Differentially Private Range Queries with Correlated Input Perturbation

Prathamesh Dharangutte **Jie Gao** **Ruobin Gong** **Guanyang Wang**
Rutgers University
{prathamesh.d, jg1555, ruobin.gong, guanyang.wang}@rutgers.edu

Abstract

This work proposes a class of differentially private mechanisms for linear queries, in particular range queries, that leverages correlated input perturbation to simultaneously achieve unbiasedness, consistency, statistical transparency, and control over utility requirements in terms of accuracy targets expressed either in certain query margins or as implied by the hierarchical database structure. The proposed Cascade Sampling algorithm instantiates the mechanism exactly and efficiently. Our theoretical and empirical analysis demonstrates that we achieve near-optimal utility, effectively compete with other methods, and retain all the favorable statistical properties discussed earlier.

1 Introduction

In this paper we construct a class of differentially private mechanisms for linear queries, including range queries, representable as a multiplicative operation of a pre-specified workload matrix and a confidential database. Our work is strongly motivated by the application of differential privacy to the 2020 U.S. Decennial Census, providing redistricting (P.L. 94-171) as well as Demographic and Housing Characteristic (DHC) files in the forms of multi-resolutional tabular data [3]. Population tabulations across geographic resolutions follow a hierarchical system termed the "spine" [8], which orders from top to bottom geographic entities (states, counties, tracts, block groups, and blocks), with higher-level geographies partitioned by the lower-level ones. As the only high-profile deployment of differential privacy in the public domain, the particular demands from the Census and similar official data products reveal a number of crucial considerations on privacy mechanisms that are possibly shared in a broader set of practical application scenarios. We discuss these considerations, highlighting consistency and utility control as the most prominent.

Consistency (internal): The sanitized output may plausibly be viewed as having been queried directly from an input database without modification. In particular, the data output for a geographical range (e.g. a state) should be precisely the sum of data values from its constituent sub-ranges (e.g. all the counties in a state). This is an example of a broader family of logical consistency that ensures stability and absence of conflicts in the data output.

Fine-grained utility control: The mechanism accommodates custom, externally specified utility requirements expressed as accuracy targets in query margins or implied by the hierarchical database structure. For example, Census tabulations at lower and intermediate geographies, as does certain "off-spine" geographies (e.g. voting districts), must meet accuracy targets according to the relevant operational standards [40]. Moreover, population counts across larger geographical regions at a lower resolution may not be permitted to have a greater error margin compared to smaller geographical regions at a higher resolution. For example, the mean error and mean absolute errors of total population counts in the Census DHC files remain consistent at the state, county, tract, and block group levels [41]. With the exception of a number of very recent work [43, 45], fine-grained utility control has been scarcely discussed in the DP literature, as the focus has been predominantly placed on the assessment of overall utility (such as average or worst-case). The Census application raised this issue in an interesting angle – that the algorithm designer is given both custom specified, fine-grained utility targets as well as privacy budget target, and must work backward to meet both objectives.

Furthermore, unbiasedness and statistical trans-

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

parency influence both the quality and usability of the data product [18, 20]. With **unbiasedness**, the sanitized output exhibits no bias with respect to the ground truth; With **statistical transparency**, the probabilistic description of the sanitized output is analytically tractable (ideally in a closed-form) to enable reliable downstream statistical inferences. Last, it is always desirable to have **efficient implementation** – the algorithm is exact and simple to implement, with no need for approximate simulation (e.g. Markov chain Monte Carlo) nor optimization-based post-processing. We will review the Census Bureau's *TopDown algorithm* [3] as well as other DP mechanisms with respect to these considerations later.

This paper considers *input perturbation* – adding Gaussian noises at each input data items and answering queries with the perturbed data. Classic input perturbation mechanisms naturally support unbiasedness, logical consistency, and statistical transparency, and are practical to implement. However, they do not support fine utility control and typically result in poor data utility, an issue that worsens when the query range contains a large number of data elements [6]. For this reason, input perturbation methods have been largely under-utilized in practice.

