance ν^2 is calibrated to the desired privacy level, e.g. μ-GDP.

The main objective of this monograph is to study a family of correlated noise mechanisms which privatize the gradient $\widehat{\boldsymbol{g}}_t = \boldsymbol{g}_t + \widetilde{\boldsymbol{z}}_t$ using noise $\widetilde{\boldsymbol{z}}_t$ that is correlated across time. That is, $\widetilde{\boldsymbol{z}}_t$ and $\widetilde{\boldsymbol{z}}_\tau$ for $t \neq \tau$ are not required to be statistically independent.

From SGD to Prefix Sum Estimation We begin developing correlated noise mechanisms for learning by framing DP-SGD as a problem of privately estimating prefix sums. For convenience, let us assume that a learning rate $\eta > 0$ is fixed throughout. Unrolling the model parameters updates of non-private SGD yields

$$\boldsymbol{\theta}_t - \boldsymbol{\theta}_0 = -\eta \sum_{\tau=0}^{t-1} \boldsymbol{g}_\tau \,. \tag{1.4}$$

Thus, the sequence of iterates of the SGD algorithm are obtained from the prefix sums $\boldsymbol{s}_t := \sum_{\tau=0}^{t} \boldsymbol{g}_\tau \in \mathbb{R}^m$ of the gradients $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots \in \mathbb{R}^m$.

Now consider two matrices $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{G} \in \mathbb{R}^{n \times m}$ formed by stacking the prefix sums and gradients row-wise, i.e.,

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{s}_0 \\ \boldsymbol{s}_2 \\ \vdots \\ \boldsymbol{s}_{n-1} \end{pmatrix} \in \mathbb{R}^{n \times m} \quad \text{and} \quad \boldsymbol{G} = \begin{pmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_{n-1} \end{pmatrix} \in \mathbb{R}^{n \times m}. \tag{1.5}$$

Then it is easy to check that $\boldsymbol{S}$ can be obtained from $\boldsymbol{G}$ via a linear map $\boldsymbol{S} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{G}$ corresponding to the lower-triangular matrix

$$\boldsymbol{A}_{\mathsf{pre}}[t,\tau] = \begin{cases} 1 & \text{if } t \le \tau \\ 0 & \text{else;} \end{cases} \qquad \text{e.g,} \qquad \boldsymbol{A}_{\mathsf{pre}}^{4\times 4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{1.6}$$

The matrix $\boldsymbol{A}_{\mathsf{pre}}$ is also known as the *prefix sum workload matrix*.

Vanilla SGD with a non-constant learning rate and other first-order optimizers for the empirical risk minimization problem such as SGD with *momentum* lead to different lower triangular workload matrices $\boldsymbol{A}$ that may not necessarily have only binary entries; we give a few more examples in Section 3. Thus, we use $\boldsymbol{A}_{\mathsf{pre}}$ when referring specifically to prefix sums (vanilla SGD), but whenever possible, we state results in terms of a arbitrary lower-triangular workload matrix $\boldsymbol{A}$, allowing the extension to other first-order optimizers and learning rate schedules. We note that prefix sums arise naturally in other contexts such as counting items in a streaming setting or density estimation, as we discuss in Section 1.10.

In this monograph, we are interested in differentially private learning algorithms. This translates to estimating these (weighted) prefix sums with diffential privacy guarantees. We formalize this next.

General Problem: Private Weighted Prefix Sum Estimation Generalizing the above, we can consider the problem of computing differentially private estimates of *weighted* prefix sums $\boldsymbol{s}_0, \dots, \boldsymbol{s}_{n-1}$ of a sequence of vectors representing (functions of) datapoints $\boldsymbol{g}_0, \boldsymbol{g}_1, \dots, \boldsymbol{g}_{n-1} \in \mathbb{R}^m$ weighted by a *workload matrix* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$:

$$\text{Estimate} \quad \boldsymbol{s}_t \coloneqq \sum_{\tau=0}^{t} \boldsymbol{A}[t,\tau]\,\boldsymbol{g}_\tau \qquad \forall t \in [n] \quad \text{with DP.} \tag{1.7}$$

Since only the lower triangular part of $\boldsymbol{A}$ shows up in Eq. (1.7), we assume without loss of generality that $\boldsymbol{A}$ is a lower triangular matrix, so equivalently we may write $\boldsymbol{s}_t = (\boldsymbol{A}\boldsymbol{G})[t,:]$ and our goal is to estimate the rows of $\boldsymbol{S} = \boldsymbol{A}\boldsymbol{G}$ with DP. Recall that, in the case of vanilla SGD, $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ while it can have different form for other first-order optimizers. Finally, we assume throughout that the inputs $(\boldsymbol{g}_t)_{t=0}^{n-1}$ are bounded in

ℓ_2 norm:

$$\|\boldsymbol{g}_t\|_2 \leq 1\,. \tag{1.8}$$

In the context of DP-SGD, this is equivalent to taking the clip norm $\zeta = 1$ in Eq. (1.3). This assumption is without loss of generality: since the output $\boldsymbol{s}_t$ in Eq. (1.7) is a linear function of $\boldsymbol{g}_t$, we can take $\|\boldsymbol{g}_t\|_2 \leq 1$ and scale $\boldsymbol{s}_t$ by ζ instead.

Remark 1.9. We will give an alternative (though equivalent) perspective in Section 1.4.3: we can view the problem as providing DP estimates $\widehat{\boldsymbol{g}}_t = \boldsymbol{g}_t + \widetilde{\boldsymbol{z}}_t$ of the $\boldsymbol{g}_t$, with (correlated) noise $\widetilde{\boldsymbol{z}}_t$ chosen such that beneficial cancellation occurs when we compute $\boldsymbol{A}\widehat{\boldsymbol{G}}$ as post processing.

Remark 1.10. Though the matrix $\boldsymbol{A}$ is flexible enough to describe more complicated first-order optimizers and learning rate schedules, we remark that this can be achieved either by directly describing the algorithm in $\boldsymbol{A}$ as described above or by using the standard prefix sums $\boldsymbol{A}_{\mathsf{pre}}$ with the corresponding choice of inner optimizer. This can lead to different privacy-utility Pareto frontiers for machine learning applications as may be observed in Section 3.

1.3 Correlated Noise Mechanisms

We now turn our attention to the main focus of this monograph: correlated noise mechanisms. Such mechanisms can be described as producing differentially private estimates of the prefix sums $\boldsymbol{AG}$.

In particular, given a factorization $\boldsymbol{A} = \boldsymbol{BC}$ of the workload matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we define a **correlated noise mechanism** as

$$\mathcal{M}(\boldsymbol{G}) := \boldsymbol{B}\left(\boldsymbol{CG} + \boldsymbol{Z}\right) = \boldsymbol{AG} + \boldsymbol{BZ}\,, \tag{1.9}$$

where $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ are lower triangular matrices, and $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \nu^2)$ is an $n \times m$ matrix of component-wise i.i.d. Gaussian noise with appropriate variance ν^2.

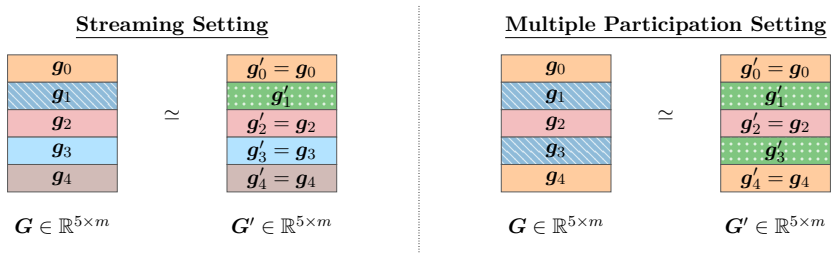


Figure 1.3: Illustration of adjacency of gradient sequences $\boldsymbol{g}_t = \nabla\ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)$ and $\boldsymbol{g}'_t = \nabla\ell(\boldsymbol{\theta}_t, \boldsymbol{x}'_t)$, where equal datapoints are indicated by having the same color cell. Following Theorem 1.11, we assume gradients are computed in the non-adaptive setting for a fixed sequence $(\boldsymbol{\theta}_t)_{t=1}^{n-1}$. **Left**: In the streaming setting, each datapoint participates only once. Thus, the sequences of gradients on adjacent datasets differ in only one item (in this example, we have $\boldsymbol{g}_1 \neq \boldsymbol{g}'_1$) and are equal everywhere else. **Right**: In the multiple-participation setting considered in Section 3, each datapoint can participate multiple times (e.g. by taking mulitple passes through the dataset). Thus, sequences of gradients on adjacent datasets can differ in multiple indices. In this example, the blue datapoint is replaced by the green one, changing both $\boldsymbol{g}_1$ and $\boldsymbol{g}_3$—note that the same (changed) datapoint is used for both $\boldsymbol{g}_1$ and $\boldsymbol{g}_3$. In this section, we focus on the streaming setting.

Correlated Noise for DP-SGD To map these correlated noise mechanisms directly to DP-SGD (Fig. 1.2, right), we can equivalently write Eq. (1.9) as

$$\mathcal{M}(\boldsymbol{G}) = \boldsymbol{A}(\boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z})\,, \tag{1.10}$$

assuming the matrix $\boldsymbol{C}$ is invertible. This equation involves two components: (1) computing the noisy gradients $\widehat{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}$, and (2) multiplying it by the workload matrix $\boldsymbol{A}$ to return $\boldsymbol{A}\widehat{\boldsymbol{G}}$.

First, the noisy gradients $\widehat{\boldsymbol{g}}_t = \widehat{\boldsymbol{G}}[t,:] = \boldsymbol{g}_t + \widetilde{\boldsymbol{z}}_t$ are computed by injecting the correlated noise

$$\widetilde{\boldsymbol{z}}_t \coloneqq (\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:] = \sum_{\tau=0}^{t} (\boldsymbol{C}^{-1})[t,\tau]\,\boldsymbol{z}_\tau\,, \tag{1.11}$$

where $\boldsymbol{z}_t \sim \mathcal{N}_m(0, \nu^2)$ is i.i.d. seed noise. This is a direct replacement to the i.i.d. noise update in DP-SGD (Line 5 in Algorithm 1.2).

Second, multiplication by the matrix $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ (for the case of vanilla SGD) is achieved by the gradient step $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\widehat{\boldsymbol{g}}_t$ (Line 6

Algorithm 1.3 DP-SGD with Correlated Noise

Inputs: Dataset D, number of steps n, learning rate η, clip norm ζ, noise variance ν^2, noise correlating matrix $\boldsymbol{C}^{-1}$ (standard DP-SGD uses $\boldsymbol{C}^{-1} = \boldsymbol{I}_{n\times n}$)

Pick an initial model $\boldsymbol{\theta}_0 \in \Theta$
for t **in** $0, 1, \ldots, n-1$ **do**
  Receive a fresh data point $\boldsymbol{x}_t \in D$
  $\boldsymbol{g}_t \leftarrow \mathsf{clip}(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t; \boldsymbol{x}_t))$ ▶ $\mathsf{clip}(\boldsymbol{v}) := \boldsymbol{v} \cdot \min\{1, \zeta / \|\boldsymbol{v}\|_2\}$
  Sample i.i.d. seed noise $\boldsymbol{z}_t \sim \mathcal{N}_m(0, \nu^2)$
  Calculate the correlated noise

$$\widetilde{\boldsymbol{z}}_t \leftarrow \sum_{\tau=0}^{t} (\boldsymbol{C}^{-1})[t, \tau]\, \boldsymbol{z}_\tau$$

  $\widehat{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t + \widetilde{\boldsymbol{z}}_t$
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \widehat{\boldsymbol{g}}_t$

in Algorithm 1.2); this can be established by unrolling the noisy SGD update, similar to Eq. (1.4). This gradient step is usually implemented programmatically using an optimizer step. The resulting algorithm is given in Algorithm 1.3. Here, the matrix $\boldsymbol{C}$ is also known as the *strategy matrix* or *encoder matrix*, while the matrix $\boldsymbol{C}^{-1}$ is known as the *noise correlating matrix*.

1.3.1 Privacy Guarantees of Correlated Noise Mechanisms

Recall that two datasets D and D' satisfy replace-one example adjacency if D' can be obtained by replacing one element of D, as per Definition 1.1. Since the correlated noise mechanism in Eq. (1.9) takes in the gradients as input, we need to reason about how adjacency of the underlying datasets affects the corresponding (clipped) gradients $\boldsymbol{g}_t = \mathsf{clip}(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t))$ and $\boldsymbol{g}'_t = \mathsf{clip}(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}'_t))$ computed using $\boldsymbol{x}_t \in D$ and $\boldsymbol{x}'_t \in D'$ respectively for $t = 0, \ldots, n-1$.

Suppose D and D' differ in the t^{th} example where $\boldsymbol{x}_t \neq \boldsymbol{x}'_t$. This affects not only the gradient $\boldsymbol{g}_t$ in step t, but also *all subsequent gradients* $\boldsymbol{g}_{t+1}, \ldots, \boldsymbol{g}_{n-1}$ via the updated model parameters $\boldsymbol{\theta}_{t+1}$. Fortunately, for

Adaptive Setting	Non-adaptive Setting
Choose $\boldsymbol{\theta}_0 \in \Theta$	Choose $\boldsymbol{\theta}_0 \in \Theta$.
Let $f_0, \dots, f_{n-1}$ be functions $f_t : \mathcal{X}^* \times \Theta^* \to \mathbb{R}^m$.	Let $f_0, \dots, f_{n-1}$ be functions $f_t : \mathcal{X}^* \times \Theta^* \to \mathbb{R}^m$.
for $t = 0, \dots, n-1$ **do**	Fix a sequence from Θ^{n-1} $(\boldsymbol{\theta}_t)_{t=1}^{n-1} \coloneqq (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{n-1})$
$\boldsymbol{f}_t = f_t\big(D, (\boldsymbol{\theta}_\tau)_{\tau=0}^t\big)$	**for** $t = 0, \dots, n-1$ **do**
$\widehat{\boldsymbol{f}}_t = \boldsymbol{f}_t + \mathcal{N}_m\big(0, \nu^2\big)$	$\boldsymbol{f}_t = f_t\big(D, (\boldsymbol{\theta}_\tau)_{\tau=0}^t\big)$
$\boldsymbol{\theta}_{t+1} \leftarrow$ an arbitrary function of $(\widehat{\boldsymbol{f}}_\tau)_{\tau=0}^t$ and $(\boldsymbol{\theta}_\tau)_{\tau=0}^t$	$\widehat{\boldsymbol{f}}_t = \boldsymbol{f}_t + \mathcal{N}_m\big(0, \nu^2\big)$
Return $\widehat{\boldsymbol{F}}_a \coloneqq \big(\widehat{\boldsymbol{f}}_0, \dots, \widehat{\boldsymbol{f}}_{n-1}\big)$	**Return** $\widehat{\boldsymbol{F}} \coloneqq \big(\widehat{\boldsymbol{f}}_0, \dots, \widehat{\boldsymbol{f}}_{n-1}\big)$

Figure 1.4: Adaptive and non-adaptive iterative procedures with state denoted by $\boldsymbol{\theta}_t$. DP-SGD can be expressed as an instance of the adaptive setting; however, Theorem 1.11 shows that we achieve the same GDP guarantee in the non-adaptive setting, which greatly simplifies privacy analysis.

the purposes of privacy analysis, it turns out that we can ignore the effect of of $\boldsymbol{x}_t$ on all future gradients $\boldsymbol{g}_{t+1}, \dots, \boldsymbol{g}_{n-1}$. In particular, as we discuss next, the privacy analysis of a correlated noise mechanism coincides with a similar non-adaptive procedure.

Adaptive vs. Non-Adaptive GDP Fig. 1.4 gives an abstract iterative procedure with state denoted by $\boldsymbol{\theta}_t$. The full sequence of states $\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{n-1}$ is fixed ahead of time in the non-adaptive setting. In contrast, the state $\boldsymbol{\theta}_{t+1}$ in the adaptive setting is allowed to depend arbitrarily on all *past* information, including the previous states $\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_t$ as well as the privatized outputs $\widehat{\boldsymbol{f}}_0, \dots, \widehat{\boldsymbol{f}}_t$. In both settings, the state $\boldsymbol{\theta}_t$ is used to compute a data-dependent output $\boldsymbol{f}_t$, whose privatized version $\widehat{\boldsymbol{f}}_t$ (via the Gaussian mechanism) is then released.

The following general theorem shows that the privacy analysis with Gaussian additive noise in the adaptive case can be reduced to the non-adaptive case, though take care to note that analogous result does not hold for all mechanisms. This result underpins the privacy analysis of all the algorithms in this monograph:

> **Theorem 1.11** (Adaptive vs. Non-Adaptive Gaussian mechanism)**.** Consider the two mechanisms defined in Fig. 1.4 with functions

$f_0, \ldots, f_{n-1} : \mathcal{X}^* \times \Theta^* \to \mathbb{R}^m$, whose (combined) ℓ_2-sensitivity as a function of the first argument is bounded by s for *any fixed inputs* to the second argument as:

$$\sup_{\boldsymbol{\theta}_0,\ldots,\boldsymbol{\theta}_{n-1}\in\Theta} \sup_{D\simeq D'} \left\| F\big(D, (\boldsymbol{\theta}_t)_{t=0}^{n-1}\big) - F\big(D', (\boldsymbol{\theta}_t)_{t=0}^{n-1}\big) \right\|_{\mathrm{F}} \le s\,, \quad \text{with}$$

$$F\big(D, (\boldsymbol{\theta}_t)_{t=0}^{n-1}\big) \coloneqq \begin{bmatrix} f_0(D, \boldsymbol{\theta}_0) & \cdots & f_{n-1}\big(D, (\boldsymbol{\theta}_t)_{t=0}^{n-1}\big) \end{bmatrix}^\top \in \mathbb{R}^{n\times m}$$

denoting a matrix whose rows are the outputs of $f_0, \ldots, f_{n-1}$, and $\|\boldsymbol{M}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} \boldsymbol{M}[i,j]}$ denotes the Frobenius norm of the matrix $\boldsymbol{M}$.[7] Then, for any fixed noise variance $\nu^2 > 0$, the output $\widehat{\boldsymbol{F}}_a$ of the adaptive setting satisfies the same GDP guarantee as the output $\widehat{\boldsymbol{F}}$ of the non-adaptive setting.

Let $\mathsf{clip}(\cdot)$ denote the clipping funtion $\mathsf{clip}_\zeta(\cdot)$ defined in Eq. (1.3) instantiated with $\zeta = 1$. Then Theorem 1.11 captures DP-SGD with independent noise (Algorithm 1.2): it is simply an adaptive procedure where the function f_t denotes a computation of the clipped gradient

$$f_t\big(D, (\boldsymbol{\theta}_\tau)_{\tau=0}^{t}\big) = \mathsf{clip}\big(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)\big)\,,$$

while the state update is the gradient update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\, \widehat{\boldsymbol{f}}_t$. In particular, note that $f_t\big(D, (\boldsymbol{\theta}_\tau)_{\tau=0}^{t}\big)$ depends only on current model $\boldsymbol{\theta}_t$ and not on the previous θ_τ's for $\tau < t$. We need the additional generality of Theorem 1.11 to establish the privacy guarantee of Algorithm 1.3, as we will momentarily see in Theorem 1.14.

In the adaptive setting, changing one example $\boldsymbol{x}_t$ can influence all of the following gradients, $\boldsymbol{g}_t, \ldots, \boldsymbol{g}_{n-1}$. Fortunately, Theorem 1.11 allows us to instead analyze the privacy properties of the corresponding non-adaptive setting; here, by design, changing the data point $\boldsymbol{x}_t$ can *only* change the gradient $\boldsymbol{g}_t$, while all other gradients are unchanged.

Hence, we can easily extend the definition of adjacency to sequences of gradients by assuming the non-adaptive setting. In the streaming setting where each example is processed only once, we say that two sequences of gradients $\boldsymbol{G} = (\boldsymbol{g}_t)_{t=0}^{n-1}$ and $\boldsymbol{G}' = (\boldsymbol{g}'_t)_{t=0}^{n-1}$ are adjacent if we

[7] This is the worst-case ℓ_2 sensitivity in the non-adaptive setting. The Frobenius norm of a matrix is exactly the $\|\cdot\|_2$ norm of the vector obtained by flattening it, and is used to compute the ℓ_2 sensitivity of matrix-valued maps.

have that $\boldsymbol{g}_\tau = \boldsymbol{g}'_\tau$ for all $\tau \in [n]$, except possibly at some step t (where $\boldsymbol{x}_\tau \neq \boldsymbol{x}'_\tau$).

GDP Bound of Correlated Noise Mechanisms We are now ready to tackle the privacy analysis of correlated noise mechanisms.

In order to apply Theorem 1.11, our first step will be to derive a privacy analysis for the correlated noise mechanism of Eq. (1.9) in the non-adaptive setting. The key ingredient is bounding the sensitivity induced by the strategy matrix:

$$\mathsf{sens}(\boldsymbol{C}) \coloneqq \mathsf{sens}(\boldsymbol{G} \mapsto \boldsymbol{C}\boldsymbol{G}) = \max_{\boldsymbol{G} \simeq \boldsymbol{G}'} \left\|\boldsymbol{C}(\boldsymbol{G} - \boldsymbol{G}')\right\|_{\mathrm{F}} . \tag{1.12}$$

(Since the map $\boldsymbol{G} \mapsto \boldsymbol{C}\boldsymbol{G}$ returns a matrix, its ℓ_2-sensitivity is computed using the Frobenius norm.) This quantity is tightly related to the maximum column norm of $\boldsymbol{C}$:

$$\|\boldsymbol{C}\|_{\mathsf{col}} \coloneqq \max_{t \in [n]} \|\boldsymbol{C}[:, t]\|_2 .$$

Lemma 1.12. Under replace-one-example adjacency (Definition 1.1) with gradients clipped to norm 1 (cf. Eq. (1.8)), we have

$$\mathsf{sens}(\boldsymbol{C}) = 2 \cdot \|\boldsymbol{C}\|_{\mathsf{col}} .$$

Proof. Suppose $\boldsymbol{G}, \boldsymbol{G}'$ differ in the t^{th} row (cf. Fig. 1.5). Recall that $\boldsymbol{G}$ ($\boldsymbol{G}'$, respectively) is formed by stacking the vectors $\boldsymbol{g}_t \in \mathbb{R}^n$ ($\boldsymbol{g}'_t \in \mathbb{R}^n$, respectively) row-wise. Therefore $\boldsymbol{C}(\boldsymbol{G} - \boldsymbol{G}') = \boldsymbol{C}[:, t](\boldsymbol{g}_t - \boldsymbol{g}'_t)^\top \in \mathbb{R}^{p \times n}$. Since each row of $\boldsymbol{G}, \boldsymbol{G}'$, i.e., $\boldsymbol{g}_t, \boldsymbol{g}'_t$ for all $t \in [n]$ is bounded in ℓ_2 norm, we have the ℓ_2 norm of their difference, $\boldsymbol{\delta}_t \coloneqq \boldsymbol{g}_t - \boldsymbol{g}'_t$, bounded as $\|\boldsymbol{\delta}_t\|_2 \leq 2$ by the triangle inequality. Thus, we have

$$\begin{aligned} \mathsf{sens}(\boldsymbol{C}) = \max_{\boldsymbol{G} \simeq \boldsymbol{G}'} \left\|\boldsymbol{C}(\boldsymbol{G} - \boldsymbol{G}')\right\|_{\mathrm{F}} &= \max_{t \in [t],\ \|\boldsymbol{\delta}_t\|_2 \leq 2} \left\|\boldsymbol{C}[:, t]\, \boldsymbol{\delta}_t^\top\right\|_{\mathrm{F}} \\ &\overset{(*)}{=} \max_{t \in [t]} \|\boldsymbol{C}[:, t]\|_2 \cdot \max_{\|\boldsymbol{\delta}\|_2 \leq 2} \|\boldsymbol{\delta}\|_2 \;=\; 2 \cdot \|\boldsymbol{C}\|_{\mathsf{col}} , \end{aligned} \tag{1.13}$$

where $(*)$ follows from using $\|\boldsymbol{u}\boldsymbol{v}^\top\|_{\mathrm{F}} = \|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2$ for any vectors $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^m$:

$$\left\|\boldsymbol{u}\boldsymbol{v}^\top\right\|_{\mathrm{F}}^2 = \sum_{i \in [n]} \sum_{j \in [m]} \left(\boldsymbol{u}[i]\, \boldsymbol{v}[j]\right)^2 = \sum_{i \in [n]} \boldsymbol{u}[i]^2 \sum_{j \in [m]} \boldsymbol{v}[j]^2 = \|\boldsymbol{u}\|_2^2 \, \|\boldsymbol{v}\|_2^2 .$$

This completes the proof of Lemma 1.12. □

We will see in the starred Section 1.7 that the constant 2 in Lemma 1.12 is specific to the replace-one-example adjacency.

With this lemma in hand, the privacy result for the non-adaptive streaming case is straightforward:

Lemma 1.13. Fix a noise multiplier $\sigma > 0$. Consider the replace-one-example adjacency (Definition 1.1) of gradients $\boldsymbol{G}, \boldsymbol{G}' \in \mathbb{R}^{n \times m}$ in the streaming non-adaptive setting, i.e.

(a) any adjacent $\boldsymbol{G} \simeq \boldsymbol{G}'$ satisfy $\boldsymbol{g}_\tau = \boldsymbol{g}'_\tau$ for all rows $\tau \in [n]$ except possibly at some row $t \in [n]$, and

(b) the rows $\boldsymbol{g}_t, \boldsymbol{g}'_t$ are clipped to norm 1 (cf. Eq. (1.8)).

Then, the mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z})$ for any matrices $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$ (possibly non-lower-triangular and non-invertible) and i.i.d. Gaussian noise $\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(0, \nu^2)$ satisfies $\frac{1}{\sigma}$-GDP if we choose the noise standard deviation as $\nu = 2\sigma \|\boldsymbol{C}\|_{\mathsf{col}}$.

Proof. The mechanism $\mathcal{M}$ is a simple post-processing of the mechanism

$$\mathcal{M}'(\boldsymbol{G}) \coloneqq \boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}, \tag{1.14}$$

as $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B} \cdot \mathcal{M}'(\boldsymbol{G})$. So it suffices to prove the GDP guarantee for $\mathcal{M}'$. We have (after flattening the matrices to vectors), that $\mathcal{M}'$ is instance of the Gaussian mechanism (Definition 1.5) and so the result follows from Lemma 1.6 using the sensitivity bound from Lemma 1.12. □

Finally, we extend this non-adaptive privacy guarantee to DP-SGD with correlated noise:

Theorem 1.14. Fix a noise multiplier σ and consider Algorithm 1.3 with an invertible lower triangular strategy matrix $\boldsymbol{C}$, clip norm $\zeta = 1$, and noise standard deviation $\nu = 2\sigma \|\boldsymbol{C}\|_{\mathsf{col}}$. Then, the privatized gradients $(\widehat{\boldsymbol{g}}_t)_{t \in [n]}$ and iterates $(\boldsymbol{\theta}_t)_{t \in [n]}$ produced by Algorithm 1.3 satisfy $\frac{1}{\sigma}$-GDP under replace-one-example adjacency

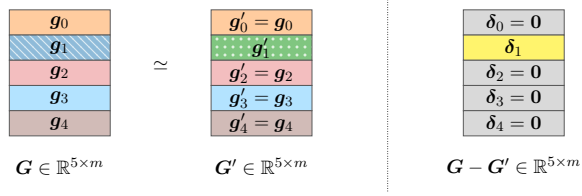


Figure 1.5: For two adjacent sequences of gradients $\boldsymbol{G} \simeq \boldsymbol{G}'$ in the streaming setting, we have that $\boldsymbol{G} - \boldsymbol{G}'$ is non-zero only for one row.

in the streaming setting.

Proof. We instantiate Algorithm 1.3 as an adaptive iterative process in the framework of Theorem 1.11. Concretely, we take f_t to be functions which return the rows of $\boldsymbol{CG}$:

$$f_t(D, (\boldsymbol{\theta}_\tau)_{\tau=0}^t) = (\boldsymbol{CG})[t,:] \quad \text{where} \quad \boldsymbol{g}_\tau = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_\tau, \boldsymbol{x}_\tau) \tag{1.15}$$

for $\boldsymbol{x}_\tau \in D$. Since $\boldsymbol{C}$ is a lower triangular matrix, we have that $(\boldsymbol{CG})[t,:]$ depends only on the previous gradients $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_t$, which in turn depend (only) on $(\boldsymbol{\theta}_0, \cdots, \boldsymbol{\theta}_t)$ and D. It follows then that $\widehat{\boldsymbol{f}}_t = (\boldsymbol{CG} + \boldsymbol{Z})[t,:]$. We instantiate the state update of Algorithm 1.3 with

$$\widehat{\boldsymbol{g}}_t \leftarrow (\boldsymbol{C}^{-1}\widehat{\boldsymbol{F}}_a)[t,:] = (\boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]\,,$$
$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\widehat{\boldsymbol{g}}_t.$$

Since $\boldsymbol{C}^{-1}$ is lower-triangular, $\widehat{\boldsymbol{g}}_t$ is post-processing of $\widehat{\boldsymbol{f}}_0, \ldots, \widehat{\boldsymbol{f}}_t$, the first t rows of $\widehat{\boldsymbol{F}}_a$, and so this is a valid update. By the same post-processing argument, the privatized gradients $(\widehat{\boldsymbol{g}}_t)_{t\in[n]}$ and iterates $(\boldsymbol{\theta}_t)_{t\in[n]}$ satisfy the same privacy guarantee as $\widehat{\boldsymbol{F}}_a$. Thus, with this choice of the functions f_t (and appropriate update), we have that Algorithm 1.3 is in fact an instance of the adaptive setting of Fig. 1.4.

Hence, applying Theorem 1.11, it remains to analyze this choice of f_t in the non-adaptive setting. By construction, the output of Fig. 1.4 in the non-adaptive setting is

$$\widehat{\boldsymbol{F}} = \boldsymbol{CG} + \boldsymbol{Z}\,.$$

This is exactly the mechanism analyzed by Lemma 1.13 taking $\boldsymbol{B} = \boldsymbol{I}$, and the conditions (a)-(b) are satisfied by the assumption of replace-one-example adjacency, the choice of clipping parameter $\zeta = 1$, and

non-adaptivity. Finally, the choice of ν matches, and so the result follows. □

1.3.2 Baseline Correlated Noise Mechanisms

The family of correlated noise mechanisms defined by Eq. (1.9) includes two natural baselines to privately estimate the weighted prefix sums as in Eq. (1.7): adding noise to the input or the output.

Baseline I: Input Perturbation The first baseline corresponds to the strategy matrix $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$, which is the choice taken by standard DP-SGD in Algorithm 1.2. Here, we *privatize each input* with the Gaussian mechanism $\widehat{\boldsymbol{g}}_t = \boldsymbol{g}_t + \boldsymbol{z}_t$ with $\boldsymbol{z}_t \sim \mathcal{N}_m(0, \nu^2)$ for some standard deviation ν. Then, it follows that the post-processed (weighted) prefix sums

$$\widehat{\boldsymbol{s}}_t = \sum_{\tau=0}^{t} \boldsymbol{A}[t,\tau]\, \widehat{\boldsymbol{g}}_\tau = \sum_{\tau=0}^{t} \boldsymbol{A}[t,\tau]\, \boldsymbol{g}_\tau + \sum_{\tau=0}^{t} \boldsymbol{A}[t,\tau]\, \boldsymbol{z}_\tau \tag{1.16}$$

satisfy the *same* DP guarantees as the privatized inputs $\widehat{\boldsymbol{g}}_t$ for all choices of the workload matrix $\boldsymbol{A}$. In particular, taking the standard deviation $\nu = 2\sigma$ yields the desired $(1/\sigma)$-GDP guarantee.[8]

Baseline II: Output Perturbation The second baseline corresponds to $\boldsymbol{C} = \boldsymbol{A}$ (so that the matrix $\boldsymbol{B} = \boldsymbol{I}_{n\times n}$). This is equivalent to *privatizing each output* directly with the Gaussian mechanism to release

$$\widehat{\boldsymbol{s}}_t = \left(\sum_{\tau=0}^{t} \boldsymbol{A}[t,\tau]\, \boldsymbol{g}_\tau \right) + \boldsymbol{z}_\tau \tag{1.17}$$

for $\boldsymbol{z}_\tau \sim \mathcal{N}_m(0, \nu^2)$ with component-wise standard deviation ν. Unlike the input perturbation baseline, ν depends on the choice of the workload matrix $\boldsymbol{A}$ as it affects the sensitivity of the operation $\boldsymbol{G} \mapsto \boldsymbol{AG}$. For example, with the prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$, Lemma 1.12 yields that $\mathsf{sens}(\boldsymbol{A}_{\mathsf{pre}}) = 2\sqrt{n}$. Thus, to obtain a $(1/\sigma)$-GDP guarantee on the

[8] Again, the factor of 2 in the standard deviation ν here and in the rest of this section is specific to the replace-one notion of adjacency. This vanishes with other notions of adjacency, as we discuss in Section 1.7.

noisy prefix sums $\widehat{\boldsymbol{s}}_t$ in this case, we need a significantly larger standard deviation of $\nu = 2\sqrt{n}/\mu$ compared to the input perturbation baseline.

Thus, the best correlated noise mechanism (for a given objective) is no worse than either of the two baselines. However, an important question remains: can general correlated noise mechanisms (with $\boldsymbol{B}, \boldsymbol{C} \neq \boldsymbol{I}_{n\times n}$) *strictly* improve over the baselines? In other words, is there a strong reason to choose *non-trivial* noise correlations across time steps?

1.4 Why Correlated Noise Mechanisms?

We now develop some intuition regarding why correlated noise mechanisms might lead to improved privacy-utility tradeoffs for DP prefix sum estimation.

In particular, if $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ are the true prefix sums and $\widehat{\boldsymbol{s}}_1, \ldots, \widehat{\boldsymbol{s}}_n$ are the outputs of a correlated noise mechanism, then we will argue that the maximum prefix sum loss

$$\max_{t\in[n]} \mathbb{E}\,\|\widehat{\boldsymbol{s}}_t - \boldsymbol{s}_t\|_2^2$$

is smaller than the baseline described in the previous section at each privacy level.

1.4.1 Why Estimate Prefix Sums Directly?

We consider a simple example in Fig. 1.6 comparing a non-trivial correlated noise mechanism to the input perturbation mechanism, where $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$ (i.e., DP-SGD, which uses i.i.d. noise). Consider minimization of the simple linear function $f(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = -\boldsymbol{\theta}_1$ in $m = 2$ dimensions over the bounded set $\Theta = \{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) : \boldsymbol{\theta}_1^2 + \boldsymbol{\theta}_2^2 \leq M\}$, for a sufficiently large number M. At each step of the optimization, the gradient of the objective function is $\nabla f(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = (-1, 0)$.

The top plot in Fig. 1.6 illustrates (noisy) gradient descent with a learning rate of $\eta = 1$ under different noise conditions for $n = 40$ steps starting from $\boldsymbol{\theta}_0 = (0, 0)$. The black arrows depict the noiseless case, which produces iterates $\boldsymbol{\theta}_t^{\mathsf{GD}} = (t, 0)$ as expected. The green arrows

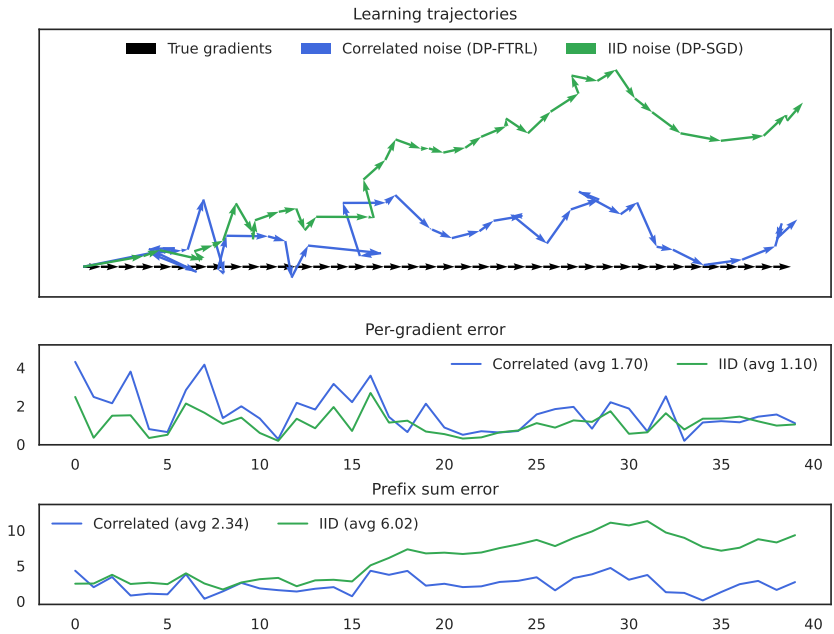


Figure 1.6: Example trajectories for learning with correlated noise vs. i.i.d. noise for $n = 40$ steps in $m = 2$ dimensions. This figure considers a simple example where the black line is the non-private gradient descent baseline with learning rate $\eta = 1$, where the true gradients obtained at each step is $\boldsymbol{g}_t = (-1, 0)$. The blue line uses a correlated noise mechanism (optimized to achieves the best worst-case performance in the prefix sums; cf. Section 2.2) and the green line corresponds to i.i.d. noise via DP-SGD (i.e., the input perturbation mechanism with $\boldsymbol{C} = \boldsymbol{C}^{-1} = \boldsymbol{I}_{n \times n}$). We calibrate these mechanisms to represent equivalent privacy guarantees at any given noise level as per Lemma 1.13; here we use Gaussian noise with $\nu = 1$. The middle plot shows that i.i.d. noise is better at estimating individual gradients (1.10 vs 1.70 average root mean squared error in estimating the individual gradients $(-1, 0)$). However, correlated noise results in a trajectory that on average stays closer to the baseline trajectory, thanks to lower error in prefix sum estimates shown in the bottom plot (2.34 vs 6.02 average root mean squared error in estimating the prefix sums $(-t, 0)$ for $t \in [40]$).

represent the sample path obtained under i.i.d. noise:

$$\boldsymbol{\theta}_t^{\mathsf{IID}} = \left(t + \sum_{\tau=0}^{t-1} w_{\tau,1}, \sum_{\tau=0}^{t-1} w_{\tau,2} \right), \quad \text{where} \quad \boldsymbol{z}_\tau \sim \mathcal{N}_2(0,1).$$

The blue arrows depict that of a correlated noise mechanism corresponding to a certain decomposition $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{B}_\star \boldsymbol{C}_\star$ (described in Section 2.2):

$$\boldsymbol{\theta}_t^{\mathsf{corr}} = (t, 0) + (\boldsymbol{B}_\star \boldsymbol{Z})[t, :].$$

where $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \|\boldsymbol{C}_\star\|_{\mathsf{col}}^2)$. Note that the noise is scaled according to Lemma 1.13 so that both mechanisms represent equivalent privacy guarantees. We can then calculate:

$$\begin{aligned} \boldsymbol{\theta}_t^{\mathsf{IID}} - \boldsymbol{\theta}_t^{\mathsf{GD}} &\sim \mathcal{N}_2(\mathbf{0}, t), \quad \text{and} \\ \boldsymbol{\theta}_t^{\mathsf{corr}} - \boldsymbol{\theta}_t^{\mathsf{GD}} &\sim \mathcal{N}_2\left(\mathbf{0}, \|\boldsymbol{C}_\star\|_{\mathsf{col}}^2 \|\boldsymbol{B}_\star[t, :]\|_2^2\right). \end{aligned}$$

The correlated noise mechanism can yield a trajectory that is closer in the worst-case to the noiseless black trajectory (in expectation) if we can have

$$\max_{t\in[n]} \|\boldsymbol{C}_\star\|_{\mathsf{col}}^2 \|\boldsymbol{B}_\star[t, :]\|^2 \le \max_{t\in[n]} \; t = n - 1 \tag{1.18}$$

for some factorization $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{B}_\star \boldsymbol{C}_\star$. This is indeed possible, and we will see a quantitative example next.

The reason behind this improvement is the direct estimation of prefix sums rather than the gradients. Indeed, while i.i.d. noise is the optimal strategy to minimize the total squared error in the noisy *gradients* $\widehat{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}$, correlated noise mechanisms are able to produce better estimates of the prefix sums $\boldsymbol{A}\widehat{\boldsymbol{G}} = \boldsymbol{A}\boldsymbol{G} + \boldsymbol{B}\boldsymbol{Z}$, resulting in better worst-case noisy *trajectories*.

1.4.2 Quantitative Benefits of Using Correlated Noise

We now build on the example of Section 1.4.1 by constructing a simple correlated noise mechanism that outperforms the input-perturbation and output-perturbation baselines. For simplicity, we compare all three approaches under a fixed privacy requirement of 1-GDP. (As we will see, our calculations will actually hold for any level of μ-GDP.)

Computing the gradient descent iterates in the previous setting reduces to estimating prefix sums with differential privacy, Problem 1.7 for the prefix-sum matrix $\boldsymbol{A}_{\mathsf{pre}}$, as in Eq. (1.6). For simplicity, suppose that the dimension of the stream is $m = 1$, that is $\boldsymbol{g} = (\boldsymbol{g}_0, \dots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^n$.

Next, we need to quantify exactly how good our private prefix sum estimates are. As we saw in Eq. (1.18), a reasonable objective is to minimize the maximum squared error between the private estimate $\widehat{s}_t = \mathcal{M}(\boldsymbol{g})[t]$ and the actual prefix sum s_t.[9] Formally, we wish to minimize

$$\max_{t\in[n]} \mathbb{E}(\widehat{s}_t - s_t)^2 = \max_{t\in[n]} \mathbb{E}\Big(\left(\boldsymbol{B}\boldsymbol{z}\right)[t,:]\Big)^2 = \nu^2 \max_{t\in[n]} \|\boldsymbol{B}[t,:]\|_2^2\,, \quad (1.19)$$

the maximum squared row norm of $\boldsymbol{B}$ scaled by the noise variance necessary to achieve 1-GDP. Note that we have dropped the dependence on $\boldsymbol{C}$ via Theorem 1.14.

To build some intuition why correlated noise helps, let us consider a simple family of correlated noise mechanisms parameterized by $0 \le \lambda < 1$ with

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 & \dots & \dots & 0 \\ \lambda & 1 & 0 & \ddots & \ddots & 0 \\ \lambda^2 & \lambda & 1 & \ddots & \ddots & 0 \\ \lambda^3 & \lambda^2 & \lambda & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \end{pmatrix}. \quad (1.20)$$

These constructions provides a simple interpolation between the baselines introduced in Section 1.3: $\lambda = 0$ recovers input perturbation, and $\lambda = 1$ would recover output perturbation (though for technical reasons, our analysis requires $\lambda < 1$).

[9] Other objectives are possible, as we discuss in Section 2

It can be easily checked that

$$\boldsymbol{C}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -\lambda & 1 & 0 & \ddots & 0 \\ 0 & -\lambda & 1 & \ddots & 0 \\ 0 & 0 & -\lambda & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \end{pmatrix}. \tag{1.21}$$

Then, we get

$$\boldsymbol{B} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1-\lambda & 1 & 0 & \ddots & 0 \\ 1-\lambda & 1-\lambda & 1 & \ddots & 0 \\ 1-\lambda & 1-\lambda & 1-\lambda & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \end{pmatrix}. \tag{1.22}$$

To achieve 1-GDP, we need from Lemma 1.13 that $\nu \geq 2\left\|\boldsymbol{C}\right\|_{\mathsf{col}}$. Since the first column of $\boldsymbol{C}$ has the highest norm, we have

$$\left\|\boldsymbol{C}\right\|_{\mathsf{col}}^2 = \sum_{t=0}^{n-1} \lambda^{2t} = \frac{1-\lambda^{2n}}{1-\lambda^2} \tag{1.23}$$

and so for any n we can achieve 1-GDP with $\nu^2 = 4(1-\lambda^{2n})/(1-\lambda^2)$. Similarly, the last row of $\boldsymbol{B}$ has the highest norm, so the maximum error from Eq. (1.19) is

$$\begin{aligned} \max_{t\in[n]} \mathbb{E}(\widehat{s}_t - s_t)^2 &= \nu^2\left(1 + (n-1)(1-\lambda)^2\right) \\ &= 4\left(\frac{1-\lambda^{2n}}{1-\lambda^2}\right)\left(1 + (n-1)(1-\lambda)^2\right) \qquad (1.24) \\ &\leq 4\left(\frac{1}{1-\lambda^2}\right)\left(1 + (n-1)(1-\lambda)^2\right). \qquad (1.25) \end{aligned}$$

Both baselines correspond to sub-optimal choices for λ in this loss expression: input perturbation Eq. (1.16) corresponds to $\lambda = 0$ while the output perturbation baseline Eq. (1.17) corresponds to $\lambda \to 1$ and $\nu^2 = \Theta(n)$.

We can do better. One could minimize Eq. (1.24) to choose the optimal λ for a specific n; instead, to obtain an asymptotic result, we consider the upper bound of Eq. (1.25) with $\lambda = 1 - 1/\sqrt{n}$. With this choice, direct computation shows

$$\max_{t\in[n]} \mathbb{E}(\widehat{s}_t - s_t)^2 \leq \frac{4(2n-1)}{2\sqrt{n}-1} = \Theta(\sqrt{n}).$$

This substantially improves on the $\Theta(n)$ error of the baselines, and is in fact the optimal maximum error (up to constants) for the one-parameter family defined in Eqs. (1.20) and (1.22).

In fact, as we shall see in Section 2, it is possible to attain a significantly lower objective value of $\Theta(\text{poly}\ln(n))$ with appropriately-defined correlated noise mechanisms.

This is our first example of a general pattern we will follow regularly: we first design a class of correlated noise mechanisms, generally in terms of some parameterization or structural assumption on $\boldsymbol{C}$ or $\boldsymbol{C}^{-1}$, and then show how to find mechanisms in that class that minimize a certain notion of error, subject to a constraint on privacy.

Remark 1.15 (Error Analysis in $m > 1$ Dimensions)**.** The error analysis for the mechanism defined above directly extends to $m > 1$ dimensions. First, we note from Lemma 1.13 that the noise calibration does not depend on the dimension. Second, the noise is independent across dimensions, so that the error Eq. (1.19) simply adds up across dimensions, leading to an error which is m times worse. In Section 2 and later, we will consider the case of $m > 1$ in full generality.

Remark 1.16 (Non-Square Factorizations)**.** We could define a correlated noise mechanism based on a non-square factorization $\boldsymbol{A} = \boldsymbol{BC}$ with $\boldsymbol{B} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$. The privacy guarantee of Lemma 1.13 holds without any modification. All algorithms and key results, such as the correlated noise DP-SGD (Algorithm 1.3) and its privacy guarantee (Theorem 1.14) can be adapted to work

with the pseudoinverse $\boldsymbol{C}^{\dagger}$ instead of $\boldsymbol{C}^{-1}$. We stick to square and invertible $\boldsymbol{B}, \boldsymbol{C}$ for ease of presentation.

Remark 1.17 (Lower Triangular Factorizations)**.** The factors $\boldsymbol{B}$ and $\boldsymbol{C}$ are both lower-triangular in the above example. At times, it may be computationally convenient to work with non-lower-triangular matrices, particularly when considering implementing sampling from the stream $\boldsymbol{C}^{-1}\boldsymbol{Z}$. Conveniently, we can always get a lower triangular factorization from any factorization without altering either the error (as defined above) or the privacy properties of the mechanism. This can be achieved, for example, by taking the LQ decomposition of $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{Q}$ with a lower-triangular matrix $\boldsymbol{L}$ and an orthonormal matrix $\boldsymbol{Q}$. We then set $\boldsymbol{B}' = \boldsymbol{L}$ and $\boldsymbol{C}' = \boldsymbol{L}^{-1}\boldsymbol{A}$. It follows from the rotational invariance of Gaussian distribution that we get the same error bound and privacy properties whether we use the factors $\boldsymbol{B}', \boldsymbol{C}'$ or $\boldsymbol{B}, \boldsymbol{C}$. Thus, we will restrict ourselves to lower triangular factorizations for the rest of the monograph.

1.4.3 A Noise-Cancellation View of Correlated Noise Mechanisms

To gain intuition for how correlated noise can improve the estimation error, we can give a noise cancellation view for a mechanism define by matrix $\boldsymbol{C}$; this interpretation applies to any $\boldsymbol{C}$, but for concreteness we focus on the one-parameter family defined in Eq. (1.20). We consider the resulting private estimates $\widehat{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}$ of the data $\boldsymbol{G}$, where $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \nu^2)$. In this case, $\boldsymbol{C}^{-1}$ is as in Eq. (1.21).

This directly gives us that

$$\begin{pmatrix} \widehat{\boldsymbol{g}}_0 \\ \widehat{\boldsymbol{g}}_1 \\ \vdots \\ \widehat{\boldsymbol{g}}_{n-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{n-1} \end{pmatrix} + \begin{pmatrix} 1 & 0 & \dots & 0 \\ -\lambda & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{n-1} \end{pmatrix}, \tag{1.26}$$

which can be rewritten as

$$\widehat{\boldsymbol{g}}_t = \begin{cases} \boldsymbol{g}_1 + \boldsymbol{z}_1 & t = 0 \\ \boldsymbol{g}_t + \boldsymbol{z}_t - \lambda \boldsymbol{z}_{t-1} & t \geq 1\,. \end{cases}$$

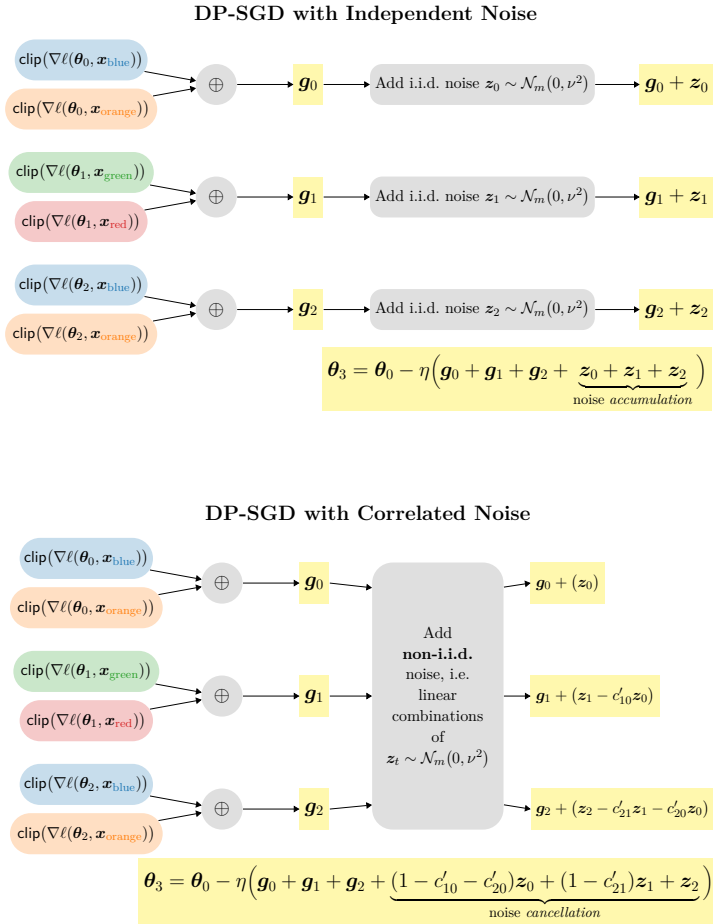


Figure 1.7: An illustration of the noise accumulation in standard DP-SGD (with independent noise), which corresponds to $\boldsymbol{C}^{-1} = \boldsymbol{I}_{n\times n}$. On the other hand, DP-SGD with correlated noise (with $\boldsymbol{C}^{-1} \neq \boldsymbol{I}_{n\times n}$) can cancel a part of the injected noise due to *anti*-correlations. In the lower plot, the inverse of the strategy matrix is given by $\boldsymbol{C}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -c'_{10} & 1 & 0 \\ -c'_{20} & -c'_{21} & 1 \end{pmatrix}$.

In the context of private stochastic gradient descent (Section 1.2), $\boldsymbol{g}_t$ is the gradient obtained in time step t. Then, the privatized gradient $\widehat{\boldsymbol{g}}_t$ can be interpreted as *canceling out* a λ-fraction of the noise $\boldsymbol{z}_{t-1}$ injected in the previous time step; see Fig. 1.7 for an illustration for general $\boldsymbol{C}$ matrices.

In general, a large λ leads to better estimation at a fixed variance ν^2 (of each component of the noise $\boldsymbol{z}$) as more noise is canceled out. However, this leads to a larger $\|\boldsymbol{C}\|_{\mathsf{col}}$, implying an increased privacy cost (Lemma 1.13). Thus, to guarantee a given level of privacy such as 1-GDP, this in turn requires a larger variance ν^2. In general, the correction applied via the correlated noise needs to be balanced with the privacy constraint to obtain improved privacy-utility tradeoffs over the baselines.

As we will see in the rest of this monograph, this is generally possible in a range of prefix sum estimation and machine learning tasks. In particular, we study details of how to optimally choose the noise correlations for various objectives and under various constraints on the correlation structure such that we obtain efficient implementations.

1.5 The Design Space of Correlated Noise Mechanisms

The design space of correlated noise mechanisms for gradient-based learning algorithms involves a complex interplay of several factors, requiring careful consideration of the workload, desired privacy guarantees, acceptable error levels, and computational constraints. We give a brief description of each of these design factors and describe how Sections 2 to 4 disentangle these factors, culminating in a set of practical recommendations by the end of the monograph.

A practitioner interested in employing correlated noise mechanisms in their model training tasks must make five key design decisions shown in Fig. 1.8.

1. **Workload.** The starting point is the workload matrix $\boldsymbol{A}$ to factorize. When implementing DP-SGD with correlated noise (Algorithm 1.3), we only need to specify the noise correlating matrix $\boldsymbol{C}^{-1}$; the privacy calibration depends on the matrix $\boldsymbol{C}$

alone (Theorem 1.14). On the other hand, the $\boldsymbol{A}$ play a key role (via $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1}$) in the surrogate objective (e.g. Eq. (1.19)) we optimize to find the $\boldsymbol{C}$ matrix.

The viewpoint of Section 1.2 suggests a heuristic to select the workload matrix implied by the base (non-private) optimizer. For example, vanilla SGD corresponds to the prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ (Eq. (1.6)). We will use this a canonical example throughout this monograph. All key considerations we discuss generalize directly to other first-order optimization algorithms whose iterates can be obtained as linear combinations of gradients. This class includes momentum variants such as heavy ball and Nesterov momentum—we briefly discuss these other workloads in Section 3.