In this work we consider input perturbation with *correlated Gaussian noise*, which reduces error magnitude and offers fine control over utility while harnessing its many strengths. The proposed mechanism carefully couples the item-wise noises to allow queries at different hierarchical levels to conform to a *uniform* accuracy standard while achieving *near-optimal* overall utility objectives, both theoretically and empirically. We show for 1D range queries, the proposed mechanism achieves optimal mean square error and near-optimal worst-case and expected worst-case errors when compared to prevailing (ϵ, δ)-DP mechanisms. The special error correlation structure supports a linear time efficient implementation called the Cascade Sampling algorithm. Importantly, the fine control over data utility at different levels of geography is inherent to the design of the proposed mechanism, rather than reliant on optimization-based post-processing which may destroy transparency and render unpredictable accuracy. Our proposal generalizes to other hierarchical and multidimensional linear query settings.

2 Problem Formulation

Given a confidential data vector $\boldsymbol{x}$ of dimension n, and a workload matrix $\boldsymbol{W}$ of dimension $p \times n$, we would like to report a (possibly) noisy version of the query answer $\boldsymbol{Wx}$ while preserving the privacy of individual data elements in $\boldsymbol{x}$. $\boldsymbol{W}$ is an incidence matrix with rows corresponding to queries and columns corresponding to data elements. Specifically, we consider an (ε, δ)-differentially private mechanism $\mathcal{W}_{\varepsilon,\delta}$ which satisfies for any two neighboring databases $\boldsymbol{x}, \boldsymbol{x}'$, $||\boldsymbol{x}-\boldsymbol{x}'||_1 \leq 1$, and any set D of output values

$$\mathbb{P}\left[\mathcal{W}_{\varepsilon,\delta}(\boldsymbol{x}) \in D\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{W}_{\varepsilon,\delta}(\boldsymbol{x}') \in D\right] + \delta.$$

The linear query framework models many scenarios in practice. Three are particularly relevant to this work. *Predicate counting queries* report the number of database rows that satisfy the given predicate q, which are encoded into the rows of the workload matrix $\boldsymbol{W}$. *Range queries* report the sum of elements (or coordinates; x_i) that fall inside a given range, such as a time interval $[\ell, r]$ (e.g. streaming data) or a two-dimensional geographic area, with the structure of the range reflected in $\boldsymbol{W}$. *Contingency tables* are multidimensional histograms of entities satisfying certain composite attributes in a database. They can be regarded as a special type of high-dimensional range query, and are used extensively by statistical agencies for data processing and dissemination.

2.1 Definitions

Definition 2.1. A mechanism $\mathcal{W}_{\varepsilon,\delta}$ is *unbiased* if $\mathbb{E}(\mathcal{W}_{\varepsilon,\delta}(\boldsymbol{x})) = \boldsymbol{Wx}$, where the expectation is taken over the randomness of $\mathcal{W}_{\varepsilon,\delta}$.

That is, unbiasedness forbids a privacy mechanism from injecting systematic drift into the data output.

Definition 2.2 ([20], Def. 3)**.** A privacy mechanism $\mathcal{W}$ is *statistically transparent* if the conditional distribution of its output given the input, $p_{\xi}(\mathcal{W} = w \mid \boldsymbol{x} = x)$, is analytically available up to p and ξ, where ξ is the parameter for p (both tuning and auxiliary).

Statistical transparency is not frequently discussed in the literature of private mechanism design, but it is crucial if the sanitized output is subject to further data analysis as it provides the basis for valid statistical uncertainty quantification [20].

Definition 2.3. A mechanism $\mathcal{W}_{\varepsilon,\delta}$ operating on the data vector $\boldsymbol{x}$ is internally *consistent* if with probability one (over the randomness of $\mathcal{W}_{\varepsilon,\delta}$) there exists a vector $\boldsymbol{x}'$ such that $\mathcal{W}_{\varepsilon,\delta}(\boldsymbol{x}) = \boldsymbol{Wx}'$.

First defined for contingency tables and generalizable to any query, consistency requires the sanitized query to be a legitimate output of the intended query applied to a potential input database [4, 26, 6]. It is particularly important if the sanitized output enters directly into downstream decisions and is expected to be free of internal logical conflicts. In the literature, external consistency has also been discussed.