2. **Error Metric.** The practitioner also determines the error metric as a proxy to measure the utility of the correlated noise mechanisms. The error metric and the sensitivity (which depends on the participation pattern) together determine the objective we optimize to find the factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$. For instance, the objective Eq. (1.24) in the example of Section 1.4 is a product of the sensitivity term with the error term; its general form is as shown in Fig. 1.8.

   We use the worst-case expected error in prefix sums (as in Eq. (1.19) of Section 1.4) as a default error metric for concreteness. We briefly discuss the use of the average expected error across prefix sums towards the end of Section 2 (and return to it in Section 4). Section 3 also considers using the learning objectives as the utility measures for simple problems such as mean estimation and linear regression.

3. **Participation Patterns.** The third factor is to determine the *participation pattern* of datapoints:

   > Do we process each datapoint only once (i.e., the streaming setting of Section 1.2) or do we make multiple passes over the data? In the latter case, are there any restric-

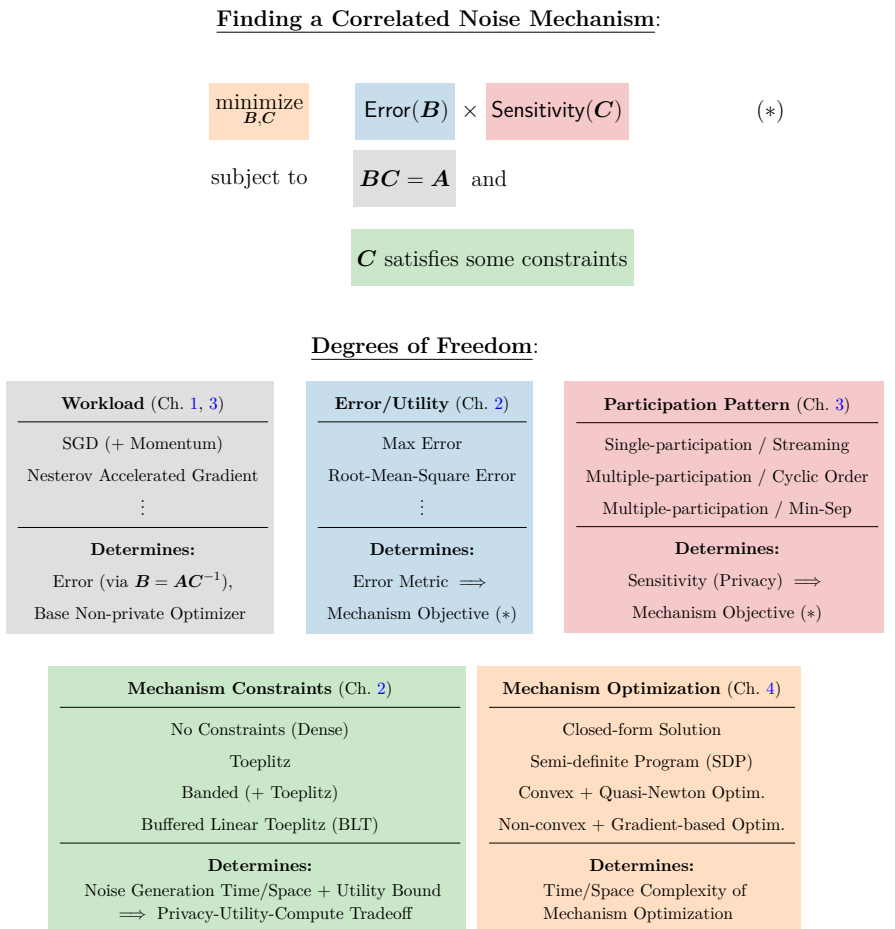


Figure 1.8: An outline of the topics defined and covered in each of the sections of this monograph.

tions on the order in which the datapoints are processed (e.g. cyclic order)?

As illustrated in Fig. 1.3, this determines the effect of changing one datapoint (in an adjacent dataset) on the output of the mechanism. In other words, the participation pattern is key to quantifying the sensitivity, and hence, the privacy guarantee, of the correlated noise mechanism.

The streaming setting is straightforward to handle, as we saw in Lemma 1.13. Expanding upon the illustration of Fig. 1.3, Section 3 explains the challenges and nuances of tightly bounding the sensitivity in different multiple-participation scenarios, as well as its efficient computation in Section 3.

4. **Mechanism Constraints.** The practitioner also decides the class of factorizations $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ to optimize over. This design decision can lead to different privacy-utility-compute tradeoffs of the correlated noise mechanism. For example, generating the correlated noise $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$ (or equivalently, $(\boldsymbol{B}\boldsymbol{Z})[t,:]$) in step t can take $O(nm)$ time per-step, and simply storing the correlating noise matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ would require $O(n^2)$ space. This is impractical for long training runs (i.e., with n large, especially for larger parameter model where m can range in billions). We explore families of structured matrices that allow reduced space and time complexity of this noise generation in Section 2. However, if the class of matrices is too restrictive, it can have sub-optimal utility.

   We will see in Section 2 that it is possible to attain favourable privacy-utility-compute tradeoffs with carefully designed families of structured matrices. We then adapt these mechanisms to the multiple-participation setting in Section 3.

5. **Mechanism Optimization.** Finally, the practitioner has to determine the optimization algorithm to find the factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ over the determined set of structured matrices to minimize the objective (which is the product of the error metric

and sensitivity); see Fig. 1.8. This is a one-time cost, provided the obtained factors $\boldsymbol{B}, \boldsymbol{C}$ can be cached. Section 4 discusses the approaches to perform this optimization and approximations required to make to practical and scalable.

1.6 The Privacy Unit: Example-level vs. User-level DP

As we discussed at the end of Section 1.1, the definition of differential privacy can be instantiated at different granularity by specifying the unit of change between adjacent D and D'. This includes

(i) **example-level DP**, where D and D' differ by one example (as specified by the adjacency notion); and

(ii) **user-level DP** refers to the setting where D and D' differ by all examples derived from a single user or entity.

In practice, the privacy unit must be selected based on the problem at hand. For example, if multiple examples derived from a single user or entity might have common features or attributes, user-level DP can align more closely with the intuitive notion of individual privacy protection. This is generally the case for AI models that are trained or fine-tuned directly on user-written emails, documents, or text messages. In this case, example-level DP may fail to adequately mitigate the risk of information leakage due to the inherent inter-dependencies among a user's examples.

To a large extent, the correlated noise mechanisms we develop are agnostic to the privacy unit. For the sake of clarity and simplicity in exposition, we default to example-level DP in this monograph. Notably, the algorithmic modifications made to adapt independent noise DP-SGD from example-level to user-level DP are largely applicable to correlated noise mechanisms.

We can adapt Algorithms 1.2 and 1.3 to user-level DP as follows. It is convenient to describe the dataset D as a collection of user datasets:

$$D = \{D_u \,:\, u \in [N]\}\,,$$

where $D_u = \{\boldsymbol{x}_{u,0}, \boldsymbol{x}_{u,1}, \ldots\}$ is the dataset of user $u \in [N]$. We define the loss on one user's data as the average loss over

$$\ell(\boldsymbol{\theta}, D_u) := \frac{1}{|D_u|} \sum_{\boldsymbol{x} \in D_u} \ell(\boldsymbol{\theta}, \boldsymbol{x}) \,.$$

Finally, in each iteration t, we sample/select a user u_t and calculate the clipped user gradient:

$$\boldsymbol{g}_t = \mathsf{clip}_\zeta(\nabla \ell(\boldsymbol{\theta}_t, D_{u_t})) \,.$$

For the purposes of privacy analysis, this $\boldsymbol{g}_t$ can actually be *any clipped function* of the user data such as a stochastic user gradient $\nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_{u_t})$ for some arbitrary $\boldsymbol{x}_{u_t} \in D_{u_t}$. Another popular choice in the context of federated learning is a clipped *pseudo-gradient*. This is the delta of k stochastic gradient steps on D_{u_t}:

$$\begin{aligned} \boldsymbol{g}_t &= \mathsf{clip}_\zeta \left(\frac{1}{\eta} (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t,k}^{(u)}) \right) \,, \quad \text{where} \\ \boldsymbol{\theta}_{t,l+1}^{(u)} &= \boldsymbol{\theta}_{t,l}^{(u)} - \eta \nabla \ell(\boldsymbol{\theta}_{t,l}^{(u)}, \boldsymbol{x}_{u,l}) \text{ for } l = 0, \ldots, k-1 \,, \end{aligned} \tag{1.27}$$

and $\boldsymbol{x}_{u,l}$ sampled/selected from D_u and the stochastic gradient steps are initialized at $\boldsymbol{\theta}_{t,0}^{(u)} = \boldsymbol{\theta}_t$.

Importantly, the streaming assumption in user-level DP translates to each user appearing only once during training. If any user's data is used more than once (even if each example might be processed only once), the streaming assumption is violated for the purpose of user-level DP. In such a case, we have to use the multiple-participation techniques of Section 3.

1.7 Other Notions of Adjacency*

While we use the replace-one adjacency (Definition 1.1) in this section, in practical deployment and literature, two other preferences are the *zero-out* adjacency and *add-or-remove* adjacency.

Zero-out Adjacency

The zero-out adjacency is usually used when working with algorithms operating on example gradients. In this setting, the zero-out adjacency

says $D \simeq D'$ if D and D' are the same, except one example in D is replaced with a special example in D' whose gradients are zero everywhere or vice-versa:

Definition 1.18. We say two datasets $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $D' = \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_N\}$ are **zero-out adjacent** (denoted $D \simeq D'$) if $\boldsymbol{x}_i = \boldsymbol{x}_i$ for all $i \in [n] \setminus \{j\}$ for some index $j \in [n]$ and $\boldsymbol{x}_j = \perp$ or $\boldsymbol{x}'_j = \perp$, where $\perp$ is a special *null* element such that $\nabla \ell(\boldsymbol{\theta}, \perp) = \mathbf{0}$ for all $\boldsymbol{\theta}$.

Replace-one vs. Zero-out Adjacency To compare replace-one and zero-out, consider $f(D) = \sum_{\boldsymbol{x} \in D} \nabla \ell(\boldsymbol{\theta}, \boldsymbol{x})$, the model gradient summed over a dataset D, where the parameter $\boldsymbol{\theta}$ is 1-dimensional and the loss $\boldsymbol{\theta} \mapsto \ell(\boldsymbol{\theta}, \boldsymbol{x})$ is 1-Lipschitz, i.e., the per-example gradients are bounded as $|\nabla \ell(\boldsymbol{\theta}, \boldsymbol{x})| \leq 1$.

Then, we have that $\mathsf{sens}(f) = 2$ under the replace-one adjacency, because when replacing an example, a gradient can switch from e.g. 1 to -1. Under the zero-out adjacency, however, we have $\mathsf{sens}(f) = 1$, since the replaced gradient can switch from 1 to 0, but not to -1. A consequence of the doubled sensitivity is that an algorithm that is μ-GDP with respect to the replace-one adjacency is $\mu/2$-GDP with respect to the zero-out adjacency and vice-versa, giving us an easy way to directly compare the two definitions.

In particular, we have the following analogue to Lemma 1.12:

Lemma 1.19. Under zero-out adjacency (Definition 1.18) with gradients clipped to norm 1 (cf. Eq. (1.8)), we have $\mathsf{sens}(\boldsymbol{C}) = \|\boldsymbol{C}\|_{\mathsf{col}}$.

Proof. Same as the proof of Lemma 1.12, except that $\|\boldsymbol{\delta}\|_2 \leq 1$ due to the different notion of the adjacency. □

This leads to analogous privacy results in the zero-out case:

Theorem 1.20. Consider the setting of Theorem 1.14. The gradients and iterates of DP-SGD with correlated noise (Algorithm 1.3) satisfy $\frac{1}{\sigma}$-GDP in the zero-out-example adjacency in the streaming setting if we take $\nu = \sigma \, \|\boldsymbol{C}\|_{\mathsf{col}}$.

Advantages of Zero-out Adjacency and Add-or-Remove Adjacency

Since zero-out and replace-one is only a factor of 2 off from each other, one might wonder why zero-out, which is restricted to gradient-like queries, is used instead of replace-one in practice. One reason is that zero-out is more comparable to the popular add-or-remove-one adjacency described below:

Definition 1.21. Two datasets D, D' are adjacent under the **add-or-remove adjacency** if $D = D' \cup \{\boldsymbol{x}\}$ for $D' = D \cup \{\boldsymbol{x}\}$ for some example $\boldsymbol{x}$.

The add-or-remove-one adjacency is popular in part because it is very natural. For instance, it gives privacy guarantees to a user who might want opt their data out of model training. The add-or-remove model is also more natural for analyzing privacy amplification by Poisson sampling, discussed in Section 3.4. However, for model training, add-or-remove adjacency is somewhat inconvenient because adjacent datasets have different sizes. In turn, the dataset size itself becomes a private quantity, which greatly complicates privacy analyses. Zero-out adjacency maintains semantic similarity to add-or-remove-one, but maintains that D and D' are the same size and e.g. batches are formed the same way for both datasets, i.e., effectively allows us to publish the dataset size without violating DP guarantees.

For these reasons, we will use the zero-out notion of adjacency later in Section 3.

Remark 1.22 (Event-Level Privacy)**.** In the setting of continual counting and in settings such as Example 1.27, the notion of adjacency usually considered is the *event-level privacy*: there is exactly one time step where the neighboring streams differ. Thus, in the streaming machine learning setting, event-level privacy of gradients coincides with example-level (or user-level) privacy of their underlying datasets.

1.8 Other Common Notions of Differential Privacy*

In this chapter, we discuss two differential privacy (DP) definitions, Gaussian DP and approximate DP, which are more commonly reported in practice. We state a method for converting a Gaussian DP guarantee (or its equivalent description via a noise multiplier for the Gaussian mechanism) to another notion known as zero-concentrated DP. In particular, this conversion can be done by either computing a simple formula, or calling an existing open-source library.

1.8.1 Zero-Concentrated DP (zCDP)

Zero-concentrated differential privacy (zCDP) is defined in terms of the Rényi divergence:

Definition 1.23. The **α-Rényi divergence** between two distributions P and Q for $\alpha > 1$ is defined as:

$$R_\alpha(P, Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{X \sim Q} \left(\frac{P(X)}{Q(X)} \right)^\alpha .$$

The definition extends to $\alpha \in \{1, \infty\}$ by continuity.

Definition 1.24. A mechanism $\mathcal{M}$ is ρ-zCDP if for all $D \simeq D'$, it holds that

$$R_\alpha(\mathcal{M}(D), \mathcal{M}(D')) \leq \rho\alpha$$

for all $\alpha > 1$.

We can convert from GDP (or its equivalent noise multiplier description) to zCDP using the following result:

Lemma 1.25. Any μ-GDP mechanism is $(\mu^2/2)$-zCDP. In particular, the Gaussian mechanism with noise multiplier σ is $(1/2\sigma^2)$-zCDP.

1.9 Additional Proofs*

Earlier on page 11, we gave a proof sketch of Lemma 1.6 in the sense that it was only for the 1-dimensional case. However, in the case of correlated

noise mechanisms, we would be applying the Gaussian mechanism on high-dimensional vectors. We now give a general proof of the GDP bound of the Gaussian mechanism (Lemma 1.6) in high-dimensions, which at a high level, uses the isometry of multi-variate Gaussian distribution to reduce the general high-dimensional case to 1-dimensional case. For readers familiar with the proof of (ε, δ)-DP for the Gaussian mechanism, the proof is analogous.

Proof of Lemma 1.6. The proof proceeds similarly to the 1-dimensional case. Consider a pair of worst-case adjacent datasets $D \simeq D'$ such that $\|f(D) - f(D')\|_2 = \mathsf{sens}(f)$. If $f(D)$ and $f(D')$ are closer, it only makes the mechanism outputs $\mathcal{M}(D), \mathcal{M}(D')$ more indistinguishable. We will exhibit a randomized map g such that $\mathcal{M}(D) \overset{\mathrm{d}}{=} g(Z)$ and $\mathcal{M}(D) \overset{\mathrm{d}}{=} g(Z + \mu)$ for $Z \sim \mathcal{N}(0, 1)$, as required by Definition 1.3.

Denote $\mu = 1/\sigma$ and $\boldsymbol{u}$ as the unit vector along $f(D') - f(D)$:

$$\boldsymbol{u} := \frac{f(D') - f(D)}{\|f(D') - f(D)\|_2} = \frac{1}{\mathsf{sens}(f)}(f(D') - f(D)) \,.$$

The idea is to construct g such that $\mathbb{E}[g(Z)] = f(D)$ and that $g(Z + \mu)$ will introduce a bias along the unit vector $\boldsymbol{u}$, yielding $f(D')$ in expectation. All other directions are not relevant, as both distributions are essentially Gaussian noise with identical variance.

Concretely, consider the randomized function $g : \mathbb{R} \to \mathbb{R}^m$ given by

$$g(s) := f(D) + \sigma \cdot \mathsf{sens}(f) \cdot \left(s\boldsymbol{u} + \left(\boldsymbol{I}_{m\times m} - \boldsymbol{u}\boldsymbol{u}^\top\right)\boldsymbol{\xi}\right) \text{ for } \boldsymbol{\xi} \sim \mathcal{N}_m(0, 1) \,.$$

If we take $s = Z \sim \mathcal{N}(0, 1)$, then the second term is just component-wise i.i.d. Gaussian noise with variance $\nu^2 = \sigma^2 \,\mathsf{sens}(f)^2$, since

$$Z\boldsymbol{u} + \left(\boldsymbol{I}_{m\times m} - \boldsymbol{u}\boldsymbol{u}^\top\right)\boldsymbol{\xi} \overset{\mathrm{d}}{=} \mathcal{N}_m(0, 1) \,,$$

by the rotational invariance of the Gaussian distribution. Thus, for $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 \,\mathsf{sens}(f)^2)$, we get $g(Z) \overset{\mathrm{d}}{=} f(D) + \boldsymbol{z} \overset{\mathrm{d}}{=} \mathcal{M}(D)$, as required. Instead, if we plug in $s = Z + \mu$, only the mean changes, while the variance remains unchanged:

$$\begin{aligned} g(Z + \mu) &= f(D) + \sigma \cdot \mathsf{sens}(f)\ (\mu\boldsymbol{u}) + \boldsymbol{z} \\ &= f(D) + \mathsf{sens}(f) \cdot \boldsymbol{u} + \boldsymbol{z} = f(D') + \boldsymbol{z} \,, \end{aligned}$$

since $\mu = 1/\sigma$ and $\mathsf{sens}(f) \cdot \boldsymbol{u} = f(D') - f(D)$. □

1.10 Chapter Notes*

We give some additional context and details on some of the topics covered here.

1.10.1 The Workload and Strategy Matrices

In the context of correlated noise mechanisms, the matrix $\boldsymbol{A} \in R^{n \times n}$ is called the *workload matrix* or *query matrix*. Similarly, in the factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$, the matrix $\boldsymbol{C}$ is referred to as the *strategy matrix*. This terminology originates from the database literature, where early research on correlated noise mechanisms (detailed in Section 1.11) focused on counting records (or bins) satisfying a given predicate in a database (or histogram). A concrete example is provided in the upcoming Example 1.27.

We map this problem to our notation. Let $\boldsymbol{G} \in \mathbb{R}^{n \times 1}$ represent the input database with n records (here, the dimension is $m = 1$). Each counting query to the database can be denoted by a vector $\boldsymbol{a} \in \{0, 1\}^n$, where $\boldsymbol{a}[t] = 1$ indicates the records to be counted. The result of this query is simply $\boldsymbol{a}^\top \boldsymbol{G}$, the dot product of $\boldsymbol{a}$ and the database $\boldsymbol{G}$. Early work on correlated noise mechanisms considered answering a batch of queries denoted by vectors $\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{N-1}$, which can be arranged row-wise into a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times n}$. The goal is to estimate $\boldsymbol{A}\boldsymbol{G}$ under differential privacy (we take $N = n$ in our setting). A key motivation of the early work was to show how answering all the queries collectively—with appropriate noise correlation—yields better private estimates than answering them individually (which corresponds to the output perturbation mechanism).

In databases, the term "workload" refers to a set of database requests or queries that share common characteristics (e.g., on the same table or histogram), allowing for the application of workload management controls to optimize system performance. Since the $\boldsymbol{A}$ matrix denotes a collection of the queries $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{N-1}$ on the same input $\boldsymbol{G}$, it is naturally called the *workload matrix*.

Similarly, database management systems optimize query execution by grouping and refactoring queries to avoid redundant work. In cor-

related noise mechanisms, the matrix $\boldsymbol{C}$ captures a *strategy* for performing some shared work $\boldsymbol{CG}$ on the input $\boldsymbol{G}$ before adding noise $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \nu^2)$ for DP. Thus, it is analogously referred to as the *strategy matrix*.

1.10.2 DP-FTRL: DP-SGD with Correlated Noise

DP-SGD with correlated noise, as given in Algorithm 1.3 is referred to in the literature as *DP-FTRL*. This stands for a differentially private version of the *follow-the-regularized-leader* (FTRL) algorithm. Given a sequence of past gradients $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{t-1}$, the FTRL algorithm (also known as *dual averaging* or *lazy mirror descent*) sets the next iterate $\boldsymbol{\theta}_t$ as

$$\boldsymbol{\theta}_t := \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{ \boldsymbol{s}_t^\top \boldsymbol{\theta} + \frac{1}{2\eta} \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2 \right\}, \quad \text{where} \quad \boldsymbol{s}_t = \sum_{\tau=0}^{t-1} \boldsymbol{g}_\tau$$

is the prefix sum of gradients, $\boldsymbol{\theta}_0$ is the first iterate, and Θ is the constraint set. For an unconstrained optimization problem with $\Theta = \mathbb{R}^m$, these iterates coincide exactly with the SGD iterates in Eq. (1.4). FTRL differs from SGD when Θ is a strict subset of $\mathbb{R}^m$ or when other penalties are used in the place of the squared Euclidean norm $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2$.

Historically, the name DP-FTRL has been used to emphasize that correlated noise mechanisms are used to privately estimate the prefix sums $(\boldsymbol{s}_t)_{t=0}^{n-1}$ (as opposed to independent noise DP-SGD with $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$). Throughout this monograph, we stick with the nomenclature *DP-SGD with correlated noise*.

1.10.3 More Examples of Weighted Prefix Sums

Weighted prefix sums are ubiquitous in data science, in addition to the setting of gradient descent considered in Section 1.2. We give a few more examples here.

Example 1.26 (Online k-means Clustering)**.** In this example, given input data that consists of a stream of points P in $\mathbb{R}^d$, the goal is to output after the arrival of a point a set S of k points (called the

centers) in $\mathbb{R}^d$ in order to minimize the following cost function:

$$\text{cost}(P, S) = \sum_{p \in P} \min_{s \in S} \text{dist}(p, s)^2,$$

where $\text{dist}(p, s)$ is the Euclidean distance between p and point s while remaining differentially private with respect to the stream of datapoints. There is a reduction of this problem to a finite collection prefix sum problem, where each problem is d-dimensional.

Example 1.27 (Range Queries and CDF Estimation)**.** In this problem, we are given n data points, $D = (x_1, \ldots, x_n) \in [0, C]^n$ for some fixed $C \in \mathbb{R}$. The goal is to construct a data structure so that, given a query range $[a, b] \in [0, C]$, the output is an estimate of

$$R_{[a,b]} = |\{x_i \in D \text{ such that } a \leq x_i \leq b\}|.$$

A notable application of range queries in the estimation of the cumulative distribution function (CDF) $F(b) := \mathbb{P}(X \leq b)$ of a random variable X. In particular, if the data points $x_1, \ldots, x_n$ are i.i.d. samples of the random variable X, then the empirical CDF estimator $\widehat{F}(b) = \frac{R_{[0,b]}}{n}$ can be computed via a range query. In general, range queries are relevant in applications ranging from online analytics such as financial markets or sensor networks to contact tracing, mobility analysis and urban planning.

One can reduce range queries to prefix sum computation, allowing us to leverage correlated noise mechanisms for better differentially private estimators. In particular, we divide the domain $[0, C]$ into disjoint buckets, each with size equal to the allowed accuracy. Then, each bucket stores the number of points that are in the range covered by the bucket. To estimate the value of $R_{[a,b]}$ for a given $0 \leq a \leq b \leq C$, we use the prefix sum until the bucket that contains a and the prefix sum until the bucket that contains b. The difference of the two gives the estimate of $R_{[a,b]}$.

1.11 Bibliographic Notes

Differential Privacy Differential privacy was introduced in the celebrated work of Dwork, McSherry, Nissim, and Smith [2016]. The Gaussian mechanism was introduced subsequently by Dwork, Kenthapadi, McSherry, Mironov, and Naor [2006] in which they also introduced the relaxation of differential privacy, now commonly known as *approximate differential privacy*. Gaussian differential privacy was introduced by Dong, Roth, and Su [2022] and zero-concentrated differential privacy by Bun and Steinke [2016] and Dwork and Rothblum [2016]. Lemma 1.25 was shown as Proposition 1.6 in Bun and Steinke [2016]. The proof of Theorem 1.11 is due to Denisov, McMahan, Rush, Smith, and Guha Thakurta [2022].

Gradient Descent with DP Differentially private gradient descent was first introduced by Song, Chaudhuri, and Sarwate [2013]. Bassily, Smith, and Thakurta [2014b] showed that it achieves optimal empirical risk guarantee and Abadi, Chu, Goodfellow, McMahan, Mironov, Talwar, and Zhang [2016] developed a better privacy accounting method (via the "moments accountant") using the Gaussian mechanism, making differentially private SGD (DP-SGD) practically usable on deep nets.

Correlated Noise Mechanisms for Learning The use of correlated noise for differentially private learning was first studied in the context of online learning by Jain, Kothari, and Thakurta [2012], and was later by Thakurta and Smith [2013] who considered *follow-the-approximate leader*. The variant considered in this monograph was first defined in Kairouz, McMahan, Song, Thakkar, Thakurta, and Xu [2021a] and later improved in a series of works by Denisov, McMahan, Rush, Smith, and Guha Thakurta [2022], Choquette-Choo, McMahan, Rush, and Thakurta [2023b], Choquette-Choo, Dvijotham, Pillutla, Ganesh, Steinke, and Thakurta [2024a]. In particular, Denisov, McMahan, Rush, Smith, and Guha Thakurta [2022] proved Theorem 1.11 and these works framed correlated noise mechanism as deriving a differentially private version of the popular follow-the-regularized leader (FTRL) algorithm, also known as dual averaging. This family of algorithms was, in turn,

developed through a series of works by Nesterov [2009], Xiao [2009], McMahan [2011] and Duchi, Agarwal, and Wainwright [2011].

Correlated Noise Mechanisms for Prefix Sum Estimation Computing prefix sum under the constraints of differential privacy was first studied by Dwork, Naor, Pitassi, and Rothblum [2010] and Chan, Shi, and Song [2011]. These works used a particular hand-crafted correlations, now known as the *binary tree mechanism*; we discuss this further in Section 2.7. The general framework of correlated noise mechanisms we defined in Section 1.3 (i.e., using a factorization $\boldsymbol{A} = \boldsymbol{BC}$) was first introduced concurrently by Li, Miklau, Hay, McGregor, and Rastogi [2015] under the name *matrix mechanism* and by Nikolov, Talwar, and Zhang [2016] as the *factorization mechanism*.

The use of the matrix mechanism for prefix sum estimation was first observed independently and concurrently by Denisov, McMahan, Rush, Smith, and Guha Thakurta [2022] and Fichtenberger, Henzinger, and Upadhyay [2023]. The weighted version of the prefix sum was first studied by Bolot, Fawaz, Muthukrishnan, Nikolov, and Taft [2013], and recently improved by Henzinger, Upadhyay, and Upadhyay [2023], Henzinger and Upadhyay [2025].

Large-Scale Practical Deployments of DP The United States Census Bureau used DP for the 2020 Census to provide demographic insights [US Census Bureau, 2021]. Google used DP in the Chrome browser to analyze user behavior [Erlingsson et al., 2014], while Apple used DP in iOS and MacOS [Apple, 2017]. In particular, correlated noise mechanisms have been industrially deployed to train next-word-prediction models for mobile keyboards by Google [Xu, Zhang, Andrew, Choquette-Choo, Kairouz, McMahan, Rosenstock, and Zhang, 2023]. For best practices involving DP in machine learning, see Ponomareva, Hazimeh, Kurakin, Xu, Denison, McMahan, Vassilvitskii, Chien, and Thakurta [2023, Sec 5.3.3].

Adjacency Notions and Privacy Units For a detailed discussion of adjacency notions and privacy unit, we refer to Ponomareva, Hazimeh,

Kurakin, Xu, Denison, McMahan, Vassilvitskii, Chien, and Thakurta [2023]. In particular, we refer to their Section 2.1 and 5.1 for how to select a privacy unit. A discussion on the pros and cons of add-or-remove-one adjacency vis-à-vis replace-one adjacency in the context of privacy amplification by sampling can be found in [Steinke, 2022, Sec. 6]. The resulting user-level DP learning algorithm obtained by modifying independent noise DP-SGD (Algorithm 1.2) with Eq. (1.27) is also known as "DP-FedAvg". This algorithm was proposed by McMahan, Ramage, Talwar, and Zhang [2018] to make the standard federated averaging algorithm satisfy user-level DP guarantees. In fact, much of the prior literature on correlated noise mechanisms in the learning setting, starting from Kairouz, McMahan, Song, Thakkar, Thakurta, and Xu [2021a], was motivated by user-level DP in the federated learning context; see Example 3.1 in Section 3 for more background context.

We note that it is also possible to promote example-level DP guarantees to user-level DP guarantees by using generic group privacy reductions [Vadhan, 2017, Lemma 2.2]. While such reductions are usually not as tight as the approach presented in Section 1.6, it is possible to develop tighter user-level privacy accounting techniques for the example-level DP algorithm of DP-SGD with independent noise [Charles, Ganesh, McKenna, McMahan, Mitchell, Pillutla, and Rush, 2025].

Other Remarks The reduction of online k-means clustering to a finite collection of prefix sum problem was shown in Dupré la Tour, Henzinger, and Saulpic [2024].

2

Correlated Noise Mechanisms for Streaming Prefix Sums

We develop some correlated noise mechanisms in the simplified *streaming setting* to convey the key ideas. As we introduced in Section 1, the streaming setting refers to the setting where each data point participates in training only once.

Recall that our goal is to factorize a lower-triangular workload matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ into $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$. We describe a general approach but instantiate it specifically with unweighted prefix sum matrix, $\boldsymbol{A}_{\mathsf{pre}} \in \{0,1\}^{n \times n}$, which is a lower-triangular matrix of all ones (see also Eq. (1.6)):

$$\boldsymbol{A}_{\mathsf{pre}}[i,j] = \begin{cases} 1 & i \geq j \\ 0 & \text{otherwise.} \end{cases}$$

Recall from Section 1.2 that privatizing the iterates $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1}$ obtained from the iterations of stochastic gradient descent (SGD; see Algorithm 1.1) with correlated noise mechanisms corresponds to factorizing the matrix $\boldsymbol{A}_{\mathsf{pre}}$.

For all theorems and lemmas stated without proof, detailed references that include proofs are given in the bibliographic notes of Section 2.12 at the end of the section.

2.1 Design Considerations

The primary design considerations for a correlated noise mechanism are its computational cost and its effectiveness in improving the utility of the algorithm. We recall the setup and describe each in turn.

Setup As described in Section 1, our goal is to privately compute the prefix sums $\boldsymbol{G} \mapsto \boldsymbol{A}\boldsymbol{G}$ for an input sequence $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ (representing gradients). Given a factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ of a weighted prefix sum matrix with lower triangular factors $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$, consider a correlated noise mechanism $\mathcal{M}$ that returns

$$\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}) = \boldsymbol{A}(\boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}), \tag{2.1}$$

where $\boldsymbol{Z}$ is component-wise i.i.d. Gaussian noise. In Section 1.4.3, we saw this was equivalent to a mechanism $\mathcal{M}'$ which computes DP estimates of $\boldsymbol{G}$, $\mathcal{M}'(\boldsymbol{G}) = \widehat{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z}$, and then applying the workload as post-processing, $\boldsymbol{A}\widehat{\boldsymbol{G}} = \boldsymbol{A}\mathcal{M}'(\boldsymbol{G}) = \mathcal{M}(\boldsymbol{G})$.

We know from Lemma 1.13 that $\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(0, \nu^2)$ for

$$\nu = \mathsf{sens}(\boldsymbol{C}) \cdot \sigma = 2 \left\|\boldsymbol{C}\right\|_{\mathsf{col}} \sigma \tag{2.2}$$

yields a $(1/\sigma)$-GDP mechanism in the streaming setting, where $\left\|\boldsymbol{C}\right\|_{\mathsf{col}}$ denotes the maximum column norm of the matrix $\boldsymbol{C}$. Thus, throughout this section, we choose ν following Eq. (2.2); then σ is interpreted as a noise multiplier parameter that is independent of the factorization used.

2.1.1 Time and Space Complexity of Noise Generation

As we saw in Algorithm 1.3 in Section 1, private optimization using the correlated noise mechanism requires us to compute the noisy gradient

$$\widehat{\boldsymbol{g}}_t = \boldsymbol{g}_t + \left(\boldsymbol{C}^{-1}\boldsymbol{Z}\right)[t, :] = \boldsymbol{g}_t + \sum_{\tau=0}^{t} (\boldsymbol{C}^{-1})[t, \tau]\ \boldsymbol{Z}[\tau, :] \tag{2.3}$$

in each step t of the algorithm, where $\boldsymbol{Z}$ is component-wise i.i.d. Gaussian noise.[1] Consequently, a critical design consideration in correlated noise

[1] The summation over τ in the second expression in Eq. (2.3) runs only until $\tau = t$ because $\boldsymbol{C}^{-1}$ is lower triangular. This follows from $\boldsymbol{C}$ being lower triangular.

mechanisms is the time and space complexity of the **noise generation** process of Eq. (2.3). In the worst case, it can take $O(mn)$ time to compute a sum of up to n vectors $\boldsymbol{Z}[\tau, :] \in \mathbb{R}^m$. Similarly, it can take up to $O(n^2)$ space to simply represent the matrix $\boldsymbol{C}^{-1}$ (or $O(n^3)$ time to compute, given $\boldsymbol{C}$).

In typical AI and machine learning settings, the model dimension m can be significantly larger than the number of steps n. Indeed, m can vary between a few millions for small on-device models to several billions or more for transformer language models. On the other hand, differentially private training entails large batches, so the number of steps n is usually a few tens of thousands or smaller. Thus, the $O(mn)$ time complexity of noise generation is usually more of a bottleneck than the $O(n^2)$ memory. In some cases, even a noise generation cost $O(mn)$ may be tractable, as discussed in Section 4 and Remark 4.16. Further, as we will see in this section, it is often possible to do better with carefully designed structured matrices.

The cost of finding a factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ is also relevant. This is a one-time cost as the matrices $\boldsymbol{B}, \boldsymbol{C}$ can be cached for future use. In this section, we only consider the cost of noise generation, as it is incurred in every step of the correlated noise mechanism. We will revisit the factorization cost in Section 4.

2.1.2 Loss Metric

While the ultimate test of a correlated noise mechanism is its learning performance (e.g., accuracy for classification tasks), it is nevertheless useful to track the error in the prefix sum estimate as a surrogate measure of utility. Fig. 1.6 illustrates the error in the prefix sums in a learning setting.

A natural metric is the maximum squared error in the estimation of any prefix sum; we considered this in Eq. (1.19) in Section 1. In the literature, it is also common to consider its square root:

> **Definition 2.1.** The **unnormalized max loss** (also known as the **unnormalized ℓ_∞ loss**) of a mechanism $\mathcal{M}(\boldsymbol{G})$ estimating the

(weighted) prefix sum $\boldsymbol{AG}$ of an input $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ is defined as

$$\mathcal{L}_\infty(\mathcal{M}) \coloneqq \max_{t\in[n]} \sqrt{\mathbb{E}\,\|(\boldsymbol{AG} - \mathcal{M}(\boldsymbol{G}))[t,:]\|_2^2}\,.$$

It turns out that the max loss for the correlated noise mechanisms has a simple characterization in terms of the maximum row norm $\|\boldsymbol{B}\|_{\mathsf{row}}$ of $\boldsymbol{B}$ and the maximum column norm $\|\boldsymbol{C}\|_{\mathsf{col}}$ of $\boldsymbol{C}$:[2]

Theorem 2.2. Consider a correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{CG}+\boldsymbol{Z})$ for $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0,\nu^2)$ with $\nu = \mathsf{sens}(\boldsymbol{C})\cdot\sigma = 2\,\|\boldsymbol{C}\|_{\mathsf{col}}\,\sigma$ for noise multiplier σ, following Lemma 1.12. Then, we have that its max loss, denoted as $\mathcal{L}_\infty(\boldsymbol{B},\boldsymbol{C})$, equals

$$\mathcal{L}_\infty(\boldsymbol{B},\boldsymbol{C}) = 2\sqrt{m}\sigma\ \ \|\boldsymbol{B}\|_{\mathsf{row}}\,\|\boldsymbol{C}\|_{\mathsf{col}}\ .$$

Proof Sketch. Note that $\boldsymbol{AG} - \mathcal{M}(\boldsymbol{G}) = \boldsymbol{BZ} \sim \mathcal{N}(\boldsymbol{0}, \nu^2\boldsymbol{BB}^\top)$ is oblivious to the input $\boldsymbol{G}$. Recalling $\boldsymbol{s}_t \coloneqq \boldsymbol{A}[t,:]\boldsymbol{G}$ and $\widehat{\boldsymbol{s}}_t = \mathcal{M}(\boldsymbol{G})[t,:]$, we can measure the ℓ_2 norm of the error of the t prefix sum estimate $\widehat{\boldsymbol{s}}_t$ as $\|\boldsymbol{s}_t - \widehat{\boldsymbol{s}}_t\|_2^2 = \|(\boldsymbol{B}[t,:]\,\boldsymbol{Z})\|_2^2$, and so

$$\mathbb{E}_{\mathcal{M}}\,\|\boldsymbol{s}_t - \widehat{\boldsymbol{s}}_t\|_2^2 = \mathbb{E}_{\boldsymbol{Z}}\,\|(\boldsymbol{B}[t,:]\,\boldsymbol{Z})\|_2^2 = m\nu^2\left(\sum_{\tau\in[n]} \boldsymbol{B}[t,\tau]^2\right)$$

because $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0,\nu^2)$ is the only source of randomness in $\mathcal{M}$. Therefore, we get that

$$\begin{aligned}\mathcal{L}_\infty(\boldsymbol{B},\boldsymbol{C}) &= \sqrt{m}\nu\ \max_{t\in[n]} \sqrt{\sum_{\tau\in[n]} \boldsymbol{B}[t,\tau]^2} \\ &= 2\sqrt{m}\sigma\,\|\boldsymbol{C}\|_{\mathsf{col}}\,\|\boldsymbol{B}\|_{\mathsf{row}}\end{aligned}$$

as required. □

Since the dimension m is a fixed constant for a given problem, and the noise multiplier σ is independent of the factorization $\boldsymbol{A} = \boldsymbol{BC}$ defining the correlated noise mechanism, we will state our results in

[2] In general, we use the notation $\|\boldsymbol{M}\|_{p\to q} \coloneqq \max_{\boldsymbol{u}\neq\boldsymbol{0}} \frac{\|\boldsymbol{Mu}\|_q}{\|\boldsymbol{u}\|_p}$ to denote the induced matrix norm.

terms of the **(normalized) max loss**; because this loss is the primary function considered when designing mechansimms, we will refer to the normalized max loss as simply "max loss" going forward.

Definition 2.3 (Max Loss)**.** For a correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z})$ for $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \nu^2)$, we define the (normalized) max loss by

$$\bar{\mathcal{L}}_\infty(\boldsymbol{B}, \boldsymbol{C}) \coloneqq \|\boldsymbol{B}\|_{\mathsf{row}} \, \|\boldsymbol{C}\|_{\mathsf{col}} \, . \tag{2.4}$$

Other loss metrics can be used; we discuss root-mean-squared-loss (RMS-loss) in Section 2.8. Throughout, for a factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$, we will use the term *loss* when we take into account the privacy of the mechanism, e.g. scaling the noise by the sensitivity of $\boldsymbol{C}$, and *error* when we measure to the magnitude of the noise $\boldsymbol{B}\boldsymbol{Z}$ introduced in our estimates of $\boldsymbol{A}\boldsymbol{G}$. Further, the loss generally scales linearly in the noise multiplier σ as in Theorem 2.2, and hence when evaluating the quality of a mechanism, we can ignore this term, leading to the notion of *normalized loss*. Hence, in general we have

$$\mathsf{loss}(\boldsymbol{B}, \boldsymbol{C}) = \mathsf{error}(\boldsymbol{B}) \cdot \mathsf{sens}(\boldsymbol{C}). \tag{2.5}$$

The error term can be varied, e.g. depending on whether we select the maximum per-iteration error (as in this section) or the mean error. In the streaming setting, $\mathsf{sens}(\boldsymbol{C}) = 2\,\|\boldsymbol{C}\|_{\mathsf{col}}$, as in Eq. (1.13), but in Section 3 we will generalize this.

2.1.3 Baseline Mechanisms

Recall the two baseline mechanisms of Section 1: the input perturbation ($\boldsymbol{C} = \boldsymbol{I}_{n\times n}$) and output perturbation ($\boldsymbol{C} = \boldsymbol{A}$). We review the complexity of noise generation and the utility bounds for both these cases.

Input Perturbation With $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$ as the identity matrix, the noise generation process in Eq. (2.3) is $(\boldsymbol{C}^{-1}\boldsymbol{Z})\,[t,:] = \boldsymbol{Z}[t,:] \sim \mathcal{N}_m(0, \sigma^2)$. In other words, it simplifies to generating independent Gaussian noise. This corresponds to the DP-SGD algorithm in the learning setting.

The per-step noise generation time is $O(m)$, which is required to sample m-dimensional white noise $\boldsymbol{Z}[t,:]$. This noise also requires $O(m)$ memory. As established in Section 1, its normalized max loss for the unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$ is $\bar{\mathcal{L}}_\infty(\boldsymbol{A}_{\mathsf{pre}}, \boldsymbol{I}) = \Theta(\sqrt{n})$.

Output Perturbation The $\boldsymbol{C} = \boldsymbol{A}$ baseline corresponds to perturbing the output $\boldsymbol{AG}$ with independent noise $\boldsymbol{Z}$. For the unweighted prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$, the noise generation process in Eq. (2.3) takes the form

$$\left(\boldsymbol{A}_{\mathsf{pre}}^{-1}\boldsymbol{Z}\right)[t,:] = \boldsymbol{Z}[t,:] - \boldsymbol{Z}[t-1,:]\,.$$

This follows, for example, from the formula for $\boldsymbol{C}^{-1}$ in Eq. (1.21).

The time complexity of the noise generation process remains $O(m)$ per-step even for output perturbation. Thus, in each step t, we need to store the noise $\boldsymbol{Z}[t-1,:]$ sampled from the previous time step in addition to $\boldsymbol{Z}[t,:]$; we still have an $O(m)$ memory overhead. Its utility bound $\bar{\mathcal{L}}_\infty(\boldsymbol{I}, \boldsymbol{A}_{\mathsf{pre}}) = \Theta(\sqrt{n})$ is the same as input perturbation, as we saw in Section 1.

The time and space complexities of these baselines are the best possible: $O(m)$ memory is required to store the input vector $\boldsymbol{g}_t \in \mathbb{R}^m$ and $O(m)$ time is required to process it. However, the $\Theta(\sqrt{n})$ utility is suboptimal as we saw in Section 1.4, where we showed that a (normalized) max loss of $O(n^{1/4})$ is achievable. We show that this can, in fact, be significantly improved to $\Theta(\ln(n))$. We summarize the mechanisms we study in this section in Table 2.1.

2.2 Dense Mechanism

The poor utility of the baselines stems from their disregard for the error metric. This can be remedied by optimizing for the normalized max loss $\bar{\mathcal{L}}_\infty(\boldsymbol{B}, \boldsymbol{C})$ directly when finding $\boldsymbol{B}, \boldsymbol{C}$:

$$\min\left\{\|\boldsymbol{B}\|_{\mathsf{row}}\|\boldsymbol{C}\|_{\mathsf{col}} \;:\; \begin{array}{c}\boldsymbol{BC} = \boldsymbol{A}, \quad \text{and} \\ \boldsymbol{B}, \boldsymbol{C} \text{ are lower-triangular}\end{array}\right\}. \tag{2.6}$$

While this optimization problem is non-convex, it can be cast as a semi-definite program, enabling high-quality solutions for small problems

Table 2.1: Summary of mechanisms considered in Section 2. Here, n is the number of steps, m is the model dimension, and c is an absolute constant that can change in each row. We describe the maximum time and space complexity of producing the correlated noise $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$ for any iteration $t \in [n]$. For most real-world applications, $m > n$ (possibly by orders of magnitude), and the time complexity becomes the dominating concern. The $O(m+n)$ space complexity for Toeplitz mechanisms holds for an arbitrary Toeplitz mechanism; when the Toeplitz coefficients can be easily (re)computed on the fly, e.g. as with the max-loss-optimal factorization of Theorem 2.5, the space requirement reduces to $O(m)$. The Banded Toeplitz and BLT mechanisms are in fact Toeplitz and so could be implemented with the $O(nm)$ time and $O(m+n)$ space, but the given time/space complexities achieved by specializing noise generation are almost always preferable.

Mechanism	**Time**	**Space**	**max loss $\bar{\mathcal{L}}_\infty$**
Input perturbation	$O(m)$	$O(m)$	$\Theta(\sqrt{n})$
Output perturbation	$O(m)$	$O(m)$	$\Theta(\sqrt{n})$
Dense	$O(nm)$	$O(m+n^2)$	$\frac{1}{\pi}\ln(n)+c$
Toeplitz	$O(nm)$	$O(m+n)$	$\frac{1}{\pi}\ln(n)+c$
b-Banded Toeplitz	$O(bm)$	$O(bm)$	See Theorem 2.13
d-buffer BLT, $d = \operatorname{poly}\ln(n)$	$O(dm)$	$O(dm)$	$\frac{1}{\pi}\ln(n)+c$

(with a few thousand steps or less); we discuss considerations around scalability of this approach in Section 4.

We refer to the mechanism defined by the resulting matrices, $\boldsymbol{B}$ and $\boldsymbol{C}$ as the *dense mechanism*. (These matrices are dense; in contrast, some of the mechanisms we consider later are either sparse or can be parameterized using a small number of parameters.)

Complexity of Noise Generation To compute the linear combination of noises $\sum_{\tau=0}^{t}\left(\boldsymbol{C}^{-1}\right)[t,\tau]\,\boldsymbol{Z}[\tau,:]$ in the noise generation process of Eq. (2.3) in step t requires $O(mt)$ time to generate and then sum over t white noise vectors in $\mathbb{R}^m$. Thus, the worst-case per-step time complexity of noise generation is $O(mn)$.

Storing the matrix $\boldsymbol{C}^{-1}$ requires $O(n^2)$ space. We assume the source i.i.d. noise $\boldsymbol{Z}[\tau,:]$ can be generated on the fly (e.g., via a random seed

hashed with τ), and hence materializing the noise $\boldsymbol{Z}[\tau, :]$ only takes space $O(m)$.

Utility and Error Bounds For the unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$, the optimal max loss from Eq. (2.6) can be precisely characterized with upper and lower bounds matching up to a small additive constant:

Theorem 2.4. Let $\boldsymbol{B}^\star, \boldsymbol{C}^\star$ be the minimizers of the optimization problem from Eq. (2.6) when $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ is the unweighted prefix sum matrix. Let

$$\mathsf{Opt} = \bar{\mathcal{L}}_\infty(\boldsymbol{B}^\star, \boldsymbol{C}^\star) = \|\boldsymbol{B}^\star\|_{\mathsf{row}} \|\boldsymbol{C}^\star\|_{\mathsf{col}} \tag{2.7}$$

denote the minimal normalized max loss. Then, we have the nearly matching upper and lower bounds:

$$\frac{\ln(2n+1)}{\pi} \leq \mathsf{Opt} \leq 1 + \frac{\ln(n)}{\pi},$$

where $\pi \approx 3.14$ is the ratio of a circle's circumference to its diameter and $\ln(\cdot)$ is the natural logarithm function.

While the full proof of this statement is out of the scope of this monograph, we give a detailed attribution of this theorem and the other upcoming results of this section in the bibliographic notes of Section 2.12. While Theorem 2.4 gives the asymptotic behavior of the max loss, we can numerically find high-precision solutions for small n (smaller than a few thousands; we discuss algorithms and their scalability issues in Section 4). We such solutions, we can numerically compute the optimal max loss for specific n (together with a certificate of optimality, such as a duality gap).

Summary The optimized dense mechanism's normalized max loss of $\ln(n)/\pi + c$ (for a small numerical constant c) is significantly better than that the $\Theta(\sqrt{n})$ error obtained from the baselines when factorizing $\boldsymbol{A}_{\mathsf{pre}}$. However, the time complexity of $O(mn)$ (and to a lesser extent, the space complexity of $O(n^2)$) can be prohibitive when the number of steps n is large. The mechanisms we discuss in the rest of the section will address this issue.

2.3 Toeplitz Mechanism

We start by addressing the high $O(n^2)$ space complexity of the optimized dense mechanism. While this is generally less of a concern than the $O(nm)$ time complexity in typical machine learning settings, it forms a good starting point to develop mechanisms with improved time complexity.

A common trick to overcome the $O(n^2)$ memory required to store an $n \times n$ matrix is to assume that it is *Toeplitz*, where each diagonal parallel to the main diagonal is a constant.[3] An $n \times n$ lower-triangular Toeplitz matrix $\boldsymbol{C}$ can be described in terms of its first column of n numbers:

$$\boldsymbol{C} = \begin{pmatrix} c_0 & 0 & 0 & \cdots & 0 \\ c_1 & c_0 & 0 & \cdots & 0 \\ c_2 & c_1 & c_0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{pmatrix}. \tag{2.8}$$

Toeplitz matrices offer not only reduced memory usage but also two remarkable properties that make them ideal for this application: (a) the optimal Toeplitz factorization can be determined analytically, and (b) they *almost* achieve the optimal max loss of Theorem 2.4, namely up to a small additive constant.

It is most interesting to restrict the strategy matrix $\boldsymbol{C}$ to be Toeplitz (and lower triangular) when the workload matrix $\boldsymbol{A}$ is also Toeplitz; in this case the matrix $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1}$ is also Toeplitz (and lower triangular). The unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$ satisfies this requirement. As we shall see in the upcoming Section 3, other common first-order optimizers also lead to workload matrices $\boldsymbol{A}$ that are Toeplitz and element-wise non-negative.

[3] Another common trick is to assume a low-rank factorization; we will return to that in Section 2.5.

Mechanism Definition The max-loss-optimal Toeplitz mechanism in terms of the max loss is the solution to the optimization problem:

$$\min \left\{ \|\boldsymbol{B}\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\mathsf{col}} : \begin{array}{c} \boldsymbol{BC} = \boldsymbol{A}, \quad \text{and} \\ \boldsymbol{B}, \boldsymbol{C} \text{ lower-triangular \& Toeplitz} \end{array} \right\}. \tag{2.9}$$

As in the earlier case, this is a non-convex optimization problem, which can be solved numerically. We restrict our discussion to the unweighted prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ in the following. For this matrix, Problem (2.9) admits an analytical solution for this case:

Theorem 2.5 (Max-Loss-Optimal Toeplitz Factorization)**.** For the unweighted prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$, the optimization problem from Eq. (2.9) is minimized by Toeplitz matrices $\boldsymbol{B}_{\mathsf{Toep}} = \boldsymbol{C}_{\mathsf{Toep}} = \boldsymbol{A}_{\mathsf{pre}}^{1/2}$, the square root matrix of $\boldsymbol{A}_{\mathsf{pre}}$.[4] In particular, their first column $c_0^\star, c_1^\star, \ldots, c_{n-1}^\star$ is given by

$$c_t^\star = (-1)^t \binom{-1/2}{t} = \begin{cases} 1\,, & \text{if } t = 0, \\ \Theta(t^{-1/2})\,, & \text{else}\,, \end{cases} \tag{2.10}$$

where we denote the generalized binomial coefficient $\binom{p}{t} := \prod_{\tau=0}^{t-1} \frac{p-\tau}{t-\tau}$ for $t \in \mathbb{N}$ and non-integer $p \in \mathbb{R}$. Moreover, $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$ is also a Toeplitz matrix whose first column $c_0', c_1', \ldots, c_{n-1}'$ is given by

$$c_t' = (-1)^t \binom{1/2}{t} = \begin{cases} 1\,, & \text{if } t = 0, \\ -\Theta(t^{-3/2})\,, & \text{else}\,. \end{cases} \tag{2.11}$$

Interestingly, the entries of the optimal factorization, as given in Eq. (2.10), are independent of the size n of the problem. See Section 2.9 for the key idea behind the proof.

Remark 2.6. The generalized binomial coefficients $\binom{-1/2}{t}$ and $\binom{1/2}{t}$ from Theorem 2.5 can directly be computed with standard library functions such as `scipy.special.binom`. We also have the alter-

[4] The square root $\boldsymbol{A}^{1/2}$ of the lower triangular matrix $\boldsymbol{A}$ with positive diagonal entries is the unique lower triangular matrix with positive diagonal entries such that $\boldsymbol{A} = \boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}$.

native expressions

$$c_t^\star = 2^{-2t}\binom{2t}{t} = \left(\frac{2t-1}{2t}\right) c_{t-1}^\star\,,$$

where the latter recursion starts from the base case of $c_0^\star = 1$. Similarly, using the relation $\boldsymbol{C}_{\mathsf{Toep}}^{-1} = \boldsymbol{A}_{\mathsf{pre}}^{-1}\boldsymbol{C}_{\mathsf{Toep}}$, we can express the coefficients c_t' of $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$ as

$$c_t' = \begin{cases} 1, & \text{if } t = 0, \\ c_{t+1}^\star - c_t^\star, & \text{if } t > 0. \end{cases}$$

Complexity of Noise Generation As with the dense mechanism, we assume we can re-generate the previous noises $\boldsymbol{z}_{t-1}, \boldsymbol{z}_{t-2}, \ldots$ in each iteration (using e.g. their random seeds). This approach suffers from a $O(mn)$ time complexity of the per-step noise generation, which is the same as the dense mechanism, as well as the same $O(m)$ memory requirement. For a general Toeplitz matrix, only $O(n)$ rather than $O(n^2)$ memory is required to store the strategy matrix $\boldsymbol{C}$. For the specific factorization of Theorem 2.7, the entries of $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$ can be generated as needed from Eq. (2.11) (e.g., using the standard library functions as in Remark 2.6), so in this case we do not need even the $O(n)$ space to store them.

It is possible to obtain an improved $O(mn\ln(n))$ time complexity by implementing the (left) multiplication by a Toeplitz matrix using the fast Fourier transform. Unfortunately, this requires $O(mn)$ space complexity to materialize all previous rows of the noise matrix $\boldsymbol{Z}$, which is generally prohibitive for typical AI and machine learning models.

Utility and Error Bounds This mechanism is near-optimal among the class of *all* (possibly non-Toeplitz) factorizations of the unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$ up to a small additive constant:

Theorem 2.7. Let $\mathsf{Opt}_{\mathsf{Toep}} = \bar{\mathcal{L}}_\infty(\boldsymbol{A}_{\mathsf{pre}}^{1/2}, \boldsymbol{A}_{\mathsf{pre}}^{1/2})$ denote the normalized max loss of the optimal lower triangular Toeplitz factors

$\boldsymbol{B}_{\mathsf{Toep}} = \boldsymbol{C}_{\mathsf{Toep}} = \boldsymbol{A}_{\mathsf{pre}}^{1/2}$ from Theorem 2.5. We have,

$$\mathsf{Opt}_{\mathsf{Toep}} \leq \frac{\gamma + \ln(n)}{\pi} + 1\,,$$

where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. In particular, we have $\mathsf{Opt}_{\mathsf{Toep}} - \mathsf{Opt} \leq 1$, where Opt is the best achievable normalized max loss by any mechanism as defined in Theorem 2.4.

Proof Sketch. The maximum row norm of $\boldsymbol{A}_{\mathsf{pre}}^{1/2}$ is that of its last row, while its maximum column norm is attained by its first column; this can be observed by the structure of the Toeplitz matrix in Eq. (2.8). Thus, we have,

$$\left\|\boldsymbol{A}_{\mathsf{pre}}^{1/2}\right\|_{\mathsf{row}}^2 = \left\|\boldsymbol{A}_{\mathsf{pre}}^{1/2}\right\|_{\mathsf{col}}^2 = \sum_{t=0}^{n-1} \binom{-1/2}{t}^2 \leq 1 + \sum_{k=1}^{n-1} \Theta\left(\frac{1}{k}\right).$$

Standard result of the harmonic sum gives $\sum_{k=1}^{n-1} \frac{1}{k} \leq \ln(n-1) + \gamma + 1/(2(n-1)) \leq \ln(n) + \gamma$, while a careful analysis of the coefficient $\binom{-1/2}{t} = \frac{1}{2^{2t}}\binom{2t}{t}$ reveals that the constant hidden in the big-Θ is $\frac{1}{\pi}$. □

In Section 2.6, we also verify empirically that the max-loss-optimal Toeplitz mechanism nearly matches the dense mechanism in terms of empirical performance.

Remark 2.8 (Lower Triangular and Toeplitz Factorizations)**.** Recall from Remark 1.17 that we restricted ourselves to lower triangular factorizations because given any factorization $\boldsymbol{BC} = \boldsymbol{A}$, there exists a lower triangular factorization $\boldsymbol{B'C'} = \boldsymbol{A}$ such that $\bar{\mathcal{L}}_\infty(\boldsymbol{B}, \boldsymbol{C}) = \bar{\mathcal{L}}_\infty(\boldsymbol{B'}, \boldsymbol{C'})$. However, this construction does not preserve the Toeplitz structure of the matrices. That is, optimal lower triangular and Toeplitz factorization (Eq. (2.9)) may attain worse normalized max loss than the optimal Toeplitz factorization *without* the lower triangular constraint. Fortunately, we know these are close from Theorem 2.7 as the latter always lies in the interval $[\mathsf{Opt}, \mathsf{Opt}_{\mathsf{Toep}}]$, and $\mathsf{Opt}_{\mathsf{Toep}} - \mathsf{Opt} \leq 1$ (independent of n).

Column Normalization to Improve the Mechanism We discuss a heuristic known as column normalization that can further improve the

max loss of the Toeplitz mechanism. It is motivated by the observation that optimal dense $\boldsymbol{C}^\star$ from Theorem 2.4 generally have equal column norms:

Lemma 2.9. There exists optimal solutions $\boldsymbol{B}^\star, \boldsymbol{C}^\star$ to the dense mechanism satisfying Eq. (2.7) (from Theorem 2.4) such that $\boldsymbol{C}^\star$ is column-normalized, i.e. $\|\boldsymbol{C}^\star[:,t]\|_2 = \|\boldsymbol{C}^\star\|_{\mathsf{col}}$ for all $t \in [n]$.

Unfortunately, Toeplitz matrices $\boldsymbol{C}$ (including those defined by Eq. (2.8)) cannot in general satisfy this property. This can be remedied by column normalization:

Definition 2.10 (Column Normalization)**.** Given a correlated noise mechanism based on a factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ with $\boldsymbol{C}$ invertible, its column normalized version is given by the factorization $\boldsymbol{A} = \boldsymbol{B}_{\mathsf{norm}}\boldsymbol{C}_{\mathsf{norm}}$ with

$$\boldsymbol{C}_{\mathsf{norm}}[:,t] = \frac{\boldsymbol{C}[:,t]}{\|\boldsymbol{C}[:,t]\|_2}, \quad \text{and} \quad \boldsymbol{B}_{\mathsf{norm}} = \boldsymbol{A}\boldsymbol{C}_{\mathsf{norm}}^{-1}.$$

As we discussed in Theorem 2.7, the suboptimality of the original max-loss-optimal Toeplitz mechanism is quite small. Yet, it can be shown that this mechanism can be improved slightly by column normalization at the same time and memory cost. (Note that the resulting $\boldsymbol{C}_{\mathsf{norm}}$ matrix is no longer Toeplitz.)

Theorem 2.11. Let $\mathsf{Opt}_{\mathsf{norm}} = \bar{\mathcal{L}}_\infty(\boldsymbol{B}_{\mathsf{norm}}, \boldsymbol{C}_{\mathsf{norm}})$ denote the normalized max loss of the column normalized version $\boldsymbol{C}_{\mathsf{norm}}$ of the max-loss-optimal Toeplitz mechanism's strategy matrix $\boldsymbol{A}_{\mathsf{pre}}^{1/2}$ and its corresponding factor $\boldsymbol{B}_{\mathsf{norm}} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}_{\mathsf{norm}}^{-1}$. We have,

$$\mathsf{Opt}_{\mathsf{norm}} \leq \frac{\ln(n)}{\pi} + 1,$$

In particular, this removes an extra γ/π term in Theorem 2.7, matching the upper bound of the dense mechanism in Theorem 2.4.

Summary The max-loss-optimal Toeplitz mechanism gives a tight *additive approximation* to Opt: it has the right asymptotic dependence of

$\ln n$ including the leading multiplicative constant of $1/\pi$. Its space complexity is also the best possible. The main drawback of this mechanism is the time complexity of noise generation. The next two mechanisms attain a better time complexity of noise generation, at the cost of increased space complexity.

2.4 Banded Toeplitz Mechanism

The banded Toeplitz mechanism reduces the per-step cost of noise generation by requiring that the strategy matrix $\boldsymbol{C}$ be *sparse* (in addition to lower triangular and Toeplitz) to allow for efficient noise generation (as we see in the upcoming Algorithm 2.1). Since the optimal Toeplitz coefficients from Theorem 2.5 are monotonically decreasing, it is natural to require that only the first b Toeplitz coefficients are non-zero. Such matrices are a special instance of the class of *banded matrices*:

Definition 2.12 (b-Banded Matrix)**.** A lower triangular matrix $\boldsymbol{M}$ is said to be *b-banded* if $\boldsymbol{M}[t, \tau] = 0$ for all $t - \tau \geq b$.[5]

This banded structure lends itself to more efficient noise generation algorithms, as we will momentarily see.

Mechanism Definition The banded Toeplitz mechanism aims to find the factorization with the smallest max loss subject to the band sparsity of the strategy matrix and Toeplitz constraints:

$$\min \left\{ \|\boldsymbol{B}\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\mathsf{col}} : \begin{array}{c} \boldsymbol{BC} = \boldsymbol{A}, \quad \boldsymbol{C}[t, \tau] = 0 \ \forall\, t - \tau \geq b\,, \\ \boldsymbol{B}, \boldsymbol{C} \text{ lower-triangular \& Toeplitz} \end{array} \right\}. \quad (2.12)$$

For example, for $n = 5$ and $b = 3$, such a mechanism can be defined by 3 parameters (c_0, c_1, c_2) as

$$\boldsymbol{C} = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 \\ c_1 & c_0 & 0 & 0 & 0 \\ c_2 & c_1 & c_0 & 0 & 0 \\ 0 & c_2 & c_1 & c_0 & 0 \\ 0 & 0 & c_2 & c_1 & c_0 \end{bmatrix}.$$

[5] Since $\boldsymbol{M}$ is lower triangular we have that $\boldsymbol{M}[t, \tau] = 0$ for all $t < \tau$ as well.

Algorithm 2.1 Noise Generation with the Banded Toeplitz Mechanism

Input: A b-banded lower-triangular Toeplitz matrix $\boldsymbol{C}$ whose first column has non-zero entries $c_0, c_1, \ldots, c_{b-1}$, i.i.d. noise $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$.

Output: $\widetilde{\boldsymbol{z}}_t = (\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$ for each t.

1: **for** $t = 0, \ldots, n-1$ **do**

2: Define $\boldsymbol{z}_t = \boldsymbol{Z}[t,:] \in \mathbb{R}^m$ and

$$\widetilde{\boldsymbol{z}}_t = \boldsymbol{z}_t - \frac{1}{c_0}\left(\sum_{\tau=1}^{\min\{t,b-1\}} c_\tau \widetilde{\boldsymbol{z}}_{t-\tau}\right)$$

3: **Yield** correlated noise $\widetilde{\boldsymbol{z}}_t$

Similar to the optimal factorization problem of Eq. (2.6), this is a non-convex optimization problem; we return to the optimization aspect in Section 4. We assume for now that a suitable factorization is available.

Complexity of Noise Generation A b-banded and lower-triangular linear system can be solved efficiently: each output element can be sequentially computed in $O(b)$ time, independent of the size n of the problem. The same approach can be adapted for the noise generation $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$, as show in Algorithm 2.1. In each step t of the algorithm, we need to maintain the $b-1$ previous outputs $\widetilde{\boldsymbol{z}}_1, \ldots, \widetilde{\boldsymbol{z}}_{t-b+1}$, leading to a space complexity of $O(mb)$. Each update in Line 2 simply takes a linear combination of these b previous outputs, so its time complexity is $O(mb)$ as well.

Utility and Error Bounds As previously, we focus on the unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$ for utility bounds. While precise max loss bounds for the solution of the optimization problem from Eq. (2.12) have not been established (to the best of our knowledge), we can quantify the error of a feasible solution which is obtained by "sparsifying" the optimal Toeplitz factorization:

Theorem 2.13. Fix the number of steps n and the number of bands $1 \leq b < n$. Define the matrix $\widehat{\boldsymbol{C}}_{\mathsf{band}} \in \mathbb{R}^{n \times n}$ to be the lower-

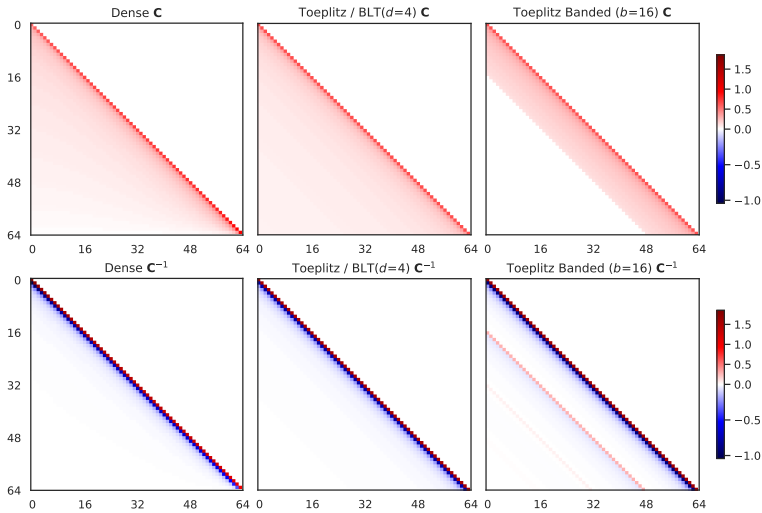


Figure 2.1: Examples of mechanisms as defined by $\left(\boldsymbol{C}, \boldsymbol{C}^{-1}\right)$ pairs for $n = 64$: **(Left)** A dense (arbitrary lower-triangular) matrix (Section 2.2), optimized for RMSE (see Section 2.8); **(Middle)** The max-loss-optimal Toeplitz mechanism of Theorem 2.5, which is extremely well approximated (to the point of visual indistinguishability in this plot) by a BLT of order $d = 4$; **(Right)** A banded Toeplitz mechanism optimized for max loss (Section 2.4). Optimizing mechanisms for scenarios with multiple participations can produce substantially different mechanisms, see Fig. 3.3.

triangular and Toeplitz matrix whose first column is made up of $c_0^\star, c_1^\star, \ldots, c_{b-1}^\star, 0, \ldots, 0$, where $c_t^\star$ is the optimal Toeplitz coefficient defined in Eq. (2.10) of Theorem 2.5. Letting $\widehat{\boldsymbol{B}}_{\mathsf{band}} = \boldsymbol{A}_{\mathsf{pre}}\widehat{\boldsymbol{C}}_{\mathsf{band}}^{-1}$, we have

$$\bar{\mathcal{L}}_\infty(\widehat{\boldsymbol{B}}_{\mathsf{band}}, \widehat{\boldsymbol{C}}_{\mathsf{band}}) \leq O\left(\sqrt{\left(\frac{n}{b} - 1\right)\ln b + \ln^2 b}\right).$$

The above bound shows that the error decreases monotonically with an increasing number of bands b. In particular, $b = n$ bands are optimal in the streaming setting, recovering the max-loss-optimal Toeplitz mechanism of Section 2.3. It is more practical to consider a small number of bands such as $b = O(1)$ or $b = O(\operatorname{poly}\ln(n))$. Unfortunately, the max loss bound is suboptimal in this regime; the bound is comparable to the $\Theta(\sqrt{n})$ bound obtained by the baseline mechanisms and is exponentially worse than the optimal $\Theta(\ln n)$ scaling of Theorem 2.4.

Remark 2.14 (Bandedness and Amplification by Sampling*)**.** While the banded Toeplitz mechanism appears to be suboptimal in terms of the max loss, it has two key advantages in the learning setting. First, the banded Toeplitz mechanism allows for significantly improved privacy guarantees via amplification by sampling, as we discuss in Section 3.4. Second, the banded Toeplitz mechanism allows for a natural and interpretable structure in the multi-participation setting. In both these cases, it is advantageous to take $b < n$—we discuss these factors further in Sections 3 and 4 (in particular, Fig. 4.3 shows the empirical optimal number of bands). These reasons make the banded Toeplitz mechanism a compelling option in practice. We refer to Section 4.4 for rules of thumb regarding the selection of different mechanisms.

Summary While the banded Toeplitz mechanism improves the time complexity of noise generation from $O(mn)$ of the max-loss-optimal Toeplitz mechanism to $O(mb)$, it comes at the cost of a (potentially) exponentially worse max loss. This would suggest that this mechanism requires a large number of bands b to attain competitive empirical per-

formance in the streaming setting, which in turn could make the $O(mb)$ space complexity prohibitively large. The next mechanism overcomes these limitations.

2.5 Buffered Linear Toeplitz (BLT) Mechanism

The Buffered Linear Toeplitz (BLT) mechanism takes a different approach to improving the time complexity of the max-loss-optimal Toeplitz mechanism in Section 2.3 while maintaining its near-optimal max loss.

We develop some intuition. Suppose the matrix $\boldsymbol{C}^{-1}$ takes the form

$$\boldsymbol{C}_\lambda^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -\lambda & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -\lambda^{n-1} & -\lambda^{n-2} & \cdots & 1 \end{pmatrix}$$

for a parameter $\lambda \in [0, 1)$.[6]

The exponential decay in the diagonals of $\boldsymbol{C}_\lambda^{-1}$ leads to an efficient recursive implementation of the noise generation. In particular, we can compute $\widetilde{\boldsymbol{z}}_t = (\boldsymbol{C}_\lambda^{-1}\boldsymbol{Z})[t, :]$ from $\boldsymbol{z}_t = \boldsymbol{Z}[t, :]$ recursively as

$$\widetilde{\boldsymbol{z}}_0 = \boldsymbol{z}_0 \quad \text{and} \quad \widetilde{\boldsymbol{z}}_t = \boldsymbol{z}_t - \lambda \widetilde{\boldsymbol{z}}_{t-1}\,. \tag{2.13}$$

Unfortunately, the optimal Toeplitz coefficients of $\boldsymbol{C}^{-1}$, which scale as $c'_0 = 1$ and $c'_t \approx -t^{-3/2}$ for $t > 1$ for the case of $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ (from Theorem 2.5), do not admit a similar efficiently-implementable recursion.[7]

The BLT mechanism of order d instead *approximates* the optimal Toeplitz coefficients $c^\star_t$ (e.g. from Theorem 2.5 for the unweighted prefix

[6] This is related to the one-parameter Toeplitz matrix considered in Eq. (1.20). The difference is that we parameterized the strategy matrix $\boldsymbol{C}$ with an exponentially decaying sequence in Section 1, while we now parameterize $\boldsymbol{C}^{-1}$. We will see in Lemma 2.16 that this relationship can be made precise.

[7] In particular, there exist no finite collection of numbers $q_1, \ldots, q_d \in \mathbb{R}$ for any $d < \infty$ such that $\sum_{i=0}^{d} q_i c'_{t-i} = 0$ holds (with $q_0 = 1$) for all $t \geq d$. This characterizes the general class of linear recurrences, for which an efficiently implementable recursion like Eq. (2.13) exists. We return to this in Section 2.9.

sum workload $\boldsymbol{A}_{\mathsf{pre}}$) with a linear combination of exponentials:

$$c_t^\star \approx \sum_{i=1}^{d} \alpha_i \lambda_i^{t-1} \tag{2.14}$$

for $t > 0$ using some scale parameters $\alpha_1, \ldots, \alpha_d$ and decay parameters $\lambda_1, \ldots, \lambda_d \in [0, 1)$.

There are two reasons to utilize this approximation. First, it turns out that the BLT parameterization from Eq. (2.14) can effectively approximate decreasing non-negative sequences, including the optimal Toeplitz coefficients $c_t^\star$. We briefly discuss this in Remark 2.15 below and present a more detailed intuition in Section 2.9. Secondly, the BLT parameterization allows an efficient recursion for noise generation $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t, :]$ that can be implemented (as shown in the upcoming Lemma 2.17) with a per-step time complexity and total space complexity of $O(md)$. This can be viewed as d buffers of the same size m as the model parameters. In practice, a small constant order d (e.g. $d < 5$) can provide near-optimal performance empirically (see also Section 2.6), so this is effectively only a small constant-factor overhead of storing and computing on the model of size m.

Remark 2.15 (BLT and Fourier Approximations*)**.** For readers familiar with the Fourier transform, it is instructional to view its parallels with the BLT parameterization from Eq. (2.14). Any sequence $(c_t)_{t=0}^\infty$ can be expressed in the Fourier basis $\omega \mapsto \left(\exp(\iota\,\omega t)\right)_{t=0}^\infty$ (where $\iota = \sqrt{-1}$ is the imaginary unit) as

$$c_t = \frac{1}{2\pi} \int_{-\pi}^{\pi} F_c(\omega) \exp(\iota\,\omega t)\ \mathrm{d}t\,,$$

where $F_c : [-2\pi, 2\pi] \to \mathbb{C}$ is the discrete-time Fourier transform of the sequence $(c_t)_{t=0}^\infty$. It is common practice to approximate this integral as a sum over d points $\omega_1, \ldots, \omega_d \in \mathbb{C}$ with weights $\beta_1, \ldots, \beta_d \in \mathbb{C}$:

$$c_t \approx \sum_{i=1}^{d} \beta_i \exp(\iota\,\omega_i t)\,. \tag{2.15}$$

> The BLT approximation in Eq. (2.14) is analogous to this, with the key distinction that is a linear combination of real and decreasing exponential functions $\exp(-\mu_i t)$ (where $\mu_i = \ln(1/\lambda_i) > 0$), making it suitable to approximate decreasing functions.

Mechanism Definition A BLT mechanism of order-d is parameterized by a scale parameter $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{R}^d$ and a decay parameter $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d) \in [0, 1)^d$. It represents the Toeplitz matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ as

$$\mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \coloneqq \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \sum_{i=1}^d \alpha_i & 1 & \cdots & \cdots & 0 \\ \sum_{i=1}^d \alpha_i \lambda_i & \sum_{i=1}^d \alpha_i & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^d \alpha_i \lambda_i^{n-2} & \sum_{i=1}^d \alpha_i \lambda_i^{n-3} & \sum_{i=1}^d \alpha_i \lambda_i^{n-4} & \cdots & 1 \end{pmatrix}. \tag{2.16}$$

With slight abuse of nomenclature, we refer to matrices $\mathsf{BLT}(\boldsymbol{a}, \boldsymbol{\lambda})$ parameterized in this fashion as "BLT matrices". The best parameters $\boldsymbol{a}, \boldsymbol{\lambda}$ can be found numerically as solutions to the optimization problem:

$$\min \left\{ \|\boldsymbol{B}\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\mathsf{col}} : \begin{array}{c} \boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda}),\ \boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1}, \\ \boldsymbol{\alpha} \in \mathbb{R}_+^d, \boldsymbol{\lambda} \in [0, 1)^d \end{array} \right\}, \tag{2.17}$$

where $\mathbb{R}_+$ denotes the set of positive real numbers. Note that we constrain the scale parameters $\boldsymbol{\alpha}$ to be non-negative in Problem (2.17). This is because the optimal Toeplitz coefficients $c_t^\star$ we wish to approximate are non-negative for typical workload matrices $\boldsymbol{A}$ encountered in machine learning. For instance, we see from Theorem 2.5 that this is true for the unweighted prefix sum workload $\boldsymbol{A}_{\mathsf{pre}}$.

While the optimization problem (2.17) is non-convex, it turns out that we can obtain empirically high-quality solutions within a few seconds for n up to several billions. We return to the optimization aspect in Section 4 and assume for now that suitable parameters $\boldsymbol{\alpha}, \boldsymbol{\lambda}$ are available.

Noise Generation BLT noise generation is efficiently implementable via a suitable recursion. While we wish to compute[8] $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$ for $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, it is instructive to first consider how to compute $\boldsymbol{C}[t,:]\boldsymbol{Z}$ for $t \in [n]$. Recalling $\boldsymbol{z}_t = \boldsymbol{Z}[t,:]$ is the t^{th} row of $\boldsymbol{Z}$, we have

$$\boldsymbol{C}[t,:]\boldsymbol{Z} = \boldsymbol{z}_t + \sum_{\tau=1}^{t}\left(\sum_{i=1}^{d}\alpha_i\lambda_i^{\tau-1}\right)\boldsymbol{z}_{t-\tau} = \boldsymbol{z}_t + \sum_{i=1}^{d}\alpha_i\boldsymbol{m}_{t,i} \tag{2.18}$$

where we have d memory buffers $\boldsymbol{m}_{t,i}$ for $i \in 1, \dots d$ given by $\boldsymbol{m}_{0,i} \coloneqq \boldsymbol{0}$ and for $t \geq 1$, and

$$\boldsymbol{m}_{t,i} \coloneqq \sum_{\tau=1}^{t}\lambda_i^{\tau-1}\boldsymbol{z}_{t-\tau} = \boldsymbol{z}_{t-1} + \lambda_i\boldsymbol{z}_{t-2} + \cdots + \lambda_i^{t-1}\boldsymbol{z}_0 \in \mathbb{R}^m.$$

The key to an efficient implementation is the following recursion akin to Eq. (2.13):

$$\boldsymbol{m}_{0,i} = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{m}_{t+1,i} = \boldsymbol{z}_t + \lambda_i\boldsymbol{m}_{t,i}\,.$$

This recipe can directly be used to generate the noise $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$ since $\boldsymbol{C}^{-1}$ exists and is also BLT:

Lemma 2.16 (Inverse BLT parameterization)**.** Any matrix $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \in \mathbb{R}^{n\times n}$ that BLT-parameterized matrix is invertible. Further, suppose the scale parameters are positive (i.e. $\alpha_i > 0$ for all i) with $\sum_{i=1}^{d}\alpha_i < 1$, and the decay parameters $\boldsymbol{\lambda} \in (0,1)^d$ are unique (i.e. $\lambda_i \neq \lambda_j$ for $i \neq j$). Then, there exist parameters $\widehat{\boldsymbol{\alpha}} \in \mathbb{R}^d$ and $\widehat{\boldsymbol{\lambda}} \in [-1,1]^d$ such that $\boldsymbol{C}^{-1} = \mathsf{BLT}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$. Furthermore, the map $(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \mapsto (\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ is continuously differentiable and can be implemented in $O(d^3)$ time.

The scale parameter $\boldsymbol{\alpha}$ of the BLT is positive element-wise while the corresponding parameter $\widehat{\boldsymbol{\alpha}}$ of the inverse BLT is element-wise negative. This mimics the optimal Toeplitz coefficients of Theorem 2.5: we have that the coefficients $c_t^\star = \Theta(t^{-1/2})$ of the Toeplitz matrix $\boldsymbol{C}_{\mathsf{Toep}}$ are positive, while the corresponding coefficients $c_t' = -\Theta(t^{-3/2})$ of $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$ are negative for $t \geq 1$.

[8] The matrix $\mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is invertible. Indeed, as it is a lower triangular matrix with all ones along the diagonal, we have that all of its eigenvalues equal 1.

Algorithm 2.2 Noise Generation with BLTs

Input: Degree-d BLT $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, i.i.d. noise $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$.
Output: $\widetilde{\boldsymbol{z}}_t = (\boldsymbol{C}^{-1}\boldsymbol{Z})[t, :]$ for each t.

1: Initialize buffer $\boldsymbol{M}_0 = \boldsymbol{0}_{m \times d}$
2: **for** $t = 0, \ldots, n-1$ **do**
3: $\quad \widetilde{\boldsymbol{z}}_t = \boldsymbol{z}_t - \boldsymbol{M}_t \boldsymbol{\alpha}$ with $\boldsymbol{z}_t = \boldsymbol{Z}[t, :] \in \mathbb{R}^m$ ▶ *Noise generation*
4: $\quad \boldsymbol{M}_{t+1} = \boldsymbol{M}_t \boldsymbol{\Lambda} + \widetilde{\boldsymbol{z}}_t \boldsymbol{1}_d^\top$ with $\boldsymbol{\Lambda} = \mathsf{diag}(\boldsymbol{\lambda})$ ▶ *Buffer update*
5: $\quad$ **Yield** correlated noise $\widetilde{\boldsymbol{z}}_t$

The parameters $\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}}$ of the inverse BLT can be computed using an eigenvalue decomposition of a non-symmetric $d \times d$ matrix, a common sub-routine in numerical software. Therefore, Lemma 2.16 allows us to use Eq. (2.18) for efficient noise generation.

Alternatively, instead of deriving the BLT parameters for $\boldsymbol{C}^{-1}$, we can directly compute $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t, :]$ from the BLT parameterization of $\boldsymbol{C}$. This approach, given as Algorithm 2.2, may be preferable in practice as it avoids potential numerical issues in the computation of the inverse parameters.

Lemma 2.17 (Correctness of BLT noise generation)**.** The output $\widetilde{\boldsymbol{Z}} = (\widetilde{\boldsymbol{z}}_0, \ldots, \widetilde{\boldsymbol{z}}_{n-1}) \in \mathbb{R}^{n \times m}$ of Algorithm 2.2 with inputs $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ and a matrix $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$ satisfies $\widetilde{\boldsymbol{Z}} = \boldsymbol{C}^{-1}\boldsymbol{Z}$.

Proof Sketch. We start by noting that the memory buffer $\boldsymbol{M}_t$ satisfies

$$\boldsymbol{M}_t = \begin{pmatrix} \sum_{\tau=1}^{t} \lambda_1^{\tau-1} \widetilde{\boldsymbol{z}}_{t-\tau} & \cdots & \sum_{\tau=1}^{t} \lambda_d^{\tau-1} \widetilde{\boldsymbol{z}}_{t-\tau} \end{pmatrix} \in \mathbb{R}^{m \times d}.$$

This can be proved, for instance, by induction. Next, we use this formula together with Line 3 of Algorithm 2.2 to expand out

$$\boldsymbol{z}_t = \widetilde{\boldsymbol{z}}_t + \boldsymbol{M}_t \boldsymbol{\alpha} = \widetilde{\boldsymbol{z}}_t + \sum_{i=1}^{d} \alpha_i \sum_{\tau=1}^{t} \lambda_i^{\tau-1} \widetilde{\boldsymbol{z}}_{t-\tau}.$$

Comparing this with Eq. (2.18) gives $\boldsymbol{Z} = \boldsymbol{C}\widetilde{\boldsymbol{Z}}$. Thus, we have $\widetilde{\boldsymbol{Z}} = \boldsymbol{C}^{-1}\boldsymbol{Z}$ and Algorithm 2.2 gives the correctly correlated noise. □

Complexity of Noise Generation Noise generation with a d-buffer BLT requires $O(md)$ space to store the memory buffer $\boldsymbol{M}_t$ of Algo-

rithm 2.2. Its per-step time-complexity is $O(md)$ to generate the correlated noise and update the buffer. It typically suffices to take $d \ll n$ to get competitive performance (see the error bound below). This is a huge improvement in the running time of $O(mn)$ of the max-loss-optimal Toeplitz mechanism and the banded Toeplitz mechanism.

Utility and Error Bounds As previously, we give utility bounds for the unweighted prefix sum workload $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$. It turns out to be sufficient to take $d = \mathrm{poly}(\ln(n))$ to get a competitive approximation guarantee:

Theorem 2.18. Fix the number of steps n, and error term $\delta > 0$. There exists a d-buffer BLT matrix $\boldsymbol{C}_{\mathsf{blt}}$ with $d = O(\ln^2(n/\delta))$ and its corresponding factor $\boldsymbol{B}_{\mathsf{blt}} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}_{\mathsf{blt}}^{-1}$ such that

$$\bar{\mathcal{L}}_\infty(\boldsymbol{B}_{\mathsf{blt}}, \boldsymbol{C}_{\mathsf{blt}}) \leq \mathsf{Opt}_{\mathsf{Toep}} + \delta\,,$$

where $\mathsf{Opt}_{\mathsf{Toep}} = \bar{\mathcal{L}}_\infty(\boldsymbol{A}_{\mathsf{pre}}^{1/2}, \boldsymbol{A}_{\mathsf{pre}}^{1/2})$ is the optimal normalized max loss achievable by any Toeplitz factorization as defined in Theorem 2.7.

Theorem 2.18 implies that $\bar{\mathcal{L}}_\infty(\boldsymbol{B}_{\mathsf{blt}}, \boldsymbol{C}_{\mathsf{blt}}) \leq \ln(n)/\pi + \mathsf{constant}$ achieves the optimal asymptotic rate of $\ln(n)$ and optimal leading constant $1/\pi$. To get this *additive* approximation on the optimal max loss Opt, the explicit construction used in Theorem 2.18 requires $d = O(\ln^2(n))$ buffers, leading to a space and time complexity of $O(m \ln^2(n))$. However, directly optimizing $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ for a specific n can in practice produce BLTs with equivalent performance and substantially smaller number d of dimensions. We discuss the direct optimization of BLTs in Section 4.3.2.

Remark 2.19 (Column Normalization of BLTs)**.** Column normalization, defined in Definition 2.10, can also yield improvements to the max loss BLT mechanism. In particular, given a fixed number of steps n, the efficient noise generation approach of Algorithm 2.2 can be extended to column-normalized BLTs at the same time and space complexity. We leave the details as an exercise to the reader.

Summary The BLT mechanism attains a favorable tradeoff between noise generation complexity and utility. It is the only mechanism (as of this writing) that simultaneously admits an *additive* max loss guarantee and a poly(ln(n)) time and space complexity of noise generation.

2.6 Empirical Comparison of the Mechanisms

We numerically compute the optimal strategy within each class of factorizations defined above, and report the max loss for different values of the number of steps n in the streaming setting in Fig. 2.2.[9]

In addition to the correlated noise mechanism described in this section, we also compare the max loss of the so called *binary tree mechanism* (annotated as Tree Aggregation in Fig. 2.2), historically, the first mechanism for estimating prefix sums with poly $\log(n)$ error and $\log(n)$ space and time requirement. This mechanism also has a correlated noise mechanism perspective. Since it is not the central aspect of this monograph, we cover it in more detail in Section 2.7.

The empirical findings closely mirror the theoretical max loss bounds presented so far.

1. First, we note the sub-optimality of the baselines from the top plot of Fig. 2.2. Indeed, the input/output perturbation lines are identical and have a slope of 0.5 (up to an error of 10^{-5}) in the log-log plot. This indicates a $\sqrt{n}$ max loss scaling, as established.[10]

2. Second, tree aggregation (Section 2.7) which achieves the optimal $O(\ln n)$ rate, but its leading constant is sub-optimal. This leads to significantly worse empirical performance than methods like the BLT mechanism that can attain an additive performance guarantee. Third, the BLT mechanism gives a tight approximation to the dense mechanism and its upper bound in the top plot.

[9] In the setting that we consider here, the recommended number of bands b for the banded Toeplitz mechanism is equal to the number of steps n (see also Remark 2.14), and hence banded Toeplitz mechanism of Section 2.4 is equivalent to max-loss-optimal Toeplitz mechanism of Section 2.3. Thus, we omit the banded Toeplitz mechanism from this comparison.

[10] The log-log plot of $f(n) = c\,x^a$ versus n appears as a straight line with slope a.

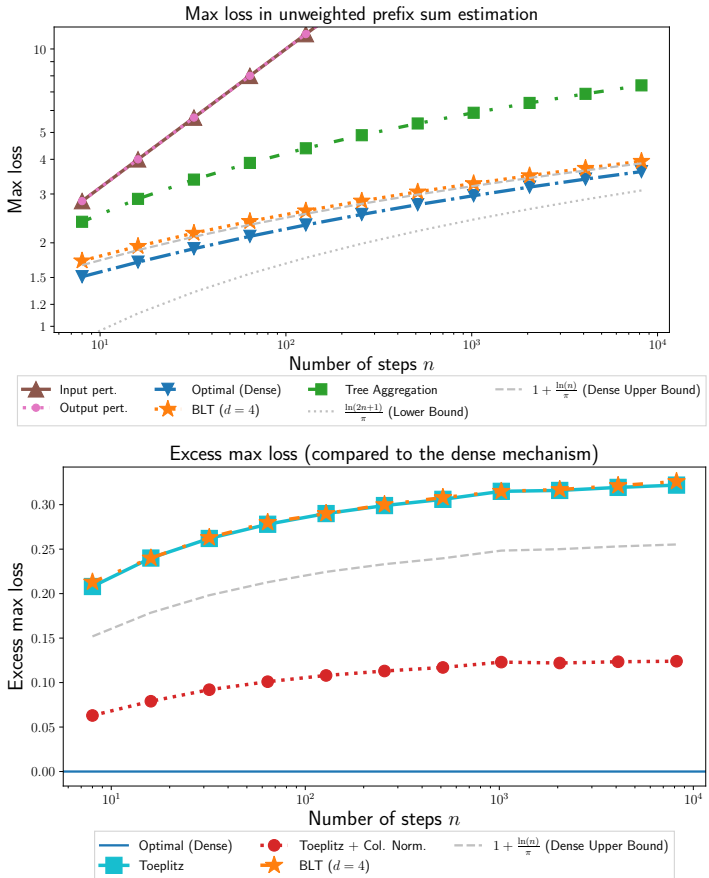


Figure 2.2: An empirical comparison of the max loss for various mechanisms in the context of streaming unweighted prefix sum estimation. **Top**: A log-log plot of the max loss of the baselines (input and output perturbation), tree aggregation (Section 2.7; optimal rate but suboptimal leading coefficient), BLT (Section 2.5; optimal rate and leading coefficient) with $d = 4$ buffers, the optimal dense mechanism (Section 2.2). For reference, we also plot the lower/upper bounds on the dense max loss from Theorem 2.4. The BLT parameters are optimized for the max loss (as described in Section 4). **Bottom**: A comparison of the excess max loss $\bar{\mathcal{L}}_\infty(\boldsymbol{B}, \boldsymbol{C}) - \bar{\mathcal{L}}_\infty(\boldsymbol{B}^\star, \boldsymbol{C}^\star)$ of mechanisms defined by the factorization $\boldsymbol{A}_{\text{pre}} = \boldsymbol{B}\boldsymbol{C}$ compared to the dense mechanism $\boldsymbol{A}_{\text{pre}} = \boldsymbol{B}^\star\boldsymbol{C}^\star$. Only mechanisms that attain the optimal leading constant (i.e., additive approximation guarantee), including column normalization (Definition 2.10) are shown. See Section 2.11 for the exact numerical values plotted here. Note the excess max loss for all of these mechanisms is $o(n)$.

We turn to the bottom plot of Fig. 2.2 for more detail. Here, we see that the BLT mechanism's performance is nearly indistinguishable from the max-loss-optimal Toeplitz mechanism up to the resolution of this plot, while being computationally more efficient. We also note that column normalization improves the max loss of the Topelitz mechanism, while maintaining the same running time.

In conclusion, the theoretical bounds presented in this section capture the true empirical behavior. For this streaming prefix sum setting, we recommend using the dense mechanism when its run-time overhead is not prohibitive, and the BLT mechanism otherwise. We will revisit these recommendations in the context of training AI and machine learning models in Section 4.4.

2.7 Tree Aggregation*

The tree aggregation mechanism alleviates the high space and time complexity of the optimized dense mechanism by leveraging *sparsity*. Also known as the *binary tree* mechanism, it constructs a factorization $\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{C}_{\mathsf{tree}} = \boldsymbol{A}_{\mathsf{pre}}$ with sparse rectangular matrices $\boldsymbol{B}_{\mathsf{tree}} \in \{0,1\}^{n\times(2n-1)}$ and $\boldsymbol{C}_{\mathsf{tree}} \in \{0,1\}^{(2n-1)\times n}$ whose non-zero entries are given by a binary tree data structure, as illustrated in Fig. 2.3.[11]

Mechanism Definition Consider an input sequence $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}$ where $n = 2^k$ as the leaves of a (complete) binary tree.[12]. Each intermediate node of the tree is assigned the sum of the leaves of its sub-tree. In the example of Fig. 2.3, we have

$$\begin{aligned}
&\boldsymbol{s}_{0,1} = \boldsymbol{s}_0 + \boldsymbol{s}_1, \quad \boldsymbol{s}_{2,3} = \boldsymbol{s}_2 + \boldsymbol{s}_3,\\
&\boldsymbol{s}_{4,5} = \boldsymbol{s}_4 + \boldsymbol{s}_5, \quad \boldsymbol{s}_{6,7} = \boldsymbol{s}_6 + \boldsymbol{s}_7,\\
&\boldsymbol{s}_{0,3} = \sum_{t=0}^{3} \boldsymbol{g}_t, \quad \boldsymbol{s}_{4,7} = \sum_{t=4}^{7} \boldsymbol{g}_t, \quad \text{and} \quad \boldsymbol{s}_{0,7} = \sum_{t=0}^{7} \boldsymbol{g}_t.
\end{aligned}$$

[11] While we construct a factorization with rectangular $\boldsymbol{B}_{\mathsf{tree}}, \boldsymbol{C}_{\mathsf{tree}}$ matrices, we can obtain square matrices $\boldsymbol{B}'_{\mathsf{tree}}, \boldsymbol{C}'_{\mathsf{tree}}$ by dropping some unused rows of $\boldsymbol{C}_{\mathsf{tree}}$ and columns of $\boldsymbol{B}_{\mathsf{tree}}$, as will see soon.

[12] This is just for the ease of presentation. For n which is not a power of two, we can simply pad our input sequence with zero vectors to length $n' = 2^{\lceil \log_2(n) \rceil}$

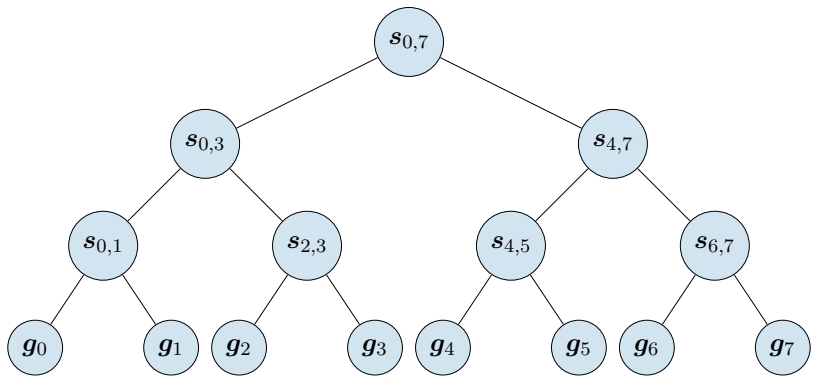


Figure 2.3: The **tree aggregation** mechanism generates privatized prefix sums using a binary tree data structure. The input vectors $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_7$ are arranged as leaves of the binary tree and each non-leaf node contains the sum of the leaves within its sub-tree. For instance, $\boldsymbol{s}_{0,1} = \boldsymbol{g}_0 + \boldsymbol{g}_1$, while $\boldsymbol{s}_{4,7} = \boldsymbol{g}_4 + \cdots + \boldsymbol{g}_7$. The root node $\boldsymbol{s}_{0,7} = \boldsymbol{g}_0 + \cdots + \boldsymbol{g}_7$ represents the sum of all input vectors. The core idea behind this mechanism is that any prefix sum $\boldsymbol{g}_0 + \cdots + \boldsymbol{g}_{t-1}$ can be expressed as a sum of at most $\log_2(n)$ nodes. This is achieved using a dyadic partition of the interval $[0, t-1]$. For example, $\boldsymbol{g}_0 + \cdots + \boldsymbol{g}_6 = \boldsymbol{s}_{0,3} + \boldsymbol{s}_{4,5} + \boldsymbol{g}_6$ corresponds to the partition $[0, 6] = [0, 3] \cup [4, 5] \cup [6]$. Once we privatize each intermediate $\boldsymbol{s}_{i,j}$ with white Guassian noise $\boldsymbol{z}_{i,j}$ as $\widehat{\boldsymbol{s}}_{i,j} = \boldsymbol{s}_{i,j} + \boldsymbol{z}_{i,j}$, we can compute any prefix sum by adding at most $\log_2(n)$ noise vectors.

Every intermediate node is of the form $\boldsymbol{s}_{j2^\ell,(j+1)2^\ell-1}$ for some $j \geq 0$ and ℓ such that its sub-tree contains 2^l leaves. In fact, it represents a *dyadic interval*: the $(\ell+1)^{\text{st}}$ bit (from the right) in the binary representation of each of the leaves in the sub-tree is the same. For instance, the node $\boldsymbol{s}_{4,5}$ corresponds to $\ell = 1, j = 2$, and its leaves are $4 = (1\underline{0}0)_2$ and $5 = (1\underline{0}1)_2$ with a common bit $\underline{0}$ (underlined). Similarly, the node $\boldsymbol{s}_{0,3}$ corresponds to $\ell = 2, j = 0$ and $\boldsymbol{s}_{4,7}$ corresponds to $\ell = 2, j = 1$.

The main idea underlying the binary tree mechanism is that any prefix sum $\boldsymbol{g}_0 + \cdots + \boldsymbol{g}_{t-1}$ can be expressed as a sum of at most $\log_2(n)$ nodes. This is achieved using a maximal *dyadic partition* of the interval $[0, t-1]$. For example, $\boldsymbol{g}_0 + \cdots + \boldsymbol{g}_6 = \boldsymbol{s}_{0,3} + \boldsymbol{s}_{4,5} + \boldsymbol{g}_6$ corresponds to the partition $[0, 6] = [0, 3] \cup [4, 5] \cup [6]$.

This procedure corresponds to a certain factorization $\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{C}_{\mathsf{tree}} = \boldsymbol{A}_{\mathsf{pre}}$. Indeed, the matrix $\boldsymbol{C}_{\mathsf{tree}} \in \{0,1\}^{(2n-1)\times n}$ gives us all intermediate nodes in the tree, while the matrix $\boldsymbol{B}_{\mathsf{tree}} \in \{0,1\}^{n\times(2n-1)}$ sums up the nodes forming a maximal dyadic partition of any prefix sum. To concretely write out these matrices, let us index the $2n-1$ nodes of the tree using a *postorder traversal*. For the example of Fig. 2.3, this is

$$\boldsymbol{g}_0, \boldsymbol{g}_1, \boldsymbol{s}_{0,1}, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{s}_{2,3}, \boldsymbol{s}_{0,3}, \boldsymbol{g}_4, \boldsymbol{g}_5, \boldsymbol{s}_{4,5}, \boldsymbol{g}_6, \boldsymbol{g}_7, \boldsymbol{s}_{6,7}, \boldsymbol{s}_{4,7}, \boldsymbol{s}_{0,7}\,.$$

Denoting $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n\times m}$ as the input matrix, we define $\boldsymbol{C}_{\mathsf{tree}}$ such that $(\boldsymbol{C}_{\mathsf{tree}}\boldsymbol{g})[p, :]$ returns the value of the p^{th} node in the postorder traversal. Then, we can express $\boldsymbol{C}_{\mathsf{tree}} = \boldsymbol{P}_k$ based on the recursive definition $\boldsymbol{P}_0 = (1) \in \mathbb{R}^{1\times 1}$ and

$$\boldsymbol{P}_{i+1} = \begin{pmatrix} \boldsymbol{P}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_i \\ \boldsymbol{1} & \boldsymbol{1} \end{pmatrix}.$$

Indeed, the two $\boldsymbol{P}_{i-1}$'s give the postorder traversal of the left and right sub-trees respectively, while the final row of ones produces the sum of the entire sub-tree, completing the postorder traversal.

Finally, we take $\boldsymbol{B}_{\mathsf{tree}}[t, p] = 1$ if the corresponding node p in the postorder traversal is used in the computation of the prefix sum $\boldsymbol{g}_0 + \cdots + \boldsymbol{g}_{t-1}$. For the example of Fig. 2.3, we have (with the index nodes

of the rows and columns shown in orange):

$$\boldsymbol{B} = \begin{array}{c} \\ g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} g_0 & g_1 & s_{0,1} & g_2 & g_3 & s_{2,3} & s_{0,3} & g_4 & g_5 & s_{4,5} & g_6 & g_7 & s_{6,7} & s_{4,7} & s_{0,7} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

The columns corresponding to some nodes such as $\boldsymbol{G}_1$ and $\boldsymbol{s}_{2,3}$ are all zeros because they never appear in any maximal dyadic partition; such nodes can be dropped to obtain a square factorization.

Complexity of Noise Generation Note that each row of $\boldsymbol{B}_{\mathsf{tree}}$ contains at most $\lceil \log_2(n) \rceil$ non-zeros, as this is the largest size of any maximal dyadic partition of $[0, t-1]$ for $1 \leq t \leq n$. Thus, we can compute $(\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{Z})[t,:]$ in $O(m \lceil \log_2(n) \rceil)$ time by summing up these non-zero noise vectors. This immediately gives us the correlated noise $\boldsymbol{C}_{\mathsf{tree}}^{-1}\boldsymbol{Z}$, since

$$(\boldsymbol{C}_{\mathsf{tree}}^{-1}\boldsymbol{Z})[t,:] = (\boldsymbol{A}_{\mathsf{pre}}^{-1}\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{Z})[t,:] = (\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{Z})[t,:] - (\boldsymbol{B}_{\mathsf{tree}}\boldsymbol{Z})[t-1,:]\,.$$

This follows from the formula for $\boldsymbol{A}_{\mathsf{pre}}^{-1}$, which can be inferred from Eq. (1.21). Similarly, we need to materialize $\lceil \log_2(n) \rceil$ noise vectors in $\mathbb{R}^m$, leading to a space complexity of $O(m \lceil \log_2(n) \rceil)$.

Utility and Error Bounds The tree aggregation mechanism can attain the optimal error up to a multiplicative factor:

Theorem 2.20. For matrices $\boldsymbol{B}_{\mathsf{tree}}$ and $\boldsymbol{C}_{\mathsf{tree}}$ obtained by the tree aggregation algorithm, we have that

$$\bar{\mathcal{L}}_\infty(\boldsymbol{B}_{\mathsf{tree}}, \boldsymbol{C}_{\mathsf{tree}}) \leq \sqrt{\lceil \log_2(n) \rceil \left(1 + \lceil \log_2(n) \rceil\right)}\,.$$

Proof Sketch. We have already argued that the maximum number of non-zero entries in any row of $\boldsymbol{B}_{\mathsf{tree}}$ is $\lceil \log_2(n) \rceil$, so that $\|\boldsymbol{B}_{\mathsf{tree}}\|_{\mathsf{row}}^2 \leq \lceil \log_2(n) \rceil$. We can also argue (e.g. by induction) that the maximum

number of non-zero entries in any column of $\boldsymbol{C}_{\mathsf{tree}}$ is $\lceil \log_2(n) \rceil + 1$, leading to $\|\boldsymbol{C}_{\mathsf{tree}}\|_{\mathsf{col}}^2 \leq 1 + \lceil \log_2(n) \rceil$. □

Thus, tree aggregation has the same asymptotic $\Theta(\ln n)$ max loss as the optimized dense mechanism (Theorem 2.4). However, it is suboptimal by a multiplicative factor of approximately $\pi / \ln 2 \approx 4.5$. Several improvements have been proposed to improve this leading constant (see Bibliographic Notes in Section 2.12), yet all of them remain suboptimal. In this problem, the multiplicative constants in the error have a larger practical impact on the mechanism's utility than an additive error: this is also illustrated by the empirical comparisons of Section 2.6.

Summary In summary, the binary tree-based mechanisms have a near-optimal time and space complexity of $O(m \ln n)$, improving greatly over the dense mechanism. This even better than the BLT mechanism's $O(m \ln^2 n)$ by a $\ln n$ factor. However, while it has the correct $\Theta(\ln n)$ asymptotic max loss guarantee, the BLT mechanism attains better empirical performance due to the additive approximation guarantee.

2.8 Other Loss Metrics*

In this section, we mostly focussed on max-loss; however, in many machine learning applications of prefix sums, another common loss metric which is considered is the root mean squared loss: the root-mean-squared-error (RMSE) of $\boldsymbol{B}$, scaled by the sensitivity of $\boldsymbol{C}$.

Definition 2.21. The **unnormalized root mean squared loss** of a mechanism $\mathcal{M}(\boldsymbol{G})$ estimating the (weighted) prefix sum $\boldsymbol{AG}$ of an input $\boldsymbol{G} \in \mathbb{R}^{n \times m}$ is defined as

$$\mathcal{L}_2(\mathcal{M}) \coloneqq \sqrt{\mathbb{E}_{\mathcal{M}} \left[\frac{1}{n} \sum_{t=0}^{n-1} \|(\boldsymbol{AG} - \mathcal{M}(\boldsymbol{G}))[t,:]\|_2^2 \right]}.$$

This is essentially a scaled version of the ℓ_2 error of the prefix sums, while the max loss measures the ℓ_∞ norm of the error. The scaling factor above is chosen to ensure that the RMS-loss and max loss are of

the same scale, as we see in the upcoming Lemma 2.24. Before that, we start with a closed form expression for the RMS-loss.

Theorem 2.22. Consider a correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{G}+\boldsymbol{Z})$ for $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0,\nu^2)$ with $\nu = \mathsf{sens}(\boldsymbol{C})\cdot\sigma = 2\,\|\boldsymbol{C}\|_{\mathsf{col}}\,\sigma$. Then, the unnormalized RMS-loss satisfies

$$\mathcal{L}_2(\boldsymbol{B},\boldsymbol{C}) = 2\sqrt{\frac{m}{n}}\,\sigma\ \ \|\boldsymbol{B}\|_{\mathrm{F}}\,\|\boldsymbol{C}\|_{\mathsf{col}}\ .$$

Proof Sketch. As in the proof of Theorem 2.2, we have $\boldsymbol{A}\boldsymbol{G} - \mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \nu^2\boldsymbol{B}\boldsymbol{B}^\top)$. Therefore, we get,

$$\mathbb{E}_{\mathcal{M}}\|(\mathcal{M}(\boldsymbol{G})[t,:] - \boldsymbol{A}[t,:]\boldsymbol{G})\|_2^2 = \mathbb{E}_{\boldsymbol{Z}}\|(\boldsymbol{B}[t,:]\,\boldsymbol{Z})\|_2^2 = m\nu^2\left(\sum_{\tau=0}^{t}\boldsymbol{B}[t,\tau]^2\right)$$

because $\boldsymbol{Z} \sim \mathcal{N}(0,\nu^2)^{n\times m}$. Therefore, from the linearity of expectation,

$$\begin{aligned} n\cdot\mathcal{L}_2(\boldsymbol{B},\boldsymbol{C})^2 &= \mathbb{E}_{\mathcal{M}}\sum_{t=0}^{n-1}\|(\boldsymbol{A}\boldsymbol{G} - \mathcal{M}(\boldsymbol{G}))[t,:]\|_2^2 \\ &= \sum_{t=0}^{n-1}\mathbb{E}_{\mathcal{M}}\|(\boldsymbol{A}\boldsymbol{G} - \mathcal{M}(\boldsymbol{G}))[t,:]\|_2^2 \\ &= m\nu^2\sum_{t=0}^{n-1}\sum_{\tau=0}^{t}\boldsymbol{B}[t,\tau]^2 \\ &= m\nu^2\,\|\boldsymbol{B}\|_{\mathrm{F}}^2\ , \end{aligned}$$

where the last equality follows from $\boldsymbol{B}$ being lower triangular. The result follows be re-arranging the above equation and plugging in $\nu = 2\,\|\boldsymbol{C}\|_{\mathsf{col}}\,\sigma$. □

Analogous to the normalized max loss in Eq. (2.4), we also define (normalized) root mean-squared loss:

Definition 2.23. Consider a correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) =$

$\boldsymbol{B}(\boldsymbol{C}\boldsymbol{G}+\boldsymbol{Z})$ for $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0,\nu^2)$. Then, we define RMS-loss by

$$\bar{\mathcal{L}}_2(\boldsymbol{B},\boldsymbol{C}) \coloneqq \frac{1}{\sqrt{n}} \|\boldsymbol{B}\|_{\mathrm{F}} \|\boldsymbol{C}\|_{\mathsf{col}} \,. \tag{2.19}$$

The normalized RMS-loss is always upper bounded by the normalized max loss:

Lemma 2.24. For any matrices $\boldsymbol{B},\boldsymbol{C} \in \mathbb{R}^{n\times n}$, we have $\bar{\mathcal{L}}_2(\boldsymbol{B},\boldsymbol{C}) \le \bar{\mathcal{L}}_\infty(\boldsymbol{B},\boldsymbol{C})$.