For example, state-level populations must be exactly reported per their constitutional purpose for reapportionment – an "invariant" requirement enforced externally; see [19, 9]. This is not discussed in this paper.

Remark 2.4. For linear queries, consistency requires the sanitized output to be in the column space of $\boldsymbol{W}$. Any logical relationship embodied in $\boldsymbol{W}$ (e.g. one range being the union of two disjoint ranges) is mirrored in $\mathcal{W}_{\varepsilon,\delta}$. Additive mechanisms of the form $\boldsymbol{W}\boldsymbol{x}+\boldsymbol{e}$ may not automatically obey consistency, unless $\boldsymbol{e}$ is guaranteed to be in the column space of $\boldsymbol{W}$. The same is true for exponential mechanisms unless the range is intentionally restricted [39].

3 Correlated input perturbation mechanism

This section presents the design of the correlation matrix, an efficient algorithm to sample from this distribution, and the resulting privacy and utility guarantees. Proof of our results are postponed to Appendix B and extensions to general binary tree and 2-D data are discussed in Appendix D.

3.1 Correlation matrix

To simplify matters, we consider n data points on a one-dimensional line and assume that n is a power of 2, denoted as $n = 2^k$. Extensions will be discussed in Section D. Each data point will be represented by its binary form, utilizing k bits, as leaves of a perfect binary tree with height k. Nodes in the tree receive labels based on their positions in a level-order traversal. For example, the root node is labelled as $\varnothing$, the nodes at depth 1 will have labels "0" and "1", and so on.

Our objective is to allocate Gaussian random noises, denoted as $\{X_I\}_{I\in\{0,1\}^k}$, to every data point (i.e. leaf node). The noise imposed on each internal node is the sum of the noises of its two children. With our labeling convention, this relationship can be succinctly represented as $X_\star = X_{\star 0} + X_{\star 1}$, where $\star$ stands for any binary sequence (including the empty one) with a length less than k. If all the noises on the leaf nodes are independently and identically distributed (i.i.d.), we anticipate that the noise introduced at the root will have a Gaussian variance of $\Theta(n)$. However, by carefully coupling the Gaussian variables on the leaf nodes, we can establish *uniform* variance across *all* nodes in the binary tree. The structure of our correlation matrix is recursively defined below:

Definition 3.1. J_i is the all-one matrix of size $2^i \times 2^i$. $\boldsymbol{C}_i$ is a square matrix of size $2^i \times 2^i$:

$$\boldsymbol{C}_1 := \begin{pmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{pmatrix},\ \boldsymbol{C}_{i+1} := \begin{pmatrix} \boldsymbol{C}_i & -\frac{J_i}{2^{2i+1}} \\ -\frac{J_i}{2^{2i+1}} & \boldsymbol{C}_i \end{pmatrix}. \tag{1}$$

Definition 3.2. For $n = 2^k$ data points identified by binary representation, our correlated noise mechanism is defined as $\mathsf{Noise} = \sigma Z$. Here, $Z \sim \mathbb{N}(\boldsymbol{0}, \boldsymbol{C}_k)$, and σ depends on a later-specified privacy budget. This mechanism applies to any private data vector $\boldsymbol{x}$, resulting in the output $\boldsymbol{x} + \mathsf{Noise}$.

The structure outlined in Definition 3.2 has a clear recursive pattern. One can split the 2^k data points into a left subtree, where labels begin with 0, and a right subtree, where labels begin with 1. The collection of points within each group mirrors a perfect binary tree of depth $k-1$ with a covariance matrix of $\sigma^2\boldsymbol{C}_{k-1}$. Points belonging to different groupings have a slightly negative correlation of -2^{-2k+1}. Similarly, points in the left subtree can be further divided based on those starting with 00 and those beginning with 01. Each of these smaller sub-groupings exhibits a covariance of $\boldsymbol{C}_{k-2}$, and any pair of points from these groups share a correlation of -2^{-2k+3}. This process can be recursively applied until each group is reduced to a single data point.

The next result shows that the variance of each internal node is the same as that of every leaf node.