Proof. Since each row norm is at most the largest row norm, we have,

$$\|\boldsymbol{B}\|_{\mathrm{F}}^2 = \sum_{t\in[n]} \|\boldsymbol{B}[t,:]\|_2^2 \le n \max_{t\in[n]} \|\boldsymbol{B}[t,:]\|_2^2 = n \|\boldsymbol{B}\|_{\mathsf{row}}^2 \,.$$

Thus, we have that

$$\bar{\mathcal{L}}_2(\boldsymbol{B},\boldsymbol{C}) = \frac{1}{\sqrt{n}} \|\boldsymbol{B}\|_{\mathrm{F}} \|\boldsymbol{C}\|_{\mathsf{col}} \le \|\boldsymbol{B}\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\mathsf{col}} = \bar{\mathcal{L}}_\infty(\boldsymbol{B},\boldsymbol{C})$$

as required. □

2.9 An Approximation Theory Viewpoint*

The properties of an (infinite) Toeplitz matrix with first column given by $\boldsymbol{c} = (c_0, c_1, \ldots)$ can be understood through its **ordinary generating function**, which is the formal power series:[13]

$$f_{\boldsymbol{c}}(x) \coloneqq \sum_{t=0}^{\infty} c_t x^t \,.$$

For instance, the generating function of the all-ones worklad matrix $\boldsymbol{A}_{\mathsf{pre}}$ is

$$f_{\mathsf{pre}}(x) = \sum_{t=0}^{\infty} x^t = \frac{1}{1-x} \,.$$

The Toeplitz coefficients $\boldsymbol{c} = (c_0, c_1, \ldots)$ can be reconstructed from the Taylor expansion of the generating function around $x = 0$:

$$f_{\boldsymbol{c}}(x) = \sum_{t=0}^{\infty} \frac{f_{\boldsymbol{c}}^{(t)}(0)}{t!} x^t = \sum_{t=0}^{\infty} c_t x^t \quad \iff \quad c_t = \frac{f_{\boldsymbol{c}}^{(t)}(0)}{t!} \,,$$

[13] The term "formal" here refers to the fact that we regard x as a placeholder symbol rather than a number, meaning that we disregard issues of convergence.

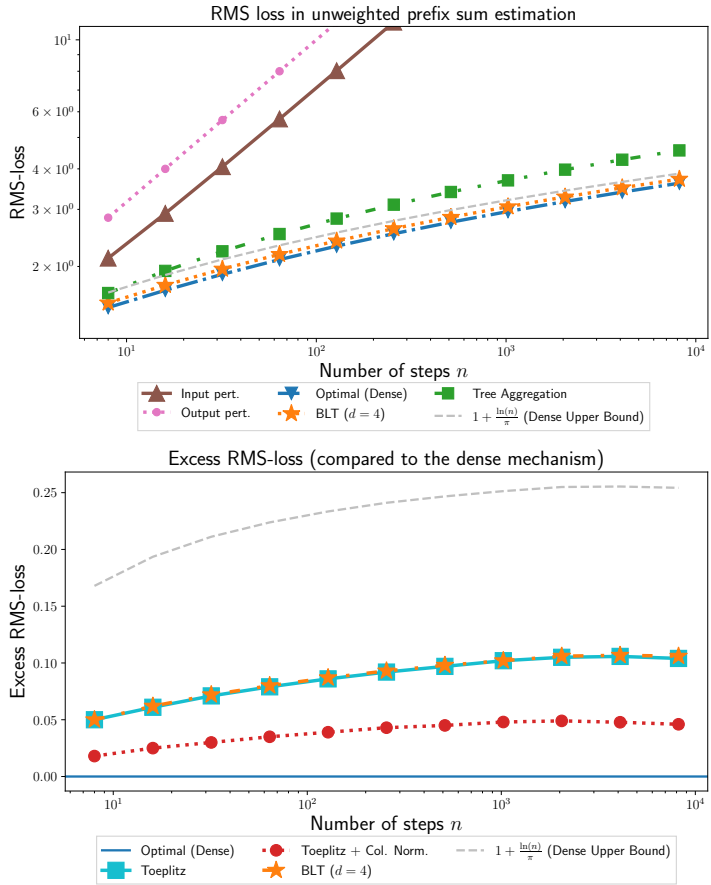


Figure 2.4: The RMS-loss counterpart to Fig. 2.2. We empirically compare various mechanisms in the streaming unweighted prefix sum estimation in terms of their RMS-loss. The upper bound of $1 + \ln(n)/\pi$ on the max loss is also a valid upper bound on the RMSE by Lemma 2.24. See Section 2.11 for the exact numerical values plotted here.

where $f_{\boldsymbol{c}}^{(t)}$ denotes the t^{th} derivative of the function $f_{\boldsymbol{c}}$.

The key relationship governing this connection is that the product of two Toeplitz matrices is equivalent to the product of the generating functions of their respective coefficients:

Lemma 2.25. Consider real-valued sequences $\boldsymbol{a} = (a_t)_{t=0}^{\infty}, \boldsymbol{b} = (b_t)_{t=0}^{\infty}, \boldsymbol{c} = (c_t)_{t=0}^{\infty}$. The following properties are equivalent:

(i) The sequence $\boldsymbol{a}$ is the convolution of $\boldsymbol{b}$ and $\boldsymbol{c}$, i.e. $a_t = \sum_{\tau=0}^{t} b_t c_{t-\tau}$;

(ii) For any size $n > 0$, the $n \times n$ lower triangular Toeplitz matrices $\boldsymbol{M_a}, \boldsymbol{M_b}, \boldsymbol{M_c}$ with respective first columns given by sequences $(a_t)_{t=0}^{n-1}, (b_t)_{t=0}^{n-1}, (c_t)_{t=0}^{n-1}$ satisfy $\boldsymbol{M_a} = \boldsymbol{M_b M_c}$;

(iii) Their respective generating functions $f_{\boldsymbol{a}}, f_{\boldsymbol{b}}, f_{\boldsymbol{c}}$ satisfy $f_{\boldsymbol{a}}(x) = f_{\boldsymbol{b}}(x) f_{\boldsymbol{c}}(x)$.

Thus, the factorization $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{BC}$ underlying a correlated noise mechanism can be understood by looking at the implied factorization of the generating function $f_{\mathsf{pre}} = 1/(1-x)$ of the workload matrix $\boldsymbol{A}_{\mathsf{pre}}$.

The optimal Toeplitz factorization of Section 2.3 with $\boldsymbol{B}_{\mathsf{Toep}} = \boldsymbol{C}_{\mathsf{Toep}} = \boldsymbol{A}_{\mathsf{pre}}^{1/2}$ corresponds to the factorization

$$f_{\mathsf{pre}}(x) = \frac{1}{1-x} = \frac{1}{\sqrt{1-x}} \cdot \frac{1}{\sqrt{1-x}} = f_{\boldsymbol{b}}(x)\, f_{\boldsymbol{c}}(x)\,. \tag{2.20}$$

Indeed, the coefficients $c_t^\star$ of Theorem 2.5 are the Taylor coefficients of

$$f_{\boldsymbol{c}^\star}(x) = \frac{1}{\sqrt{1-x}} = \sum_{t=0}^{\infty} \binom{-1/2}{t} (-x)^t = \sum_{t=0}^{\infty} c_t^\star x^t\,.$$

Similarly, the first column of $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$ can be found as the Taylor coefficients of

$$\frac{1}{f_{\boldsymbol{c}^\star}(x)} = \sqrt{1-x} = \sum_{t=0}^{\infty} (-1)^t \binom{1/2}{t} x^t\,,$$

yielding Eq. (2.11).

Generating Function Approximations and Utility Bounds Given any generating function $r(x)$, we can obtain a factorization

$$f_{\boldsymbol{b}}(x) = \frac{r(x)}{1-x}, \quad \text{and} \quad f_{\boldsymbol{c}}(x) = \frac{1}{r(x)}\,,$$

such that the $n \times n$ Toeplitz matrix $\boldsymbol{C}^{-1}$ is made up of the first n Taylor coefficients of $r(x)$ and $\boldsymbol{B} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}$ for any size $n > 0$. The optimal factorization Eq. (2.20) corresponds to $r(x) = \sqrt{1-x}$. Thus, we can expect that $r(x) \approx \sqrt{1-x}$ will lead to a good factorization with tighter approximations leading to better factorizations in terms of the max loss:

Theorem 2.26. Fix a size $n \in \mathbb{N}$ and consider a complex-valued function $r : \{x \in \mathbb{C} : |x| < 1\} \to \mathbb{C}$ defined on the open unit disc in the complex plane. Define the $n \times n$ Toeplitz matrices $\boldsymbol{B}_r$ and $\boldsymbol{C}_r$ whose first columns are given by the first n Taylor coefficients of the generating functions $f_{\boldsymbol{b}}(x) = r(x)/(1-x)$ and $f_{\boldsymbol{c}}(x) = 1/r(x)$ respectively. Then, we have:

$$\bar{\mathcal{L}}_\infty(\boldsymbol{B}_r, \boldsymbol{C}_r) \leq \bar{\mathcal{L}}_\infty(\boldsymbol{A}_{\mathsf{pre}}^{1/2}, \boldsymbol{A}_{\mathsf{pre}}^{1/2}) + O\big(n \cdot \mathsf{err}(r)\big)\,.$$

where $\mathsf{err}(r)$ is the approximation error of $r(x) \approx \sqrt{1-x}$:

$$\mathsf{err}(r) \coloneqq \max_{x \in \mathbb{C}\,:\,|x| = 1-n^{-1}} \left| r(x) - \sqrt{1-x} \right|,$$

and is assumed to satisfy $\mathsf{err}(r) < 1$. Here, the notation $|x|$ denotes the *absolute value* or *modulus* of the complex number $x \in \mathbb{C}$.

Examples The baselines of input perturbation ($r(x) = 1$ for all x) and output perturbation ($r(x) = 1 - x$) are clearly poor approximations to $\sqrt{1-x}$. Indeed, we showed $\mathsf{err}(r) = \Theta(\sqrt{n})$ in this case in Section 1 and Section 2.1.3. (Note that Theorem 2.26 only gives an upper bound.)

The b-banded Toeplitz mechanism with coefficients $c_0, \ldots, c_{b-1}$ corresponds to the (inverse) polynomial of degree $b-1$:

$$f_{\boldsymbol{c}}(x) = \frac{1}{r(x)} = \sum_{t=0}^{b-1} c_t x^t\,.$$

Then, $\mathsf{err}(r)$ is related to how well a degree-b polynomial can approximate the function $1/\sqrt{1-x}$ at $|x| = 1 - n^{-1}$. Unfortunately, polynomial approximations can be quite bad, especially around the poles[14] of the

[14] For a rational function in reduced form, the *poles* are the values of the function where the denominator is equal to zero.

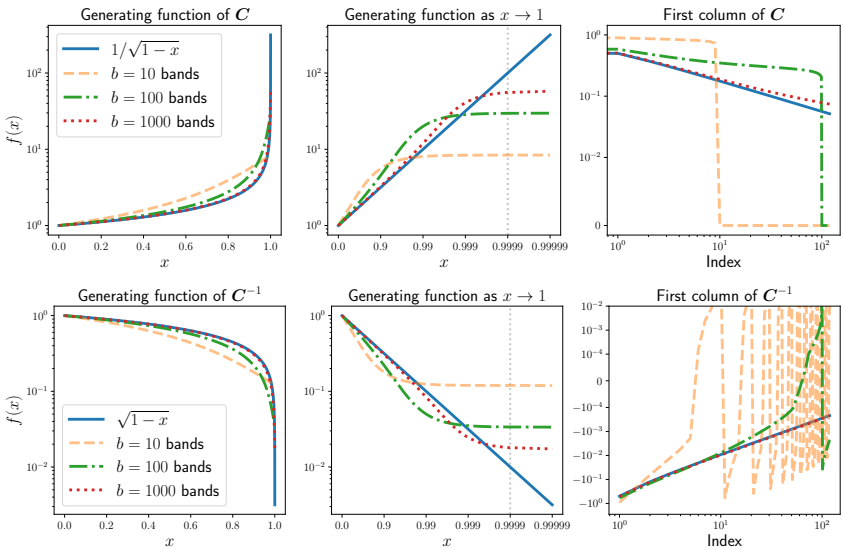


Figure 2.5: An illustration of the polynomial generating function approximation (corresponding to $\boldsymbol{C}$) provided by the banded Toeplitz mechanism compared to the optimal generating function of $1/\sqrt{1-x}$ of the max-loss-optimal Toeplitz mechanism (**top**) and the correspond approximation provided by $\boldsymbol{C}^{-1}$ to $\sqrt{1-x}$ (**bottom**). The generating function corresponding to the b-banded $\boldsymbol{C}$ matrix is a degree-$(b-1)$ polynomial: $f(x) = \sum_{t=0}^{b-1} c_t x^t$. Here, the Toeplitz coefficients c_t are optimized to minimize Eq. (2.12) (using the techniques of Section 4) for $n = 10^4$. Notice the approximation quality of the generating function as $x \to 1$ (middle plots) and the Toeplitz coefficients (right plots), are not as tight as the BLT mechanism, shown in Fig. 2.6. The dotted line in the middle plot shows $x = 1 - n^{-1}$; Theorem 2.26 shows that the approximation error in the generating function at $|x| = 1 - n^{-1}$ (in the complex plane) determines the max loss.

function being approximated; $1/\sqrt{1-x}$ has a pole at 1. Indeed, it turns out that approximating $1/\sqrt{1-x}$ with a degree-b polynomial up to error δ requires b to be larger than $1/\mathrm{poly}(\delta)$. See Fig. 2.5 for an illustration.

We now turn to the BLT mechanism. The generating function of $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is given by

$$r(x) = 1 + \sum_{i=1}^{d} \alpha_i x + \sum_{i=1}^{d} \alpha_i \lambda_i x^2 + \cdots = 1 + \sum_{i=1}^{d} \frac{\alpha_i x}{1 - \lambda_i x}\,. \tag{2.21}$$

This is a rational function of the form

$$r(x) = \frac{p(x)}{q(x)}, \quad \text{where} \quad p(x) \coloneqq \sum_{i=0}^{d} p_i x^i \quad \text{and} \quad q(x) \coloneqq \sum_{i=0}^{d} q_i x^i$$

with $p(0) = q(0) \neq 0$. Rational approximations generally give much tighter approximations than polynomials. For instance, rational Padé approximations are known to be much tighter than polynomial Taylor approximations. In particular, there is a rational function $r(x)$ of degree d such that

$$\sup_{x \in [0,1]} |r(x) - \sqrt{1-x}| \leq O\big(\exp(-\sqrt{d})\big)\,. \tag{2.22}$$

The error vanishes exponentially in the degree d, and is significantly better than polynomial approximations. This is illustrated in Fig. 2.6.

While this result holds only on the real line, there exists a degree-d rational function r over the complex plane with $\mathsf{err}(r) = O\big(\exp(-\sqrt{d})\big)$. This explains the competitive utility bound of the BLT mechanism (Theorem 2.18). In particular, it suffices to take a degree of $d = O(\ln^2(n/\delta))$ so that $\mathsf{err}(r) = O(\delta/n)$, leading to an additive error of δ.

Generating Functions and Efficient Implementation A broad class of Toeplitz coefficients that admit efficient noise generation algorithms correspond to the class of constant-recurrent sequences:

Definition 2.27. A sequence $(c_t)_{t=0}^{\infty}$ is called a constant-recurrent sequence of order d if there exist numbers $q_1, \ldots, q_d \in \mathbb{R}$ such that

$$c_t + \sum_{i=1}^{d} q_i c_{t-i} = 0$$

for all $t \geq d$.

This broad class includes many special cases such as arithmetic and geometric progressions, the Fibonacci series, and many others. More relevant to correlated noise mechanisms, banded Toeplitz coefficients (and eventually periodic sequences in general), as well as BLT coefficients are special cases. This follows from inspecting their generating function, based on the following equivalent representations:

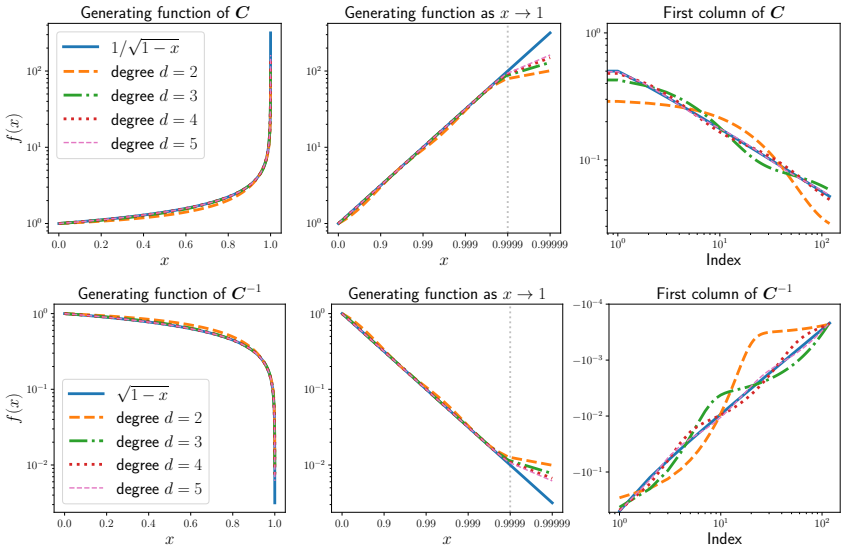


Figure 2.6: An illustration of the rational generating function approximation provided by the BLT mechanism compared to the optimal generating function of $1/\sqrt{1-x}$ of the max-loss-optimal Toeplitz mechanism (**top**) and the corresponding approximation its inverse provides to $\sqrt{1-x}$ (**bottom**). We plot the degree-d BLT approximation obtained by solving Problem 2.17 for $n = 10^4$ (using the techniques of Section 4). We get a tight approximation for x less than $\approx 1 - n^{-1}$ (shown by the dotted line in the middle plot), with tighter approximation for larger degree d. Notice that the approximation quality of the generating function as $x \to 1$, and of the optimal Toeplitz coefficients (rightmost plots) is significantly tighter than that the banded $\boldsymbol{C}$ from Fig. 2.5.

Theorem 2.28. The following properties are equivalent:

(a) $(c_t)_{t=0}^{\infty}$ is a constant-recurrent sequence of order-d.

(b) The generating function $f_c(x) = \sum_{t=0}^{\infty}$ of $(c_t)_{t=0}^{\infty}$ is a rational function $f_c(x) = p(x)/q(x)$, where $q(0) = 1$, $\mathsf{deg}(q) \leq d$ and $\mathsf{deg}(p) < d$.

(c) There exists a matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times d}$ and vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ such that $c_t = \boldsymbol{u}^\top \boldsymbol{\Lambda}^t \boldsymbol{v}$ for all $t \geq 0$.

For the BLT mechanism, we have that $c_0, c_1, c_2, \ldots$ forms a constant-recurrent sequence (i.e., omitting the first term $c_0 = 1$). As a function

of the BLT parameters $\boldsymbol{\alpha}, \boldsymbol{\lambda}$, we have $\boldsymbol{\Lambda} = \mathsf{diag}(\boldsymbol{\lambda})$, $\boldsymbol{v} = \boldsymbol{\alpha}$ and $\boldsymbol{u} = (1, \ldots, 1)$ is the vector of ones. While the sequence $c_0, c_1, \ldots$ (including the first term c_0) can be written as a constant-recurrent sequences of order-$(d+1)$, it is computationally advantageous to treat it separately.

A key advantage of constant-recurrent coefficients is that they are amenable to efficient recursive noise generation. Given a constant-recurrent sequence $c_t = \boldsymbol{u}^\top \boldsymbol{\Lambda}^t \boldsymbol{v}$ (in the matrix-power representation of Theorem 2.28(c)) that makes up the first column of the matrix $\boldsymbol{C}$, we can compute $\widetilde{\boldsymbol{Z}} = \boldsymbol{C}^{-1}\boldsymbol{Z}$ with a small modification of Algorithm 2.2. In particular, we replace Lines 3 and 4 with

$$\widetilde{\boldsymbol{z}}_t = \boldsymbol{z}_t - \boldsymbol{M}_t \boldsymbol{v}\,, \quad \text{and} \quad \boldsymbol{M}_{t+1} = \boldsymbol{M}_t \boldsymbol{\Lambda} + \widetilde{\boldsymbol{z}}_t \boldsymbol{u}^\top\,,$$

which requires $O(md + d^2)$ time and space. This is worse than Algorithm 2.2 by an additive factor of d^2 on both counts. When the matrix $\boldsymbol{\Lambda}$ is diagonalizable, this recurrence can effectively be reduced to Algorithm 2.2. For example, if $\boldsymbol{\Lambda}$ has all real and unique eigenvalues, we can diagonalize it as $\boldsymbol{\Lambda} = \boldsymbol{V}\mathsf{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$. Then, we have

$$c_t = \boldsymbol{u}^\top \boldsymbol{\Lambda}^t \boldsymbol{v} = \widetilde{\boldsymbol{u}}^\top \mathsf{diag}(\boldsymbol{\lambda})^t\, \widetilde{\boldsymbol{v}}\,,$$

where $\widetilde{\boldsymbol{u}} = \boldsymbol{V}^\top \boldsymbol{u}$ and $\widetilde{\boldsymbol{v}} = \boldsymbol{V}^{-1}\boldsymbol{v}$.

Importantly, the optimal coefficients $(c_t^\star)_{t=0}^\infty$ of the max-loss-optimal Toeplitz mechanism defined in Theorem 2.5 do not form a constant-recurrent sequence. Indeed, their generating function of $1/\sqrt{1-x}$ is not a rational function. Thus, they are not amenable to the efficient recursive noise generation procedure outlined above.

2.10 Closed Form Expressions for the BLT Mechanism*

Here, we give closed form expressions for the sensitivity and error for the BLT mechanism (Section 2.5). These expressions are useful in numerically optimizing the parameters of the mechanism to minimize the max loss; see Section 4.3.2.

Lemma 2.29. Suppose we are given parameters $\boldsymbol{\alpha}, \tilde{\boldsymbol{\alpha}} \in \mathbb{R}_+^d$ and $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in [0,1)^d$ such that $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ and $\boldsymbol{C}^{-1} = \mathsf{BLT}(-\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\lambda}})$ are inverses of each other, assuming they exist (see Lemma 2.16

for precise conditions). Further, for $\lambda \in [0, 1)$ define the n-term geometric sum

$$\gamma_n(\lambda) := \sum_{t=0}^{n-1} \lambda^t = \frac{1 - \lambda^n}{1 - \lambda}\,.$$

Then, we have the following:

(a) The single epoch sensitivity is given by

$$\|\boldsymbol{C}\|_{\mathsf{col}}^2 = 1 + \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \alpha_j \gamma_{n-1}(\lambda_i \lambda_j)\,.$$

(b) For $n \geq 2$, the maximum row norm of $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1}$ is

$$\|\boldsymbol{B}\|_{\mathsf{row}}^2 = n + 2\sum_{i=1}^{d} \tilde{\alpha}_i \Gamma_i^{(n)} + \sum_{i=1}^{d} \sum_{j=1}^{d} \tilde{\alpha}_i \tilde{\alpha}_j \Gamma_{i,j}^{(n)}\,,$$

where we define

$$\Gamma_i^{(n)} = \frac{1}{1 - \tilde{\lambda}_i}\big(n - \gamma_n(\tilde{\lambda}_i)\big)\,,$$
$$\Gamma_{i,j}^{(n)} = \frac{1}{(1 - \tilde{\lambda}_i)(1 - \tilde{\lambda}_j)}\Big(n - \gamma_n(\tilde{\lambda}_i) - \gamma_n(\tilde{\lambda}_j) + \gamma_n(\tilde{\lambda}_i \tilde{\lambda}_j)\Big)\,.$$

(c) The Frobenius norm of $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1}$ is given by

$$\frac{1}{n}\|\boldsymbol{B}\|_{\mathrm{F}}^2 = \frac{n+1}{2} + \frac{2}{n}\sum_{i=1}^{d} \tilde{\alpha}_i \hat{\Gamma}_i^{(n)} + \frac{1}{n}\sum_{i=1}^{d} \tilde{\alpha}_i \tilde{\alpha}_j \hat{\Gamma}_{i,j}^{(n)}\,,$$

where we define

$$S_i^{(n)} = 1 - \tilde{\lambda}_i + \frac{\tilde{\lambda}_i}{1-\tilde{\lambda}_i}\left(n - \gamma_n(\tilde{\lambda}_i)\right),$$

$$S_{i,j}^{(n)} = 1 - \tilde{\lambda}_i\tilde{\lambda}_j + \frac{\tilde{\lambda}_i\tilde{\lambda}_j}{1-\tilde{\lambda}_i\tilde{\lambda}_j}\left(n - \gamma_n(\tilde{\lambda}_i\tilde{\lambda}_j)\right),$$

$$\hat{\Gamma}_i^{(n)} = 1 + \frac{1}{1-\tilde{\lambda}_i}\left(\frac{n(n-1)}{2} - S_i^{(n)}\right),$$

$$\hat{\Gamma}_{i,j}^{(n)} = 1 + \frac{1}{(1-\tilde{\lambda}_i)(1-\tilde{\lambda}_j)}\left(\frac{n(n-1)}{2} - S_i^{(n)} - S_j^{(n)} + S_{i,j}^{(n)}\right).$$

Further, each of these quantities can be computed in $O(d^2)$ floating point operations.

2.11 Numerical Comparison in the Streaming Setting*

Tables 2.2 and 2.3 give the raw numbers used to plot Figs. 2.2 and 2.4 respectively. Each column corresponds to a different factorization, with "Identity" and "Workload" refers to the Input Perturbation and Output Perturbation mechanism introduced in Section 2.1.3. Streaming H2 and Full H2 refer to different variants of the tree aggregation approach Section 2.7. BLT was presented in Section 4.3.2. Toeplitz and Col-Norm. Toep were presented in Section 2.3. Dense was presented in Section 2.2.

Table 2.2: Max Error for different matrix factorizations.

n	Identity	Workload	Streaming H2	Full H2	BLT	Toeplitz	Col-Norm. Toep.	Dense
8	2.828	2.828	NaN	2.382	1.723	1.718	1.573	1.51
16	4.0	4.0	NaN	2.881	1.944	1.944	1.783	1.704
32	5.657	5.657	NaN	3.381	2.168	2.167	1.997	1.905
64	8.0	8.0	NaN	3.883	2.391	2.389	2.212	2.111
128	11.314	11.314	NaN	4.384	2.61	2.61	2.428	2.32
256	16.0	16.0	NaN	4.886	2.832	2.831	2.645	2.532
512	22.627	22.627	NaN	5.387	3.054	3.052	2.863	2.746
1024	32.0	32.0	NaN	5.888	3.273	3.273	3.081	2.958
2048	45.255	45.255	NaN	6.389	3.494	3.493	3.299	3.177
4096	64.0	64.0	NaN	6.89	3.716	3.714	3.518	-
8192	90.51	90.51	NaN	7.391	3.939	3.935	3.737	-

Table 2.3: RMSE for different matrix factorizations.

n	Identity	Workload	Streaming H2	Full H2	BLT	Toeplitz	Col-Norm. Toep.	Dense
8	2.121	2.828	2.178	1.656	1.544	1.544	1.512	1.494
16	2.915	4.0	2.663	1.938	1.751	1.75	1.714	1.689
32	4.062	5.657	3.156	2.227	1.964	1.963	1.922	1.892
64	5.701	8.0	3.652	2.518	2.18	2.179	2.135	2.1
128	8.031	11.314	4.151	2.81	2.398	2.397	2.35	2.311
256	11.336	16.0	4.65	3.102	2.617	2.616	2.567	2.524
512	16.016	22.627	5.15	3.394	2.837	2.836	2.784	2.739
1024	22.638	32.0	5.65	3.686	3.057	3.057	3.003	2.955
2048	32.008	45.255	6.15	3.978	3.278	3.277	3.221	3.172
4096	45.26	64.0	6.65	4.269	3.499	3.498	3.44	-
8192	64.004	90.51	7.15	4.56	3.72	3.718	3.66	-

2.12 Bibliographic Notes

Bounds on the max loss $\bar{\mathcal{L}}_\infty(\boldsymbol{B}, \boldsymbol{C})$ for $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{BC}$ have been studied extensively, starting with the work of Kwapień and Pełczyński [1970]. Since then many results have improved the leading constants. Theorem 2.2 has been known in the literature from, for example, Nikolov, Talwar, and Zhang [2016], with an explicit construction appearing in Edmonds, Nikolov, and Ullman [2020].

Dense Mechanism The bounds of Theorem 2.4 use the property that the optimization problem (2.6) defines a norm $\gamma_2(\cdot)$ of a matrix:

$$\gamma_2(\boldsymbol{M}) \coloneqq \min \left\{ \|\boldsymbol{B}\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\mathsf{col}} \,:\, \boldsymbol{BC} = \boldsymbol{M} \right\} .$$

The fact that this is a norm was first proved in an unpublished manuscript of Haagerup [1980], who showed its equivalence to the Hadamard norm. (This was subsequently also shown in many works.)

The upper bound in Theorem 2.4 was proved by Mathias [1993, Corollary 3.5], with the bound

$$\gamma_2(\boldsymbol{A}_{\mathsf{pre}}) \leq \frac{1}{2} + \frac{1}{2n} \sum_{t=1}^{n} \left| \csc\left(\frac{(2t-1)\pi}{2n} \right) \right| ,$$

where $\boldsymbol{A}_{\mathsf{pre}}$ is the lower triangular matrix of all ones (cf. Eq. (1.6)) and $\csc(\cdot)$ is the trigonometric cosecant function. This is at most[15]

[15]This can be shown by upper bounding the sum with an integral [cf. Mathias, 1993, Sec. 3].

$1 + \ln(n)/\pi$. The proof of Mathias [1993] is non-constructive and uses the triangle inequality to bound $\gamma_2(\boldsymbol{A}_{\mathsf{pre}})$ with the γ_2-norms of two other matrices, which are themselves bounded by other means. Recently, Henzinger and Upadhyay [2025] gave a constructive factorization that matches the upper bound of Mathias [1993] and also has the property that the columns of $\boldsymbol{C}$ all have the same ℓ_2 norm. We note that Mathias [1993, Corollary 3.5] also gives a lower bound:

$$\gamma_2(\boldsymbol{A}_{\mathsf{pre}}) \geq \frac{1}{2n} + \frac{1}{2n}\sum_{t=1}^{n}\left|\csc\left(\frac{(2t-1)\pi}{2n}\right)\right|,$$

where the first term is $1/(2n)$ instead of $1/2$ in the upper bound. Theorem 2.4 instead gives a tighter lower bound established by Matoušek, Nikolov, and Talwar [2020] by using the fact that $|\csc(x)| \geq x^{-1}$:

$$\gamma_2(\boldsymbol{A}_{\mathsf{pre}}) \geq \frac{1}{2n}\sum_{t=1}^{n}\left|\csc\left(\frac{(2t-1)\pi}{2(2n+1)}\right)\right| \geq \frac{2n+1}{\pi n}\sum_{t=1}^{n}\frac{1}{2t-1} \geq \frac{\ln(2n+1)}{\pi}.$$

The dual characterization of the γ_2-norm is useful in constructing lower bounds. First shown by Haagerup [1980] and also appeared later in Haagerup and Pisier [1993] and Lee, Shraibman, and Špalek [2008, Theorem 9], we have:

$$\gamma_2(\boldsymbol{M}) = \max\left\{\|\mathsf{diag}(\boldsymbol{p})^{1/2}\ \boldsymbol{M}\ \mathsf{diag}(\boldsymbol{q})^{1/2}\|_* : \begin{matrix}\boldsymbol{p},\boldsymbol{q} \in \mathbb{R}^n_+ \text{ with} \\ \boldsymbol{p}^\top\mathbf{1} = 1 = \boldsymbol{q}^\top\mathbf{1}\end{matrix}\right\},$$

where $\|\boldsymbol{M}\|_*$ denotes the nuclear norm of the matrix $\boldsymbol{M}$. The lower bound from Theorem 2.4, established by Matoušek, Nikolov, and Talwar [2020], follows from taking $\boldsymbol{p} = \boldsymbol{q} = \mathbf{1}_n/n$ in the optimization problem defining the dual norm, and explicitly computing the eigenvalues of the matrix $\boldsymbol{M} = \boldsymbol{A}_{\mathsf{pre}}$. In fact, a bound on the trace norm $\|\cdot\|_*$ appearing in the dual norm was explicitly first computed by Elliott [1953] and subsequently improved by a series of works [Gregory and Karney, 1969, Hoffman, 1971, Strang, 2022].

Tree Aggregation Tree aggregation, known also as the binary tree mechanism, was proposed independently by Hay, Rastogi, Miklau, and Suciu [2010], Dwork, Naor, Pitassi, and Rothblum [2010], Chan, Shi, and

Song [2011]. These were among the first correlated noise mechanisms for differential privacy. A variant of the binary tree mechanism was also proposed by Smith, Thakurta, and Upadhyay [2017], although that does not fit into the the class of correlated noise mechanisms we study in this monograph (Eq. (2.1)). Tree aggregation is also known as the *hierarchical histogram* method, especially in the context of range queries, as in Example 1.27, see for example Hay, Rastogi, Miklau, and Suciu [2010], Qardaji, Yang, and Li [2013], Cormode, Kulkarni, and Srivastava [2019], Li, Hay, Miklau, and Wang [2014].

Follow up work made several improvements to the tree aggregation approach of Section 2.7. For instance, Honaker [2015] proposed to replace the noise correlation matrix $\boldsymbol{B}_{\mathsf{tree}}$ from Theorem 2.20 with the matrix[16] $\boldsymbol{B}^{\star}_{\mathsf{tree}} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{\dagger}_{\mathsf{tree}}$ for its better error properties; this comes at the cost of an increased $O(mn)$ memory for noise generation. This same construction appeared earlier in the range query literature Hay, Rastogi, Miklau, and Suciu [2010], Li, Miklau, Hay, McGregor, and Rastogi [2015] as well, where it was shown to be the best linear unbiased estimator for the workload query answers. As another example, Andersson and Pagh [2023] improve the error bound of Theorem 2.20 by a factor 2 by carefully discarding rows/columns from $\boldsymbol{B}_{\mathsf{tree}}/\boldsymbol{C}_{\mathsf{tree}}$ with large norm. This construction preserves the $m\log_2(n)$ time and space complexity of noise generation, but the max loss is still suboptimal by a factor of approximately $\pi/(2\ln 2) \approx 2.26$. Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024] propose a tree construction which can get a constant factor of $1 + o(1)$, i.e., arbitrarily close to optimal. Their procedure involves building a recursive tree-like factorization over a base BLT factorization.

The Max-Loss-Optimal Toeplitz Mechanism We can trace the factorization $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{A}^{1/2}_{\mathsf{pre}}\boldsymbol{A}^{1/2}_{\mathsf{pre}}$ of the prefix sum matrix (that we studied in Section 2.3) at least as far back as Bennett [1977]. Fichtenberger, Henzinger, and Upadhyay [2023] were the first to utilize this factorization for differentially private computation of streaming prefix sums. The optimality of this mechanism for the objective Eq. (2.9) among the

[16] Here, $\boldsymbol{M}^{\dagger}$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{M}$.

class of all lower triangular and Toeplitz factorizations was established by Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024]. (Previously, it was only known that it is optimal in the asymptotic $n \to \infty$ regime.)

Fichtenberger, Henzinger, and Upadhyay [2023] also first established the bound on the max loss of the max-loss-optimal Toeplitz mechanism, where the optimal $1/\pi$ factor comes from Chen and Qi [2005]. Later, Henzinger, Upadhyay, and Upadhyay [2023] showed an almost tight bound on the RMS-loss. In Theorem 2.7, we present the bound of Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024, Lemma 2.1]. Finally, column normalization is motivated by Lemma 2.9, which was observed by Yuan, Yang, Zhang, and Hao [2016].

Banded Toeplitz Mechanism Banded strategy matrices $\boldsymbol{C}$ were considered by Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a] in the machine learning context for their compatibility with amplification by sampling, and the banded Toeplitz matrices by Kalinin and Lampert [2024], McKenna [2024] to improve the factorization efficiency. We discuss amplification by sampling in Section 3 and the factorization considerations in Section 4. The error bound we present in Theorem 2.13 is due to Kalinin and Lampert [2024, Theorem 6].[17]

The BLT Mechanism The BLT mechanism was introduced by Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024], including the error bound Theorem 2.18 which corresponds to their Theorem 4.6. Their Lemma 5.2 parameterized a mechanism $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ and $\boldsymbol{C}^{-1} = \mathsf{BLT}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ via the pair $(\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}})$, allowing efficient noise generation from $\boldsymbol{C}^{-1} = \mathsf{BLT}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ following the approach of Eq. (2.18) (their Algorithm 1).

The BLT mechanism was extended from the streaming setting to the machine learning setting by McMahan, Xu, and Zhang [2024], including the algorithm for noise generation directly from $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ (our

[17] While they state bounds on the ℓ_2 error, Kalinin and Lampert [2024] actually upper bound it by the max loss and prove their bounds on the latter.

Algorithm 2.2 is a restatement of their Algorithm 3). Lemma 2.16 regarding the inverse BLT parameterization is due to McMahan and Pillutla [2025].

For more discussion on the fact that rational Padé approximations are known to be much tighter than polynomial Taylor approximations [e.g., Baker Jr and Gammel, 1961]. The approximation stated in Eq. (2.22) was given by Newman [1964].

The Approximation Theory Viewpoint The approximation theory viewpoint presented in Section 2.9 was utilized by Fichtenberger, Henzinger, and Upadhyay [2023] to develop the max-loss-optimal Toeplitz mechanism and by Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024] for the BLT mechanism. In particular, Theorem 2.26 is a simplified (and slightly looser) version of Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024, Proposition 4.1].

These developments are based on classical ideas from approximation theory; we highlight a few important connections. The classical Prony interpolation, developed by Gaspard Riche de Prony in 1795, is an approximate decomposition of sequence (or a function) into a sum of complex exponentials as in Eq. (2.15). Newman [1964] gave an approximation of $|x|$ using a rational function. This can be used to construct a rational approximation to $\sqrt{1-x}$, forming the basis for the proof of the error bound of the BLT mechanism (Theorem 2.18). Finally, Braess and Hackbusch [2005] show that it is possible to approximate $1/t \approx \sum_{i=1}^{d} \alpha_t \lambda_i^t$ (with α_i, λ_i real) up to an error $\exp(-\Omega(\sqrt{d}))$. The BLT approximation task is closely related—we wish to approximate the function $t^{-3/2}$, as exhibited by the coefficients (c'_t) of $\boldsymbol{C}_{\mathsf{Toep}}^{-1}$, as in Theorem 2.5.

Finally, for more details on Theorem 2.28 and the equivalent representations of constant-recurrent sequences, we refer to the excellent monograph by Corless, Ida, and Hong [2011].

3

Correlated Noise Mechanisms for Learning Problems

In this chapter, we build on the correlated noise mechanism introduced in the previous chapters to make them applicable to practical AI and machine learning settings. Recall from Section 1.2 that we wish to find model parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$ optimizing the objective

$$\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathsf{data}}} \left[\ell \left(\boldsymbol{\theta}, \boldsymbol{x} \right) \right], \tag{3.1}$$

where $\ell(\boldsymbol{\theta}, \boldsymbol{x})$ is the loss of making a prediction with model parameters $\boldsymbol{\theta}$ on a datapoint $\boldsymbol{x}$, and $\mathbb{P}_{\mathsf{data}}$ is an underlying data distribution.

Throughout, we give bounds on the *suboptimality* of a model $\boldsymbol{\theta}$, also known as the *excess population risk* of $\boldsymbol{\theta}$. For a mechanism $\mathcal{M}$ that outputs θ,

$$R(\mathcal{M}) := \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathsf{data}}} \left[\ell(\boldsymbol{\theta}, \boldsymbol{x}) \right] - \min_{\boldsymbol{\theta}^* \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathsf{data}}} \left[\ell(\boldsymbol{\theta}^*, \boldsymbol{x}) \right] . \tag{3.2}$$

In Sections 1 and 2, we mainly focused on the streaming setting where each data point is processed only once. We relax this assumption and treat the general multiple-participation setting in this section, allowing each data point to participate multiple times in training (e.g. by making multiple passes through the dataset).

The multiple-participation setting is also highly relevant in the context of user-level DP (as opposed to our default privacy unit of

example-level DP). Here, multiple participations of *any* of a user's data violates the streaming assumption, even if each datapoint is processed only once (see Section 1.6 for details).

The main difference between the streaming and multiple-participation settings is the sensitivity computation. As illustrated in Fig. 1.3, changing one data point (to an adjacent dataset) can change multiple gradients (even in the non-adaptive setting we use for analysis per Theorem 1.11), increasing the sensitivity of the operation. The key challenge in the multiple-participation setting turns out to give tight and efficient bounds on the sensitivity. We define data processing abstractions that map to practical scenarios but where the sensitivity is not too large, and give algorithms to efficiently compute this sensitivity.

Outline We start this section with the potential advantages that correlated noise mechanisms can offer in the learning setting in Section 3.1 (and hence motivating the setting of multiple participation). Section 3.2 gives examples of other first order-optimization algorithms and their reduction to (weighted) prefix sum estimation as in Eq. (1.7). Next, Section 3.3 takes a deep dive into the multiple-participation setting, including the challenges, data processing patterns for tight sensitivity calculations, and efficient computational algorithms.

Changing gears, Section 3.5 surveys learning-theoretic guarantees for correlated noise mechanisms, and how they can help over using independent noise. We end the section by discussing how the privacy guarantees of correlated noise mechanisms can be amplified by accounting for the random sampling of data points in Section 3.4. As in Section 2, detailed pointers to missing proofs can be found in the bibliographic notes of Section 3.7.

3.1 Motivation

Correlated noise mechanisms are generally never worse (in terms of downstream learning performance at any given privacy level) and often much better than using independent noise in learning settings. In this section, we highlight the factors behind this Pareto-dominance of correlated noise over independent noise.

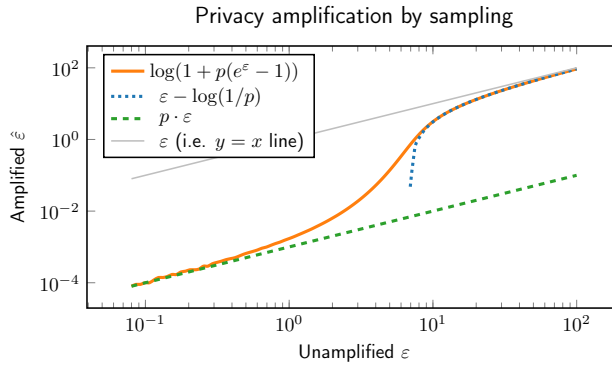


Figure 3.1: Plots for theoretical bounds on amplification of a generic (ε, δ)-DP with Poisson sampling (in orange), where a batch of data is formed by including each example i.i.d. with probability p, leading to $(\hat{\varepsilon}, p\delta)$-DP with $\hat{\varepsilon} = \log(1 + p(e^\varepsilon - 1))$. This plot shows $p = 10^{-3}$. Note the shape of this curve: as $\varepsilon \to 0$, then $\hat{\varepsilon} \approx p\,\varepsilon$. On the other hand, when $\varepsilon \to \infty$, then $\hat{\varepsilon} \approx \varepsilon - \log(1/p)$. In all cases, amplification leads to an improvement over the unamplified guarantee denoted by the solid gray line.

Privacy amplification by sampling In the centralized training setting, we can choose to form batches of data randomly. This typically has no adverse impact on the training process, and can even be helpful. However, the additional randomness from sampling can be beneficial for privacy. Intuitively, this added randomness will typically further increase the uncertainty of an adversary about whether a dataset D or another adjacent dataset D' was used in training. This phenomenon is also known as *privacy amplification by sampling*.

Privacy amplification guarantees for DP-SGD (i.e., the input perturbation baseline of $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$) are well understood when the data batches are chosen via *Poisson sampling*: in each iteration, each example is included in the batch independently with probability $p = B/N$, chosen so that the expected batch size is B. While the precise details of the amplification analysis are beyond the scope of this monograph, we note that existing libraries can compute exact privacy guarantees for DP-SGD with sampling using only the number of steps n, the sampling probability p, and the noise multiplier σ. Indeed, these amplification guarantees are central to achieving practical privacy-utility trade-offs for DP-SGD, as we discuss in Section 3.1. (We provide a deeper dive

into amplification in Section 3.4.)

Drawbacks of Independent Noise Mechanisms Prior to the introduction of correlated noise mechanisms, the *de facto* standard for differentially private deep learning was DP-SGD with independent noise (Algorithm 1.2). This algorithm adds independent Gaussian noise to each stochastic gradient update, and is an instantiation of the input perturbation baseline of Eq. (1.16). To achieve good practical performance (in terms of downstream learning performance) at moderate privacy budgets, DP-SGD often requires privacy amplification by sampling. Unfortunately, as we detail next, the data sampling assumptions used for theoretical analyses are often violated in practice.

Privacy amplification analyses typically assume that batches of data are constructed either by *Poisson sampling* or *fixed-size sampling with replacement*. These sampling methods can in some regimes give (amplified) privacy guarantees competitive with correlated noise mechanisms while remaining tractable to compute. However, machine learning pipelines typically use fixed-size sampling *without* replacement (usually implemented by shuffling the data).

In the academic literature, it has been a common practice to report privacy guarantees based on Poisson sampling even when the actual models are trained with fixed-size data batches sampled with replacement. This leads to a mismatch between the claimed privacy guarantee and the actual one satisfied by the trained model. Recent work has shown that there is a real gap in the level of DP offered by these two data processing patterns. That is, *the gap is not due to the lack of a tight analysis for shuffling, but a real difference in the level of privacy protection*. Therefore, this practice should not be considered acceptable, at least in real applications (see Section 3.7 for references).

Finally, and perhaps most importantly, privacy amplification is challenging or impossible when it is not feasible to obtain a uniformly random sample of data points (whether through Poisson sampling or otherwise) due to domain-specific constraints. We review two such examples.

Example 3.1 (Federated Learning)**.** Cross-device federated learning refers to a distributed learning setting where several data silos such as smartphones (referred to as "clients") collaborate to learn a single model while keeping their training data decentralized. The typical learning algorithm, federated averaging, samples a few *available* clients and constructs a stochastic gradient estimator based on their data. In industrial federated learning systems, the availability of a client varies diurnally, and is determined by external factors such as the device being idle, connected to an unmetered Wi-Fi network, and charging.

Thus, ensuring that clients are subsampled precisely and uniformly at random from a large population is complex and hard to verify (similar concerns apply to random shuffling). This makes amplification-by-sampling arguments, and thus, amplified DP-SGD (or its federated variant), infeasible. In this federated learning setting, correlated noise mechanisms can maintain provable privacy guarantees with utility comparable to or better than amplified DP-SGD, even with arbitrary changes in client availability.

Example 3.2 (Learning with Distribution Drift)**.** Distribution drift in learning problems arises from non-stationary data generating distributions. This drift can be independent of the learning process, as seen in continual learning where user behavior shifts over time. For instance, discussions on social networks could change with viral trends and current affairs, while health data collected at hospitals could change with emerging health challenges and evolving medical practices. Alternatively, drift can be driven by feedback mechanisms where agents strategically adapt their behavior to deployed decision systems for favorable outcomes. For instance, in loan application scenarios where AI models may be used to predict the likelihood of a default, applicants may manipulate model-relevant factors to improve their scores, causing data distribution drift. The model thus actively shapes the observed data rather than passively observing a static environment.

In both scenarios, models should adapt to data drift. This

> is best achieved by treating data as a stream and continuously learning from recent data while discarding outdated information. This approach necessitates respecting the data's temporal ordering, precluding methods like random batch sampling or using randomly shuffled datasets, which disregard this crucial temporal structure.

Advantages of Correlated Noise Mechanisms In both these examples, amplification by sampling/shuffling is not practically feasible. In the absence of amplification arguments, DP-SGD with independent noise suffers from a huge drop in downstream task utility (e.g. accuracy for classification problems) at small or moderate privacy budgets.[1] In such instances, DP-SGD with correlated noise (Algorithm 1.3), known also as "DP-FTRL" (see also Section 1.10.2), has been found to achieve competitive performance *without* the need for privacy amplification.

In particular, correlated noise DP-SGD *without* amplification outperforms independent noise DP-SGD *with* amplification for moderate to large privacy budgets. We refer to Fig. 3.2 for an example. This method guarantees strict privacy regardless of sampling constraints, even in scenarios where random sampling is limited or infeasible, while delivering good downstream task performance. In cases where sampling is viable, we can provide amplified privacy guarantees for correlated noise mechanisms (Section 3.4), offering the best of both worlds. Indeed, correlated noise *with* amplification uniformly outperforms amplified independent noise DP-SGD for all privacy budgets.

3.2 Learning Problems as Weighted Prefix Sums

Recall the setting of stochastic gradient descent (SGD) from Section 1.2: in step t of a learning algorithm, we receive a batch (per-sample clipped) gradient $\boldsymbol{g}_t \in \mathbb{R}^m$ calculated under the current model parameters $\boldsymbol{\theta}_t$. The SGD updates is then computed from the prefix sums of the gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$; which is obtained as linear map $\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{G}$

[1] For example, privacy amplification by sampling improves the classification accuracy in the CIFAR-10 setting of Fig. 3.2 by around 10 points for $\varepsilon \approx 7$ in Kairouz, McMahan, Song, Thakkar, Thakurta, and Xu [2021a, Fig. 1b].

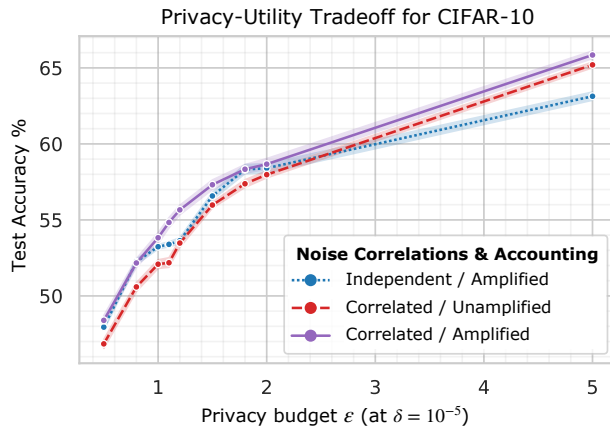


Figure 3.2: Privacy-utility tradeoff of CIFAR-10: This plot shows the classification accuracy of DP-SGD with independent noise vs. correlated noise calibrated to achieve $(\varepsilon, 10^{-5})$-DP with or without privacy amplification by sampling at different privacy budgets ε. We make two key observations. First, correlated noise *without* amplification outperforms independent noise *with* amplification at $\varepsilon \geq 3$. Second, correlated noise *with* amplification uniformly outperforms independent noise at all ε. Figure reproduced from Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a, Fig. 1a] with permission; we refer the reader there for details.
Caveat: While these accuracies are not state-of-the-art for the CIFAR-10 benchmark, they are representative of the general performance gain that can be expected from correlated noise. Indeed, these numbers are reported on a small convolutional neural network with 6 conv layers, one dense layer and around half a million parameters, trained with a batch size of $B = 500$. Significantly higher accuracies can be obtained on the same CIFAR-10 dataset under the same privacy budgets with larger batch sizes (e.g. $B = 65536$), larger models (e.g. a 40-layer Wide-ResNet with 9 million parameters), and data augmentation [De, Berrada, Hayes, Smith, and Balle, 2022].

with the workload matrix

$$\boldsymbol{A}_{\mathsf{pre}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \in \{0,1\}^{n \times n} . \tag{3.3}$$

More generally, SGD with momentum and weight decay can be described by a linear map as well. This update is defined by the recursion

$$\begin{aligned} \boldsymbol{v}_{t+1} &= \beta \boldsymbol{v}_t + \eta \boldsymbol{g}_t \\ \boldsymbol{\theta}_{t+1} &= (1-\lambda)\boldsymbol{\theta}_t - \boldsymbol{v}_{t+1} , \end{aligned} \tag{3.4}$$

where $\eta > 0$ is the *learning rate*, $\beta \in [0,1)$ is the *momentum parameter*, $\lambda \in [0,1)$ is the *weight decay parameter*.

We first consider the recursion in Eq. (3.4) case by case.

Case $\beta = \lambda = 0$ When $\beta = \lambda = 0$, then Eq. (3.4) reduces to the recursion

$$\boldsymbol{v}_{t+1} = \eta \boldsymbol{g}_t \quad \text{and} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{v}_{t+1} .$$

This is exactly the iterates of SGD, as we saw in Algorithm 1.1.

Case $\beta, \lambda \neq 0$ Unrolling the recursion in Eq. (3.4) with $\boldsymbol{v}_0 = \boldsymbol{0}$ in this case, we get[2]

$$\begin{aligned} \boldsymbol{\theta}_t &= (1-\lambda)^t \boldsymbol{\theta}_0 - \eta \sum_{\tau=0}^{t-1} a_{t-1-\tau} \boldsymbol{g}_\tau , \quad \text{where} \\ a_t &= \sum_{\tau=0}^{t} \beta^\tau (1-\lambda)^{t-\tau} . \end{aligned} \tag{3.5}$$

The weights a_t are an the exponentially decaying function of $t \in \mathbb{N}$. Thus, SGD with momentum and weight decay can be obtained from

[2]This can be established using, e.g., an induction argument. We leave the details as an exercise to the reader.

the weighted prefix sum $\boldsymbol{A}_{\mathsf{mom}}\boldsymbol{G}$ with the Toeplitz workload matrix

$$\boldsymbol{A}_{\mathsf{mom}} = \begin{pmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{pmatrix} \tag{3.6}$$

where the a_t's are defined in Eq. (3.5).

Remark 3.3. Other flavors of momentum, such as the one arising from Nesterov's accelerated gradient method, also admit similar representations as weighted prefix sums. We leave the derivation of the corresponding workload as an exercise to the reader.

We will focus our discussion in this section mainly on the unweighted prefix sum workload matrix $\boldsymbol{A}_{\mathsf{pre}}$ from Eq. (3.3) for concreteness, although all aspects of our discussions hold for more general non-negative Toeplitz and lower triangular workload matrices $\boldsymbol{A}$ such as the momentum matrix $\boldsymbol{A}_{\mathsf{mom}}$ from Eq. (3.6).

3.3 Multiple-Participation Correlated Noise Mechanisms

The results of Sections 1 and 2 are directly applicable for learning problems in the *streaming setting* where the data is processed in a single pass.[3] Further, we also considered stochastic gradient algorithms with a batch size of one. As we discussed in Remark 1.8, these assumptions are typically violated in practical AI model training scenarios. Indeed, we usually process each example multiple times in the learning process; we refer to this as the *multiple-participation setting*. Moreover, we process mini-batches of examples in each stochastic gradient step.

The version of DP-SGD with correlated noise we consider is given in Algorithm 3.1; it generalizes Algorithm 1.3 to (expected) mini-batch sizes B.[4] The key difference relative to non-private SGD is that we *clip the per-example gradients* before averaging them (Line 5), and then

[3] A complete pass over the data is termed an *epoch*.

[4] The batch size may only hold in expectation, as Poisson sampling or other sampling approaches Section 3.4 may produce a randomly-sized batch.

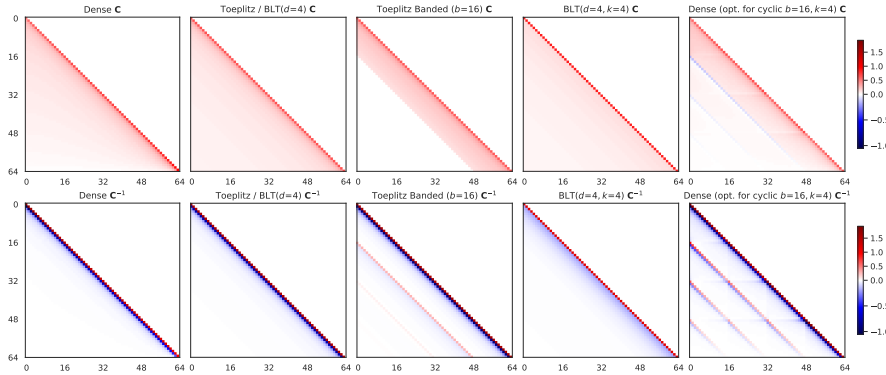


Figure 3.3: An expanded version of Fig. 2.1 which includes additional mechanisms optimized for multiple participations. As before, the first three mechanisms are optimized for single-participation, though the banded mechanism is also optimal for $b \geq 16$ Min-Sep participation, as sensitivity simply scales linearly with the number of participations, Eq. (3.21). The BLT mechanism in the fourth column is optimized to minimize max loss under $b = 16$ Min-Sep participation Eq. (3.17); the final dense mechanism is optimized for $k = 4$ training epochs with $b = 16$, each with the same cyclic data order, Eq. (3.16). Note unlike any of the other matrices, some elements of dense $\boldsymbol{C}$ are, in fact, negative.

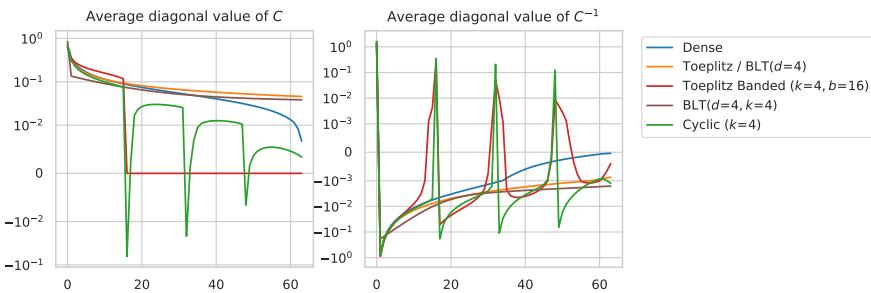


Figure 3.4: An alternative visualization of the matrices in Fig. 3.3; we represent each matrix by the mean value on each diagonal from the main diagonal and down. Of course, for the Toeplitz mechanisms this representation fully captures the matrix, but it is a lossy summary for the other classes. Nevertheless, these values for $\boldsymbol{C}^{-1}$ in particular, provide an indication of how a mechanism cancels noise; in fact, in the case of the banded and cyclic matrices, the mechanisms not only cancel previously added noise, but sometimes (on a period corresponding to b), re-add previously added noise as indicated by positive values below the main diagonal of $\boldsymbol{C}^{-1}$.

Algorithm 3.1 Mini-batch DP-SGD with Correlated Noise

Inputs: Dataset D, number of steps n, learning rate η, batch size B, noise variance ν^2, clip norm ζ, noise correlating matrix $\boldsymbol{C}^{-1}$ (for standard DP-SGD with independent noise, take $\boldsymbol{C}^{-1} = \boldsymbol{I}_{n\times n}$).

1: Define $\mathsf{clip}_\zeta(\boldsymbol{v}) := \boldsymbol{v} \cdot \min\{1, \zeta/\left\|\boldsymbol{v}\right\|_2\}$
2: Pick an initial model $\boldsymbol{\theta}_0 \in \Theta$
3: **for** $t = 0, 1, \ldots, n-1$ **do**
4: Receive a batch $\mathcal{B}_t \subseteq D$ of (expected) size B
5: $\boldsymbol{g}_t \leftarrow \frac{1}{B}\sum_{\boldsymbol{x}\in\mathcal{B}_t} \mathsf{clip}_\zeta(\nabla\ell(\boldsymbol{\theta}_t; \boldsymbol{x}))$ ▶ *Eq. (1.3) for* $\mathsf{clip}_\zeta(\cdot)$
6: Sample $\boldsymbol{z}_t \sim \mathcal{N}_m(0, \nu^2)$ ▶ *Sample i.i.d. seed noise*
7: $\widetilde{\boldsymbol{z}}_t \leftarrow \frac{1}{B}\sum_{\tau=0}^{t-1} \boldsymbol{C}^{-1}[t,\tau]\boldsymbol{z}_{t,\tau}$ ▶ *Correlated noise* $(\boldsymbol{C}^{-1}\boldsymbol{Z})[t,:]$
8: $\widehat{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t + \widetilde{\boldsymbol{z}}_t$ ▶ *Privatized* ***average*** *gradient.*
9: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\widehat{\boldsymbol{g}}_t$ ▶ *Or any first-order update*
10: **Return** $\boldsymbol{\theta}_n$

add (correlated) noise (Line 8). Critically, we then use the DP estimate of the *average* gradient in the first-order update (Line 9). Following standard convention, we assume that the noise variance ν^2 is calibrated to the sum of gradients. However, since $\boldsymbol{g}_t$ is calculated as the average mini-batch gradient (rather than their sum), we scale down the noise $\widetilde{\boldsymbol{z}}_t$ by a B factor in Line 7 to ensure correct noise calibration.

Remark 3.4. Fixing the number of training iterations n, the (expected) batch size B is a hyperparameter that plays an essential role in privacy-utility tradeoffs as well as compute cost. The prefix-sum max loss of the *average* gradient $\widehat{\boldsymbol{g}}_t$ is a reasonable proxy for learning performance. In the zero-out model, we have this *unnormalized* max loss is

$$\begin{aligned}\mathcal{L}_\infty(\widehat{\boldsymbol{g}}; \boldsymbol{B}, \boldsymbol{C}) &= \max_{t\in[n]} \sqrt{\mathbb{E}\left\|\sum_{\tau=0}^{t}\boldsymbol{g}_\tau - \sum_{\tau=0}^{t}\widehat{\boldsymbol{g}}_\tau\right\|_2} \\ &= \frac{\sigma m}{B}\left\|\boldsymbol{B}\right\|_{\mathsf{row}}\mathsf{sens}(\boldsymbol{C}) \end{aligned} \tag{3.7}$$

following Theorem 2.2 and Eq. (2.5). Hence, if we could increase the batch size without changing anything else (e.g., if we had infinite compute and an infinite dataset), we could drive the max loss down to be arbitrarily small. Of course, this requires nB examples, and so if our dataset is of fixed size N, some example must participate at least $k = \left\lceil \frac{nB}{N} \right\rceil$ times, and this will increase the sensitivity $\mathsf{sens}(\boldsymbol{C})$. Formally defining and then computing $\mathsf{sens}(\boldsymbol{C})$ in multiple-participation settings will be a principal topic in this section.

Once we account for the increased sensitivity (or equivalently, the impact of the additional participations under sampling), significant benefit from increasing the batch size remains. In fact, if compute cost was not an issue, we would always simply process the entire training dataset in every batch. *In the extreme of full-batch gradient descent, neither correlated noise nor privacy amplification via sampling offer any advantage.*[5] Fig. 3.5 is representative, showing the value of increasing batch size is pronounced for both the banded Toeplitz mechanism of Section 2.4 and vanilla DP-SGD with independent noise. Yet, we find that both correlated noise and amplification can (independently or jointly) lead to significant improvements in compute-constrained settings, where using the full batch for each gradient update is infeasible. We will revisit full-batch training in Remark 4.10.

In order to run this algorithm, we need to specify how to set the noise standard deviation ν^2 as a function of the problem parameters and the desired privacy level in this multiple-participation regime. It turns out that the analysis of Section 1 straightforwardly generalizes to larger mini-batch sizes. These techniques are, however, insufficient to handle the multiple-participation setting.

Recall from Section 1 (more specifically Section 1.3.1) that the streaming setting combined with Theorem 1.11 made it straightforward

[5] Proposition 3.11 shows that independent noise achieves the optimal max loss in the full-batch setting, meaning that noise correlations do not give any additional benefits. Similarly, there is no added uncertainty from sampling when we use the full batch, so there is no amplification by sampling in this regime.

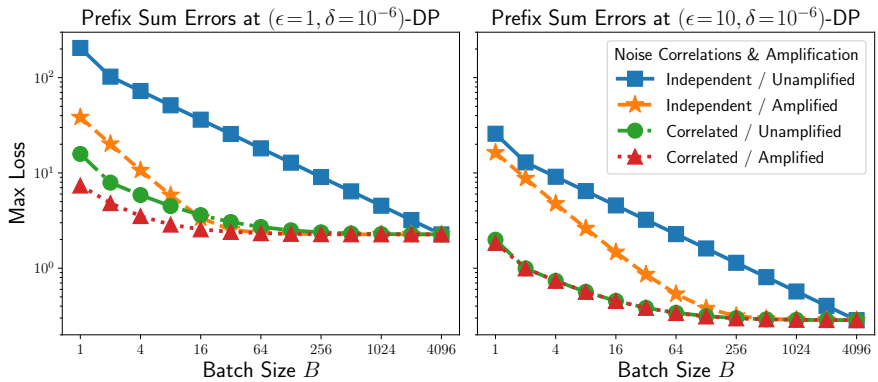


Figure 3.5: Effect of the batch size: Fixing the number of iterations $n = 2048$ and dataset size $N = 4069$, and increasing the batch size can substantially decrease the max loss in the average gradient of DP-SGD (Algorithm 3.1, Line 8) for both correlated noise (in this case, the banded Toeplitz mechanism of Section 2.4) and independent noise, both with privacy amplification from Poisson sampling (Section 3.4) and in unamplified scenarios. Because $N/n = 2$, the number of epochs k is equal to $B/2$. Thus $B = 1$ and $B = 2$ correspond to the streaming setting, while $B = 4096$ corresponds to full-batch gradient descent. For the banded Toeplitz mechanism with cyclic Poisson sampling, the number of bands was empirically optimized to minimize the loss; using fewer bands increases the benefits of amplification, while more bands allows more flexibility in designing correlated noise. At $\varepsilon = 10$, correlated noise provides most of the benefit, but at $\varepsilon = 1$ amplification yields significant additional improvements for small batch sizes.

to translate adjacency of datasets into adjacency of gradients: since each data point is processed only once, two sequences of gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ and $\boldsymbol{G}' = (\boldsymbol{g}'_0, \ldots, \boldsymbol{g}'_{n-1}) \in \mathbb{R}^{n \times m}$ arising from adjacent datasets D and D' can differ in at most one element, i.e., $\boldsymbol{g}_\tau = \boldsymbol{g}'_\tau$ for all $\tau \in [n]$ except possibly for one index t (see Theorem 1.11 for more discussion).

However, model training typically involves making $k > 1$ passes over the data.[6] A change of one data point in an adjacent dataset leads to a change in k gradients in the input $\boldsymbol{G}$. The goal of this section is to extend the results of the preceding sections to the multiple-participation setting.

3.3.1 Multiple-Participation Training Setup

Suppose we have a dataset of N data points, which appear in n iterations in batches of size B. We are interested in the multiple-participation setting where $nB/N > 1$, which implies that at least one data point is used at least twice. Further, we let k denote the maximum number of times a data point might participate in training. For example, k can refer to the number of epochs in settings where that can be tracked.

The input to our correlated noise mechanism is a sequence of gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ obtained from a dataset $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of data points. We assume that each gradient $\boldsymbol{g}_t$ is obtained as the total gradient over a batch $I_t \subset [N]$ of $|I_t| = B$ data points (evaluated at the current model $\boldsymbol{\theta}_t$):

$$\boldsymbol{g}_t = \frac{1}{B} \sum_{i \in I_t} \nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_i)\,. \tag{3.8}$$

Throughout this section, we assume a *gradient norm bound* of 1:

$$\|\nabla \ell(\boldsymbol{\theta}, \boldsymbol{x})\|_2 \leq 1 \quad \text{for all } \boldsymbol{\theta} \in \Theta \text{ and } \boldsymbol{x} \in \mathcal{X}\,. \tag{3.9}$$

In practice, this can be achieved by the so-called *gradient clipping* operation; see Algorithm 3.1, Line 5. As mentioned in Section 1, the case of clipping to a different norm can simply be handled by scaling.

[6] Other forms of multiple participation are possible, e.g., in federated learning where data can be processed in arbitrary order depending on device availability. We discuss this further in Section 3.3.3.

Adjacent Datasets and Sequences We define adjacent sequences of gradients based on adjacency of the underlying datasets, using the *zero-out* notion of the adjacency (following much prior work in this setting; see Section 3.7). Section 1.7 provides additional discussion of this choice, but for the remainder we only need the definition, which we restate for convenience:

Definition 3.5 (Zero-out Adjacency of Datasets)**.** Two datasets $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $D' = \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_N\}$ are **zero-out adjacent** (denoted $D \simeq D'$) if $\boldsymbol{x}_i = \boldsymbol{x}_i$ for all indices $i \in [n]$ except possibly at some index $j \in [n]$. At this index j, we have either $\boldsymbol{x}_j = \perp$ or $\boldsymbol{x}'_j = \perp$, where "$\perp$" is a special *null* element such that $\nabla \ell(\boldsymbol{\theta}, \perp) = \mathbf{0}$ for all $\boldsymbol{\theta}$.

Replacing an example by $\perp$ is a convenient way to capture the removal of a data point without changing the dataset size or how examples are grouped into batches. This *zero-out* notion of adjacency generally yields sensitivities that are a factor of 2 smaller than *replace-one adjacency* which we use in Sections 1 and 2.

Recalling Theorem 1.11, we may assume a *fixed* sequence of model iterates $(\boldsymbol{\theta}_t)_{t=1}^{n-1}$ when defining adjacent sequences of gradients:

Definition 3.6 (Zero-out Adjacency of Gradients)**.** Two sequences of gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ and $\boldsymbol{G}' = (\boldsymbol{g}'_0, \ldots, \boldsymbol{g}'_{n-1}) \in \mathbb{R}^{n \times m}$ of gradients are *adjacent* if for each $t \in [n]$, we have that

$$\boldsymbol{g}_t = \sum_{i \in I_t} \nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_i) \quad \text{and} \quad \boldsymbol{g}'_t = \sum_{i \in I_t} \nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x}'_i)$$

are evaluated for the same sequence of batch indices $I_0, \ldots, I_{n-1} \subset [n]$ and the same sequence of models $(\boldsymbol{\theta}_t)_{t=1}^{n-1}$ on adjacent datasets $D = \{\boldsymbol{x}_i\}_{i=1}^N$ and $D' = \{\boldsymbol{x}'_i\}_{i=1}^N$. We say that $\boldsymbol{G}, \boldsymbol{G}'$ are zero-out adjacent if the underlying datasets $D \simeq D'$ are zero-out adjacent; we denote this as $\boldsymbol{G} \simeq \boldsymbol{G}'$.

Participation Patterns Let I_t be the set of batch indices that is picked at time t. We say that a data point $\boldsymbol{x}_i$ *participates* in step t if $i \in I_t$. Note that if a data point can participate in up to k batches $I_{t_1}, \ldots, I_{t_k}$, then $\boldsymbol{G}$ and $\boldsymbol{G}'$ can differ in up to k gradients $\boldsymbol{g}_{t_1}, \ldots, \boldsymbol{g}_{t_k}$.

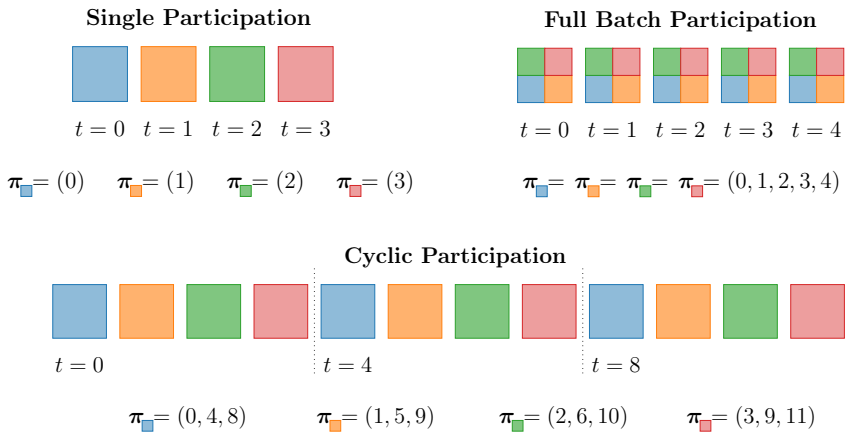


Figure 3.6: Illustration of participation patterns where each color represents a data point. **Top Left**: The streaming setting with $N = 4$ data points over $n = 4$ steps with a batch size of $B = 1$ for one epoch ($k = 1$). **Top Right**: Full batch participation of $N = 4$ data points over $n = 5$ steps (the batch size is $B = N$ and number of epochs are $k = n$). **Bottom**: Cyclic participation of $N = 4$ data points over $n = 12$ steps with a batch size of $B = 1$, leading to $k = nB/N = 3$ epochs (separated by the dotted line).

It is useful to define the sequence $\boldsymbol{\pi}_i$ of all steps where $\boldsymbol{x}_i$ participates in training:

$$\boldsymbol{\pi}_i := \left(t \in [n] \,:\, i \in I_t\right). \tag{3.10}$$

We refer to $\boldsymbol{\pi}_i$ as the *participation pattern* of data point $\boldsymbol{x}_i$. For instance, $\boldsymbol{\pi}_i = [n]$ indicates that data point i participates in each step, while $|\boldsymbol{\pi}_i| = 1$ corresponds to the setting where $\boldsymbol{x}_i$ only appears once during training. Finally, if data point i participates in the ℓ^{th} step of each cyclic pass over the data, we have (assuming the batch size B divides the dataset size N):

$$\boldsymbol{\pi}_i = \left(l,\ l + \frac{N}{B}, l + 2\frac{N}{B},\ \ldots,\ l + (k-1)\frac{N}{B}\right). \tag{3.11}$$

A natural baseline is to simply "restart" the mechanism at the end of each epoch (assuming the algorithm can run in epochs). Then, we can simply compose the privacy loss across epochs using Lemma 3.19. We will instead derive tight sensitivity bounds for the entire multiple-participation training algorithm as one mechanism. This will recover the "restarting mechanism" as a special case; we return to this in Section 3.3.6.

As we will see in the next section (Section 3.3.2), tight sensitivity bounds in the multiple-participation setting requires us to impose restrictions on the allowed participation patterns. We will define specific participation patterns in Section 3.3.3 for further study.

3.3.2 Multiple-Participation Sensitivity: Definition and Challenges

Given a stream of gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ as input, the correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}$ is an instance of the Gaussian mechanism with i.i.d. Gaussian noise $\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(0, \nu^2)$ (Definition 1.5). Thus, following Lemma 1.6, we only need to bound the sensitivity of the linear map $\boldsymbol{G} \mapsto \boldsymbol{C}\boldsymbol{G}$ to design mechanisms suited to the multiple-participation case. Recall we did this for the single-participation streaming setting in Theorem 1.14; we now generalize this result. Similar to that case, we can simply consider a non-adaptive setting, thanks to Theorem 1.11.

Definition 3.7 (ℓ_2 Sensitivity Induced by the Strategy Matrix)**.** The ℓ_2-sensitivity of the matrix-valued map $\boldsymbol{G} \mapsto \boldsymbol{C}\boldsymbol{G}$ is

$$\mathsf{sens}(\boldsymbol{C}) = \sup_{\boldsymbol{G} \simeq \boldsymbol{G}' \in \mathbb{R}^{n \times m}} \left\| \boldsymbol{C}\boldsymbol{G} - \boldsymbol{C}\boldsymbol{G}' \right\|_{\mathrm{F}} , \tag{3.12}$$

where the Frobenius norm $\|\boldsymbol{M}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} \boldsymbol{M}[i,j]^2}$ generalizes the ℓ_2-norm of Definition 1.4 to matrix-valued maps.

Given a bound on this sensitivity, we can get a GDP guarantee, generalizing Lemma 1.13 and Theorem 1.14:

Lemma 3.8. Fix a noise multiplier $\sigma > 0$. Consider the nonadaptive multi-participation setting, i.e.

(a) we have that $\boldsymbol{G} \simeq \boldsymbol{G}'$ are adjacent in the zero-out sense, as defined in Definition 3.6, and

(b) the rows $\boldsymbol{g}_t, \boldsymbol{g}'_t$ are clipped to norm 1 (cf. Eq. (1.8) or Eq. (3.9)).

Then, the mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z})$ for any matrices $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$ (possibly non-lower-triangular and non-invertible) and i.i.d. Gaussian noise $\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(0, \nu^2)$ satisfies $\frac{1}{\sigma}$-GDP if we choose the noise standard deviation as $\nu = \sigma\, \mathsf{sens}(\boldsymbol{C})$.

Recall that Theorem 1.11 in Section 1 reduces the adaptive case to the nonadaptive case. Then, similar to Theorem 1.14, this directly leads to a GDP bound on Algorithm 3.1:

Theorem 3.9. Fix a noise multiplier σ and consider Algorithm 3.1 with an invertible lower-triangular strategy matrix $\boldsymbol{C}$, clip norm $\zeta = 1$, any batch size $B > 0$, and noise standard deviation $\nu = \sigma\, \mathsf{sens}(\boldsymbol{C})$. Then, the privatized gradients $(\widehat{\boldsymbol{g}}_t)_{t \in [n]}$ and iterates $(\boldsymbol{\theta}_t)_{t \in [n]}$ produced by Algorithm 3.1 satisfy $\frac{1}{\sigma}$-GDP under the zero-out adjacency of input datasets (Definition 3.5), even in the multi-participation setting.

Comparing Theorem 3.9 for the multi-participation case to to Theorem 1.14 for streaming setting, we note that the sensitivity $\mathsf{sens}(\boldsymbol{C})$

does the heavy lifting in capturing the multiple-participation aspect. This sensitivity bound of Definition 3.7 is required to hold over all pairs of adjacent sequences of gradients $\boldsymbol{G} \simeq \boldsymbol{G}'$ as in Definition 3.6. In the (non-adaptive version of the) streaming setting, $\boldsymbol{G}$ and $\boldsymbol{G}'$ can differ only in one row, and this difference is limited by the gradient norm bound. Thus, we recover $\mathsf{sens}(\boldsymbol{C}) = \|\boldsymbol{C}\|_{\mathsf{col}}$ as in Theorem 1.14.[7] We can have $\mathsf{sens}(\boldsymbol{C}) > \|\boldsymbol{C}\|_{\mathsf{col}}$ in the multi-participation setting, as multiple rows of $\boldsymbol{G}$ can change in an adjacent dataset (corresponding to the participation of the underlying example that changes).

If we do not impose any restrictions on the allowed participation patterns for the multiple-participation setting, a single data point can participate in every single iteration in the worst case. Thus, it is possible to have $\boldsymbol{G} \simeq \boldsymbol{G}'$ that differ in *every row* by an ℓ_2 distance of up to 1 (due to the gradient norm bound assumption of Eq. (3.9)). By our definition of adjacency, it follows for the simpler case of $m = 1$ dimension that two sequences $\boldsymbol{G}, \boldsymbol{G}' \in \mathbb{R}^{n \times 1}$ are *adjacent* (in the absence of participation pattern restrictions) if $\|\mathsf{vec}(\boldsymbol{G} - \boldsymbol{G}')\|_\infty \leq 1$, where $\mathsf{vec}(\cdot)$ treats its matrix input of shape $n \times 1$ as an n-dimensional vector.

We then have for the ℓ_2-sensitivity

$$\begin{aligned} \mathsf{sens}(\boldsymbol{C}) &= \max_{\|\mathsf{vec}(\boldsymbol{G}-\boldsymbol{G}')\|_\infty \leq 1} \|\boldsymbol{C}(\boldsymbol{G} - \boldsymbol{G}')\|_{\mathrm{F}} \\ &= \max_{\|\boldsymbol{u}\|_\infty \leq 1} \|\boldsymbol{C}\boldsymbol{u}\|_2 =: \|\boldsymbol{C}\|_{\infty \to 2} \,, \end{aligned} \tag{3.13}$$

where the Frobenius norm of Eq. (3.12) reduces to the ℓ_2 norm for $m = 1$ dimension in Eq. (3.13). Here, the notation $\|\boldsymbol{C}\|_{\infty \to 2}$ denotes the $\infty \to 2$ matrix operator norm (also known as the induced matrix norm). In general, for $p, q \in [1, \infty]$, the $p \to q$ operator norm of a matrix is defined as:

$$\|\boldsymbol{C}\|_{p \to q} := \max_{\boldsymbol{u} \neq \boldsymbol{0}} \frac{\|\boldsymbol{C}\boldsymbol{u}\|_q}{\|\boldsymbol{u}\|_p} \,. \tag{3.14}$$

[7] The multiplicative factor of 2 in Theorem 1.14 is due to the replace-one adjacency, and vanishes if we consider the zero-out adjacency as in this section. See Section 1.7 for a detailed discussion on different notions of adjacency.

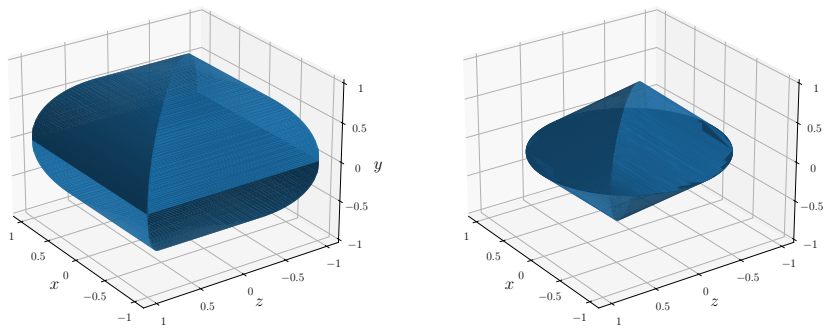


Figure 3.7: Operator norm balls: The norm balls of the operator norms $\|\boldsymbol{M}\|_{\mathsf{col}} = \|\boldsymbol{M}\|_{1\to 2}$ (left) and $\|\boldsymbol{M}\|_{\infty\to 2}$ (right) of the symmetric matrix $\boldsymbol{M} = \begin{pmatrix} x & y \\ y & z \end{pmatrix}$ plotted as a function of (x, y, z). The left norm ball $\{\boldsymbol{M} : \|\boldsymbol{M}\|_{1\to 2} \le 1\}$ is much bigger than the right one $\{\boldsymbol{M} : \|\boldsymbol{M}\|_{\infty\to 2} \le 1\}$, meaning that more matrices satisfy the constraint $\|\boldsymbol{M}\|_{1\to 2} \le 1$ than the constraint $\|\boldsymbol{M}\|_{\infty\to 2} \le 1$; see Lemma 3.10. Figures adapted from Grimmer [2022] with permission.

Challenges Unfortunately, the sensitivity in Eq. (3.13) that we obtained in the absence of participation restrictions suffers from some major difficulties. First, computing the operator norm $\|\boldsymbol{C}\|_{\infty\to 2}$ for general $\boldsymbol{C}$ matrices is NP-hard.[8] Second, the multiple-participation sensitivity can be significantly larger than the single-participation sensitivity $\|\boldsymbol{C}\|_{\mathsf{col}}$. In particular, the multiple-participation sensitivity can be worse by a factor of the number n of steps (see Section 3.6 for a proof):

Lemma 3.10. For any matrix $\boldsymbol{C} \in \mathbb{R}^{p\times n}$, we have,

$$\|\boldsymbol{C}\|_{\mathsf{col}} \le \|\boldsymbol{C}\|_{\infty\to 2} \le n\,\|\boldsymbol{C}\|_{\mathsf{col}}\,.$$

This large spike in the sensitivity means that independent noise achieves the optimal max loss in this setting (see Section 3.6 for a proof):

[8] To be precise, the corresponding decision problem is NP-hard: does there exist a vector $\boldsymbol{u} \in \mathbb{R}^n$ with $\|\boldsymbol{u}\|_\infty \le 1$ such that $\|\boldsymbol{C}\boldsymbol{u}\|_2 \le \kappa$ for some given constant $\kappa > 0$?

Proposition 3.11. We have that

$$\min_{\boldsymbol{C}\in\mathbb{R}^{n\times n}} \left\{ \left\| \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1} \right\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\infty\to 2} \right\} = n\,,$$

and this minimum is attained at $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$. That is, the correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}$ in $m = 1$ dimension that achieves the optimal max loss with the worst-case multi-participation sensitivity (Eq. (3.13)) is independent noise $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$.

However, in contrast to Proposition 3.11, empirically, we observe that carefully designed correlated noise mechanisms lead to significant practical improvements (e.g., in Figs. 3.2 and 3.5). A key reason for the limitations Eq. (3.13) and the approach of this section is that, without participation restrictions, the matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$ can differ arbitrarily in every row. This setup fails to reflect practical scenarios, where examples typically participate a few times during training, but not in every iteration.

In particular, as discussed in Remark 3.4, when sufficient compute resources are available, increasing the batch size is generally beneficial, often pushing us into the multi-participation regime. So, a natural approach would be to use full-batch gradient descent (where $\boldsymbol{G} \simeq \boldsymbol{G}'$ may differ in every row). However, empirical evidence suggests that the benefit of using full-batch gradient descent in terms of loss is not significant to warrant using the extra compute resource. This can be also seen in Fig. 3.5, where a batch size of $B = 256$ achieves nearly the same loss as using the full dataset ($B = 4096$) while requiring $16\times$ less compute, even without amplification.

These discussions suggests that we should seek reasonable restrictions on the allowed participation patterns (and sometimes also on the matrix $\boldsymbol{C}$) to design correlated noise mechanisms for the multi-participation setting. In particular, we have the following desiderata:

(a) **Implementation is practical**: The restrictions on participation patterns can be implemented efficiently in real-world ML training pipelines.

(b) **Computing $\mathsf{sens}(\boldsymbol{C})$ is tractable**: The multiple-participation sensitivity can be computed (or tightly bounded from above) in

polynomial time in the problem parameters, namely the number of training samples, n, and the model dimension, m, for at least some classes of strategy matrices $\boldsymbol{C}$. Preferably, it can be computed in $O(n)$ time or less and is independent of the model dimension m.

(c) **Larger batches give improvements:** Finally, we seek restrictions where the unnormalized max loss (Eq. (3.7)) improves as we increase the batch size (and compute cost) — otherwise, we would be better off just using a smaller batch size and staying in the streaming setting. (These can typically only be verified empirically, as in Fig. 3.5.)

Next, we will define participation patterns that satisfy these criteria and derive sensitivity bounds.

3.3.3 Practical Participation Patterns

Recall that the participation pattern $\boldsymbol{\pi} \subset [n]$ of a data point is the sequence of steps in which it participates in training; see Eq. (3.10) for a definition. We define the set of all "allowed" participation patterns as an abstraction to impose restrictions on how we process the data:

Definition 3.12 (Participation Schema)**.** A *participation schema* $\Pi \subset 2^{[n]}$ is a subset of possible participation patterns, called the *allowed* participation patterns. That is, two sequences of gradients $\boldsymbol{G} = (\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}) \in \mathbb{R}^{n \times m}$ and $\boldsymbol{G}' = (\boldsymbol{g}'_0, \ldots, \boldsymbol{g}'_{n-1}) \in \mathbb{R}^{n \times m}$ are adjacent under schema Π if and only if there exists a participation pattern $\boldsymbol{\pi} \in \Pi$ such that:

(a) $\boldsymbol{g}_t = \boldsymbol{g}'_t$ for all $t \notin \boldsymbol{\pi}$;

(b) $\|\boldsymbol{g}_t - \boldsymbol{g}'_t\|_2 \leq 1$ for all $t \in \boldsymbol{\pi}$ (following Eq. (3.9))

We denote this as $\boldsymbol{G} \overset{\Pi}{\simeq} \boldsymbol{G}'$.

In other words, the participation schema imposes restrictions on how many and which gradients in the stream $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{n-1}$ can be affected by changing one data point. This lets us map adjacency of datasets D, D' into adjacency of the corresponding gradients $\boldsymbol{G}, \boldsymbol{G}' \in \mathbb{R}^{n \times m}$.

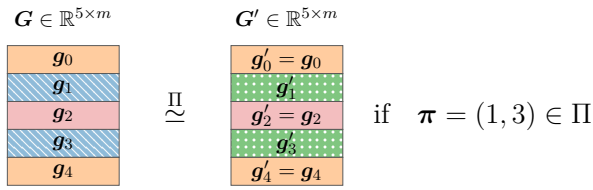


Figure 3.8: Two inputs $G, G' \in \mathbb{R}^{n\times m}$ (with $n = 5$) are adjacent if they can only differ in the rows contributed by one data point. In this illustration, the blue data point in G is replaced by the green data point in G'. The participation schema Π controls all allowed participation patterns. So, we have that $G \overset{\Pi}{\simeq} G'$ are adjacent under schema Π only if the participation pattern $\pi = (1,3)$ that encodes the rows where G, G' differ is allowed as per Π (i.e., only if $\pi = (1,3) \in \Pi$).

We can now formally define the sensitivity of the matrix C (representing the linear map $G \mapsto CG$) calibrated to a participation schema:

Definition 3.13 (Participation-Calibrated Sensitivity)**.** The ℓ_2-sensitivity of the matrix $C \in \mathbb{R}^{n\times n}$ restricted to participation schema $\Pi \subseteq 2^{[n]}$ and dimension m is defined as

$$\mathsf{sens}(C, \Pi, m) = \max_{G, G' \in \mathbb{R}^{n\times m}} \left\{ \|CG - CG'\|_{\mathrm{F}} \,:\, G \overset{\Pi}{\simeq} G' \right\} . \quad (3.15)$$

When $\mathsf{sens}(C, \Pi, m)$ is independent of the dimension m, we will omit m and simply write $\mathsf{sens}(C, \Pi)$.

Simple Examples We now consider some basic participation schemas arising in the machine learning setting. See Fig. 3.6 for an illustration of the corresponding participation patterns.

(a) **Streaming**: Each data point appears only once in training, so that $|\pi| = 1$ for each $\pi \in \Pi$. This conforms to the schema

$$\Pi^{\mathsf{single}} := \{(0), (1), \ldots, (n-1)\} .$$

Following Theorem 1.14, we have $\mathsf{sens}(C, \Pi^{\mathsf{single}}) = \|C\|_{\mathsf{col}}$. Crucially, this is independent of the dimension m. In this case, we simply use the notation $\mathsf{sens}(C)$. For sufficiently large datasets and limited compute resources, this can be a very practical approach. However, when possible, we will generally achieve better

privacy/utility tradeoffs by using larger batches, as we saw in Remark 3.4. This generally requires multiple participations.

(b) **Full batch**: Each data point participates in each step of training, so that $\Pi^{\mathsf{full}} = \{[n]\}$. This yields $\mathsf{sens}(\boldsymbol{C}, \Pi^{\mathsf{full}}, 1) = \|\boldsymbol{C}\|_{\infty\to 2}$. As discussed in Remark 3.4, this full-batch training approach yields optimal privacy-utility tradeoffs, but at a very large cost in compute, since the total number of gradient evaluation is $Bn = BN$). We will generally be able to do almost as well with much smaller batches with carefully chosen participation restrictions; see Fig. 3.5.

Next, we consider two non-trivial participation schema that satisfy the desiderata of Section 3.3.2.

Cyclic Participation The first new participation pattern we consider is tailored to centralized[9] training pipelines. We loop over fixed batches of examples in some predefined fixed order.[10] Therefore, each data point occurs in a fixed iteration in each epoch, so the only types of participation patterns allowed are as in Eq. (3.11); see Fig. 3.6 for an illustration. For a dataset of size N and batch size B, each pass (epoch) comprises of $b = N/B$ steps (assuming N divides B), and so for k epochs, we have the participation schema

$$\Pi^{\mathsf{cyclic}}_{b,k} = \left\{ (l, l+b, l+2b, \ldots, l+(k-1)b) \ : \ l \in [b] \right\}. \tag{3.16}$$

[9]We refer to training models in a data-center (possibly in a distributed fashion) as *centralized*. The key property of centralized training is that the full training dataset is always available, and hence can be accessed in any reasonable fashion; this is in contrast to the federated setting of Example 3.1, where for example the data on any particular device might be unavailable for arbitrary periods of time, e.g. when the device is offline.

[10]The cyclic participation setting is practical in that it can easily be implemented in realistic centralized training pipelines. While we lose some generality as we are not allowed to shuffle the data in each epoch, we can introduce some randomness by shuffling the data *once* at the start of training. It is not known if tight sensitivity bounds can be derived for the shuffle-every-epoch approach; we return to this open problem in Section 5.

As we see in the upcoming Section 3.3.4, it is computationally efficient to evaluate the multiple-participation sensitivity in this case under mild conditions.

Minimum Separation (Min-Sep) Participation While it is possible to make cyclic passes through the data in centralized model training, this is not possible in federated learning, where the client availability is determined by external factors (see Example 3.1). However, it is often possible to prevent any client from participating twice in quick succession. Each client remembers the last step τ in which it participated, and participates in training again only when the current step t satisfies $t \geq \tau + b$, for a minimum separation parameter b. Importantly, no client is forced to check in with the server during a narrow (and unknown to the device) time window for a specific step, as required by cyclic participation. This leads to a generalization of cyclic participation that inherits its favorable traits (as we will discuss further in the coming sections) but is also practical for federated learning.

Definition 3.14 (Participation with Minimum Separation)**.** A participation pattern $\boldsymbol{\pi}$ satisfies *b-minimum-separation (Min-Sep)* with at most k participations if if: (a) every pair of indices $t, \tau \in \boldsymbol{\pi}$ satisfy $t = \tau$ or $|t - \tau| \geq b$, i.e., subsequent participations of any data point are at least b steps apart, and (b) $|\boldsymbol{\pi}| \leq k$, i.e., each data point participates at most k times. The corresponding participation schema is the collection of all participation patterns that satisfy the b-Min-Sep condition:

$$\Pi_{b,k}^{\mathsf{minSep}} = \left\{ \boldsymbol{\pi} \subset [n] \,:\, \begin{array}{c} |\boldsymbol{\pi}| \leq k, \text{ and} \\ \forall t, \tau \in \boldsymbol{\pi} : \, t = \tau \text{ or } |t - \tau| \geq b \end{array} \right\}. \quad (3.17)$$

We given an illustration of the minimum separation condition in Fig. 3.9. We usually refer to the participation pattern and schema from Definition 3.14 as "*b-Min-Sep participation*" with the understanding that at most k participations are allowed throughout (for all participation schemas we consider).

We assume throughout that k is feasible, that is, that k participations are in fact possible while satisfying a minimum separation of b. The

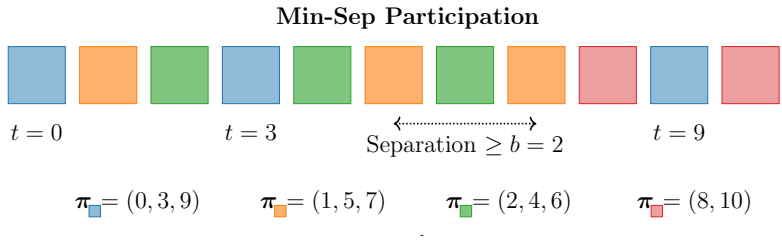


Figure 3.9: Illustration of the Min-Sep participation pattern. Each color represents a data point, as in Fig. 3.6. This setting corresponds to $N = 4$ data points participating in batches of size $B = 1$ over $n = 11$. Each point participates at most $k = 3$ times with a minimum separation of $b = 2$. Observe that any two subsequent participations of any given data point are at least $b = 2$ steps apart.

maximum value of $k \in \mathbb{N}$ is then determined by the early-and-often pattern:

$$\boldsymbol{\pi}^* := (0, b, 2b, \dots (k-1)b) \quad \text{subject to } (k-1)b < n. \tag{3.18}$$

We do not in general assume that b divides n, and so the last participation $(k-1)b$ in $\boldsymbol{\pi}^*$ could occur at any iteration between $t = n - b$ and the last iteration $t = n - 1$, both included.

Observe that $\Pi^{\mathsf{cyclic}}_{b,k} \subset \Pi^{\mathsf{minSep}}_{b,k}$. In other words, cyclic participation is a special case of Min-Sep participation. As a consequence, if $\boldsymbol{G} \simeq \boldsymbol{G}$ as per the cyclic participation schema $\Pi^{\mathsf{cyclic}}_{b,k}$, then $\boldsymbol{G} \simeq \boldsymbol{G}'$ as per the Min-Sep schema $\Pi^{\mathsf{minSep}}_{b,k}$ as well. Thus, we will focus on computing the sensitivity for the Min-Sep participation as this directly yields bounds on the sensitivity for cyclic participation.

We also remark that $\left|\Pi^{\mathsf{minSep}}_{b,k}\right| \gg \left|\Pi^{\mathsf{cyclic}}_{b,k}\right| = b$, where the number of Min-Sep participation patters can grow as n^k (see Eq. (3.23)). This distinction will crucially matter in Section 3.3.5, when designing practical algorithms to compute the sensitivity efficiently.

3.3.4 Expressions for Multiple-Participation Sensitivity

We now give various expressions for the participation-calibrated sensitivity of Definition 3.13. These will directly lead to efficient algorithms to compute the sensitivity in Section 3.3.5.

Generic Sensitivity Bounds We start by explicitly relating the sensitivity to the participation schema. Recall that we use the notation $(\boldsymbol{C}^\top \boldsymbol{C})[t, \tau]$ to denote the $(t, \tau)^{\text{th}}$ entry of the matrix $\boldsymbol{C}^\top \boldsymbol{C}$.

Lemma 3.15. For any participation schema Π and any lower-triangular matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$, we have:

(a) For any dimension $m \geq 1$, we have the upper bound

$$\mathsf{sens}\,(\boldsymbol{C}, \Pi, m)^2 \leq \max_{\boldsymbol{\pi} \in \Pi} \sum_{t,\tau \in \boldsymbol{\pi}} \left| (\boldsymbol{C}^\top \boldsymbol{C})[t, \tau] \right| .$$

(b) Suppose that

$$\min_{t,\tau \in \boldsymbol{\pi}} (\boldsymbol{C}^\top \boldsymbol{C})[t, \tau] \geq 0 \qquad \text{for all } \boldsymbol{\pi} \in \Pi. \tag{3.19}$$

(This is true when $\boldsymbol{C}^\top \boldsymbol{C}$ is element-wise non-negative.) Then, we get a dimension-independent bound:

$$\mathsf{sens}(\boldsymbol{C}, \Pi)^2 = \max_{\boldsymbol{\pi} \in \Pi} \sum_{t,\tau \in \boldsymbol{\pi}} (\boldsymbol{C}^\top \boldsymbol{C})[t, \tau] . \tag{3.20}$$

That is, Part (a) holds with equality.

We discuss the interpretation of this result and its implications, before giving the proof. Unlike the streaming setting, the sensitivity generally depends on the dimension m. For general any general strategy matrix $\boldsymbol{C}$, we have an upper bound on the sensitivity (Part (a)). However, a fortunate exception arises when $\boldsymbol{C}^\top \boldsymbol{C}$ is element-wise non-negative (Part (b)). In such cases, the sensitivity becomes independent of the dimension m, and the upper bound holds with equality.

Fortunately, most practical scenarios involve $\boldsymbol{C}^\top \boldsymbol{C}$ being non-negative. This is often because $\boldsymbol{C}$ itself is non-negative, a condition met by mechanisms like the max-loss-optimal Toeplitz mechanism (Section 2.3), BLT (Section 2.5), and tree aggregation (Section 2.7). Even when optimizing over $\boldsymbol{C}$ without enforcing non-negativity, we often find that $\boldsymbol{C}^\top \boldsymbol{C}$ is either non-negative or has negligible negative entries. Therefore, imposing non-negativity on $\boldsymbol{C}^\top \boldsymbol{C}$ results in solutions that are equally competitive (i.e., the non-negative solution attains nearly iden-

tical utility in terms of max loss and learning performance at any given privacy level). We return to practical considerations around optimizing for the $\boldsymbol{C}$ matrix in Section 4.

It is also instructional to rewrite Eq. (3.20) as

$$\mathsf{sens}(\boldsymbol{C}, \Pi)^2 = \max_{\boldsymbol{\pi} \in \Pi} \sum_{t,\tau \in \boldsymbol{\pi}} \langle \boldsymbol{C}[:,t], \boldsymbol{C}[:,\tau] \rangle \,.$$

This expression measures the similarity between the columns $\boldsymbol{C}[:,t]$ and $\boldsymbol{C}[:,\tau]$ corresponding to any two participations $t, \tau \in \boldsymbol{\pi}$ of any example. For the independent noise setting of $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$, this inner product is 1 when $t = \tau$ and 0 otherwise, leading to a (squared) sensitivity of $\max_{\boldsymbol{\pi}\in\Pi} |\boldsymbol{\pi}| = k$, the maximum number of participations.

Finally, the expression of Lemma 3.15 can directly be used to compute the sensitivity with a brute-force enumeration of all possible $\boldsymbol{\pi} \in \Pi$. This is tractable when $|\Pi|$ is small, e.g., $O(n)$; we discuss this further in Section 3.3.5.

We now give the proof of Lemma 3.15.

Proof of Lemma 3.15. Consider two gradient matrices $\boldsymbol{G} \overset{\Pi}{\simeq} \boldsymbol{G'} \in \mathbb{R}^{n\times m}$ that are adjacent as per the schema Π (Definition 3.12). By Definition 3.13, their difference $\boldsymbol{U} := \boldsymbol{G} - \boldsymbol{G'}$ must have non-zero rows indexed by some participation pattern $\boldsymbol{\pi} \in \Pi$, which we denote (by slight abuse of notation) as $\boldsymbol{\pi}(\boldsymbol{U})$, so $\forall t \in [n], \quad \boldsymbol{U}[t,:] \neq \boldsymbol{0} \Rightarrow t \in \boldsymbol{\pi}(\boldsymbol{U})$. Below, we denote the rows of $\boldsymbol{U}$ as $\boldsymbol{u}_t := \boldsymbol{U}[t,:]$ (which satisfy $\|\boldsymbol{u}_t\|_2 \leq 1$ by assumption Eq. (3.9)) and $\boldsymbol{M} := \boldsymbol{C}^\top \boldsymbol{C}$. Then, using the fact that $\|\boldsymbol{X}\|_{\mathrm{F}}^2 = \mathsf{Tr}(\boldsymbol{X}^\top \boldsymbol{X})$ for any matrix $\boldsymbol{X}$, we get,

$$\begin{aligned}
\|\boldsymbol{CG} - \boldsymbol{CG'}\|_{\mathrm{F}}^2 &= \left\| \sum_{t \in \boldsymbol{\pi}(\boldsymbol{U})} \boldsymbol{C}[:,t]\, \boldsymbol{u}_t^\top \right\|_{\mathrm{F}}^2 \\
&= \mathsf{Tr}\left(\sum_{t,\tau \in \boldsymbol{\pi}(\boldsymbol{U})} \boldsymbol{u}_t \boldsymbol{C}[:,t]^\top \boldsymbol{C}[:,\tau] \boldsymbol{u}_\tau^\top \right) \\
&\overset{(a)}{=} \sum_{t,\tau \in \boldsymbol{\pi}(\boldsymbol{U})} \boldsymbol{M}[t,\tau]\; \boldsymbol{u}_\tau^\top \boldsymbol{u}_t
\end{aligned}$$

where (a) used the linearity of the trace (to interchange the trace and summation) and the cyclic property (to group $\boldsymbol{u}_\tau$ and $\boldsymbol{u}_t$). Next, using

(b) triangle inequality, (c) the Cauchy-Schwarz inequality, and (d) the norm bound Eq. (3.9),

$$\begin{aligned}\|\boldsymbol{C}\boldsymbol{G}-\boldsymbol{C}\boldsymbol{G}'\|_{\mathrm{F}}^2 &\overset{(b)}{\leq} \sum_{t,\tau\in\boldsymbol{\pi}(\boldsymbol{U})} |\boldsymbol{M}[t,\tau]| \cdot \left|\boldsymbol{u}_\tau^\top \boldsymbol{u}_t\right| \\ &\overset{(c)}{\leq} \sum_{t,\tau\in\boldsymbol{\pi}(\boldsymbol{U})} |\boldsymbol{M}[t,\tau]| \cdot \underbrace{\|\boldsymbol{u}_\tau\|_2 \cdot \|\boldsymbol{u}_t\|_2}_{\leq 1} \\ &\overset{(d)}{\leq} \sum_{t,\tau\in\boldsymbol{\pi}(\boldsymbol{U})} |\boldsymbol{M}[t,\tau]| \ .\end{aligned}$$

Finally, by the definition of participation-calibrated sensitivity, we have

$$\mathsf{sens}(\boldsymbol{C},\Pi,m)^2 \leq \max_{\boldsymbol{G}\overset{\Pi}{\simeq}\boldsymbol{G}'} \sum_{t,\tau\in\boldsymbol{\pi}(\boldsymbol{G}-\boldsymbol{G}')} |\boldsymbol{M}[t,\tau]| = \max_{\boldsymbol{\pi}\in\Pi} \sum_{t,\tau\in\boldsymbol{\pi}} |\boldsymbol{M}[t,\tau]| \ ,$$

yielding the claimed upper bound of Part (a).

Next, we note that we can drop the absolute values in the expression of Part (a) when $\min_{t,\tau\in\boldsymbol{\pi}} \boldsymbol{M}[t,\tau] \geq 0$ for each $\boldsymbol{\pi}\in\Pi$. Thus, to show Part (b), it suffices to exhibit a $\boldsymbol{G}\simeq\boldsymbol{G}'$ such that the inequalities (b), (c) and (d) above are tight for a participation pattern $\boldsymbol{\pi}^*\in\Pi$ that maximizes the upper bound of Part (a). We fix $\boldsymbol{G}\in\mathbb{R}^{n\times m}$ arbitrary and set $\boldsymbol{G}'=\boldsymbol{G}+\boldsymbol{U}$ where the non-zero rows of $\boldsymbol{U}\in\mathbb{R}^{n\times m}$ are indexed by $\boldsymbol{\pi}^*$ and each such row is equal a fixed (arbitrary) unit vector. □

Sensitivity for Cyclic and Min-Sep Participation Lemma 3.15 provides a clear connection between the sensitivity $\mathsf{sens}(\,\cdot\,,\Pi)$ and the participation schema Π. However, its maximization over Π makes it less suitable for direct algorithmic implementation with $|\Pi|$ is large. To address this, we focus on the most practical participation schemes and mechanisms, as discussed in Section 2 and Section 3.3.3. Specifically, we consider cyclic and Min-Sep participation combined with the banded or Toeplitz/BLT mechanisms. By simplifying the sensitivity expression for these settings, we can develop efficient algorithms to compute it.

Lemma 3.16. Let $\Pi^{\mathsf{minSep}}_{b,k}$ be the participation schema as defined in Eq. (3.17) and $\Pi^{\mathsf{cyclic}}_{b,k}$ be as defined in Eq. (3.11). For a matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ such that $\forall \boldsymbol{\pi} \in \Pi,\ \min_{t,\tau\in\boldsymbol{\pi}} (\boldsymbol{C}^\top \boldsymbol{C})[t,\tau] \geq 0$, as per the assumption in Eq. (3.19) (for example, when $\boldsymbol{C}^\top \boldsymbol{C}$ is element-wise non-negative), we have the following:

(a) **Lower bound**: For any integers $b, k > 0$, we have

$$\mathsf{sens}\left(\boldsymbol{C}, \Pi^{\mathsf{minSep}}_{b,k}\right)^2 \geq \mathsf{sens}\left(\boldsymbol{C}, \Pi^{\mathsf{cyclic}}_{b,k}\right)^2 \geq \sum_{i,j=0}^{k-1} (\boldsymbol{C}^\top \boldsymbol{C})[ib, jb]\,.$$

(b) **Banded**: If $\boldsymbol{C}$ is lower-triangular and $\widetilde{b}$-banded (Definition 2.12) with $\widetilde{b} \leq b$ bands (where b is the minimum separation parameter), we have for $\Pi \in \{\Pi^{\mathsf{cyclic}}_{b,k}, \Pi^{\mathsf{minSep}}_{b,k}\}$ that

$$\mathsf{sens}\,(\boldsymbol{C}, \Pi)^2 = \max_{\boldsymbol{\pi}\in\Pi} \sum_{t\in\boldsymbol{\pi}} \|\boldsymbol{C}[:,t]\|_2^2 \ \leq k\ \|\boldsymbol{C}\|^2_{\mathsf{col}}\,. \tag{3.21}$$

(c) **Toeplitz and Monotonic**: If $\boldsymbol{C}$ is lower-triangular and Toeplitz as in Eq. (2.8) with non-negative and non-increasing entries $c_0 \geq c_1 \geq \cdots \geq c_{n-1} \geq 0$ in the first column, then the expression of (a) holds with equality throughout.