Theorem 3.3. *Consider $n = 2^k$ data points identified by their binary representation as described earlier. Assuming the noise mechanism is defined as per Definition 3.2, every node in the binary tree, including both leaf and internal nodes, has a marginal distribution of $\mathbb{N}(0, \sigma^2)$. This implies that each node shares an identical variance.*

3.2 Cascade Sampling algorithm

This section introduces an efficient algorithm of running time $O(n)$, where $n = 2^k$ represents the total number of data points, for generating samples from our defined noise mechanism, specifically $\mathbb{N}(0, \sigma^2 C_k)$ as defined in Formula (1). This method significantly improves standard Gaussian generation methods in scalability, as supported by theoretical and numerical evidence.

Assuming $\sigma = 1$ without loss of generality, the standard method for sampling an n-dimensional Gaussian $\mathbb{N}(\mu, \Sigma)$ involves Cholesky decomposition of the covariance matrix $\Sigma = LL^\top$, followed by transforming a standard Gaussian vector ($\mathbf{x} \sim \mathbb{N}(0, \mathbb{I}_n)$) using $L\mathbf{x}+\mu$. This process has a high computational cost, primarily due to the $O(n^3)$ expense of the Cholesky decompo-

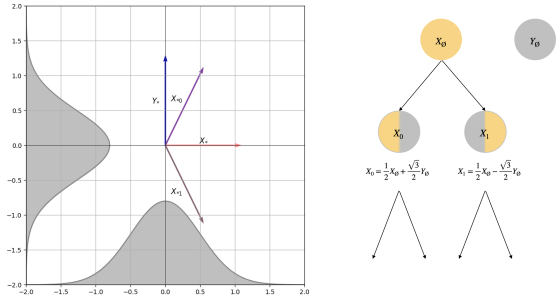


Figure 1: Left: Illustration of noise allocation to sibling nodes and their parent within a binary tree. Here $X_\star$ denotes the noise applied to a node labeled $\star$, and $Y_\star$ is another standard Gaussian independent of $X_\star$. The noise values for the children nodes are $X_{\star 0} := X_\star/2 + \sqrt{3}Y_\star/2$ and $X_{\star 1} := X_\star/2 - \sqrt{3}Y_\star/2$. Right: Noise allocation on the top two levels of a binary tree.

sition, with additional costs for sampling ($O(n)$) and transformation ($O(n^2)$).

Luckily, the recursive formula for covariance shown in (1) enables us to produce the required noise in $\Theta(n)$ time. We call our method the Cascade Sampling algorithm (Algorithm 1) – it begins by sampling the noise at the highest level and propagates to the bottom most leaves, ensuring that the relationship $X_\star = X_{\star 0} + X_{\star 1}$ as well as the covariance matrix in (1) are both preserved. A simple yet crucial observation is the following: Given i.i.d. random variables $X, Y \sim \mathbb{N}(0,1)$, define

$$X_0 = \frac{1}{2}X + \frac{\sqrt{3}}{2}Y, \quad X_1 = \frac{1}{2}X - \frac{\sqrt{3}}{2}Y. \tag{2}$$

Then X_0, X_1 are also $\mathbb{N}(0,1)$, sum up to X, and have correlation -0.5; Figure 1 is a visualization. Initially, a standard normal is sampled for the root's noise, then the noise for its direct descendants (depth 1 nodes) is determined using equation (2). This process is repeated, applying equation (2) for each subsequent level, to assign noise to all nodes. The complete process is outlined in Algorithm 1.

If one unit of cost is attributed to the sampling of a univariate normal variable and to each arithmetic operation (including addition or multiplication), it becomes clear that the cost of Algorithm 1 is linear in the total number of data points, dramatically improving the $O(n^3)$ cost of the standard sampling algorithm using Cholesky decomposition.

Proposition 3.4. *Executing Algorithm 1 for a given input depth k incurs a cost of $\Theta(n)$, where $n = 2^k$ is the total count of data points (leaf nodes).*

Our next result shows that the leaves have the covariance structure described in Definition 3.2.