(d) **Banded and Toeplitz**: If $\boldsymbol{C}$ is banded (as in Part (b)) and Toeplitz, and b divides n, then the expression of (b) also holds with equality. Regardless of whether b divides n, we always have for $\Pi \in \{\Pi^{\mathsf{cyclic}}_{b,k}, \Pi^{\mathsf{minSep}}_{b,k}\}$ that

$$(k-1)\ \|\boldsymbol{C}\|^2_{\mathsf{col}} \leq \mathsf{sens}(\boldsymbol{C}, \Pi)^2 \leq k\ \|\boldsymbol{C}\|^2_{\mathsf{col}}\,.$$

Part (a) gives lower bounds on the sensitivity, and these bounds hold with equality with Toeplitz and monotonic $\boldsymbol{C}$ matrices (Part (c)). When the $\boldsymbol{C}$ matrix is banded, Part (b) gives us an expression that can be written as a linear program. As we shall see in Section 3.3.5, this expression can be evaluated efficiently with a dynamic programming algorithm. Finally, in the special case of Part (d), we get a closed form expression for the sensitivity. We now give elementary proofs of these

expressions.

Proof of Lemma 3.16. We prove each part in turn.

Part (a): First note that $\Pi_{b,k}^{\mathsf{cyclic}} \subset \Pi_{b,k}^{\mathsf{minSep}}$. This implies that the maximum over the former is a lower bound on the maximum over the latter in the expression of Lemma 3.15(b). That is,

$$\mathsf{sens}\left(\boldsymbol{C}, \Pi_{b,k}^{\mathsf{minSep}}\right)^2 \geq \mathsf{sens}\left(\boldsymbol{C}, \Pi_{b,k}^{\mathsf{cyclic}}\right)^2$$

yielding the first inequality. The second inequality follows from plugging in the feasible early-and-often participation pattern $\boldsymbol{\pi} = (0, b, 2b, \ldots, (k-1)b)$ from Eq. (3.18) into the expression of Lemma 3.15 for a valid lower bound.

Part (b): Lemma 3.15 again gives us

$$\mathsf{sens}(\boldsymbol{C}, \Pi) = \max_{\boldsymbol{\pi} \in \Pi} \sum_{t,\tau \in \boldsymbol{\pi}} \langle \boldsymbol{C}[:,t], \boldsymbol{C}[:,\tau] \rangle \,. \tag{3.22}$$

We make two observations:

(i) for $\widetilde{b}$-banded $\boldsymbol{C}$, we have $\langle \boldsymbol{C}[:,t], \boldsymbol{C}[:,\tau] \rangle = 0$ for all $|t-\tau| > \widetilde{b}$; and

(ii) for any b-Min-Sep participation pattern $\boldsymbol{\pi}$, we have for every $t, \tau \in \boldsymbol{\pi}$ that $t = \tau$ or $|t-\tau| > b$.

Putting these together with $\widetilde{b} < b$, we conclude that the terms corresponding to $t = \tau$ are the only possibly non-zero terms in Eq. (3.22). This gives the claimed equality.

We can get the claimed upper bound by using $|\boldsymbol{\pi}| \leq k$ (this applies for both $\Pi_{b,k}^{\mathsf{minSep}}$ and $\Pi_{b,k}^{\mathsf{cyclic}}$):

$$\max_{\boldsymbol{\pi} \in \Pi_{b,k}^{\mathsf{minSep}}} \sum_{t \in \boldsymbol{\pi}} (\boldsymbol{C}^\top \boldsymbol{C})[t,t] \;\leq\; k \cdot \max_t \left\{ (\boldsymbol{C}^\top \boldsymbol{C})[t,t] \right\} = k \;\|\boldsymbol{C}\|_{\mathsf{col}}^2$$

by the definition of $\|\cdot\|_{\mathsf{col}}$.

Part (c): The proof follows from arguing that the maximum over $\Pi_{b,k}^{\mathsf{minSep}}$ in Eq. (3.22) is attained by the early-and-often participation pattern $\boldsymbol{\pi}_\star = (0, b, 2b \ldots, (k-1)b)$, i.e., the k participations occur as early as possible. Intuitively, this is because the Toeplitz coefficients c_t's are monotonically non-increasing—we refer to Kalinin and Lampert [2024, Thm. 2] for a full proof. Since $\boldsymbol{\pi}_\star \in \Pi_{b,k}^{\mathsf{cyclic}} \subset \Pi_{b,k}^{\mathsf{minSep}}$, it follows that $\boldsymbol{\pi}_\star$ attains the maximum in Eq. (3.22) for $\Pi_{b,k}^{\mathsf{cyclic}}$ as well.

Part (d): Recall that $\boldsymbol{C}$ is a lower-triangular Toeplitz matrix. Then due to the Toeplitz structure of $\boldsymbol{C}$, we have that $\|\boldsymbol{C}[:,t]\|_2$ (for $1 \le t < n - \widetilde{b}$) is equal to $\|\boldsymbol{C}\|_{\mathsf{col}}$ for the all but the last $\widetilde{b}$ columns of $\boldsymbol{C}$. On the other hand, the last $\widetilde{b}$ columns of $\boldsymbol{C}$ can have a smaller norm. Next, we plug this into the expression of Part (b). If b divides n, then it is possible to have a $\boldsymbol{\pi}$ such that it indexes only columns with norm equal to $\|\boldsymbol{C}\|_{\mathsf{col}}$, yielding a squared sensitivity of $k\,\|\boldsymbol{C}\|_{\mathsf{col}}^2$. In general, the worst-case $\boldsymbol{\pi}$ can index at most one column whose norm is less than $\|\boldsymbol{C}\|_{\mathsf{col}}$, yielding the claimed lower bound. □

Remark 3.17 (Uniform Sensitivity Across Participation Patterns)**.** In the streaming setting, there exists an optimal strategy matrix $\boldsymbol{C}$ that is column normalized, meaning that the column norms of each column are equal (see Definition 2.10 and Lemma 2.9). The natural translation to the multiple-participation setting is for the sensitivity for all $\boldsymbol{\pi} \in \Pi$ to be the same (see the upcoming Conjecture 4.7). As in the streaming setting, Toeplitz $\boldsymbol{C}$ matrices in general do not satisfy this desiderata. However, banded $\boldsymbol{C}$ matrices can be column normalized to satisfy this property for both cyclic and Min-Sep participation schemas. We return to this in Section 4.

3.3.5 Efficient Algorithms for Multiple-Participation Sensitivity

Correctly calibrating the noise magnitude ν in the multiple-participation settings requires computing (or tightly bounding) the participation-calibrated sensitivity of Definition 3.13. We now give various algorithms to compute this sensitivity and analyze their time complexity. We

again focus on the practical settings of cyclic and Min-Sep participation combined with the banded Toeplitz and BLT mechanisms. We also make the simplifying assumption that $\boldsymbol{C}^\top \boldsymbol{C}$ is element-wise non-negative, which lets us use the dimension-independent results of Lemmata 3.15(b) and 3.16.

Brute Force Computation The simplest approach to compute the sensitivity is to evaluate Eq. (3.20) of Lemma 3.15(b) by enumerating all possible $\boldsymbol{\pi} \in \Pi$ and returning the maximum. Its time complexity is the sum of the time taken by each of its two steps:

- **Matrix Multiplication**: Computing $\boldsymbol{C}^\top \boldsymbol{C}$ takes $O(n^3)$ time in general. This can be sped up to $O(n^2 \ln(n))$ for a Toeplitz matrix $\boldsymbol{C}$ using Fast Fourier Transform (FFT). Both of these approaches are only feasible for a small number of steps n. If $\boldsymbol{C}$ is $\widetilde{b}$-banded (and possibly non-Toeplitz), then $\boldsymbol{C}^\top \boldsymbol{C}$ can be computed in $O(n\widetilde{b}^2)$ making it practical for moderate n (as long as $\widetilde{b}$ is small).

  Further improvements are possible for cyclic participation: we only need to the compute the entries $(\boldsymbol{C}^\top \boldsymbol{C})[t, \tau]$ such that $t, \tau \in \boldsymbol{\pi}$ for some pattern $\boldsymbol{\pi} \in \Pi$. Indeed, these are the only entries that appear in Lemma 3.15(b). For cyclic participation, these $\Theta(nk)$ entries can be computed $O(n^2 k)$ time (see Fig. 3.10) in general making it practical for moderate n (as long as the number of epochs k is small).

- **Exhaustive search**: Searching for the best $\boldsymbol{\pi} \in \Pi$ takes $|\Pi|$ time. This step is feasible for cyclic participation on small to moderate sized datasets, since $|\Pi_{b,k}^{\mathsf{cyclic}}| = b = N/B$, which is the number of steps in each epoch. (Recall that N is the dataset size and B is the batch size.) Unfortunately, for Min-Sep participation, we have

$$|\Pi_{b,k}^{\mathsf{minSep}}| \leq n(n-b)(n-2b)\cdots(n-(k-1)b) \leq O(n^k)\,. \quad (3.23)$$

  This is not tractable for even moderate values of n and k.

Overall, the brute force approach can be tractable only for moderately sized problems with cyclic participation.

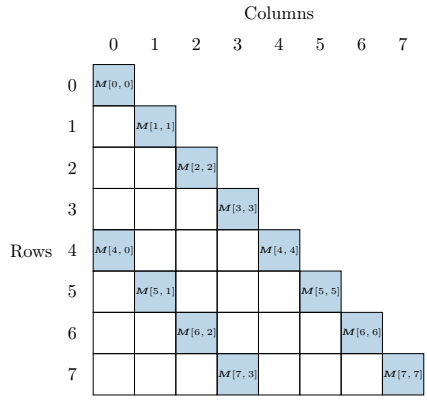


Figure 3.10: Brute-force computation of the multi-participation sensitivity cyclic participation requires us to evaluate only $\Theta(nk)$ entries of the symmetric matrix $\boldsymbol{M} = \boldsymbol{C}^\top \boldsymbol{C}$, highlighted here in blue. This figure illustrates the required entries for $n = 9, b = 4, k = 2$. The upper triangle does not have to be evaluated as the matrix $\boldsymbol{M}$ is symmetric, and has thus been omitted from this figure.

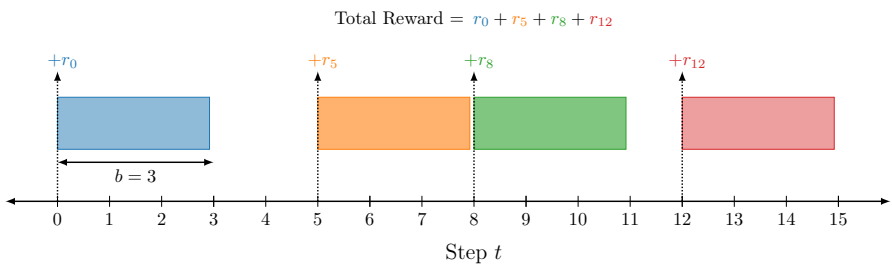


Figure 3.11: Multiple-participation sensitivity of banded matrices via dynamic programs: Given here is an illustration of Problem (3.24). The goal here is to place $k = 4$ non-overlapping blocks of width $b = 3$ over $n = 16$ steps such that we maximize the total reward, where we receive reward r_t from placing a block starting at step t. The (squared) multiple-participation sensitivity of a banded matrix $\boldsymbol{C}$ under Min-Sep participation can be solved by such a dynamic program with $r_t = \|\boldsymbol{C}[:, t]\|_2^2$, as per Lemma 3.16(b). This problem can be solved in $O(kn)$ time and space with the dynamic program of Algorithm 3.2.

Algorithm 3.2 Dynamic program for maximizing Eq. (3.24)

Input: Non-negative numbers $r_0, \dots, r_{n-1}$, separation b, number of indices k

Initialize buffer $\boldsymbol{M} \in \mathbb{R}^{n \times (k+1)}$ and set its first column $\boldsymbol{M}[:, 0] = \mathbf{0}$
▶ *$\boldsymbol{M}[t, \ell]$ will store the maximum reward from selecting l indices over steps $t, \dots, n-1$. We treat $\boldsymbol{M}[t, \ell] = 0$ for t, ℓ out of bounds.*
for $l = 1, \dots, k$ **do**
  for $t = n-1, \dots, 0$ **do**
    $R_1 = r_t + \boldsymbol{M}[t+b, \ell-1]$ ▶ *Select t as the l^{th} index*
    $R_2 = \boldsymbol{M}[t+1, \ell]$ ▶ *Ignore t for the l^{th} index*
    $\boldsymbol{M}[t, \ell] = \max\{R_1, R_2\}$

Output: $\boldsymbol{M}[0, k]$

Dynamic programming for banded matrices In the case of multiple-participation, we can compute the sensitivity exactly for a $\widetilde{b}$-banded $\boldsymbol{C}$ matrix under b-Min-Sep participation $\Pi_{b,k}^{\mathsf{minSep}}$ (where $\widetilde{b} \le b$) defined by Eq. (3.17) using a dynamic program. Consider the linear maximization problem

$$h(\boldsymbol{r}) := \max_{\boldsymbol{\pi} \in \Pi_{b,k}^{\mathsf{minSep}}} \sum_{t \in \boldsymbol{\pi}} r_t \,, \tag{3.24}$$

for some non-negative $\boldsymbol{r} = (r_0, r_1, \dots, r_{n-1})$. In particular, the expression of Lemma 3.16(b) is a special case of this problem with $r_t = \|\boldsymbol{C}[:, t]\|_2^2$.

The term $\sum_{t \in \boldsymbol{\pi}} r_t$ is known as the *reward*. Maximizing the objective of Eq. (3.24) requires us to "select" k indices $\boldsymbol{\pi} \subset [n]$ to maximize the total reward $\sum_{t \in \boldsymbol{\pi}} r_t$ such that any two distinct indices $t, \tau \in \boldsymbol{\pi}$ are at least b apart (as required by the minimum separation constraint). This is a classical scheduling problem (see Fig. 3.11), which can be efficiently solved by the textbook dynamic program of Algorithm 3.2 in $O(nk)$ time and space.

Closed-form expression for Toeplitz & monotonic matrices Consider a Toeplitz matrix $\boldsymbol{C}$ whose first column coefficients are monotonically non-increasing and non-negative as in the max-loss-optimal

Toeplitz mechanism (Section 2.3) or the BLT mechanism (Section 2.5). Lemma 3.16(c) lets us compute the sensitivity in these cases for both cyclic and Min-Sep participation. First note that every entry of $\boldsymbol{C}^\top \boldsymbol{C}$ is non-negative. Therefore, sens is independent of m and we have

$$\mathsf{sens}\left(\boldsymbol{C}, \Pi_{b,k}^{\mathsf{minSep}}\right)^2 = \sum_{i,j=0}^{k-1} (\boldsymbol{C}^\top \boldsymbol{C})[ib, jb] = \left\| \textstyle\sum_{j=0}^{k-1} \boldsymbol{C}[:, jb] \right\|_2^2 . \quad (3.25)$$

This can be evaluated in $O(nk)$ time, making it highly practical.

3.3.6 Restarted Mechanisms*

We can extend a streaming correlated noise mechanism to multiple epochs by simply restarting the mechanism at the end of each epoch. In practice, this is inferior to directly designing a multi-epoch correlated noise mechanism using the machinery in this section (see Section 3.7 for more discussion); however, restarted mechanism provides a simple way to prove bounds for multiple participation.

In particular, in the case of restarted mechanism, the privacy loss then simply composes (sequentially) over the k epochs (as per the upcoming Lemma 3.19). This requires a noise multiplier (and hence, the max loss) that is $\sqrt{k}$ times larger, when compared to the single-epoch setting. Our goal in this section is to demonstrate that the multiple-participation sensitivity can, by interpreting a restarted mechanism as a special case of cyclic participation, obtain the same bounds.

We note that general multi-participation correlated noise algorithms, such as the banded Toeplitz mechanism, are not restarted mechanisms (whose strategy matrix has a sawtooth shape as in Fig. 3.12). Thus, using the multi-participation sensitivity directly lets us design substantially better mechanisms.[11]

Mathematically, we can describe the restarted mechanism by a *tensorization* operation, as illustrated in Fig. 3.12:

[11] Restarted mechanisms can apply in some settings not covered by existing multi-participation theory, such as shuffled passes, i.e., where the dataset is shuffled at the start of the epoch. It is not known if multiple-participation sensitivity can be extended to this setting, as we discuss in Section 5.

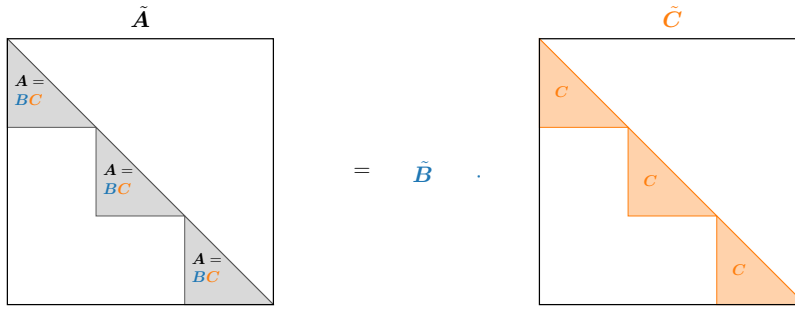


Figure 3.12: A restarted mechanism with k restarts can, as per Definition 3.18, be constructed from factorizations of the block diagonals components.

Definition 3.18 (Restarted Mechanism)**.** Let $\boldsymbol{A}_{\mathsf{pre}}$ denote a $kn \times kn$ workload matrix. The k-restarted version of a correlated noise mechanism over n steps based on the factorization $\boldsymbol{A}_{\mathsf{pre}}[:n, :n] = \boldsymbol{BC}$ of the first n rows and n columns of $\boldsymbol{A}_{\mathsf{pre}}$ is given by $\boldsymbol{A}_{kn\times kn} = \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{C}}$ with

$$\widetilde{\boldsymbol{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_{k\times k}, \quad \text{and} \quad \widetilde{\boldsymbol{B}} = \boldsymbol{A}_{\mathsf{pre}}(\boldsymbol{C}^{-1} \otimes \boldsymbol{I}_{k\times k}),$$

where $\boldsymbol{I}_{k\times k}$ is the $k \times k$ identity matrix, and "$\otimes$" denotes the Kronecker product. If $\boldsymbol{C}$ is not invertible, $\widetilde{\boldsymbol{B}}$ can be defined using a suitable pseudoinverse $\boldsymbol{C}^{\dagger}$ instead.

We can expand out the definition to see why it corresponds operationally to restarting the mechanism: the noise $\widetilde{\boldsymbol{C}}^{-1}\boldsymbol{Z}$ injected by the restarted mechanism is correlated by the matrix

$$\widetilde{\boldsymbol{C}}^{-1} = \boldsymbol{C}^{-1} \otimes \boldsymbol{I}_{k\times k} = \begin{pmatrix} \boldsymbol{C}^{-1} & & \\ & \ddots & \\ & & \boldsymbol{C}^{-1} \end{pmatrix},$$

which is simply a block-diagonal matrix. Thus, the noise injected in any given epoch is independent of the noise injected in any other epoch—this is equivalent to freezing the results of the previous epoch and restarting the mechanism.

Restarts vs. Multiple-participation Sensitivity We compare the restarted mechanisms with directly computing the multiple-participation sensitivity under cyclic participation (the case of Min-Sep participation is similar). We show through a simple calculation that the multiple-participation framework we have developed so far can recover the special case of restarted mechanisms.

The key tool to give a differential privacy guarantee for a k-restarted mechanism is composition. We recall the standard composition property of GDP:

> **Lemma 3.19** (Adaptive Composition of GDP)**.** Let $\mathcal{M}_1 : \mathcal{X}^* \to \mathcal{Y}_1$ be μ_1-GDP, and $\mathcal{M}_2 : \mathcal{X}^* \times \mathcal{Y}_1 \to \mathcal{Y}_2$ be μ_2-GDP mechanism with respect to its first argument for any fixed second argument. Then, the mechanism $\mathcal{M}(D) = (Y_1, \mathcal{M}_2(D, Y_1))$ for $Y_1 = \mathcal{M}_1(D)$ is $\sqrt{\mu_1^2 + \mu_2^2}$-GDP.

While we stated Lemma 3.19 for two mechanisms, by induction it is easy to extend to n adaptive mechanisms which have GDP parameters $\mu_0, \ldots, \mu_{n-1}$, in which case releasing all n outputs satisfies μ-GDP with $\mu = \sqrt{\sum_{t=0}^{n-1} \mu_t^2}$.

Now, returning to the restarted correlated noise mechanism, suppose we wish to obtain a μ-GDP guarantee over k epochs, where each epoch corresponds to $b = N/B$ steps with a batch size of B over N examples (see the setting in Section 3.3.1 for notation). Then, by the GDP composition property of Lemma 3.19, each epoch must satisfy $\hat{\mu}$-GDP with $\hat{\mu} = \mu/\sqrt{k}$. Thus, the seed noise $\boldsymbol{Z}$ per-epoch mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{A}(\boldsymbol{G} + \boldsymbol{C}^{-1}\boldsymbol{Z})$ has to be distributed as $\boldsymbol{Z} \sim \mathcal{N}_{b\times m}(0, \|\boldsymbol{C}\|_{\mathsf{col}}^2 / \hat{\mu}^2)$, where m is the dimension of the model space. That is, its component-wise variance is $\sigma^2_{\text{restart}} := k \|\boldsymbol{C}\|_{\mathsf{col}}^2 / \mu^2$.

On the other hand, we can view all kb steps as one run of a multi-epoch mechanism with strategy matrix $\widetilde{\boldsymbol{C}}$. Indeed, this multiple-participation correlated noise mechanism corresponds to cyclic participation schema $\Pi_{b,k}^{\mathsf{cyclic}}$ of n steps per epoch for k epochs. Thus, we can also derive privacy guarantees using the multiple-participation sensitivity of all kn steps. In this case, the component-wise variance of the seed noise $\boldsymbol{Z}$ for the correlated noise mechanism to satisfy μ-GDP is

$\sigma^2_{\mathrm{ME}} := \mathsf{sens}(\widetilde{\boldsymbol{C}}, \Pi^{\mathsf{cyclic}}_{b,k})^2/\mu^2$.

Now, it follows from the definition that both approaches turn out to be equivalent. Indeed, assuming that $\boldsymbol{C}^\top \boldsymbol{C}$ is element-wise non-negative, we have by Lemma 3.16(b) that

$$\begin{aligned}\mathsf{sens}(\widetilde{\boldsymbol{C}}, \Pi^{\mathsf{cyclic}}_{b,k})^2 &= \max_{\boldsymbol{\pi}\in\Pi^{\mathsf{cyclic}}_{b,k}} \sum_{t\in\boldsymbol{\pi}} \left\|\widetilde{\boldsymbol{C}}[:,t]\right\|_2^2 \\ &\overset{(a)}{=} \max_{l\in[b]} \sum_{j=0}^{k-1} \left\|\widetilde{\boldsymbol{C}}[:,l+jb]\right\|^2 \\ &\overset{(b)}{=} \max_{l\in[b]} k\left\|\boldsymbol{C}[:,l]\right\|_2^2 = k\cdot\|\boldsymbol{C}\|^2_{\mathsf{col}}\ ,\end{aligned}$$

where (a) follows from the definition of the schema $\Pi^{\mathsf{cyclic}}_{n,k}$ (Eq. (3.16)), while (b) follows from $\widetilde{\boldsymbol{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_{k\times k}$ (see also Fig. 3.12). Thus, we have that $\sigma^2_{\mathrm{ME}} = \sigma^2_{\mathrm{restart}}$, meaning that both approaches are equivalent.

3.4 Privacy Amplification by Sampling: A Deeper Dive

Recall also from Section 3.1 that DP-SGD with correlated noise can give much better utility than DP-SGD with independent noise in scenarios such as federated learning where amplification by sampling is not applicable. In fact, correlated noise can also be competitive with *independent noise with amplification* in some regimes as we elaborate on below.

The benefit from privacy amplification gets better as the noise multiplier σ gets larger, or equivalently, when the privacy budget ε is small[12]; see Fig. 3.1. In particular, if the noise multiplier σ is sufficiently large, the benefits of privacy amplification for DP-SGD are larger than the benefits of using correlated noise (as opposed to independent noise). That is, amplified independent noise mechanisms can outperform unamplified correlated noise mechanisms in the high-privacy regime.

However, the data processing and noise addition are independent components of the algorithm. This raises a key question:

[12]We consider amplification in the framework of (ε, δ)-DP, as it admits a tighter description of the amplified privacy guarantee than Gaussian DP.

Can correlated noise mechanisms benefit from privacy amplification by sampling (when it is feasible)?

This turns out to be technically challenging. The main reason is that the analysis of privacy amplification often relies heavily on independence of the randomness in each iteration of DP-SGD, both for the noise generation and in the sampling process. This independence lets us tightly combine the per-round privacy guarantees of DP-SGD (each step is just a Gaussian mechanism) using standard composition results such as Lemma 3.19 to get a privacy guarantee for the entire training process. Without independence, the guarantees given by composition are not valid, which prevents a straightforward amplification analysis of correlated noise under sampling. It remains an open question to give efficiently computable and near-tight privacy guarantees for general correlated noise mechanisms under amplification by sampling.[13]

We overview a simple construction that allows amplification of privacy guarantees with b-banded $\boldsymbol{C}$ matrices, i.e., $\boldsymbol{C}[t,\tau] = 0$ for all $t \geq \tau + b$ and $t < \tau$ (see Definition 2.12 of Section 2.4). This construction uses a modification of Poisson sampling that creates a Poisson subsample from a different block (where we cycle through the blocks in fixed order):

Definition 3.20 (Block-Cyclic Poisson Sampling)**.** Given a number b of blocks and a expected batch size B, we first partition the dataset D of size $|D| = N$ into b arbitrary subsets $D_0, D_1, D_2, \ldots D_{b-1}$ of equal size N/b. In iteration t, we (Poisson) sample from the dataset $D_{t \pmod b}$, with sampling probability $p' = Bb/N$, giving a batch of expected size B.

We can now give an amplified privacy guarantee using this sampling process for banded $\boldsymbol{C}$ matrices:

[13] Here, we only tackle the problem of giving an amplified privacy guarantee for a given strategy matrix $\boldsymbol{C}$. Ultimately, we wish to co-design the strategy matrix and the sampling scheme to attain the best utility at any privacy level; we get back to this in Section 5.

Theorem 3.21 (Informal)**.** Consider the block-cyclic Poisson sampling strategy with b blocks and an expected batch size B and suppose the gradients satisfy the norm bound of Eq. (3.9). In this setting, the correlated noise mechanism $\mathcal{M}(\boldsymbol{G}) = \boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}$ with $\boldsymbol{Z} \sim \mathcal{N}_{n\times m}(0, \nu^2)$ for some variance $\nu^2 > 0$ satisfies any privacy guarantee satisfied by independent noise DP-SGD (i.e. Algorithm 3.1 with $\boldsymbol{C}$ as the identity matrix) using $n' = n/b$ steps, sampling probability $p' = Bb/N$ (i.e., so that batches are formed with an expected size of B from a dataset of size N/b), and noise standard deviation $\nu' = \nu / \|\boldsymbol{C}\|_{\mathsf{col}}$.

Proof Sketch. Without loss of generality we can assume that adjacent datasets D and D' differ in the example x_0 which is in D_0. Since the gradient $\boldsymbol{g}_0$ is drawn from dataset D_0, it could differ between D and D'. However, since the matrix $\boldsymbol{C}$ is b-banded, changing $\boldsymbol{g}_0$ only affects rows $0, 1, \ldots, b-1$ of the output $\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z}$, and $\boldsymbol{g}_b, \boldsymbol{g}_{2b}, \ldots$ do not affect these rows. Similarly, the gradient $\boldsymbol{g}_b$ is also computed on a subsample from D_0 and could differ when we move to an adjacent dataset. This only affects rows $b, b+1, \ldots, 2b-1$ of $\boldsymbol{C}\boldsymbol{G}$, and $\boldsymbol{g}_0, \boldsymbol{g}_{2b}, \boldsymbol{g}_{3b}, \ldots$ do not affect these rows.

In other words, each batch drawn from D_0 affects a disjoint part of the output: the j^{th} batch can only affect rows $R_j := \{jb, \ldots, (j+1)b-1\}$ for a total of $n' = n/b$ such groups. Then the mechanism $\mathcal{M}_j(\boldsymbol{G}) := (\boldsymbol{C}\boldsymbol{G} + \boldsymbol{Z})[R_j]$ generating corresponding rows of the $\mathcal{M}(\boldsymbol{G})$ can be viewed as one run of the Gaussian mechanism with subsampling. In particular, the ℓ_2-sensitivity (assuming $\boldsymbol{G}'$ is generated as above) is

$$\left\|(\boldsymbol{C}(\boldsymbol{G} - \boldsymbol{G}'))[R_j, :]\right\|_{\mathrm{F}} = \left\|\boldsymbol{C}[R_j, 0](\boldsymbol{g}_0 - \boldsymbol{g}_0')^\top\right\|_{\mathrm{F}} \leq \|\boldsymbol{C}\|_{\mathsf{col}} .$$

This would satisfy the same privacy guarantee as a mechanism with unit ℓ_2-sensitivity, but with noise multiplier $\sigma = \nu / \|\boldsymbol{C}\|_{\mathsf{col}}$. Moreover, $\boldsymbol{g}_0$ is generated by sampling an expected B elements from $|D_0| = N/b$ choices; so the sampling probability is $p' = B/(N/b) = Bb/N$. Thus, $\mathcal{M}_j$ satisfies the same privacy guarantee as DP-SGD (whose ℓ_2-sensitivity is 1) with a noise multiplier σ and sampling probability p'. Finally, the complete mechanism $\mathcal{M}$ is an adaptive composition of $\mathcal{M}_0, \ldots, \mathcal{M}_{n'-1}$,

meaning that the original correlated noise mechanism $\mathcal{M}$ satisfies the same privacy guarantee as n' steps of DP-SGD. □

3.5 Learning Guarantees for Correlated Noise Mechanisms*

In this section, we will focus on the learning theoretic guarantees enjoyed by correlated noise mechanisms, and how they compare to independent noise mechanisms (i.e., DP-SGD or differentially private gradient descent). Note that the full batch noisy gradient descent (DP-GD) achieves optimal accuracy-privacy trade-off for empirical risk minimization, something that can also be achieved using a proper instantiation of DP-SGD (for example, with appropriate choice of minibatching and subsampling) for both stochastic convex optimization and empirical risk minimization. The benefit of using DP-SGD over DP-GD is that the former has significantly less overhead in terms of computational time.

The main message in this section is that, while we can design optimal algorithms for stochastic convex optimization (SCO) using multiple pass over the data, it is not known that it is possible in the single-epoch setting, i.e., with a single pass over the data. On the other hand, correlated noise mechanisms are known to be the best in terms of the trade-offs between privacy, utility, and computation time in the single-pass setting. In the following, we make this claim more precise, surveying generic bounds for stochastic convex problems, and more fine-grained bounds for specific problems like linear regression.

3.5.1 Generic Learning Bounds

Stochastic convex optimization with differential privacy has been studied in a variety of settings, we focus on the case of Lipschitz functions and convexity to illustrate the effect of introducing the noise correlations. Specifically, we assume $\Theta \subseteq \mathbb{R}^m$ is a convex set (where m is the dimensionality of the model space) and:

(a) **Convex**: The loss function $\ell(\cdot\,, \boldsymbol{x})$ is convex in its first parameter for all $\boldsymbol{x}$.

(b) **Lipschitz**: The loss function $\ell(\cdot\,, \boldsymbol{x})$ is 1-Lipschitz (in the Euclidean norm). That is, for all pairs $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$ and all $\boldsymbol{x} \in \mathcal{X}$, we

have

$$|\ell(\boldsymbol{\theta}_1, \boldsymbol{x}) - \ell(\boldsymbol{\theta}_2, \boldsymbol{x})| \leq \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 .$$

The first mechanism to exploit the correlated noise mechanism was the *DP-follow-the-regularized-leader* (DP-FTRL), which we collectively refer as correlated noise DP-SGD. We refer the readers to Section 1.10.2 for details. The analysis of correlated noise DP-SGD goes via the standard regret analysis of online learning, while accounting for the additional noise added due to privacy. One can show the following results for the algorithm $\mathcal{A}_{\texttt{cor-noise}}$ defined in Section 1.10.2:

Theorem 3.22. Given n data samples, $\mathcal{A}_{\texttt{DP-FTRL}}$ is (ε, δ)-DP and outputs a $\theta \in \mathbb{R}^m$ such that

- In the general setting, we have

$$R(\mathcal{A}_{\texttt{cor-noise}}) = \widetilde{O}_\delta\left(\frac{m^{1/4}}{\sqrt{\varepsilon n}}\right),$$

  where $\widetilde{O}_\delta(\cdot)$ hides polylog factors in n and δ.

- In the *realizable* setting, i.e., when $\min_{\theta \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \tau}\left[\ell(\boldsymbol{\theta}; \boldsymbol{x})\right] = 0$, the SCO guarantee can be improved to

$$R(\mathcal{A}_{\texttt{cor-noise}}) = \widetilde{\Theta}_\delta\left(\frac{1}{\sqrt{n}} + \frac{\sqrt{m}}{\varepsilon n}\right).$$

3.5.2 Detailed Learning Bounds on Specific Problems

We now review some bounds demonstrating that correlated noise can help Algorithm 1.3 attain a better objective value than independent noise in the streaming setting with an infinite clipping norm of $\zeta = \infty$. That is, the correlated noise mechanism receives as input the unclipped gradient $\boldsymbol{g}_t = \nabla \ell(\boldsymbol{\theta}_t, \boldsymbol{x}_t)$. Hence the norm, $\|\boldsymbol{g}_t\|_2$, and, thus the sensitivity, can potentially be unbounded, and so this algorithm does not satisfy differential privacy.[14] Still, studying the suboptimality bounds on the

[14] Clipping can also impact the optimization dynamics of the learning algorithm; the bibliographic notes of Section 3.7 provides some pointers.

resulting algorithms with noise calibrated to a desired privacy level under the *false* assumption that $\|\boldsymbol{g}_2\|_2 \leq 1$ sheds light on the precise effect of *introducing noise correlations* into the learning process (while avoiding technicalities due to clipping), and how it subtly differs from the prefix sum estimation.

Linear Regression

One of the simplest learning problems is linear regression. We aim to predict a target $y \in \mathbb{R}$ from input $\boldsymbol{x} \in \mathbb{R}^m$ by solving

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \ \tfrac{1}{2}\, \mathbb{E}_{(\boldsymbol{x},y)\sim \mathbb{P}_{\mathsf{data}}} (\boldsymbol{x}^\top \boldsymbol{\theta} - y)^2 \,. \tag{3.26}$$

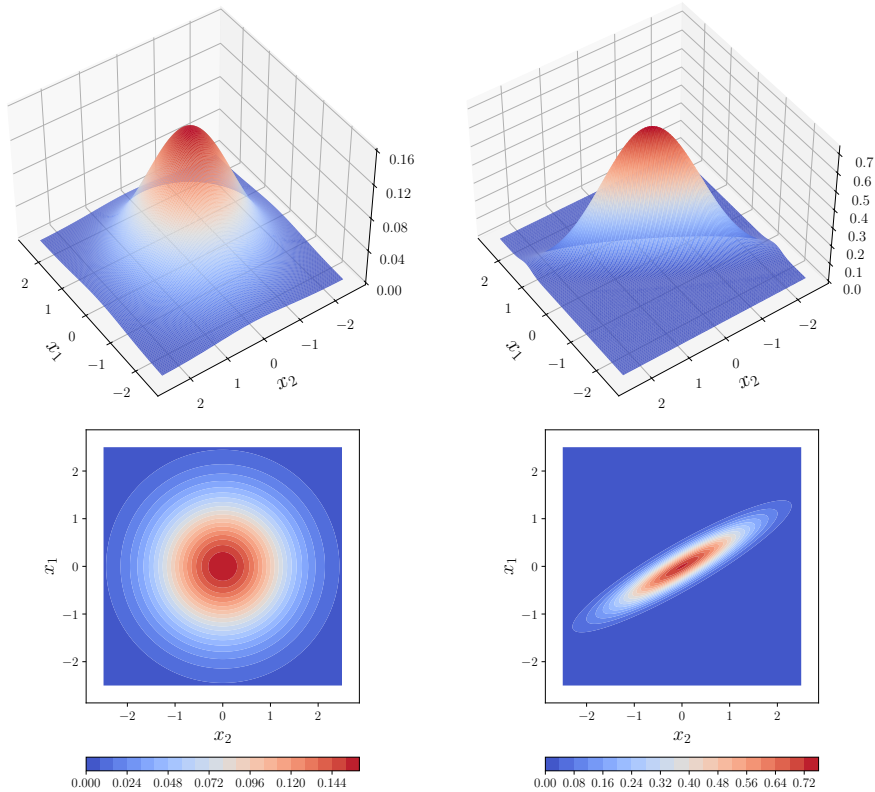
This is an instantiation of the learning problem of Section 1.2 where $\ell(\boldsymbol{\theta}, (\boldsymbol{x}, y)) = (1/2)(\boldsymbol{x}^\top \boldsymbol{\theta} - y)^2$ is the mean squared loss. This multiplicative constant $1/2$ in the squared loss ensures that the input covariance matrix $\boldsymbol{H} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] \in \mathbb{R}^{m\times m}$ is also the Hessian matrix of the objective of Eq. (3.26).

We make several simplifications to illustrate the effect of the noise correlations. First, we assume that input is Gaussian vector with full-rank covariance matrix $\boldsymbol{H}$, i.e., $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$;[15] second, we assume a realizable model, i.e., the average loss at the global minimizer $\boldsymbol{\theta}^\star$ is zero. This can only happen if $\ell(\boldsymbol{\theta}^\star, (\boldsymbol{x}, y)) = (1/2)(\boldsymbol{x}^\top \boldsymbol{\theta}^\star - y)^2 = 0$ almost surely for $(\boldsymbol{x}, y) \sim \mathbb{P}_{\mathsf{data}}$. In other words, with probability one, we have that $(\boldsymbol{x}, y) \sim \mathbb{P}_{\mathsf{data}}$ satisfies $y = \boldsymbol{x}^\top \boldsymbol{\theta}^\star$.[16] Finally, as we discussed above, we omit gradient clipping, so the correlated noise mechanisms receives as input the unclipped gradient $\boldsymbol{g}_t = \nabla \ell(\boldsymbol{\theta}_t, (\boldsymbol{x}_t, y_t))$. We calibrate the noise to a desired privacy level under the *false* assumption that $\|\boldsymbol{g}_2\|_2 \leq 1$.

The bounds depend on the ratio $m_{\mathsf{eff}} = \mathsf{Tr}(\boldsymbol{H})/\|\boldsymbol{H}\|_{2\to 2}$ of the trace of the $m \times m$ input covariance matrix $\boldsymbol{H}$ to its spectral norm; this is also

[15] The notation $\mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ denote the multivariate Gaussian with $\boldsymbol{0}$-mean and covariance $\boldsymbol{H} \in \mathbb{R}^{m\times m}$. Its probability density function is $(2\pi)^{-m/2} \det(\boldsymbol{H})^{-1/2} \exp\left(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}^{-1} \boldsymbol{x}\right)$

[16] The realizability assumption is easily lifted by assuming instead that $y = \boldsymbol{x}^\top \boldsymbol{\theta}^\star + \xi$ for i.i.d. Gaussian noise $\xi \sim \mathcal{N}(0, s^2)$. This yields an extra additive term that is independent of the DP noise in each of the bounds we give below.



(a) Gaussian with high effective dimension.

(b) Gaussian with low effective dimension.

Figure 3.13: The densities of two Gaussian distributions in $\mathbb{R}^2$ and their effective dimensions m_{eff}. The left plot depicts an isotropic Gaussian, meaning that its covariance matrix has equal eigenvalues. Its effective dimension is then $m_{\mathsf{eff}} = 2 = m$. The right plot shows a nearly low rank Gaussian, whose covariance matrix has eigenvalues $(1.2, 0.04)$. Its effective dimension $m_{\mathsf{eff}} \approx 1.03$ is strictly smaller than the ambient dimension $m = 2$.

known as its *effective dimension*. We have the alternative expression in terms of the eigenvalues $\lambda_1, \ldots, \lambda_m > 0$ of $\boldsymbol{H}$:

$$m_{\mathsf{eff}} = \frac{\sum_{i=1}^{m} \lambda_i}{\max_{i=1,\ldots,m} \lambda_i} .$$

As illustrated in Fig. 3.13, we have $1 \leq m_{\mathsf{eff}} \leq m$, with a smaller m_{eff} for approximately low rank problems, and equal to the ambient dimension m when all the eigenvalues of $\boldsymbol{H}$ are equal. It is desirable for the error of numerical algorithms to scale with effective dimension m_{eff} of the problem, rather than the ambient dimension m, which can be significantly larger. In particular, it is quite common for over-parameterized problems on real-world data to exhibit an effective dimension that is much smaller than the ambient dimension.

The advantage of correlated noise can be seen as follows. Suppose, without loss of generality that $\|\boldsymbol{H}\|_{2\to 2} = 1$. We have the following bounds in the asymptotic $n \to \infty$ regime in terms of the learning rate $0 < \eta < 1$ (assumed small enough), the GDP parameter μ (so the component-wise variance of the injected noise scales as $1/\mu^2$), and the effective dimension m_{eff}:

- Using independent noise, we have the matching upper and lower bounds on the excess population risk (3.2) (up to absolute constants):

$$\lim_{n\to\infty} R(\mathcal{M}_{\mathsf{indep},n}) = \Theta\left(\frac{\eta m}{\mu^2}\right) ,$$

  where $\mathcal{M}_{\mathsf{indep},n}$ is the output of n steps of DP-SGD where the gradients are perturbed with independent noise.

- Given a parameter $\nu > 0$, consider a variant of the max-loss-optimal Toeplitz mechanism (Section 2.3) where the first column $c_0, c_1, \ldots$ of the strategy matrix $\boldsymbol{C}$ is given by

$$c_t = (-1)^t \binom{-1/2}{t} (1-\nu)^t . \tag{3.27}$$

  This coincides with the Toeplitz coefficients in Eq. (2.10) up to a factor of $(1-\nu)^t$. (For intuition on why we have this extra factor, see the paragraph on "Interpretation" below.) The mechanism

$\mathcal{M}_{\mathsf{Toep},n}$ that uses the correlated noise mechanism over n steps with this strategy matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ satisfies the excess risk bound

$$\lim_{n\to\infty} R(\mathcal{M}_{\mathsf{Toep},n}) \leq O\left(\frac{\eta^2 m_{\mathsf{eff}}}{\mu^2} \ln^2 \frac{1}{\nu}\right)$$

for all $0 < \nu \leq \eta\lambda_{\min}(\boldsymbol{H})$, where $\lambda_{\min}(\boldsymbol{H})$ denotes the smallest eigenvalue of the covariance $\boldsymbol{H}$.

- For any correlated noise mechanism $\mathcal{M}_n$ over n steps with a Toeplitz strategy matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, we have the lower bound on the excess risk:

$$\inf_{\mathcal{M}} \left\{\lim_{n\to\infty} R(\mathcal{M}_n)\right\} \geq \Omega\left(\frac{\eta^2 m_{\mathsf{eff}}}{\mu^2}\right).$$

Interestingly, using independent noise in this case is strictly suboptimal, while the use of correlated noise is (almost) optimal: it matches the lower bound up to log factors. The bound for correlated noise is better than that of using independent noise by a problem-dependent factor that can be as large as $m/\ln m$. The gap is significant when the covariance $\boldsymbol{H}$ is approximately low rank. This is observed empirically in, e.g., overparameterized models where the features are highly correlated.

Interpretation The improvement from the ambient dimension m to the effective dimension m_{eff} sheds some light on the role played by the correlated noise in learning problems. The gradient $\boldsymbol{g}_t$ aims to counteract the effect of the noise added in previous iterations and move the trajectory back towards the minimizer $\boldsymbol{\theta}^\star$.

Unfortunately, the gradients cannot effectively cancel the noise along eigenvectors of the covariance $\boldsymbol{H}$ with small eigenvalues (i.e., low signal directions). This leads to accumulation of (independent) noise across iterations. However, this can be remedied by partial cancellation of this noise due to *anti-correlations*, as illustrated in Fig. 1.7 of Section 1.

In particular, the suboptimality $R(\mathcal{M})$ decomposes along the directions of the eigenvectors of the input covariance $\boldsymbol{H}$ as follows. Let λ_j and $\boldsymbol{v}_j$ for $j = 1, \ldots, m$ denote the eigenvalues and eigenvectors of $\boldsymbol{H}$.

Then, we can decompose the suboptimality as

$$R(\mathcal{M}) = \sum_{j=1}^{m} R_j(\mathcal{M}) \,,$$

where R_j is the suboptimality incurred in the direction given by $\boldsymbol{v}_j$, i.e., this depends on λ_j alone and not on $\{\lambda_i\}_{i \neq j}$. It can be shown that the contribution of each direction is the same if using independent noise:

$$\lim_{n\to\infty} R_j(\mathcal{M}_{\mathsf{indep},n}) = \Theta(1) \,.$$

This also holds for low signal directions with λ_j small. On the other hand, with correlated noise as in Eq. (3.27), we get that the contribution of a direction j scales with the eigenvalue λ_j:

$$\lim_{n\to\infty} R_j(\mathcal{M}_{\mathsf{Toep},n}) \leq \widetilde{O}(\lambda_j) \,,$$

where $\widetilde{O}(\cdot)$ hides constants and log factors of problem-dependent constants. That is, the contribution of lower signal directions j with small eigenvalue λ_j reduces proportionally due to the noise cancellation effect.

Finally, since the gradients of a strongly convex objective function can already provide some noise cancellation effect, the max-loss-optimal Toeplitz mechanism of Eq. (3.27) requires less aggressive anti-correlations than in the prefix sum estimation of Eq. (2.10). This explains the a damping factor of $(1-\nu)^t$ in the former.

In summary, anti-correlated DP noise can prevent noise accumulation by repeatedly cancelling out a part of the noise in low signal directions.

3.6 Proofs of Technical Results*

We give here proofs of technical results of Lemma 3.10 and Proposition 3.11.

Proof of Lemma 3.10. The claim follows from plugging in the inequality

$$\|\boldsymbol{u}\|_\infty \leq \|\boldsymbol{u}\|_1 \leq n \, \|\boldsymbol{u}\|_\infty \quad \text{for all } \boldsymbol{u} \in \mathbb{R}^n \,,$$

into the definition of the operator norm (Eq. (3.14)) and noting that $\|\cdot\|_{\mathsf{col}} = \|\cdot\|_{1\to 2}$. In particular, let $\boldsymbol{u}_1$ be such that

$$\forall \boldsymbol{u} \in \mathbb{R}^n, \quad \frac{\|\boldsymbol{C}\boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_1} \leq \frac{\|\boldsymbol{C}\boldsymbol{u}_1\|_2}{\|\boldsymbol{u}_1\|_1} \tag{3.28}$$

and $\boldsymbol{u}_\infty$ be such that

$$\forall \boldsymbol{u} \in \mathbb{R}^n, \quad \frac{\|\boldsymbol{C}\boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_\infty} \leq \frac{\|\boldsymbol{C}\boldsymbol{u}_\infty\|_2}{\|\boldsymbol{u}_\infty\|_\infty} \tag{3.29}$$

Setting $\boldsymbol{u} := \boldsymbol{u}_\infty$ in Eq. (3.28), we get

$$\frac{1}{n}\|\boldsymbol{C}\|_{\infty\to 2} = \frac{\|\boldsymbol{C}\boldsymbol{u}_\infty\|_2}{n\|\boldsymbol{u}_\infty\|_\infty} \leq \frac{\|\boldsymbol{C}\boldsymbol{u}_\infty\|_2}{\|\boldsymbol{u}_\infty\|_1} \leq \frac{\|\boldsymbol{C}\boldsymbol{u}_1\|_2}{\|\boldsymbol{u}_1\|_1} = \|\boldsymbol{C}\|_{1\to 2} = \|\boldsymbol{C}\|_{\mathsf{col}}\,.$$

The other inequality follows similarly. □

Proof of Proposition 3.11. The crux of the proof is to establish

$$f(\boldsymbol{C}) := \left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{\mathsf{row}} \|\boldsymbol{C}\|_{\infty\to 2} \geq \|\boldsymbol{A}_{\mathsf{pre}}\|_{\infty\to\infty}\,. \tag{3.30}$$

We will prove this later. Assuming that it holds for now, the next step is to note that $\|\boldsymbol{M}\|_{\infty\to\infty} = \max_{t\in[n]} \|\boldsymbol{M}[t,:]\|_1$ is the maximum absolute row sum of the matrix $\boldsymbol{M}$. For the matrix $\boldsymbol{M} = \boldsymbol{A}_{\mathsf{pre}}$, this is attained by the last row and we have $\|\boldsymbol{A}_{\mathsf{pre}}\|_{\infty\to\infty} = n$. Thus, we have, $f(\boldsymbol{C}) \geq n$.

The next step of our proof is to establish that the identity matrix $\boldsymbol{C} = \boldsymbol{I}_{n\times n}$ achieves the lower bound with equality. First, $\|\boldsymbol{A}_{\mathsf{pre}}\|_{\mathsf{row}} = \sqrt{n}$ is the largest row norm of $\boldsymbol{A}_{\mathsf{pre}}$, which is attained by the last row of $\boldsymbol{A}_{\mathsf{pre}}$. Second, we have that $\|\boldsymbol{I}\|_{\infty\to 2} = \sqrt{n}$, because of the matching upper and lower bounds

$$\|\boldsymbol{I}\|_{\infty\to 2} = \max_{\boldsymbol{u}\neq\boldsymbol{0}} \|\boldsymbol{u}\|_2 / \|\boldsymbol{u}\|_\infty \leq \sqrt{n} \quad \text{and}$$
$$\|\boldsymbol{I}\|_{\infty\to 2} \geq \|\boldsymbol{1}_n\|_2 / \|\boldsymbol{1}_n\|_\infty = \sqrt{n}\,.$$

Here, the upper bound is based on the vector norm inequality $\|\boldsymbol{u}\|_2 \leq \sqrt{n}\|\boldsymbol{u}\|_\infty$ and we plugged in $\boldsymbol{1}_n := (1,\ldots,1) \in \mathbb{R}^n$ for the lower bound. Thus, we have that

$$f(\boldsymbol{I}) = \|\boldsymbol{A}_{\mathsf{pre}}\|_{\mathsf{row}} \|\boldsymbol{I}\|_{\infty\to 2} = n\,,$$

establishing the required claim $\min_{\boldsymbol{C}} f(\boldsymbol{C}) \geq n = f(\boldsymbol{I})$.

Proving Eq. (3.30) To complete the proof, we have to show Eq. (3.30). Fix a vector $\boldsymbol{u}$ such that $\boldsymbol{C}\boldsymbol{u} \neq \boldsymbol{0}$. Noting that the max row norm can be written as an induced matrix norm, we have

$$\left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{\mathsf{row}} = \left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{2\to\infty} \geq \frac{\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}(\boldsymbol{C}\boldsymbol{u})\|_\infty}{\|\boldsymbol{C}\boldsymbol{u}\|_2} = \frac{\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{u}\|_\infty}{\|\boldsymbol{C}\boldsymbol{u}\|_2}\,.$$

Rearranging (and noting that it holds trivially for $\boldsymbol{u}$ such that $\boldsymbol{C}\boldsymbol{u} = \boldsymbol{0}$), we get the following inequality that holds for all $\boldsymbol{u} \in \mathbb{R}^n$:

$$\left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{\mathsf{row}} \|\boldsymbol{C}\boldsymbol{u}\|_2 \geq \|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{u}\|_\infty \,. \tag{3.31}$$

Fix a vector $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_\infty \leq 1$. Continuing on with the definition of $\|\cdot\|_{\infty\to 2}$, we get

$$\begin{aligned} f(\boldsymbol{C}) &= \left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{\mathsf{row}} \left(\max_{\|\boldsymbol{v}\|_\infty \leq 1} \|\boldsymbol{C}\boldsymbol{v}\|_2\right) \\ &\geq \left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}\right\|_{\mathsf{row}} \|\boldsymbol{C}\boldsymbol{u}\|_2 \\ &\geq \|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{u}\|_\infty \,, \end{aligned}$$

where the last inequality followed from Eq. (3.31). Since this inequality holds for all $\boldsymbol{u}$ such that $\|\boldsymbol{u}\|_\infty \leq 1$, we can take the largest lower bound to get

$$f(\boldsymbol{C}) \geq \max_{\|\boldsymbol{u}\|_\infty \leq 1} \|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{u}\|_\infty =: \|\boldsymbol{A}_{\mathsf{pre}}\|_{\infty\to\infty} \,.$$

This establishes Eq. (3.30). □

3.7 Bibliographic Notes

Motivation for Correlated Noise in AI The challenges in proper subsampling and its impact on privacy accounting is covered recently by Annamalai, Balle, De Cristofaro, and Hayes [2024], Chua, Ghazi, Kamath, Kumar, Manurangsi, Sinha, and Zhang [2024a,b]. For various aspects of federated learning and privacy, refer to the monograph by Kairouz et al. [2021b]. For a survey on continual learning problems where user behavior can shift over time, see Wang, Zhang, Su, and Zhu [2024]. The observation of distribution drift due to feedback mechanisms where users strategically adapt their behavior in response to a deployed system has been studied by Perdomo, Zrnic, Mendler-Dünner, and Hardt [2020].

Multi-Epoch Sensitivity Much of the material on multi-epoch correlated noise mechanisms in Section 3.3 is due to Choquette-Choo, McMahan, Rush, and Thakurta [2023b], Choquette-Choo, Ganesh,

McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a]. We give precise attributions for each part of the preceding section below.

Challenges with Multi-Epoch Sensitivity Recall from Section 3.3.2 that the multi-epoch sensitivity is equal to $\|\boldsymbol{C}\|_{\infty\to 2}$ in the absence of meaningful restrictions on the participation patterns. Tropp [2003, Sec. 4.3] showed that computing the $\|\cdot\|_{\infty\to 2}$ induced matrix norm is NP-hard, and Steinberg [2005] showed that any general $\|\cdot\|_{p\to q}$ norm for $p > q$ is NP-hard to compute.

Practical Participation Patterns The cyclical participation schema was introduced in Choquette-Choo, McMahan, Rush, and Thakurta [2023b] under the name "(k, b)-participation" while the minimum separation participation schema was introduced in Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a] as its relaxation suitable to federated learning.

Multi-Epoch Sensitivity: Bounds The sensitivity bounds of Section 3.3.4 can be attributed as follows:

- Lemma 3.15: Part (a) is due to Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a, Prop. E.1], while (b) is taken from Choquette-Choo, McMahan, Rush, and Thakurta [2023b, Eq. 3, Cor. 2.1].
- Lemma 3.16: Part (a) is elementary; its second inequality has been noted, for instance in McMahan, Xu, and Zhang [2024, Remark 3.2]. Part (b) previously appeared in Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a, Thm. 2]. Finally, Part (c) was established in Kalinin and Lampert [2024, Thm. 2], while Part (d) is a direct corollary of (b) and (d).

Multi-Epoch Sensitivity: Algorithms Next, we move on to the algorithms presented in Section 3.3.5. The brute force computation of sensitivity for cyclic participation was proposed in Choquette-Choo, McMahan, Rush, and Thakurta [2023b], while the dynamic program

for banded matrices was used by Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a]. The closed-form expression was, as we previously noted, established in Kalinin and Lampert [2024] and was used later, for instance, in McMahan, Xu, and Zhang [2024]. Recently, Kalinin, McKenna, Upadhyay, and Lampert [2025] introduced a new explicit factorization method, *Banded Inverse Square Root*, which imposes a banded structure on the inverse correlation matrix.

Finally, the tensorization process of restarted mechanisms in Section 3.3.6 was called as "Stamping" in [Choquette-Choo et al., 2023b, App. D.4]; restarting mechanisms is a commonly used trick, particularly for baselines.

Amplification by Sampling for Correlated Noise Mechanisms The amplification result of Theorem 3.21 is due to Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a]. Unfortunately, this resulting has two drawbacks: it is applicable only to banded matrices, and the result guarantees can be worse than the unamplified ones at large ε. The follow-up work of Choquette-Choo, Ganesh, Steinke, and Thakurta [2024] lifted the first drawback (but in the setting of Poisson sampling), giving a procedure that gives amplified guarantees for non-banded correlated noise mechanisms. Choquette-Choo, Ganesh, Haque, Steinke, and Thakurta [2024b] consider a different sampling scheme which allows tight guarantees through Monte Carlo sampling.

Privacy Preserving Learning Differentially private empirical risk minimization (DP-ERM) and DP stochastic convex optimization (DP-SCO) [Bassily, Smith, and Thakurta, 2014a, Bassily, Feldman, Talwar, and Thakurta, 2019, Feldman, Koren, and Talwar, 2020, Bassily, Feldman, Guzmán, and Talwar, 2020, Asi, Feldman, Koren, and Talwar, 2021a, Bassily, Guzmán, and Nandi, 2021, Zhang, Tran, and Cutkosky, 2022a, Asi, Levy, and Duchi, 2021b, Kulkarni, Lee, and Liu, 2021, Gopi, Lee, and Liu, 2022, Chaudhuri, Monteleoni, and Sarwate, 2011, Kifer, Smith, and Thakurta, 2012] are probaly the most studied problems in the theoretical DP literature. This body of work captures the optimal privacy/utility trade-offs for a large class of convex optimzation

problems. A comprehensive survey article for various methods used in practice to instantiate DP-SGD (and detailed best practices) can be found in Ponomareva, Hazimeh, Kurakin, Xu, Denison, McMahan, Vassilvitskii, Chien, and Thakurta [2023], including how one instantiate subsampling to achieve better accuracy guarantee.

Theoretical Analysis of Learning with Correlated Noise Mechanisms The generic bounds of Section 3.5.1 were shown by Kairouz, McMahan, Song, Thakkar, Thakurta, and Xu [2021a] for the non-realizable (agnostic) setting, while the one in the realizable setting was shown by Asi, Feldman, Koren, and Talwar [2023]. The detailed bounds of Section 3.5.2 were shown by Choquette-Choo, Dvijotham, Pillutla, Ganesh, Steinke, and Thakurta [2024a]. Finally, the currently best known rate for single-pass algorithms for stochastic convex optimization is in Choquette-Choo, Ganesh, and Thakurta [2024]. We refer to these original works for further details and proofs.

The analysis of Section 3.5.2 is based on Algorithm 1.3 without clipping. However, this is only an approximation as clipping does impact the optimization dynamics, see Zhang, He, Sra, and Jadbabaie [2020], Chen, Wu, and Hong [2020], Zhang, Chen, Hong, Wu, and Yi [2022b], Xiao, Xiang, Wang, and Devadas [2023a], Koloskova, Hendrikx, and Stich [2023a], Schaipp, Garrigos, Simsekli, and Gower [2024], Marshall, Xiao, Agarwala, and Paquette [2024] and the references therein. Using anti-correlated DP noise to prevent noise accumulation by repeatedly cancelling out a part of the noise in low signal directions is discussed in more details in Choquette-Choo, Dvijotham, Pillutla, Ganesh, Steinke, and Thakurta [2024a, Remark C.16].

4

Implementation Details and Practical Recommendations

We now turn to practical considerations in implementing correlated noise mechanisms. In particular, we discuss how to solve the optimization problems involved in constructing correlated noise mechanisms for the streaming setting (Section 2) and the multiple-participation setting (Section 3). We also discuss the nuances involved in efficient noise generation, including in distributed environments.

While our discussions apply more broadly to correlated noise mechanisms based on any lower triangular and Toeplitz workload matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, we use the prefix sum workload matrix, encountered in stochastic gradient descent (SGD; see Section 1.2), as a concrete example:

$$\boldsymbol{A}_{\mathsf{pre}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \in \{0,1\}^{n \times n}. \tag{4.1}$$

The main focus of this section is two fold. First, we discuss numerical optimization techniques to find a factorization $\boldsymbol{A}_{\mathsf{pre}} = \boldsymbol{B}\boldsymbol{C}$ with lower-triangular factors $\boldsymbol{B}, \boldsymbol{C}$. In particular, we setup the problem in Section 4.1, focusing on specific methods for the optimizing the

dense mechanism in Section 4.2 and parameterized mechanism (e.g. BLT) in Section 4.3. In both cases, we handle the streaming and multi-participation settings. The second objective of this section is to provide practical recommendations on the design choices in correlated noise mechanisms (Section 4.4).

4.1 Mechanism Optimization Using Numerical Methods

Recall that we considered two types of mechanisms in Section 2: (a) those that involved numerically optimizing the strategy matrix $\boldsymbol{C}$, such as the dense mechanism of Section 2.2 or the Buffered Linear Toeplitz (BLT) mechanism of Section 2.5, and (b) mechanisms with handcrafted matrices, such as the max-loss-optimal Toeplitz mechanism of Theorem 2.5 (Section 2.3).

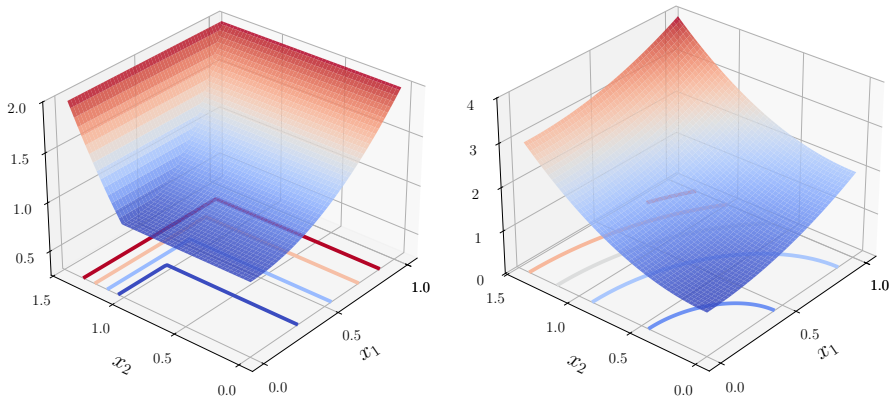
While numerical optimization of the mechanisms requires additional up-front computation cost (compared to the handcrafted mechanisms), they have a key practical advantage: they can be configured for a wider variety of settings, particularly varied participation patterns in the multiple-participation setting. Such settings are particularly relevant in the learning setting, as we discussed in Section 3.

Setting Our goal is find a lower triangular strategy matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$, such that some loss induced by the factors $\boldsymbol{B} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}$ and $\boldsymbol{C}$ is minimized. While we focused primarily on the max loss (Definition 2.3) in Sections 1 and 2, it is customary to use the root-mean square loss (RMS-loss) as an objective for numerical optimization (Definition 2.23). For the streaming setting, we showed in Theorem 2.22 that:

$$\bar{\mathcal{L}}_2(\boldsymbol{C}) := \bar{\mathcal{L}}_2(\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}, \boldsymbol{C}) = \frac{1}{\sqrt{n}} \left\| \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1} \right\|_{\mathrm{F}} \|\boldsymbol{C}\|_{\mathsf{col}} \,, \tag{4.2}$$

where we take $\boldsymbol{B} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}$ if not specified otherwise. In the multiple-participation setting, the term $\|\boldsymbol{C}\|_{\mathsf{col}}$ is replaced with the participation calibrated sensitivity as discussed in Section 3.3.3.

The RMS-loss has traditionally been the objective of choice to optimize correlated noise mechanisms, both for streaming prefix sums and for machine learning, as it better captures the "overall" loss across



(a) $\|\boldsymbol{B}\|_{\mathsf{row}}^2$, encountered in the max loss. **(b)** $\|\boldsymbol{B}\|_{\mathrm{F}}^2$, encountered in the RMS-loss.

Figure 4.1: We plot $\|\boldsymbol{B}\|_{\mathsf{row}}^2$ and $\|\boldsymbol{B}\|_{\mathrm{F}}^2$ for the matrix $\boldsymbol{B} = \begin{pmatrix} x_1 & 0 \\ 1 & x_2 \end{pmatrix}$ as a function of x_1, x_2. Notice the non-smoothness in the left plot.

all prefix sums than the max loss. Moreover, the squared norm $\|\cdot\|_{\mathrm{F}}^2$ encountered when optimizing the (square of the) RMS-loss is a smooth function,[1] while $\|\cdot\|_{\mathsf{row}}^2$ encountered in the (square of the) max loss of Eq. (2.4) is not; see Fig. 4.1 for an illustration. Practically, this makes the optimization of RMS-loss more stable.

In the upcoming sections, we focus on optimizing mechanisms whose strategy matrix $\boldsymbol{C}$ can be of two types:

1. **Dense strategies**, which are represented explicitly as a matrix $\boldsymbol{C}$. Dense strategies provide full generality and coverage over the space of strategies.

2. **Parameterized strategies**, which represents the strategy implicitly in terms of a parameter vector $\boldsymbol{\phi} \in \mathbb{R}^p$ via the parameterization $\boldsymbol{C}(\boldsymbol{\phi})$. Examples include the banded Toeplitz strategies (Section 2.4) and the BLT strategies (Section 2.5).

[1] We say a function f is smooth if it is continuously differentiable and its gradient ∇f is Lipschitz.

Remark 4.1 (Scalability)**.** While explicit dense strategy matrices $\boldsymbol{C}$ are more general, their $O(n^2)$ space and $O(n^3)$ optimization time complexity can be prohibitive for large number of steps n. Indeed, prior work has scaled up the dense mechanism only to $n \approx 10^4$ steps. For example, the numerical mechanism optimization in the open-source Jax Privacy library for dense mechanisms takes less than 10 seconds (on a GPU) for $n = 1024$ steps, but the running time increases 100× to around 15 minutes for $n = 8192$. When appropriately designed, implicitly represented strategies tend to have compact tractable representations and can be optimized efficiently for much larger values of n.

4.2 Optimizing the Dense Mechanism

The dense mechanism attempts to directly optimize the strategy matrix $\boldsymbol{C}$ to minimize the RMS-loss objective. We treat the streaming and multiple-participation settings separately.

4.2.1 Optimization in the Streaming Setting

The RMS-loss objective takes the simple form of Eq. (4.2) in the streaming setting. The key challenge arises from this objective being a non-convex function of $\boldsymbol{C}$. Fortunately, it can be rewritten as a convex optimization problem as a function of the Gram matrix $\boldsymbol{M} = \boldsymbol{C}^\top \boldsymbol{C}$. The key property is the following:

Lemma 4.2. For any two matrices $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$ with $\boldsymbol{C}$ invertible, and $\boldsymbol{M} = \boldsymbol{C}^\top \boldsymbol{C}$, we have

$$\|\boldsymbol{C}\|_{\mathsf{col}}^2 = \|\mathsf{diag}(\boldsymbol{M})\|_\infty\,, \quad \text{and} \quad \left\|\boldsymbol{A}\boldsymbol{C}^{-1}\right\|_{\mathrm{F}}^2 = \mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}^\top)\,.$$

Proof. The proof follows from direct calculation using the following two facts: (1) $\boldsymbol{M}[t,t] = \|\boldsymbol{C}[:,t]\|_2^2$, and (2) $\|\boldsymbol{Q}\|_{\mathrm{F}}^2 = \mathsf{Tr}(\boldsymbol{Q}^\top \boldsymbol{Q})$ for any real matrix $\boldsymbol{Q}$. □

The next ingredient to a convex reformulation is that the objective $\bar{\mathcal{L}}_2(\boldsymbol{C})$ is scale-invariant: $\bar{\mathcal{L}}_2(\boldsymbol{C}) = \bar{\mathcal{L}}_2(\alpha \boldsymbol{C})$ for any constant $\alpha \neq 0$.

Thus, we can fix a scale by imposing the constraint that $\|\boldsymbol{C}\|_{\mathsf{col}} = 1$, or equivalently, that $\mathsf{diag}(\boldsymbol{M}) \leq \mathbf{1}$ element-wise. Together, we end up with the following convex optimization problem:

Problem 4.3. *Find the matrix $\boldsymbol{M}_\star$ that solves the optimization problem*

$$\begin{aligned} &\text{minimize} && \mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}^\top) \\ &\textit{subject to} && \mathsf{diag}(\boldsymbol{M}) = \mathbf{1} \quad \textit{and} \quad \boldsymbol{M} \succ 0 \end{aligned} \tag{4.3}$$

and then find $\boldsymbol{C}_\star$ so that $\boldsymbol{C}_\star^\top \boldsymbol{C}_\star = \boldsymbol{M}_\star$ via e.g., Cholesky decomposition.

Here, the positive-definiteness constraint $\boldsymbol{M} \succ \mathbf{0}$ ensures that there exists a matrix $\boldsymbol{C}_\star$ such that $\boldsymbol{C}_\star^\top \boldsymbol{C}_\star = \boldsymbol{M}_\star$ and the objective $\mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A})$ is finite and convex. Finally, note that we replace the *inequality* constraint $\mathsf{diag}(\boldsymbol{M}) \leq \mathbf{1}$ with the *equality* constraint $\mathsf{diag}(\boldsymbol{M}) = \mathbf{1}$, as this does not change the solution: As we showed in Lemma 2.9 the set of optimal strategies always includes a column-normalized one. This has the added benefit of yielding an unconstrained optimization problem, as we will soon discuss. Together, we get the equivalence:

Theorem 4.4. The solution $\boldsymbol{C}_\star$ obtained from Problem 4.3 minimizes the RMS-loss, i.e.,

$$\bar{\mathcal{L}}_2(\boldsymbol{C}_\star) = \min_{\boldsymbol{C} \in \mathbb{R}^{n \times n}} \bar{\mathcal{L}}_2(\boldsymbol{C}).$$

See Fig. 3.3 (left) for an example of a mechanism optimized using this approach.

Practical Algorithms While Problem 4.3 can be written as a semi-definite program, it is possible to develop more efficient solutions using unconstrained quasi-Newton algorithms. The equality constraint $\mathsf{diag}(\boldsymbol{M}) = \mathbf{1}$ amounts to fixing the diagonal elements of $\boldsymbol{M}$ to one and not optimizing over them. (In contrast, the inequality constraint $\mathsf{diag}(\boldsymbol{M}) \leq \mathbf{1}$ is more challenging to handle, and requires constrained optimizers.)

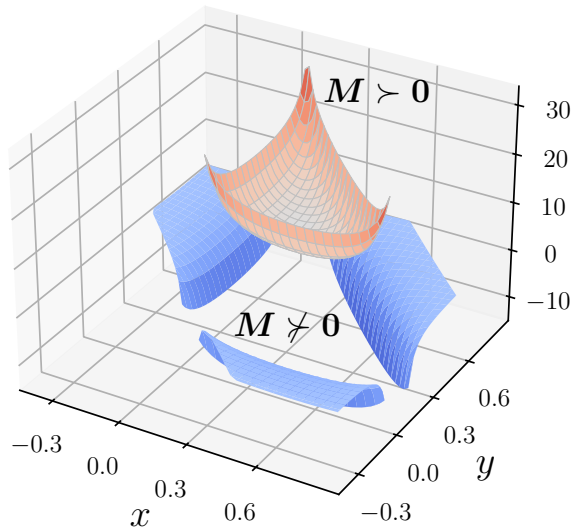


Figure 4.2: Dropping the positive-definiteness constraint of Problem 4.3: we plot $f(\boldsymbol{M}) = \mathsf{Tr}(\boldsymbol{M}^{-1})$ with $\boldsymbol{M} = \mathsf{diag}\big(x, y, 1-x-y\big)$, plotted as a function of x, y. (We have $\mathsf{Tr}(\boldsymbol{M}) = 1$ by construction.) Notice that the red region with $\boldsymbol{M} \succ \mathbf{0}$ (where the objective is convex) is disconnected from the blue regions where $\boldsymbol{M}$ fails to be positive definite. Indeed, $f(\boldsymbol{M})$ is discontinuous whenever one of the eigenvalues of $\boldsymbol{M}$ is zero. A gradient-based optimizer initialized in the red region with $\boldsymbol{M} \succ \mathbf{0}$ with appropriate safeguards (such as line search or a small enough learning rate) will generally not leave that region. Thus, this positive-definiteness constraint can be heuristically dropped in practical implementations.

Next, prior work has generally found it to be safe to ignore the positive-definiteness constraint $\boldsymbol{M} \succ \mathbf{0}$ as well (once $\mathsf{diag}(\boldsymbol{M}) = \mathbf{1}$ is imposed), as long as the optimization of $\boldsymbol{M}$ is initialized at a feasible point. The intuition behind this is illustrated in Fig. 4.2. Ignoring this constraint as a heuristic, Problem 4.3 is then a *smooth, unconstrained, and convex* optimization problem. These three properties are crucial as they allow us to leverage rapidly convergent off-the-shelf optimization algorithms to solve Problem 4.3. We recommend L-BFGS, a limited memory quasi-Newton algorithm, for its rapid empirical convergence and highly optimized numerical implementations.

Time and Space Complexity The $n \times n$ matrix $\boldsymbol{M}$ requires $O(n^2)$ memory. Thus, the total memory requirement of L-BFGS is $O(n^2)$ (assuming a small constant number of memory buffers for L-BFGS). Since the gradient $\nabla_{\boldsymbol{M}} \mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}^\top) = -\boldsymbol{M}^{-1}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{M}^{-1}$ requires computing $\boldsymbol{M}^{-1}$, the per-step time complexity is $O(n^3)$. In practice, when utilizing GPUs, it is feasible to solve this problem up to $n \approx 10^4$.

4.2.2 Optimization in the Multiple-Participation Setting

The main difference between the streaming and the multiple-participation setting lies in how sensitivity is calculated. As discussed in Sections 3.3.3 and 3.3.4, we need specialized algorithms tailored to particular participation schema to tightly compute the sensitivity. Indeed, in this case, we have that the RMS-loss of a strategy $\boldsymbol{C}$ under a participation schema Π and model dimension m is

$$\bar{\mathcal{L}}_2(\boldsymbol{C} \mid \Pi, m) = \mathsf{sens}(\boldsymbol{C}, \Pi, m) \left\| \boldsymbol{A}_{\mathsf{pre}} \boldsymbol{C}^{-1} \right\|_{\mathrm{F}}, \tag{4.4}$$

where $\mathsf{sens}(\cdot,\cdot,\cdot)$ is the participation-calibrated sensitivity (also see Definition 3.13) and the other factor $\boldsymbol{B}$ is again understood to be $\boldsymbol{B} = \boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1}$, and is omitted for brevity.

From an optimization perspective, it is convenient to use the following corollary of Lemma 3.15:

Corollary 4.5. Let $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ denote a lower-triangular matrix and let $\boldsymbol{M} = \boldsymbol{C}^\top \boldsymbol{C}$ denote its Gram matrix.

(a) For any participation schema Π, we have we

$$\mathsf{sens}\left(\boldsymbol{C}, \Pi, m\right)^2 \leq \max_{\boldsymbol{\pi} \in \Pi} \sum_{t,\tau \in \boldsymbol{\pi}} \left|\boldsymbol{M}[t,\tau]\right|. \tag{4.5}$$

(b) This upper bound is tight (and the sensitivity is independent of the dimensions m) if $\boldsymbol{M}[t,\tau] = 0$ for all $t \neq \tau \in \boldsymbol{\pi} \in \Pi$.

Lemma 3.15(b) gives more general non-negativity conditions for the upper bound to be tight. However, as discussed in the streaming setting, it is numerically easier to handle equality constraints rather than inequality constraints; we just fix $\boldsymbol{M}[t,\tau] = 0$ and do not optimize that variable. Thus, in the interest of scalability of numerical optimization, it is common to opt for this more restrictive formulation with equality constraints.

The constraint that $\boldsymbol{M}[t,\tau] = 0$ means that the noise added in two iterations t, τ where the same datapoint (or user) i can participate is *uncorrelated*. While this is a modest constraint for the cyclic participation, it imposes a b-banded structure (Definition 2.12) for b-MinSep participation (Definition 3.14). We design optimization algorithms under this assumption. In particular, since the sensitivity is then dimension-independent, we denote it by $\mathsf{sens}(\boldsymbol{C}, \Pi)$, as in Definition 3.13.

Similar to the streaming setting, this objective is scale-invariant, so we can impose the bound $\mathsf{sens}(\boldsymbol{C}, \Pi) \leq 1$. This leads us to the following optimization problem:

Problem 4.6. *Given a participation schema* Π*, (i) find the matrix* $\boldsymbol{M}_\star$ *that solves the optimization problem*

$$\begin{aligned} \underset{\boldsymbol{M} \succ 0}{\text{minimize}} \quad & \mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}^\top) \\ \textit{subject to} \quad & \sum_{t \in \boldsymbol{\pi}} \boldsymbol{M}[t,t] = 1 \ \forall \boldsymbol{\pi} \in \Pi \\ & \boldsymbol{M}[t,\tau] = 0 \ \forall \ t, \tau \in \boldsymbol{\pi}, \ \boldsymbol{\pi} \in \Pi, t \neq \tau \end{aligned} \tag{4.6}$$

and then (ii) find $\boldsymbol{C}_\star$ *so that* $\boldsymbol{C}_\star^\top \boldsymbol{C}_\star = \boldsymbol{M}_\star$ *via e.g., Cholesky decomposition.*

Note that $\mathsf{sens}(\boldsymbol{C}, \Pi) \leq 1$ is equivalent to the inequality constraint $\sum_{t \in \boldsymbol{\pi}} \boldsymbol{M}[t,t] \leq 1$ for each $\boldsymbol{\pi} \in \Pi$. Like in the streaming setting, we have replaced the inequality constraint with an *equality* in Eq. (4.6). Unlike the streaming setting, however, we do not have a proof of optimality of this step. This substitution is based on the following conjecture, which is strongly supported by empirical evidence:

Conjecture 4.7. *For every participation schema* Π *and workload* $\boldsymbol{A}$*, the unique solution* $\boldsymbol{M}_\star$ *of Problem 4.6 remains optimal when the first constraint of Eq. (4.6) is replaced with the inequality constraint* $\sum_{t \in \boldsymbol{\pi}} \boldsymbol{M}[t,t] \leq 1 \ \forall \boldsymbol{\pi} \in \Pi$.

Practically, Problem 4.6, with its equality constraints, has several advantages over the inequality constraints. First, it simplifies the problem significantly, making it closer to Problem 4.3 from the streaming setting, which is well-understood. Second, imposing these constraints improves the convergence of the optimization algorithm, allowing it to find better solutions in less time. Third, it reveals interesting and interpretable structures about the behavior of optimal strategies with different participation schemas (e.g., Proposition 4.9).

Practical Algorithms Problem 4.6 is only slightly harder to solve than its streaming version in Problem 4.3. The individual entries of $\boldsymbol{M}$ where $\boldsymbol{M}[t,\tau] = 0$ can be removed as variables and constraints from the optimization problem,[2] leaving the linear equality constraint $\sum_{t \in \boldsymbol{\pi}} \boldsymbol{M}[t,t] = 1$ as the main technical challenge to overcome. Ideally, we solve this problem in a similar fashion as in the streaming case, i.e. using an off-the-shelf rapidly-convergent optimization algorithm like L-BFGS. Since L-BFGS cannot natively handle equality constraints, some modifications are necessary, as discussed next for the case of cyclic participation.

[2] One way to achieve this is to initialize $\boldsymbol{M}[t,\tau] = 0$ and never update it during the course of optimization.

Algorithms for Cyclic Participation Constraints can be incorporated into gradient descent by *projecting* the iterates onto the constraint set. Because the constraints are linear, this can be equivalently achieved by projecting the gradients (instead of the iterates) onto the *level sets* of the constraints. Heuristically, we recommend using the same orthogonal projection strategy for L-BFGS.[3] That is, to use an out-of-the-box implementation of L-BFGS, but pass in the projected gradients (onto the level sets of the constraints) in place of the true gradients. This ensures that the iterates of the optimization respect the desired constraints, as long as the initial value does.

We emphasize that this is a *heuristic*, and we are not aware of convergence guarantees for this approach. Specifically, we use the projected gradient $\mathsf{proj}(\boldsymbol{G}^+)$ in place of the actual gradient $\boldsymbol{G}^+$ in the L-BFGS update step, as follows:

$$\mathsf{proj}(\boldsymbol{G}^+) = \arg\min_{\boldsymbol{G}} \left\{ \left\| \boldsymbol{G} - \boldsymbol{G}^+ \right\|_{\mathrm{F}}^2 \; : \; \sum_{t \in \boldsymbol{\pi}} \boldsymbol{G}[t,t] = 0 \; \forall \boldsymbol{\pi} \in \Pi \right\} . \quad (4.7)$$

This projection is easy in the case of cyclic participation $\Pi_{b,k}^{\mathsf{cyclic}}$, where the equality constraints are non-overlapping, i.e., each index t only belongs to one $\boldsymbol{\pi} \in \Pi_{b,k}^{\mathsf{cyclic}}$. In this case, $\mathsf{proj}(\boldsymbol{G}^+)$ simply modifies the diagonal entries of $\boldsymbol{G}^+$ for $|\Pi_{b,k}^{\mathsf{cyclic}}| = k$ non-overlapping subsets of entries. We refer the readers to Fig. 3.3 (far right) for an example of a mechanism optimized using this approach. In this example, some entries in the corresponding $\boldsymbol{C}$ are in fact negative, unlike with most of the other mechanisms we consider.

Algorithms for Min-Sep Participation The projection of Eq. (4.7) cannot be efficiently implemented with Min-Sep participation. In particular, it has overlapping groups (i.e. $\boldsymbol{\pi} \cap \boldsymbol{\pi}' \neq \emptyset$), so an exact projection may not be possible. Approximation projections (e.g., based on Dykstra's

[3] L-BFGS can natively handle box constraints. However, more complex constraints require projections in the Mahalanobis norm defined by L-BFGS's Hessian approximation. Instead, our heuristic uses a Euclidean projection, making it directly compatible with existing highly-optimized L-BFGS implementations (see Section 4.6 for references).

alternating projection algorithm) would also be infeasible, as their cost grows with $|\Pi|$ and $|\Pi|$ is exponentially large for the Min-Sep schema.

An alternate heuristic employed is to impose the constraint that $\mathsf{diag}(\boldsymbol{M}) = \mathbf{1}$:

Problem 4.8. *Given a participation schema* Π*, (i) find the matrix* $\boldsymbol{M}_\star$ *that solves the optimization problem*

$$\begin{aligned} \underset{\boldsymbol{M} \succ 0}{\text{minimize}} \quad & \mathsf{Tr}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}^\top) \\ \textit{subject to} \quad & \mathsf{diag}(\boldsymbol{M}) = \mathbf{1} \\ & \boldsymbol{M}[t,\tau] = 0 \;\; \forall \;\; t,\tau \in \boldsymbol{\pi},\, \boldsymbol{\pi} \in \Pi, t \neq \tau \end{aligned} \tag{4.8}$$

and then (ii) find $\boldsymbol{C}_\star$ *so that* $\boldsymbol{C}_\star^\top \boldsymbol{C}_\star = \boldsymbol{M}_\star$.

In the setting where every example participates an equal number of times, i.e. $|\boldsymbol{\pi}| = |\boldsymbol{\pi}'|$ for all $\boldsymbol{\pi}, \boldsymbol{\pi}' \in \Pi$, then $\mathsf{diag}(\boldsymbol{M}) = c\,\mathbf{1}$ implies the constraint of Eq. (4.6) for an appropriate constant c.[4] This is true for cyclic participation but not for Min-Sep participation.

Despite this difference, Problem 4.8 naturally interpolates between the streaming setting and the full batch setting. Indeed, it is nearly identical to the Problem 4.3 from the streaming setting, with the only difference being a smaller set of free variables. As a result, the same optimization routines used in the streaming setting are applicable here with minor modifications. On the other hand, analogous to Proposition 3.11, Problem 4.8 recovers independent noise under full-batch participation:[5]

Proposition 4.9. Under the full-batch setting (i.e., under participation schema $\Pi^{\mathsf{full}} = \{[n]\}$), we have:

(a) The minimizers $\boldsymbol{M}_\star$, $\boldsymbol{C}_\star$ of Problem 4.6 are diagonal matrices.

(b) Problem 4.8 is solved by $\boldsymbol{M} = \boldsymbol{I}_{n\times n} = \boldsymbol{C}$.

Proof. The proof follows from noting that only diagonal matrices are

[4] We can take $c = 1$ due to scale invariance of the objective.

[5] While computing the sensitivity in the full-batch case is NP-hard (Section 3.3.2), we can nonetheless solve Problems 4.6 and 4.8 efficiently. This is because we additionally impose $\boldsymbol{M}[t,\tau] = 0$ for all $t \neq \tau$.

allowed under the constraints of Problem 4.6 for $\Pi = \{[n]\}$, and that we require these diagonal elements to be 1 in Problem 4.8. □

Remark 4.10 (Full-batch correlated noise mechanisms)**.** In the full batch setting, where every example participates in every round, the diagonal entries of the optimal strategy $\boldsymbol{C}_\star$ solving Problem 4.6 (as described in Proposition 4.9) are generally decreasing, which means *more noise is added in the later iterations*.[6] From the machine learning perspective, this is counter-intuitive and possibly even undesirable. On the other hand, from the perspective of minimizing RMS-loss on the prefix queries, this is natural, as the earlier elements of the stream are required by more of the prefix queries, and hence need less noise.

Beyond this abnormality, Proposition 4.9 reiterates the observation of Proposition 3.11 that correlated noise offers no further improvements over independent noise in the full-batch setting. Indeed, the most notable improvements of correlated noise mechanisms over independent noise mechanisms occur in the streaming (i.e., the single-epoch) setting, and significant gains can still be obtained in large-scale compute-limited settings.

Time and Space Complexity The time and space complexity of solving Problem 4.6 with projected L-BFGS for both cyclic and b-minimum separated participation remains unchanged from the streaming setting. That is, we require $O(n^3)$ time per L-BFGS step (with a small constant number of total steps), and $O(n^2)$ space when representing the strategies as dense matrices.

For b-minimum separated participation, Problem 4.8 imposes a bandedness constraint that $\boldsymbol{M}[t, \tau] = 0$ for all $|t - \tau| \geq b$. By using parameterized strategies instead, we can reduce the time and space complexity to $O(nb^2)$ and $O(nb)$ respectively, which we discuss further in Section 4.3.

[6] The decreasing ordering of diagonal elements is a property of RMS-loss. Recall for max loss that the diagonal elements are equal, as we established in Proposition 3.11.

Remark 4.11 (Primal vs. Dual Optimization)**.** Problems 4.3-4.8 can also be effectively solved via their dual problems.[7] We recommend using the primal approach described earlier with L-BFGS for multiple reasons. First, it generally converges very rapidly (possibly due to the use of limited second order information about the curvature of the objective function). Second, the primal problem formulation can more naturally handle additional constraints on $\boldsymbol{M}$. Finally, parameterized mechanisms can only be tackled by primal-based approaches (as we see in the next section), allowing for a unified approach.

4.3 Optimizing Parameterized Mechanisms

While dense strategies provide the best utility, they can be expensive or infeasible to optimize and deploy in practice for $n \gtrsim 10^4$ steps (see Remark 4.1). This precludes their use in some regimes of practical interest. We now discuss approaches to optimizing over parameterized strategy classes such as Banded Toeplitz and Buffered Linear Toeplitz. These strategy classes have sufficient expressive capacity to represent near-optimal strategies (obtaining expected errors close to the optimal dense strategies), while their structure allows for efficient optimization and noise generation even with a large number of steps n.

Let $\boldsymbol{\phi} \in \Phi \subset \mathbb{R}^p$ represent arbitrary parameters and let $\boldsymbol{C}(\boldsymbol{\phi}) \in \mathbb{R}^{n \times n}$ denote a lower-triangular strategy matrix parameterized by $\boldsymbol{\phi}$. We focus in particular on two parameterized mechanisms introduced in Section 2:

(a) **The Banded Toeplitz mechanism** (Section 2.4): Given parameters $\boldsymbol{\phi} = (c_0, \ldots, c_{b-1})$, we parameterize the first column of $\boldsymbol{C}(\boldsymbol{\phi})$ as

$$(\boldsymbol{C}(\boldsymbol{\phi}))[:, 0] = (c_0, \ldots, c_{b-1}, 0, \ldots, 0)\,.$$

Further, we take $\boldsymbol{C}(\boldsymbol{\phi})$ to be Toeplitz, so all other columns can

[7] It turns out that these problems satisfy strong duality, so the primal and dual optimal values coincide.

be determined from the first one, e.g. for $b = 3$ and $n = 4$,

$$\boldsymbol{C} = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 \\ c_1 & c_0 & 0 & 0 & 0 \\ c_2 & c_1 & c_0 & 0 & 0 \\ 0 & c_2 & c_1 & c_0 & 0 \\ 0 & 0 & c_2 & c_1 & c_0 \end{bmatrix}.$$

(b) **The Buffered Linear Toeplitz (BLT) mechanism** (Section 2.5): Given parameters $\boldsymbol{\phi} = (\boldsymbol{\alpha}, \boldsymbol{\lambda})$, where $\boldsymbol{\alpha} \in \mathbb{R}_+^d$ is a scale parameter and $\boldsymbol{\lambda} \in [0, 1)^d$ is a decay parameter, we parameterize $\boldsymbol{C}(\boldsymbol{\phi}) = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ as in Eq. (2.16).

The goal here is then to find parameters $\boldsymbol{\phi}$ so as to minimize the RMS-loss or the max loss. We consider the max loss in addition to the RMS-loss because for Toeplitz strategies (both banded and BLT), the max loss is equal to the last-iterate loss, and is hence a smooth function that is amenable to numerical methods.[8]

Problem 4.12. *Given a participation schema* Π*, find the parameters* $\boldsymbol{\phi}_\star \in \Phi$ *that solve the optimization problem*

$$\underset{\boldsymbol{\phi} \in \Phi}{\text{minimize}} \quad \mathsf{sens}(\boldsymbol{C}(\boldsymbol{\phi}), \Pi)^2 \left\| \boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1} \right\|_{\mathcal{E}}^2, \tag{4.9}$$

where $\|\cdot\|_{\mathcal{E}} = \|\cdot\|_{\mathrm{F}} / \sqrt{n}$ *for the RMS-loss and* $\|\cdot\|_{\mathcal{E}} = \|\cdot\|_{\mathsf{row}}$ *for the max loss. For the streaming setting where* $\Pi = \Pi^{\mathsf{single}} = \{(0), (1), \ldots, (n-1)\}$*, the problem simplifies to*

$$\underset{\boldsymbol{\phi} \in \Phi}{\text{minimize}} \quad \|\boldsymbol{C}(\boldsymbol{\phi})\|_{\mathsf{col}}^2 \left\| \boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1} \right\|_{\mathcal{E}}^2. \tag{4.10}$$

Practical Algorithms We tackle Problem 4.12 with automatic differentiation coupled with off-the-shelf gradient-based optimization. Specifically, it suffices to have a function that can efficiently evaluate the objective function with differentiable operations. In order to scale to a large number of steps n, it is crucial to be able to evaluate the objective

[8] In all the cases we consider, we have that $\mathsf{sens}(\boldsymbol{C}(\boldsymbol{\phi}), \Pi)$ is independent of the model dimension m, as it satisfies the conditions of Lemma 3.15(b).

function *without ever materializing the matrices* $\boldsymbol{C}(\boldsymbol{\phi})$ *or* $\boldsymbol{A}$ *explicitly* by exploiting their special structure. Details for how this can be achieved with specific strategy classes are given in the subsections below.

In general, Problem 4.12 is a non-convex optimization problem with respect to either the strategy matrix $\boldsymbol{C}$ or its parameters $\boldsymbol{\phi}$ for the banded Toeplitz and BLT mechanisms. Thus, gradient-based optimization is not guaranteed to converge to a global optimal solution. However, in practice, we find that appropriate initialization and parameter tuning lead to high-quality solutions.[9]

Next, we describe how to evaluate the objective of Problem 4.12 for both the banded Toeplitz and BLT mechanisms.

4.3.1 Optimizing the Banded Toeplitz Mechanism

When the number of bands b' of the banded Toeplitz mechanism is chosen to be no greater than the minimum separation b under MinSep participation $\Pi_{b,k}^{\mathsf{minSep}}$, we have by Lemma 3.16(d) that the multiple-participation sensitivity $\mathsf{sens}(\boldsymbol{C}, \Pi_{b,k}^{\mathsf{minSep}})$ is $\sqrt{k}$ times the streaming sensitivity $\|\boldsymbol{C}\|_{\mathsf{col}}$, where k is the maximum number of partitions. The same relation also holds for the cyclic participation setting, where $b = N/B$ is then the number of steps in an epoch and k is the number of epochs. In both cases, k is a constant and can be ignored for optimization.

It remains to efficiently compute the error $\|\boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1}\|_{\mathcal{E}}^2$. Because the matrix $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1}$ is Toeplitz (as long as $\boldsymbol{A}$ is Toeplitz) and has coefficients $\boldsymbol{b} = \boldsymbol{C}(\boldsymbol{\phi})^{-1}\mathbf{1} \in \mathbb{R}^{\mathbf{n}}$, the expected error can be computed efficiently as

$$\left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}(\boldsymbol{\phi})^{-1}\right\|_{\mathsf{row}}^2 = \|\boldsymbol{b}\|_2^2\,, \quad \left\|\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}(\boldsymbol{\phi})^{-1}\right\|_{\mathrm{F}}^2 = \sum_{t=0}^{n-1}(n-t)b_t^2\,. \quad (4.11)$$

These expressions can be computed efficiently. First, computing the vector $\boldsymbol{b} \in \mathbb{R}^n$ requires $O(nb)$ time and $O(n)$ space by leveraging the

[9] For example, we find in the streaming setting that the gradient-based solutions obtained are competitive with the Toeplitz mechanism (Section 2.3), which is an upper bound on how well both the banded Toeplitz and BLT mechanisms can perform in the streaming setting. See Section 2.6 for empirical comparisons.

banded structure of $\boldsymbol{C}(\boldsymbol{\phi})$ (see Algorithm 2.1 in Section 2 for details). Second, given the vector $\boldsymbol{b}$, we can evaluate the expressions of Eq. (4.11) in $O(n)$ time.

The final step to optimize for Problem 4.12 using first-order optimization is to calculate the gradients of the objective w.r.t. the parameters. This can be done with automatic differentiation. The sensitivity $\|\boldsymbol{C}(\boldsymbol{\phi})\|_{\mathsf{col}}^2 = \sum_{t=0}^{b-1} c_t^2$ is clearly a differentiable function of the parameters $\boldsymbol{\phi} = (c_0, \ldots, c_{b-1})$. The error $\left\|\boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1}\right\|_{\mathcal{E}}^2$ is a function of $\boldsymbol{b}$, which is in turn a function of the $\boldsymbol{\phi}$.

Practical Considerations We find that this non-convex optimization problem is somewhat sensitive to its initialization. In practice, initializing the optimizer with $\boldsymbol{\phi}_0 = (c_0^\star, \ldots, c_{b-1}^\star)$, which are the optimal Toeplitz coefficients from Theorem 2.5, is highly effective across a wide range of settings. Then, an off-the-shelf implementation of L-BFGS with default parameters returns a high quality solution.

Furthermore, we find that it is important to tune the number of bands. One can do so without paying a privacy cost of working with real data by using the max loss or RMS-loss as a proxy for learning performance. See Fig. 4.3 which shows how the optimal number of bands varies with the batch size.

4.3.2 Optimizing the BLT Mechanism

Our high-level approach is to express the RMS-loss/max loss objective as a differentiable function of the parameters $\boldsymbol{\phi}$ that we wish to optimize over. This enables us to leverage automatic differentiation to optimize the objective with gradient-based optimization.

We first map the parameters $\boldsymbol{\phi}$ to the BLT/inverse-BLT parameters $\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}} \in \mathbb{R}^n$ and $\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}} \in [0,1)^d$ using a differentiable function such that the inverse of $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is given by $\boldsymbol{C}^{-1} = \mathsf{BLT}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$.[10] The different ways of achieving this are summarized in Fig. 4.4a. Next, we map $(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ to the sensitivity and the RMS-loss or max loss via a differentiable function, as summarized in Fig. 4.4b. By the chain rule,

[10] This is possible under the conditions imposed by Lemma 2.16.

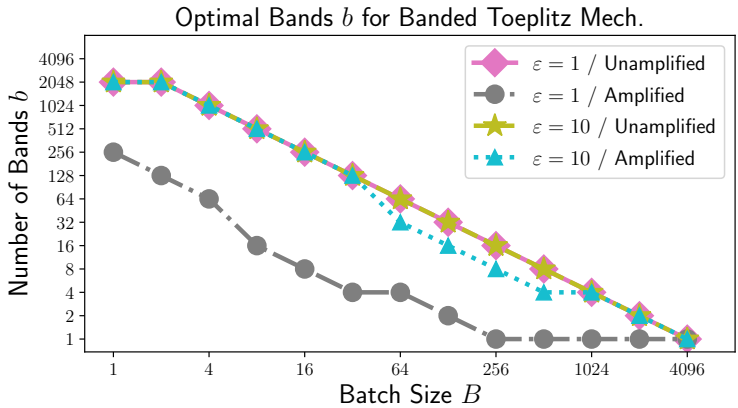


Figure 4.3: Tuning the Number of Bands: We plot the optimal number of bands b of the banded Toeplitz mechanism as a function of the batch size B. The number of bands is chosen to empirically minimize the max loss with and without amplification at various values of the privacy budget ε for (ε, δ)-DP. Throughout, we fix the number of iterations $n = 2048$ and dataset size $N = 4069$, as in Fig. 3.5. The value of ε does not matter in the unamplified scenarios, as the unnormalized max loss is simply a scaled version of $\mathsf{sens}(\boldsymbol{C}) \left\| \boldsymbol{A}_{\mathsf{pre}} \boldsymbol{C}^{-1} \right\|_{\mathsf{row}}$ (see Theorem 2.2 and 3.9). On the other hand, when accounting for amplification by sampling, the optimal number of bands varies with ε: this is because the effect of amplification is determined by the noise multiplier, as we discussed in Fig. 3.1. Recall from Fig. 3.5 that significant amplification is obtained at $\varepsilon = 1$: this figure shows that this requires a smaller number of bands. On the other hand, there is almost no amplification at $\varepsilon = 10$, so the optimal number of bands is almost identical to the unamplified one.

the composition of both these steps gives the loss as a differentiable function of the parameters $\boldsymbol{\phi}$.

We consider the streaming setting, cyclic participation, and b-minimum separated participation with at most k participations. The conditions of Lemma 3.16(c) will hold so that the sensitivity for the latter two multiple-participation settings will coincide, as in the banded case.

Parameterization Choices: From $\boldsymbol{\phi}$ to BLT Two possible BLT parameterizations $\boldsymbol{\phi}$ have been proposed such that the mapping $\boldsymbol{\phi} \mapsto (\boldsymbol{\alpha}, \boldsymbol{\lambda}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ is a differentiable mapping.

The first approach is to take $\boldsymbol{\phi} = (\boldsymbol{\alpha}, \boldsymbol{\lambda})$ as the parameters of $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ and reconstruct the inverse BLT parameters $\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}}$ from it as per Lemma 2.16; see Fig. 4.4a (left) for an illustration. This requires finding the roots of a degree-d polynomial, a step that can be performed via an eigen-decomposition of a non-symmetric matrix (known as the *companion matrix*) in $O(d^3)$ time. This subroutine is available as a differentiable function in automatic differentiation frameworks like JAX and PyTorch.

An alternate approach is to take $\boldsymbol{\phi} = (\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}})$, and reconstruct $\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}} \in \mathbb{R}^d$, see Fig. 4.4a (right) for an illustration.[11] This can be achieved in $O(d^2)$ time and memory as per the following lemma:

Lemma 4.13. Consider non-zero decay parameters $\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}} \in \mathbb{R}^d$ that are pairwise distinct.[12] Then, the unique parameters $\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}} \in \mathbb{R}^d$ that achieve $\mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathsf{BLT}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})^{-1}$ are given by:

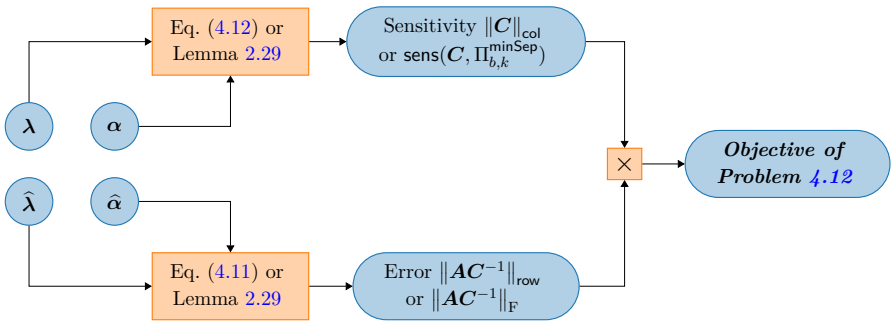
$$\alpha_i = \frac{\prod_{j=1}^{d} \lambda_i - \widehat{\lambda}_j}{\prod_{j \neq i} \lambda_i - \lambda_j}, \quad \text{and} \quad \widehat{\alpha}_i = \frac{\prod_{j=1}^{d} \widehat{\lambda}_i - \lambda_j}{\prod_{j \neq i} \widehat{\lambda}_i - \widehat{\lambda}_j}.$$

Fig. 4.4a compares these different parameterizations. From a theoretical perspective, both these parameterizations are equivalent, in that they represent the same class of BLT/inverse-BLT systems:

[11] We allow the scale parameters $\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}}$ to take both positive or negative values here. We circumvent this issue in practice with appropriate barrier functions.

[12] Specifically, $\lambda_i \neq \lambda_j$ and $\widehat{\lambda}_i \neq \widehat{\lambda}_j$ for all $i \neq j$, and $\lambda_i \neq \widehat{\lambda}_j$ for all $i, j \in [d]$.

(a) Computation graph of the differentiable mapping $\boldsymbol{\phi} \mapsto (\boldsymbol{\alpha}, \boldsymbol{\lambda}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ for different choices of the parameters $\boldsymbol{\phi}$, denoted by the double bordered nodes. **Left**: $\boldsymbol{\phi} = (\boldsymbol{\lambda}, \boldsymbol{\alpha})$. **Right**: $\boldsymbol{\phi} = (\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}})$.



(b) Computation graph of the differentiable mapping from the BLT/inverse-BLT parameters $(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ to the RMS-loss/max loss objectives.

Figure 4.4: BLT Computation Graph: We show the computation graph to compute the objective of Problem 4.12 starting with different choices of the parameters $\boldsymbol{\phi}$ of the BLT. The blue nodes denote variables (with the double bordered nodes denoting $\boldsymbol{\phi}$) while the orange nodes denote operations.

Lemma 4.14. Consider the following two BLT parameterizations:

(a) Let Φ_1 denote the set of $(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \in \mathbb{R}^d_{++} \times (0,1)^d$ that satisfy $\sum_{i=1}^d \alpha_i/\lambda_i < 1$, in addition to the conditions of Lemma 2.16;

(b) Let Φ_2 denote the set of $(\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}}) \in (0,1)^d \times (0,1)^d$ that satisfy the strict interlacing condition

$$\boldsymbol{\lambda}_1 > \widehat{\boldsymbol{\lambda}}_1 > \boldsymbol{\lambda}_2 > \widehat{\boldsymbol{\lambda}}_2 > \cdots > \widehat{\boldsymbol{\lambda}}_{d-1} > \boldsymbol{\lambda}_d > \widehat{\boldsymbol{\lambda}}_d \,.$$

Then, the set of BLT/inverse-BLT systems represented Φ_1 and Φ_2 are identical. That is, for every $\boldsymbol{\phi}_1 \in \Phi_1$, there exists $\boldsymbol{\phi}_2 \in \Phi_2$ such that $\mathsf{BLT}(\boldsymbol{\phi}_1) = \mathsf{BLT}(\boldsymbol{\phi}_2)$ and vice versa.

BLT to Loss Having obtained a differentiable mapping $\boldsymbol{\phi} \mapsto (\boldsymbol{\alpha}, \boldsymbol{\lambda}, \widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$, we now turn to expressing the loss as a function of these scale and decay parameters. We consider two approaches to achieve this:

- **Toeplitz Coefficient Materialization**: For a given time horizon n, we can materialize the first column $c_0 = 1$ and $c_t = \sum_{i=1}^d \alpha_i \lambda_i^{t-1}$ for $t \geq 1$ of the strategy matrix $\boldsymbol{C} = \mathsf{BLT}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$. Then, denoting $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{R}^n$, the sensitivity can simply be calculated as (see Eq. (3.25)):

$$\|\boldsymbol{C}\|^2_{\mathsf{col}} = \|\boldsymbol{c}\|_2^2 \,, \quad \text{and} \quad \mathsf{sens}(\boldsymbol{C}, \Pi^{\mathsf{minSep}}_{b,k}) = \left\| \sum_{j=0}^{k-1} \mathcal{B}^{jb}(\boldsymbol{c}) \right\|_2^2 \,, \tag{4.12}$$

  where $\mathcal{B}^l(\boldsymbol{c}) = \boldsymbol{C}[:, l]$ is given by the *backshift* operator:

$$\mathcal{B}^l(\boldsymbol{c}) := \left(\underbrace{0, \ldots, 0}_{l \text{ zeros}}, c_0, \ldots, c_{n-l-1} \right) \in \mathbb{R}^n \quad \text{for } l \geq 0 \,.$$

  These can be programmed as differentiable functions in automatic differentiation frameworks such as JAX or PyTorch that execute in $O(ndk)$ time. Reverse-mode automatic differentiation creates a computation graph that effective stores $\boldsymbol{c}$ in memory, leading to a $O(n)$ space complexity; see Fig. 4.4 (a).

We can then materialize the first column of $\boldsymbol{b} = (\boldsymbol{A}_{\mathsf{pre}}\boldsymbol{C}^{-1})[:, 0]$ since we have that $\boldsymbol{b}$ is the vector of prefix sums of $\boldsymbol{c}' = (\boldsymbol{C}^{-1})[:, 0]$. Finally, we use Eq. (4.11) to compute the error $\|\boldsymbol{A}\boldsymbol{C}(\boldsymbol{\phi})^{-1}\|_{\mathcal{E}}^2$. These quantities can be computed in $O(nd)$ time and $O(n)$ memory, as can their derivatives with respect to the parameters (using automatic differentiation). This can get prohibitively large, especially when the number n of steps is in the order of billions.

- **Closed-Form Expression**: In the streaming (single-participation) setting, we can do better with a little more effort—all the sensitivity and error terms in Problem 4.12 can be computed in closed form in $O(d^2)$ time and memory. This improvement is significant because, in practice, $d < 10$ typically provides sufficient accuracy for the BLT mechanism. The exact expressions are given in the appendix in Theorem 2.29.

Practical Considerations We recommend optimizing over the parameterization $\boldsymbol{\phi} = (\boldsymbol{\alpha}, \boldsymbol{\lambda})$, as in Fig. 4.4a. While the $O(d^3)$ eigendecomposition is marginally more expensive, we find that it is numerically more stable. To calculate the loss, we recommend using the closed-form expressions whenever available, i.e., for the streaming setting. This is especially advantageous with a large number n of steps. For the multi-participations setting where closed-form expressions are not available, we recommend materializing the Toeplitz coefficients.

Regardless of the parameterization (from Fig. 4.4a), it is crucial to impose a log-barrier function so that $\boldsymbol{\lambda} \in (0, 1)^d$, and $\boldsymbol{\alpha}$ is coordinate-wise positive:

$$h(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = -\sum_{i=1}^{d} \left(\log(\lambda_i) + \log(1 - \lambda_i) + \log(\alpha_i)\right) .$$

Multiple sets of BLT parameters $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$ can produce the numerically similar strategy matrix $\boldsymbol{C}$, with different (random) initializations resulting in different parameters. Interestingly, we observe that a smaller number of buffers d leads to greater numerical stability. While theory suggests that a larger d should always result in lower error, our empirical findings show that there is no benefit (in terms of RMS-loss and max

Mechanism	Strategy Optimization Complexity		
	Runtime (Per Step)	**Memory**	**Practical Limitations**
Dense	n^3	n^2	$n \lesssim 10^4$
b-Banded	bn^2	bn	$n \lesssim 10^5$
b-Banded Toeplitz	bn	n	$n \lesssim 10^7$
d-Buffered Linear Toeplitz, $k > 1$	dn	n	$n \lesssim 10^8$
d-Buffered Linear Toeplitz, $k = 1$	d	d	$n \lesssim 10^{10}$

Table 4.1: Time and space complexity of optimizing over different strategy classes in terms of the number of steps n, the maximum number of participations k, the number of bands b for banded strategy matrices (Section 2.4), and the order d for the BLT mechanism (Section 2.5). The optimal value of b for Banded and Banded Toeplitz mechanisms depends on the problem parameters, see Fig. 3.5 for an example. The $k = 1$ optimization of BLTs uses the closed-forms of the loss from Section 4.3.2, while for $k > 1$ we use the Toeplitz coefficient materialization approach (necessitating time and space $O(n)$, but scaling very well in practice).

loss) in taking d larger than 5 for this reason. In fact, $d > 10$ can actually hurt empirical performance. This also suggests that reducing d may help mitigate over-parameterization issues in this highly non-convex optimization landscape.

Using double-precision floating point arithmetic is crucial.[13] This is necessary because decay parameters λ_i that are very close to, but strictly less than, 1 are frequently encountered in practice. Finally, careful tuning of L-BFGS parameters helps improve robustness.

4.4 Choosing a Correlated Noise Mechanism: Recommendations

We now provide some concrete rules of thumb to choose various design aspects of correlated noise mechanisms in the context of AI model training problem based on various practical considerations we have discussed so far. These recommendations are summarized in Fig. 4.5.

[13]This requires an explicit adjustment in JAX and PyTorch, which default to single-precision.

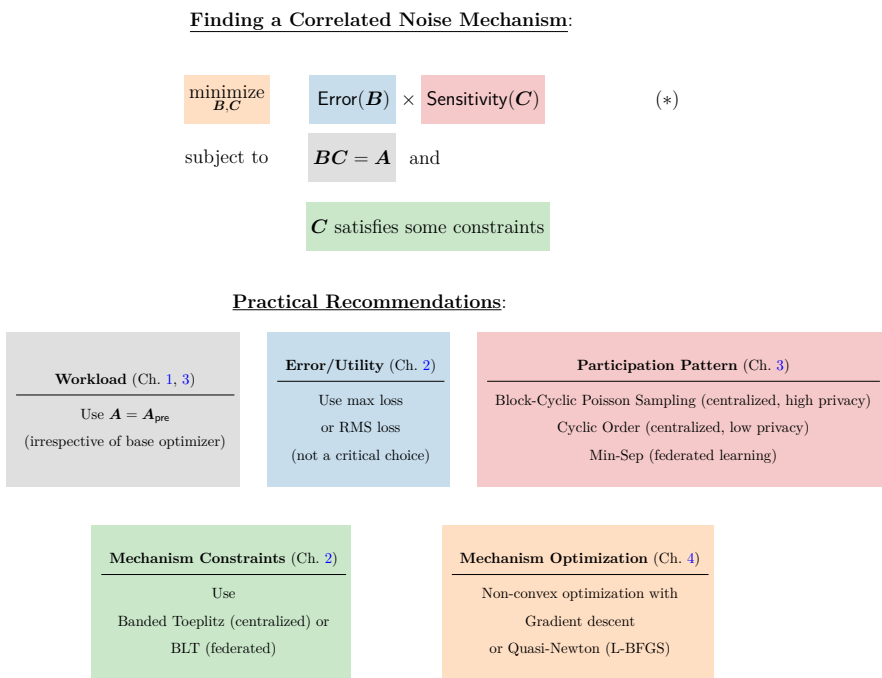


Figure 4.5: A summary of the practical recommendations for each of the design considerations highlighted in Fig. 1.8 in the context of AI model training.

4.4.1 Choosing a Workload Matrix

Recall that the workload matrix does not explicitly appear in an implementation of DP-SGD with correlated noise (Algorithm 1.3 or the batch version Algorithm 3.1). Indeed, the mechanism is fully specified by the noise-correlating matrix $\boldsymbol{C}^{-1}$ (whereas the $\boldsymbol{C}$ matrix is only needed to calibrate the noise for a particular μ-GDP guarantee). At the end of the day, our goal is good learning performance for the final trained model, for example the empirical risk $\mathbb{E}_{\boldsymbol{x}\sim D}[\ell(\boldsymbol{x}, \boldsymbol{\theta}_n)]$ (cf. Eq. (1.2)). Unfortunately, we lack a rigorous theory to estimate this performance for a given noise-correlating matrix $\boldsymbol{C}^{-1}$ used in Algorithm 3.1.[14]

In Section 1, we discussed a heuristic of selecting the workload matrix $\boldsymbol{A}$ implied by the choice of first-order optimizer, e.g. $\boldsymbol{A}_{\mathsf{pre}}$ from Eq. (1.6) for vanilla SGD and $\boldsymbol{A}_{\mathsf{mom}}$ from Eq. (3.6) for SGD with momentum. Unfortunately, this heuristic does not apply to adaptive optimizers such as Adam or AdaGrad.

We recommend an alternative heuristic: choose $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ regardless of the base (non-private) optimizer. That is, we optimize the mechanism to achieve low error on *unweighted* prefix sum estimates. We find that this heuristic yields a $\boldsymbol{C}^{-1}$ that works very well in Algorithm 3.1 even if the actual first-order update rule (Line 9 of Algorithm 3.1) corresponds to a different workload such as SGD with momentum, or even an adaptive gradient algorithm such as Adam or AdaGrad. Conversely, there is no guarantee that a mechanism selected for good performance on a particular momentum matrix $\boldsymbol{A}_{\mathsf{mom}}$ will outperform one optimized for prefix sums $\boldsymbol{A}_{\mathsf{pre}}$, even if the actual update rule corresponds exactly to the momentum workload matrix $\boldsymbol{A}_{\mathsf{mom}}$.

Recommendation: Workload Matrix

Choose $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$ (irrespective of the base non-private optimizer).

There has been some effort in getting theoretical guarantee for workload matrices arising from momentum methods, but they do not lead to space and time efficient mechanisms. Moreover, while we recommend

[14] The learning guarantees in Section 3.5 are too loose for general problems, or only apply to specific problems such as linear regression.

the above, an important research question is whether there are better choices when we consider adaptive learning algorithms, like Adam, RMSProp, etc. We return to open questions in this space in Section 5.3.

4.4.2 Choosing a Surrogate Loss

We primarily focused on the max loss objective in this monograph, although the RMS-loss is another common choice in the literature. In general, any differentiable function of the per query squared errors can be used as the objective with minor modifications to the optimization techniques discussed in this section. However, these more flexible alternatives have not been carefully studied previously. Prior work empirically comparing max loss with mean squared error is also slim, but in the experiments that have been done the training-time performance characteristics have been similar.

Recommendation: Surrogate Loss for Mechanism Optimization

Choose the max loss (Definition 2.3) or the RMS loss (Definition 2.23).

As in the case of workload matrices, the choice of a better surrogate loss still remains an interesting research question. We discuss it in more detail in Section 5.5.

4.4.3 Choosing a Participation Schema

To a large extent, the possible participation schemas will be determined by the training environment and infrastructure. Generally, we recommend choosing the first schema in the following list that is fully supported by the infrastructure:

- **Block Cyclic Poisson Sampling**, defined in Definition 3.20 should be preferred when possible, as it endows the mechanism with both the benefits of privacy amplification by sampling and noise correlation. Recall that block-cyclic sampling with $b = 1$ blocks recovers the usual data processing pattern for DP-SGD with

Poisson sampling. This schema is generally conceptually feasible in centralized training scenarios. Of course, if this sampling pattern is assumed for mechanism design and privacy accounting, it is essential that the training infrastructure correctly follows this sampling scheme; unfortunately, currently no major (DP) ML training platforms directly support such data processing.

- **Cyclic Participation**, described in Fig. 3.6, should be used in centralized training scenarios where strictly enforcing Poisson subsampling is infeasible or too inconvenient. In the low privacy regime, cyclic participation will only be slightly worse than block cyclic Poisson sampling; see also Fig. 3.1.

- **Min-Sep Participation**, which was also described in Definition 3.14 and Fig. 3.6, should be used in federated training scenarios where it is difficult to control the precise order of examples or users. It may also be applicable in centralized training scenarios under the multi-attribution model of user-level DP.

- **Single-participation** If none of the above are possible, it is generally relatively straightforward to ensure that each example contributes at most once to training (for example, training for a single epoch in an arbitrary order). For user-level privacy, we additionally need that each user contributes at most one example to training.

Recommendation: Participation Schema

Depending on the setting, choose the following:

- Centralized (non-federated) learning: Choose block-cyclic Poisson sampling as a default.

- Federated learning: Choose the Min-Sep participation schema.

For the centralized (i.e. non-federated) setting choose the max loss (Definition 2.3) or the RMS loss (Definition 2.23).

4.4.4 Choosing a Class of Strategy Matrices

Our default recommendation is to use banded Toeplitz strategies with column normalization in most settings. These strategies offer the best expected errors in both federated settings (under a b-min-sep participation schema) and centralized settings (under a cyclic participation schema with Poisson sub-sampling). Moreover, with a careful implementation, their memory overhead is generally small compared to the overall cost of the automatic differentiation to compute per-example gradients and gradient clipping + accumulation.

On the other hand, when the number of participations k is small, and a larger ε provides sufficient privacy protection, then amplification offers relatively little benefit, and so the mechanisms for the centralized un-amplified setting below may be sufficient (and allow e.g. much lower runtime costs via BLTs).

Below we provide some additional suggestions for scenarios where the default recommendation may not apply or may not be necessary.

Centralized training with many epochs and high privacy In settings where we are training a model for many epochs over the dataset (typically using a batch size that consists of a significant fraction of the total training data set size), particularly when a high privacy bar is required (e.g., 1-GDP or stronger), correlated noise offers little to no improvement over a careful implementation of DP-SGD when accounting for amplification by sampling. Full-batch gradient descent (when the full dataset is used to compute each gradient) is an extreme example of this setting. However, since DP-SGD is just a special case of matrix factorization with banded strategies, there is no need to deviate from the default recommendation here in principle.

Centralized training without amplification-via-sampling In the single-participation setting, column-normalized BLTs offer near-optimal loss, with the best-known noise-generation efficiency. Under Min-Sep participation, banded matrices with $b = \frac{n}{k}$ bands can offer small performance improvements over BLTs, but the noise generation cost for large b can be prohibitive.

In centralized training scenarios without amplification, where cyclic participation can be enforced, using dense strategies can offer a slight improvement in the expected error compared to BLTs or banded matrices, but mechanism optimization is only feasible for $n \leq 4000$ or so. Further, this approach has even more expensive noise generation for large n. In this case, the cost of noise generation, which we discuss in greater detail in Section 4.4.5, may start to outweigh the other training costs.

More than $n = 10^7$ training iterations Optimizing banded Toeplitz strategies using the techniques we have described only scales up to $n = 10^7$. Beyond this point, we recommend using BLTs instead, although the compute budget may be better spent on larger batch sizes or model sizes instead.

Recommendation: Strategy Matrix Class

Depending on the setting, choose:

- Centralized learning: choose the banded Toeplitz mechanism.
- Federated learning: choose the BLT mechanism.

4.4.5 Implementation Considerations for Noise Generation

There are several ways one can generate the correlated noise at training time, and these implementations have varying performance characteristics which we discuss in this section. Algorithm 2.1 and Algorithm 2.2 from Section 2 give two algorithms, specialized for banded Toeplitz and BLT strategies. Here, we provide a more comprehensive overview of the implementation options and their trade-offs.

Approach 1: Naive Noise Generation The simplest approach is to sample the full matrix of independent noise $\boldsymbol{Z} \sim \mathcal{N}_{n \times m}(0, \nu^2)$ in memory, and then compute $\boldsymbol{C}^{-1}\boldsymbol{Z}$ explicitly. The rows of this matrix are the correlated noise vectors and can be iteratively indexed during model training. This implementation of noise generation has an $n \times m$ time and memory overhead, and for large n and m this overhead can be

prohibitive. This approach may be feasible in very small scale problems but should otherwise be avoided.

Approach 2: Memoryless Noise Regeneration A better approach avoids the $n \times m$ memory overhead by regenerating the rows of $\boldsymbol{Z}$ in each iteration. That is, on iteration t we need to compute $\sum_{\tau=0}^{t-1} \boldsymbol{C}^{-1}[t, \tau]\boldsymbol{Z}[\tau, :]$. By keeping track of the random seeds used to sample each row of $\boldsymbol{Z}$ we can avoid keeping the entire matrix in memory at once, in favor of generating the rows on demand exactly when needed. This approach crucially relies on the fact that a pseudo-random number generator is used to sample the noise. If the random keys are discarded after use or otherwise unavailable (e.g. for security purposes), this approach would not be viable. The running time complexity of Approach 2 is the same as Approach 1, but it only has a $O(m)$ memory overhead. Prior work have used this approach and demonstrated scalability up to about $n \approx 2000$ for models with a few million parameters.

Approach 3: Low-Memory Streaming Linear Operators A third approach applies for strategy classes with specific structure that enable efficient multiplication-by-inverse algorithms, like banded Toeplitz and BLT strategies. We considered these approaches earlier in Algorithm 2.1 and Algorithm 2.2 of Section 2. In both cases, the noise generator receives i.i.d. noise vectors as a stream of inputs, and returns appropriately correlated noise vectors as outputs in a streaming fashion. To accomplish this, the streaming algorithm stores a "buffer state" in memory. For the banded Toeplitz strategy, the buffer consists of the correlated noise vectors for the previous $b - 1$ iterations, incuring a $O(bm)$ time and memory overhead. For the BLT mechanism, we require d buffers for a BLT of order d, incuring a $O(dm)$ time and memory overhead.

Distributed Noise Generation As mentioned in Section 4.4, correlated noise mechanisms are highly advantageous in large-scale training scenarios which typically occur in distributed environments with many accelerators working in tandem. By carefully splitting up the work, we can greatly reduce the time and (per-machine) memory required to

generate the correlated noise. There are several "sharding" strategies one could consider, which describe how the work should be partitioned across machines.

1. **Sharding noise like the model** is one natural option. There, each row $\boldsymbol{z} \in \mathbb{R}^m$ of the noise matrix $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$, which has the same size and structure as the model, is distributed across machines in the same way as the model. This approach is natural and can leverage existing model sharding rules that are known to be compatible with the model shape. Moreover, once the noise is generated, it can be added to the model gradient without any communication. However, for models trained with only data parallelism (where a full copy of the model is replicated on every machine), this approach does not leverage the additional machines effectively, as it duplicates the work of generating noise on each machine.

2. **Sharding noise across all machines** is a better option that ensures no duplicate work is done across machines. There, each row $\boldsymbol{z} \in \mathbb{R}^m$ of the noise matrix $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$ is evenly distributed across all machines in the environment, and all computations on these vectors respect this sharding. That is, if we have M machines, we (as evenly as possible) partition the indices $[m]$ into M groups $\{\mathcal{G}_0, \ldots, \mathcal{G}_{M-1}\}$, with each machine i on iteration t being responsible for computing the entries

$$(\widetilde{\boldsymbol{Z}}[t, j] \mid j \in \mathcal{G}_i) \qquad \text{for} \qquad \widetilde{\boldsymbol{Z}} = \boldsymbol{C}^{-1}\boldsymbol{Z}\,. \tag{4.13}$$

   For all three approaches above, this sharding strategy requires no communication between machines to compute the values of Eq. (4.13). Hence, noise computation is an entirely embarrassingly parallel computation and its running time is therefore inversely proportional to the number of machines in the training environment. Finally, the per-iteration noise vector $\widetilde{\boldsymbol{z}}_t$ must be assembled from Eq. (4.13) so that it can be added to the (clipped and aggregated) gradient which in general uses different sharding.

3. **Generating noise on CPU** is a third option worth considering,

where noise is generated on the CPUs while the forward/backward pass is done on an accelerator. At the surface, this may seem appealing because it allows the noise generation to happen asyncronously rather than sequentially. However, communicating large matrices from CPU to accelerators is typically slow on current hardware, and simple preliminary tests rule this out as a viable approach.

Remark 4.15 (Handling Complex Model Structures)**.** In the discussion above, we assumed that the model and noise are generated as flat vectors in $\mathbb{R}^m$, and that m is divisible by the number of machines. In practice, the model parameters are often represented as a collection of smaller arrays of various shapes, whose sizes may not all be divisible by the number of machines. This can be handled by appropriate flattening and padding. (The open-source software of Section 4.5 automatically implements such sub-routines.)

Remark 4.16 (Relative Cost of Noise Generation)**.** Until this section, we have discussed the costs of different correlated noise generation strategies relative to each other (e.g. in Table 2.1). While the Dense and Toeplitz mechanisms have a n times higher time complexity than Input and Output perturbation, they are not actually n times slower in practice, since correlated noise generation is only one part of the cost associated with training a model with differential privacy. The other primary cost is computing (and clipping) per-example gradients. Perhaps somewhat unintuitively, it turns out that the overhead of correlated noise generation is small relative to the other costs of DP training, *even when the number of buffers is large*. This can be attributed to careful implementation (e.g. described above) coupled with the need to use very large batch sizes to get the best utility with DP.

While the relative costs of each step are dependent on the model and compute environment, let us next consider a concrete example. For large transformer-based language models, the cost of gradient computation and clipping can be approximated as $\approx 6 \cdot m \cdot B \cdot S$

where B is the batch size and S is the sequence length. For common language models, S is typically taken to be in the thousands and B is at least as large as S for DP training. Hence this cost can often outweigh the $b \cdot m$ time complexity of noise generation required for e.g., a banded Toeplitz strategy for reasonable values of b, B, and S. As a concrete example, if $b = 100, B = S = 1000$, then noise generation requires $\leq 0.01\%$ of the total step time. On the other hand, if $S = 1$, which essentially reduces to a fully connected neural network, then noise generation requires closer to $\sim 10\%$ of the total step time.

4.5 Open-Source Software

As discussed in this section, correctly and efficiently optimizing for correlated noise mechanisms requires some care to correctly handle subtleties. Jax Privacy provides implementations of the techniques described in this section. At the time of writing, Jax Privacy provides well-tuned strategy optimization routines for dense, banded, banded toeplitz, and buffered linear toeplitz strategies. It also provides implementation of the various noise addition strategies discussed in Section 4.4.5 that can run efficiently in large distributed environments.

4.6 Bibliographic Notes

Dense Strategies Lemma 4.2, that underlies all numerical optimizations for the dense mechanism was established by Li, Miklau, Hay, McGregor, and Rastogi [2015]. Many approaches have been proposed to optimize the dense mechanism in the streaming setting, including primal approaches [Yuan, Yang, Zhang, and Hao, 2016, McKenna, Miklau, Hay, and Machanavajjhala, 2021, Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu, 2023a] and dual approaches [Denisov, McMahan, Rush, Smith, and Guha Thakurta, 2022, Choquette-Choo, McMahan, Rush, and Thakurta, 2023b]. McKenna, Miklau, Hay, and Machanavajjhala [2021] demonstrated the effectiveness of solving Problem 4.3 with L-BFGS, including the heuristic to ignore $\boldsymbol{M} \succ \mathbf{0}$. Using memoryless noise regeneration has used to demonstrate scalability up

to about $n \approx 2000$ for models with a few million parameters by several works like Denisov, McMahan, Rush, Smith, and Guha Thakurta [2022], Choquette-Choo, McMahan, Rush, and Thakurta [2023b], Choquette-Choo, Ganesh, McKenna, McMahan, Rush, Guha Thakurta, and Xu [2023a].

A note on L-BFGS L-BFGS can natively handle box constraints Zhu, Byrd, Lu, and Nocedal [1997]. However, more complex constraints require projections in the Mahalanobis norm defined by L-BFGS's Hessian approximation [Schmidt et al., 2011]. Instead, the heuristic used in this monograph is a Euclidean projection, making it directly compatible with existing highly-optimized L-BFGS implementations.

Parameterized Strategies In the counting query literature, there have been many prior works that optimize carefully parameterized strategies to overcome scalability limitations of the dense representation. Qardaji, Yang, and Li [2013] optimize the branching factor over generalized hierarchical strategies, while Li, Hay, Miklau, and Wang [2014] optimize the weighting coefficients for each level of the hierarchy. McKenna, Miklau, Hay, and Machanavajjhala [2021] propose optimizing so-called p-Identity strategies specifically for the pure-DP version of the problem, and strategies built from smaller building blocks via the Kronecker product. The approaches to optimize the BLT parameters with a $(\boldsymbol{\lambda}, \widehat{\boldsymbol{\lambda}})$ parameterization are due to Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024], McMahan, Xu, and Zhang [2024]. In particular, Lemma 4.13 results from a simplification of the result of Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024, Lem. 5.2]; the exact statement we give can be found in McMahan and Pillutla [2025]. On the other hand, the $(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\lambda}})$ parameterization approach, and the equivalence between the two parameterizations (i.e., Lemma 4.14) are from McMahan and Pillutla [2025].

Matrix mechanism in other settings Xiao, He, Zhang, and Kifer [2023b], Ding, Winslett, Han, and Li [2011], McKenna, Miklau, Hay, and Machanavajjhala [2021] propose matrix mechanisms specifically

tailored for marginal query workloads. McKenna, Miklau, Hay, and Machanavajjhala [2021] further provided optimized strategies for the broad class of conjunctive query workloads, that are applicable for domains as large as 10^9. Edmonds, Nikolov, and Ullman [2020] showed that the matrix mechanism can be applied under local DP with favorable theoretical guarantees, and McKenna, Maity, Mazumdar, and Miklau [2020] proposed a more practical variant that generalizes randomized response rather than the Gaussian mechanism.

5

Challenges and Open Questions

This monologue has explored the landscape of correlated noise mechanisms for privacy preserving machine learning, providing a structured overview of their theoretical foundations, practical applications, and privacy implications. Looking ahead, correlated noise mechanisms offer a promising frontier in privacy-preserving data analysis. As data privacy continues to grow in importance, correlated noise mechanisms will likely play a critical role in balancing the need for data utility with the imperative for individual privacy.

In the rest of this section, we explore some of the important open problems.

5.1 Client Participation in Weighted Prefix Sums

In practice and as shown in Section 3 and Section 4, private learning has two primary sources of stochasticity: (1) *client participation*, which arises through random sampling, system-driven availability, or shuffling; and (2) *noise added to ensure privacy*, such as correlated noise, i.i.d. noise, etc.

Most existing approaches in designing correlated noise mechanisms treat these two components somewhat independently. In particular,

participation is typically handled through heuristics such as defining participation schema or uniform random sampling, without regard to the implications for noise design. Conversely, correlated noise mechanisms are often built assuming idealized participation patterns, like b-MinSep, etc. This separation leads to inefficiencies: either the variance-reduction benefits of correlation are lost, or the system must be artificially constrained to preserve fragile noise structures. While the latter can be enforced, it makes the architecture a little more complicated than desired.

This raises a natural question:

> **Question 5.1.** Can we jointly design client participation strategies and correlated noise mechanisms to exploit their interaction for improved privacy-utility tradeoffs in the computation of weighted prefix sums?

5.2 Defining Factorization Losses that Reflect Learning Performance

The use of linearly-correlated noise mechanisms in (weighted) prefix sum estimation is well-understood, especially for max loss (Definition 2.3) or RMS-loss (Definition 2.23). In these two cases, the mechanism design reduces to a tractable optimization problem (more precisely, a semi-definite program). However, extending these methods to learning problems presents few challenges.

Currently, in all the works that uses correlated noise mechanisms for learning (see Section 3.7 for references), proxy losses are introduced that are typically derived from a related linear estimation problem with the *hope* that minimizing them leads to better learning performance. However, how close these proxy loss are in theoretical and empirical terms to the true learning dynamics remains an unresolved open question. In particular, while these proxies often serve as upper bounds on the learning error, whether they faithfully capture the behavior of real-world models, particularly across diverse function classes and worst-case dynamics, is still unclear.

Recent work has begun to investigate this gap between proxy losses

and actual learning performance. For example, Koloskova, McKenna, Charles, Rush, and McMahan [2023b] showed that there was not a monotonic relationship between losses imposed on typical factorization problems and learning performance, even for simple models. They proposed alternative analyses and adjusted loss functions that better track true performance. As discussed in Section 3.5.2, Choquette-Choo, Dvijotham, Pillutla, Ganesh, Steinke, and Thakurta [2024a] have provided an asymptotic characterization of performance under quadratic losses, relating them directly to properties of the noise design. These insights suggest that more faithful modeling of noise-influenced learning dynamics, especially using the linearly correlated noise mechanism, could guide improved mechanism design. By developing loss measures that more accurately reflect real learning behavior, one can hope to further improve privacy-preserving training algorithms.

An intriguing direction is to view noise mechanism design through the lens of learned optimizers. Much like learning an optimizer that performs well across tasks, we can interpret the design of noise mechanisms (e.g., DP-SGD with correlated noise) as an optimization problem over a space of algorithmic parameters (such as noise covariance structures). This analogy points to a promising opportunity: designing losses and optimization targets that better capture learning performance across tasks, without inheriting the challenges of overfitting faced by learned optimizers. By exploring richer loss parameterizations (e.g., quadratic or convex losses), one can aim to build a more robust and expressive framework for noise mechanism design.

> **Question 5.2.** How can we design proxy loss functions for linearly-correlated noise mechanisms that more faithfully capture and improve actual learning performance across a broad class of machine learning problems?

Solving this research question could ultimately lead to more effective and generalizable private learning algorithms, grounded in a better theoretical understanding of the connection between proxy objectives and learning performance.

5.3 Adaptive Private Optimization with Correlated Noise Mechanisms

The idea of casting learning algorithm design as an optimization problem over algorithmic spaces naturally leads to adaptive optimization methods such as AdaGrad, Adam, and RMSProp. These optimizers were originally motivated by meta-optimization or regret minimization frameworks, and remain central to training large-scale models like Transformer-based architectures, which benefit from per-parameter adaptive learning rates. Despite their widespread use and perceived necessity in modern machine learning workflows, integrating differentially private mechanism design, especially correlated noise mechanisms, into adaptive optimization remains an open challenge.

At the heart of this challenge lies the fundamental non-linearity introduced by adaptive optimizers: they typically perform pointwise division of gradients by functions of past gradients, breaking the linear stream-to-stream relationship between gradients and model updates that the design of correlated noise mechanisms relies on. This linearity is critical, as it ensures that loss functions (such as those quantifying the distance between noised and unnoised training trajectories) are independent of the data itself. Once this assumption fails, the proxy loss used in the optimization problem (e.g., Eq. (1.7)) becomes data-dependent, complicating both theoretical analysis and practical implementation.

> **Question 5.3.** How can linearly-correlated noise mechanisms be effectively extended to support adaptive optimization methods, either through novel regret-based analytical formulations or learned-optimizer-style numerical solutions, while preserving privacy guarantees and achieving competitive learning performance?

5.4 Stochastic Convex Optimization and Correlated Noise Mechanisms

A significant open question in differentially private optimization is whether the *stochastic convex optimization* (SCO) guarantees of correlated noise DP-SGD in the general setting can match the minimax

optimal bounds known for the realizable regime. As discussed in Section 3.5.2, Choquette-Choo, Ganesh, and Thakurta [2024] and Zhang, Tran, and Cutkosky [2022a] have shown promising advances in this direction by modifying the correlated noise DP-SGD to operate over sequences of *gradient differences* instead of raw gradients. This reformulation enables the algorithm to achieve the optimal error rates for DP-SCO previously believed to require multi-pass training or more complex mechanisms.

In particular, these works demonstrate that not only is it possible to attain the optimal DP-SCO guarantee using a *single-epoch* algorithm, but that this can be done with batch sizes as large as $\Omega(\sqrt{n})$. This is a notable improvement over standard instantiations of correlated noise DP-SGD, such as in Section 1.10.2, where each update is based on a batch of size one. The ability to scale batch sizes in this way is critical for practical applications and also reduces the number of noise-injected updates required during training.

A crucial component enabling this improvement is the use of correlated noise mechanisms applied to the sequence of gradient differences. Bounding the noise injected in this transformed domain becomes more tractable and can be done in a way that aligns with optimal privacy-utility tradeoffs. This innovation suggests that revisiting classic DP optimization pipelines with alternative sequence parameterizations may unlock further improvements.

Difference estimator has been used a lot in statistics and streaming algorithms. For example, using difference estimator has been used in the streaming algorithm literature to design more efficient robust streaming algorithms [Woodruff and Zhou, 2022]. In the context of differential privacy, by observing the difference sequence has bounded sensitivity, Fichtenberger et al. [2021] and Song et al. [2018] gave a differentially private algorithm for various graph statistics in the continual release model. However, application of these techniques are limited. This raises the following question:

> **Question 5.4.** Can the transformation to gradient-difference sequences be generalized beyond DP-SGD with correlated noise

mechanism to differentially private learning algorithms for other statistics or settings (like low-space privacy preserving algorithms or continual release algorithms for other statistics and settings), and under what conditions does this transformation preserve or improve privacy-utility tradeoffs?

5.5 Better Instantiation of DP-SGD with Correlated Noise

Extending the question of Section 5.1, and bringing together Sections 5.2 and 5.3, the bigger goal is to holistically design private training methods.

DP-SGD and its variant algorithms, such as the one outlined in Algorithm 1.3, present a flexible template for private model training. However, to turn this into a fully specified and high-performing learning algorithm, several critical design choices must be addressed as discussed in Section 3 and Section 4:

1. the data processing pattern—particularly the participation schema and its impact on privacy amplification, as seen in techniques like block Poisson sampling;

2. the structure of the strategy matrix $\boldsymbol{C}$ (or equivalently the noise-correlating matrix $\boldsymbol{C}^{-1}$), which governs how noise is injected and correlated across iterations; and

3. the choice of the first-order optimizer, which translates noisy gradient estimates into model updates.

As we saw in Section 3 and Section 4, each of these components interacts in subtle and sometimes nontrivial ways. This suggests that a piecemeal design approach may leave substantial performance gains that are not fully realized, and, hence, a lot of effort has been dedicated to understand these subtle interactions. For example, a recent thrust in the literature of privacy preserving learning has been motivated by the thesis that a *co-design* approach is necessary, one that jointly optimizes the data processing scheme, noise correlation strategy, and learning algorithm. For instance, the structured, banded design of $\boldsymbol{C}$ is essential for theoretical guarantees under block-cyclic Poisson sampling, while

the effectiveness of a proxy loss function (e.g., RMS-loss with respect to a prefix-sum workload matrix) likely depends on both the strategy matrix and the optimizer used.

Such interdependence suggests that better proxy losses, e.g. losses that are tailored to specific optimizer dynamics and workload structures, could drive more effective noise mechanism design. Furthermore, simple instantiations of DP-SGD may not suffice when moving toward adaptive optimization methods, which introduce additional complexity and opportunities for tuning noise injection and update behavior.

> **Question 5.5.** How can we develop a unified framework for the co-design of data sampling schemes, noise correlation strategies, and optimizers in DP-SGD, to achieve improved learning performance under differential privacy constraints—especially in the presence of adaptive optimization methods?

5.6 Resolving Conjecture 4.7

In designing optimal correlated noise mechanisms under a given participation schema Π, a central step involves solving an optimization problem over positive definite matrices $\boldsymbol{M}$, subject to sensitivity constraints. In analogy to the streaming setting, we consider a scale-invariant objective and normalize sensitivity via the constraint $\sum_{t\in\boldsymbol{\pi}} \boldsymbol{M}[t,t] \leq 1$ for each $\boldsymbol{\pi} \in \Pi$. In practice, however, this inequality is commonly replaced with an equality: $\sum_{t\in\boldsymbol{\pi}} \boldsymbol{M}[t,t] = 1$, which simplifies analysis and implementation.

While empirical evidence strongly supports that this substitution preserves optimality, no theoretical guarantee currently exists for the general case. Notably, in the streaming setting, this replacement is provably without loss of generality. Whether the same holds in the multi-epoch setting remains an open question. If the equality-constrained problem indeed yields an optimal solution for the original inequality-constrained formulation, it would justify current practices and simplify both the analysis and deployment of such mechanisms.

Question 5.6. Does the equality-constrained problem in Problem 4.6 yield a solution that is also optimal for the corresponding inequality-constrained version? In particular, under what conditions on the participation schema Π and workload matrix $\boldsymbol{A}$ does the solution $\boldsymbol{M}_\star$ to the equality-constrained formulation remain valid for the relaxed inequality-constrained problem?

As discussed in Section 4, resolving the above question would not only simplify the problem of multi-participation, but also improve the convergence rate of the optimization algorithms.

5.7 Correlated Noise Mechanisms for Streaming Prefix Sums

In this monograph, we primarily focused on one central application of differentially private prefix sum estimation: the training of machine learning models, particularly via private variants of stochastic gradient descent. However, the prefix sum primitive has far-reaching applications beyond private learning. It serves as a foundational tool in numerous other domains, including but not limited to histogram estimation, range query answering, shortest-path estimation on structured graphs, non-interactive local learning, graph spectral analysis, and matrix computation. In these settings, accuracy is typically assessed through metrics such as the root mean square loss (RMS-loss) or the maximum error (max loss) over the query outputs.

For the canonical prefix sum workload matrix $\boldsymbol{A} = \boldsymbol{A}_{\mathsf{pre}}$, strong theoretical foundations exist. In particular, tight upper and lower bounds on the RMS-loss are known not only in asymptotic terms but also up to constants. More generally, Liu, Upadhyay, and Zou [2024] showed that, for any positive constant p, we have asymptotically tight characterizations of the ℓ_p error defined as

$$\max_{\boldsymbol{x}\in\mathbb{R}^n} \left(\mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{x}\|_p^p\right]\right)^{1/p},$$

where $\mathcal{M}$ is a differentially private mechanism. These bounds reflect a mature understanding of the trade-offs between privacy and accuracy under a range of loss functions.

In contrast, for the max loss metric (or for $p = \omega(1)$), the picture is more nuanced. Using standard packing arguments, Dwork, Naor, Pitassi, and Rothblum [2010] showed that any $(1, o(1/\sqrt{n}))$-differentially private mechanism for the prefix sum problem must incur a max loss error of at least $\Omega(\log(n)/\varepsilon)$ in the worst case. Recently, Cohen, Lyu, Nelson, Sarlós, and Stemmer [2024] showed that, when the input vector is s-sparse, this lower bound becomes $\Omega(\min\{\log(s), \log(n)\})$. On the other hand, the best known upper bound on max loss for (ε, δ)-differential private mechanism is $O\left(\log^{3/2}(n) \cdot \sqrt{\log(1/\delta)}/\varepsilon\right)$ for input streams bounded in $[-1, 1]$.

> **Question 5.7.** The gap between best known upper and lower bound on max loss for differentially private prefix sum for n updates is $\log^{1/2}(n)$ dependence. Can we close this gap? Further, what is the optimal accuracy guarantee if the stream is s-sparse with respect to the sparsity parameter.

5.8 Amplification of Correlated Noise Mechanism Under Sliding Window

This monograph has explored the theory and practice of correlated noise mechanisms for differentially private learning, starting with their application to prefix sum estimation and culminating in their use for training large-scale models under privacy constraints.

One compelling direction for future work, which naturally extends the models discussed in this monograph, is the **sliding window model**. In many modern applications of machine learning—especially in online learning, federated learning, or real-time analytics—*more recent data are more relevant* to current predictions and model updates. In these settings, it may be desirable to estimate statistics (e.g., means, prefix sums, or gradients) over only the most recent w observations, rather than over the full history. This calls for differentially private mechanisms tailored to *sliding window estimation*, where the goal is to release private estimates of functions computed over a moving window of size w, rather than over an accumulating prefix.

From a privacy mechanism design standpoint, this introduces new

challenges and opportunities. Mechanisms must now adapt not only to the prefix structure but also to *temporal locality*—ensuring that noise is injected in a way that respects the decay in importance of older data. Correlated noise mechanisms, already well-suited to structured time dependencies, provide a promising starting point. For instance, designing strategy matrices $\boldsymbol{C}$ or workload matrices $\boldsymbol{A}$ with banded or decaying influence patterns could yield effective mechanisms for sliding window analytics. Moreover, privacy accounting in the sliding window model may require novel analyses, as traditional notions of sensitivity and amplification must be redefined for moving-window adjacency relations.

> **Question 5.8.** How can we extend correlated noise mechanisms to the sliding window model in a way that reflects the heightened importance of recent data while maintaining rigorous privacy guarantees and high utility in private estimation and training?

This question opens a rich design space that intersects with time-series privacy, streaming algorithms, and adaptive learning. Addressing it could lead to more temporally aware privacy mechanisms—particularly relevant for real-time deployment scenarios where recent user interactions carry more signal than stale historical data.

5.9 A Functional Analysis Perspective on Sensitivity

The computation of sensitivity (e.g. defined in terms of participation schema by Definition 3.13) can be viewed as an operator norm with interesting structure.

Most of the sensitivity calculations in Section 3 factor through Lemma 3.15, which can be significantly suboptimal in the presence of cancellation (e.g., via entries of mixed sign) in the matrix $\boldsymbol{C}^\top \boldsymbol{C}$. In fact, precise control of this operator-norm cancellation was a problem posed by Grothendieck in the 1950's, who showed a uniform bound in all dimensions:

> **Theorem 5.1.** Consider a matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ with entries $w_{ij} := \boldsymbol{W}[i, j]$ and suppose that it has unit ℓ_∞ to ℓ_1 operator norm (defined

in Eq. (3.14)):

$$\|\boldsymbol{W}\|_{\infty\to 1} \le 1 \quad \iff \quad \left|\sum_{i,j=0}^{n-1} w_{ij} t_i s_j\right| \le 1 \ \forall\ |t_i| \le 1, |s_j| \le 1\,.$$

Then, for every set of unit vectors $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$, $\{\boldsymbol{y}_j\}_{j=0}^{n-1}$ in a Hilbert space $\mathbb{H}$ (endowed with an inner product $\langle\cdot,\cdot\rangle_{\mathbb{H}}$), we have the uniform bound

$$\left|\sum_{i,j=0}^{n-1} w_{ij}\langle \boldsymbol{x}_i, \boldsymbol{y}_j\rangle_{\mathbb{H}}\right| \le K,$$

where K is an absolute constant known as Grothendieck's constant.

Grothendieck's constant K is known to be between 1.57 and 1.7822 in general, though in the application of Grothendieck's theorem to calculation of sensitivity one may leverage symmetry to achieve the value $\pi/2$. As has been noted previously, Grothendieck's theorem provides an alternate means of estimating the ℓ_2 sensitivity of a strategy matrix $\boldsymbol{C}$, as one may rewrite the trace which appears in the proof of Lemma 3.15 into a form to which Grothendieck's theorem applies.

If we restrict the matrices $\boldsymbol{W}$ in Theorem 5.1, K may be significantly improved. For example, the arguments of Lemma 3.15 show that for elementwise nonnegative $\boldsymbol{W}$, one may take $K = 1$ (though this represents a tiny fraction of invertible matrices). In some sense, one may view the restrictions to nonnegative $\boldsymbol{C}^\top\boldsymbol{C}$ while optimizing for nontrivial Π as primarily playing the role of *avoiding* the need to pay the $\pi/2$ cost that would otherwise be implied by Theorem 5.1.

Whether optimal or near-optimal $\boldsymbol{C}$ satisfy $\boldsymbol{C}^\top\boldsymbol{C} \ge 0$ is in many cases not known, and is in principle be dependent on the workload matrix $\boldsymbol{A}$. The presence of the Grothendieck constant in vector-to-scalar reductions is itself Π-dependent; this concern disappears in the streaming (single participation) setting. Thus the current state of theory relating $\boldsymbol{A}$, Π, and structure in $\boldsymbol{C}$ is somewhat unsatisfying: we know that in many cases of interest, we may suppress the factor of $\pi/2$ by imposing appropriate structure on $\boldsymbol{C}$, but we do not in principle know what we are losing in the optimization problem by enforcing this restriction.

Question 5.9. Are there settings of $\boldsymbol{A}, \Pi$ pairs in which loosening structure on $\boldsymbol{C}$ permits significantly improved solutions? In the case of nontrivial Π, are there large classes of matrices for which K can be lowered and yield useful extensions of the space of permissible $\boldsymbol{C}$ matrices? If not, why not?

Given the practical success of banded, Toeplitz, and BLT mechanisms for which Lemma 3.15 applies, this question is perhaps slightly academic, but represents a tempting gap in our theoretical understanding with connections to a deep result in functional analysis.

5.10 Characterizing the error of optimal BLTs

Dvijotham, McMahan, Pillutla, Steinke, and Thakurta [2024] showed that BLTs constructed via rational function approximation can achieve (up to a small additive factor) the optimal max loss of $\log(n)/\pi$ using $d = \Theta(\log^2 n)$ buffers, using the same BLT (for a given d) for all possible n. However, direct numerical optimization of the BLT parameters for a specific n (that is, the approach of Section 4.3.1) yields substantially better mechanisms. This naturally yields the following questions:

Question 5.10. Can the BLT parameters that minimize max loss for a specific n and specific number of buffers d be characterized in closed form (without numerical optimization)? Can a tight bound on max loss be given for these parameters? And most importantly, how many buffers d are necessary to achieve the optimal rate of $\log(n)/\pi$? Empirically evidence leads us to conjecture $d = \Theta(\log n)$ is sufficient.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

Joel Daniel Andersson and Rasmus Pagh. A Smooth Binary Mechanism for Efficient Private Continual Observation. *Advances in Neural Information Processing Systems*, 36, 2023.

Meenatchi Sundaram Muthu Selva Annamalai, Borja Balle, Emiliano De Cristofaro, and Jamie Hayes. To shuffle or not to shuffle: Auditing dp-sgd with shuffling. *arXiv preprint arXiv:2411.10614*, 2024.

Differential Privacy Team Apple. Learning with Privacy at Scale. 2017.

Hilal Asi, Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: Optimal rates in l1 geometry. In *International Conference on Machine Learning*, pages 393–403. PMLR, 2021a.

Hilal Asi, Daniel Asher Nathan Levy, and John Duchi. Adapting to function difficulty and growth conditions in private optimization. In *Advances in Neural Information Processing Systems*, 2021b.

Hilal Asi, Vitaly Feldman, Tomer Koren, and Kunal Talwar. Near-optimal algorithms for private online optimization in the realizable regime. In *International Conference on Machine Learning*, pages 1107–1120. PMLR, 2023.

George A Baker Jr and John L Gammel. The Padé Approximant. *Journal of Mathematical Analysis and Applications*, 2(1):21–30, 1961.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 464–473, 2014a.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014b.

Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Thakurta. Private stochastic convex optimization with optimal rates. In *Advances in Neural Information Processing Systems*, pages 11279–11288, 2019.

Raef Bassily, Vitaly Feldman, Cristóbal Guzmán, and Kunal Talwar. Stability of stochastic gradient descent on nonsmooth convex losses. *arXiv preprint arXiv:2006.06914*, 2020.

Raef Bassily, Cristóbal Guzmán, and Anupama Nandi. Non-euclidean differentially private stochastic convex optimization. In *Conference on Learning Theory*, pages 474–499. PMLR, 2021.

Grahame Bennett. Schur multipliers. 1977.

Jean Bolot, Nadia Fawaz, Shanmugavelayutham Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pages 284–295, 2013.

Dietrich Braess and Wolfgang Hackbusch. Approximation of $1/x$ by exponential sums in $[1, \infty)$. *IMA journal of numerical analysis*, 25 (4):685–697, 2005.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 635–658, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53641-4.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011.

Zachary Charles, Arun Ganesh, Ryan McKenna, H Brendan McMahan, Nicole Mitchell, Krishna Pillutla, and Keith Rush. Fine-Tuning Large Language Models with User-Level Differential Privacy. In *SaTML*, 2025.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

Chao-Ping Chen and Feng Qi. The best bounds in wallis' inequality. *Proceedings of the American Mathematical Society*, 133(2):397–401, 2005.

Xiangyi Chen, Zhiwei Steven Wu, and Mingyi Hong. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. In *NeurIPS*, 2020.

Christopher A Choquette-Choo, Arun Ganesh, Ryan McKenna, H Brendan McMahan, John Rush, Abhradeep Guha Thakurta, and Zheng Xu. (Amplified) Banded Matrix Factorization: A unified approach to private training. *Advances in Neural Information Processing Systems*, 36, 2023a.

Christopher A Choquette-Choo, H Brendan McMahan, Keith Rush, and Abhradeep Thakurta. Multi-Epoch Matrix Factorization Mecha-

nisms for Private Machine Learning. In *International Conference on Machine Learning*, volume 202, pages 5924–5963. PMLR, 2023b.

Christopher A Choquette-Choo, Krishnamurthy Dvijotham, Krishna Pillutla, Arun Ganesh, Thomas Steinke, and Abhradeep Thakurta. Correlated Noise Provably Beats Independent Noise for Differentially Private Learning. In *ICLR*, 2024a.

Christopher A Choquette-Choo, Arun Ganesh, Saminul Haque, Thomas Steinke, and Abhradeep Thakurta. Near Exact Privacy Amplification for Matrix Mechanisms. *arXiv Preprint*, 2024b.

Christopher A. Choquette-Choo, Arun Ganesh, Thomas Steinke, and Abhradeep Guha Thakurta. Privacy Amplification for Matrix Mechanisms. In *International Conference on Learning Representations*, 2024.

Christopher A Choquette-Choo, Arun Ganesh, and Abhradeep Thakurta. Optimal rates for dp-sco with a single epoch and large batches. *arXiv preprint arXiv:2406.02716*, 2024.

Lynn Chua, Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. How private are dp-sgd implementations? *arXiv preprint arXiv:2403.17673*, 2024a.

Lynn Chua, Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. Scalable dp-sgd: Shuffling vs. poisson subsampling. *Advances in Neural Information Processing Systems*, 37:70026–70047, 2024b.

Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. Lower bounds for differential privacy under continual observation and online threshold queries. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1200–1222. PMLR, 2024.

Robert Corless, Tetsuo Ida, and Hoon Hong. *The Concrete Tetrahedron: Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Springer, 2011.

Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Answering Range Queries Under Local Differential Privacy. *VLDB*, 12(10): 1126–1138, 2019.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking High-Accuracy Differentially Private Image Classification through Scale. *arXiv Preprint*, 2022.

Sergey Denisov, H Brendan McMahan, John Rush, Adam Smith, and Abhradeep Guha Thakurta. Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams. In *Advances in Neural Information Processing Systems*, volume 35, pages 5910–5924, 2022.

Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 217–228, 2011.

Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian Differential Privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 02 2022. ISSN 1369-7412. doi: 10.1111/rssb.12454. URL https://doi.org/10.1111/rssb.12454.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2011.

Max Dupré la Tour, Monika Henzinger, and David Saulpic. Making old things new: a unified algorithm for differentially private clustering. In *Proc. 41th ICML*, 2024.

Krishnamurthy Dvijotham, H. Brendan McMahan, Krishna Pillutla, Thomas Steinke, and Abhradeep Thakurta. Efficient and Near-Optimal Noise Generation for Streaming Differential Privacy. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2024.

Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51, 2016.

Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. The power of factorization mechanisms in local and central differential privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 425–438, 2020.

Joseph Frederick Elliott. The characteristic roots of certain real symmetric matrices. 1953.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *ACM SIGSAC*, pages 1054–1067, 2014.

Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: Optimal rates in linear time. In *Proc. of the Fifty-Second ACM Symp. on Theory of Computing (STOC'20)*, 2020.

Hendrik Fichtenberger, Monika Henzinger, and Lara Ost. Differentially private algorithms for graphs under continual observation. *arXiv preprint arXiv:2106.14756*, 2021.

Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. Constant Matters: Fine-grained Error Bound on Differentially Private

Continual Observation. In *International Conference on Machine Learning*, 2023.

Sivakanth Gopi, Yin Tat Lee, and Daogao Liu. Private convex optimization via exponential mechanism. *arXiv preprint arXiv:2203.00263*, 2022.

Robert Todd Gregory and David L Karney. A collection of matrices for testing computational algorithms. *(No Title)*, 1969.

Benjamin Grimmer. A Collection of Induced Norm Balls. 2022. URL https://www.ams.jhu.edu/~grimmer/Induced.pdf.

Uffe Haagerup. Decomposition of completely bounded maps on operator algebras, 1980.

Uffe Haagerup and Gilles Pisier. Bounded linear operators between c^*-algebras. *Duke Mathematical Journal*, 71(3):889–925, 1993.

Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *VLDB*, 3(1):1021–1032, 2010.

Monika Henzinger and Jalaj Upadhyay. Improved Differentially Private Continual Observation Using Group Algebra. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2025.

Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. Almost Tight Error Bounds on Differentially Private Continual Counting. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5003–5039. SIAM, 2023.

Kenneth Hoffman. *Linear algebra*. 1971.

James Honaker. Efficient use of differentially private binary trees. *Theory and Practice of Differential Privacy (TPDP 2015), London, UK*, 2:26–27, 2015.

Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially Private Online Learning. In *Conference on Learning Theory*, pages 24–1, 2012.

Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and Private (Deep) Learning without Sampling or Shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021a.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021b.

Nikita Kalinin and Christoph Lampert. Banded Square Root Matrix Factorization for Differentially Private Model Training. *arXiv Preprint*, 2024.

Nikita P Kalinin, Ryan McKenna, Jalaj Upadhyay, and Christoph H Lampert. Back to square roots: An optimal bound on the matrix factorization error for multi-epoch differentially private sgd. *arXiv preprint arXiv:2505.12128*, 2025.

Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.

Anastasia Koloskova, Hadrien Hendrikx, and Sebastian U. Stich. Revisiting Gradient Clipping: Stochastic bias and tight convergence guarantees. In *ICML*, volume 202, pages 17343–17363, 2023a.

Anastasiia Koloskova, Ryan McKenna, Zachary Charles, John Rush, and H. Brendan McMahan. Gradient descent with linearly correlated noise: Theory and applications to differential privacy. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 35761–35773. Curran Associates, Inc., 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70255afc962aca0930327c090eb7d8c5-Paper-Conference.pdf.

Janardhan Kulkarni, Yin Tat Lee, and Daogao Liu. Private non-smooth erm and sco in subquadratic steps. *Advances in Neural Information Processing Systems*, 34, 2021.

Stanisław Kwapień and Aleksander Pełczyński. The main triangle projection in matrix spaces and its applications. *Studia Mathematica*, 34(1):43–67, 1970.

Troy Lee, Adi Shraibman, and Robert Špalek. A Direct Product Theorem for Discrepancy. In *2008 23rd Annual IEEE Conference on Computational Complexity*, pages 71–80. IEEE, 2008. URL https://web.archive.org/web/20230109145533/https://www2.mta.ac.il/~adish/Pubs/Papers/DPTDiscrepancy.pdf.

Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. A data-and workload-aware algorithm for range queries under differential privacy. *arXiv preprint arXiv:1410.0265*, 2014.

Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24:757–781, 2015.

Jingcheng Liu, Jalaj Upadhyay, and Zongrui Zou. Optimality of matrix mechanism on ℓ_p^p-metric. *arXiv preprint arXiv:2406.02140*, 2024.

Noah Marshall, Ke Liang Xiao, Atish Agarwala, and Elliot Paquette. To Clip or not to Clip: the Dynamics of SGD with Gradient Clipping in High-Dimensions. *arXiv Preprint*, 2024.

Roy Mathias. The hadamard operator norm of a circulant and applications. *SIAM journal on matrix analysis and applications*, 14(4): 1152–1167, 1993.

Jiří Matoušek, Aleksandar Nikolov, and Kunal Talwar. Factorization Norms and Hereditary Discrepancy. *International Mathematics Research Notices*, 2020(3):751–780, 2020. URL https://arxiv.org/abs/1408.1376.

Ryan McKenna. Scaling up the Banded Matrix Factorization Mechanism for Differentially Private ML. *arXiv preprint arXiv:2405.15913*, 2024.

Ryan McKenna, Raj Kumar Maity, Arya Mazumdar, and Gerome Miklau. A workload-adaptive mechanism for linear queries under local differential privacy. *Proceedings of the VLDB Endowment*, 13 (11), 2020.

Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Hdmm: Optimizing error of high-dimensional statistical queries under differential privacy. *arXiv preprint arXiv:2106.12118*, 2021.

Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533. JMLR Workshop and Conference Proceedings, 2011.

Brendan McMahan and Krishna Pillutla. An Inversion Theorem for Buffered Linear Toeplitz (BLT) Matrices and Applications to Streaming Differential Privacy. *Preprint*, 2025.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning Differentially Private Recurrent Language Models. In *ICLR*, 2018.

H Brendan McMahan, Zheng Xu, and Yanxiang Zhang. A Hassle-free Algorithm for Private Learning in Practice: Don't Use Tree Aggregation, Use BLTs. In *EMNLP*, 2024.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

D. J. Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11(1):11 – 14, 1964. doi: 10.1307/mmj/1028999029. URL https://doi.org/10.1307/mmj/1028999029.

Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The Geometry of Differential Privacy: the Sparse and Approximate Cases. *SIAM Journal on Computing*, 45(2):575–616, 2016.

Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative Prediction. In *International Conference on Machine Learning*, pages 7599–7609. PMLR, 2020.

Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, 2023.

Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013.

Fabian Schaipp, Guillaume Garrigos, Umut Simsekli, and Robert Gower. SGD with Clipping is Secretly Estimating the Median Gradient. *arXiv Preprint*, 2024.

Mark Schmidt, Dongmin Kim, and Suvrit Sra. Projected Newton-type Methods in Machine Learning. 2011.

Adam D. Smith, Abhradeep Thakurta, and Jalaj Upadhyay. Is Interaction Necessary for Distributed Private Learning? In *IEEE Symposium on Security and Privacy*, pages 58–77, 2017.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

Shuang Song, Susan Little, Sanjay Mehta, Staal Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics. *arXiv preprint arXiv:1809.02575*, 2018.

Daureen Steinberg. Computation of matrix norms with applications to robust optimization. *Research thesis, Technion-Israel University of Technology*, 2, 2005.

Thomas Steinke. Composition of Differential Privacy & Privacy Amplification by Subsampling, 2022. URL https://arxiv.org/abs/2210.00597.

Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

Abhradeep Guha Thakurta and Adam Smith. (Nearly) Optimal Algorithms for Private Online Learning in Full-information and Bandit Settings. *Advances in Neural Information Processing Systems*, 26, 2013.

Joel Aaron Tropp. Topics in Sparse Approximation. 2003.

US Census Bureau. Census Bureau Sets Key Parameters to Protect Privacy in 2020 Census Results. https://www.census.gov/newsroom/press-releases/2021/2020-census-key-parameters.html, 2021. Accessed: 2025-03-21.

Salil P. Vadhan. The Complexity of Differential Privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer International Publishing, 2017.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

David P Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1183–1196. IEEE, 2022.

Hanshen Xiao, Zihang Xiang, Di Wang, and Srinivas Devadas. A Theory to Instruct Differentially-Private Learning via Clipping Bias Reduction. In *IEEE Symposium on Security and Privacy (SP)*, pages 2170–2189. IEEE, 2023a.

Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. *Advances in Neural Information Processing Systems*, 22, 2009.

Yingtai Xiao, Guanlin He, Danfeng Zhang, and Daniel Kifer. An optimal and scalable matrix mechanism for noisy marginals under convex loss functions. *Advances in Neural Information Processing Systems*, 36: 20495–20539, 2023b.

Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher A. Choquette-Choo, Peter Kairouz, H. Brendan McMahan, Jesse Rosenstock, and Yuanbo Zhang. Federated Learning of Gboard Language Models with Differential Privacy. In *ACL (Industry Track)*, pages 629–639, 2023.

Ganzhao Yuan, Yin Yang, Zhenjie Zhang, and Zhifeng Hao. Convex optimization for linear query processing under approximate differential privacy. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2005–2014, 2016.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In *ICLR*, 2020.

Qinzi Zhang, Hoang Tran, and Ashok Cutkosky. Differentially private online-to-batch for smooth losses. In *NeurIPS*, 2022a.

Xinwei Zhang, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Jinfeng Yi. Understanding Clipping for Federated Learning: Convergence and Client-Level Differential Privacy. In *ICML*, 2022b.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.