Algorithm 1: Cascade Sampling Algorithm

Input: Depth of the binary tree k, variance σ^2 determined by the privacy budget.
Output: Noise values $\{X_I\}$ for all nodes $I \in \cup_{0 \le i \le k}\{0,1\}^i$ in a binary tree.
for *each node* $\star \in \{0,1\}^i$ *at depth* $0 \le i \le k-1$ **do**
  if $\star = \varnothing$ **then**
    Assign $X_\varnothing \sim \mathbb{N}(0, \sigma^2)$
  end
  Sample $Y_\star \sim \mathbb{N}(0, \sigma^2)$ independently;
  Define the noise values for the children of $\star$:;
  $X_{\star 0} := \frac{1}{2}X_\star + \frac{\sqrt{3}}{2}Y_\star, \quad X_{\star 1} := \frac{1}{2}X_\star - \frac{\sqrt{3}}{2}Y_\star$;
end

Proposition 3.5. *For any given positive integer k, the covariance matrix of the leaf noises produced by Algorithm 1 is equal to $\sigma^2 \boldsymbol{C}_k$, where $\boldsymbol{C}_k$ is defined in Definition 3.1.*

Finally, combining Proposition 3.5 and Theorem 3.3 immediately shows that the noises on every node generated by Algorithm 1 have the same distribution $\mathbb{N}(0, \sigma^2)$. More importantly, our correlation matrix $\boldsymbol{C}_k$ has many more interesting properties that are central to privacy and utility analysis.

Remark 3.6. A special case of 1D range query considered substantially in the literature is continual counting in streaming data [6, 17, 11]. The ranges are of the form $[1, i]$ which reports the count (or sum) of values from index 1 to i. We could adapt the cascade sampling algorithm to an incremental version to handle the streaming data. Specifically, suppose we have already received $n = 2^k$ elements, and the correlated noises for the top n elements have been calculated. We consider how to generate the noise for the $n+1$-th element. This will also grow the binary tree to extend for a new root X', with the left child X_0 as the current root of the binary tree on the top n elements, and the right child X_1 to be the root of upcoming n elements. Different from the top down implementation of cascade sampling algorithm, here we have already constructed and sampled X_0, and we need to sample X_1 with negative correlation with X_0. This can be done by sampling $Y_\star \sim \mathbb{N}(0, \sigma^2)$ independently and set $X_1 := -\frac{1}{2}X_0 + \frac{\sqrt{3}}{2}Y_\star$, and $X' := X_0 + X_1$. More on sampling from a conditional Gaussian distribution can be found in [21]. Notice that we do not need to generate the entire subtree of X_1 but only need the noises along the path from X_1 to the $n+1$-th element. Thus the running time is $O(\log n)$ per element at most and $O(1)$ amortized. Details can be found in Appendix D.1.

Remark 3.7. The work by [44] considers finding the covariance matrix through an optimization procedure while constraining the variance of each workload query. As the optimization could be expensive in practice, they use approximation methods to find the desired covariance structure. For the case of equal variance on queries corresponding to nodes of a binary tree, our covariance matrix construction is a feasible solution to the optimization problem (prioritizing for privacy budget) in [43]. Our work can be seen as complementary to theirs, as our motivation comes from Census Bureaus application and identifying the recursive structure within the covariance matrix, we are able to provide a linear time sampling algorithm for this particular covariance matrix.

3.3 Privacy property

We now turn our focus to the privacy aspects of our algorithm, with a special emphasis on identifying the suitable level of noise. First, Theorem 3.8 is applicable to Gaussian noise with any covariance matrix $\boldsymbol{C}$. This can be seen as an extension of the traditional Gaussian mechanism (e.g. Appendix A of [15]) but for correlated noises. Then, Theorem 3.9 is specifically tailored for the covariance matrix outlined in Definition 3.1.

Theorem 3.8. *Let $X \in \mathcal{X}^n$ be any dataset, for neighboring X and X' let $\mathcal{M}_\sigma(X) = X + \mathsf{Noise}$ be the privacy mechanism, where $\mathsf{Noise} \sim \mathbb{N}(\mathbf{0}, \sigma^2\boldsymbol{C})$ and $\boldsymbol{C}$ is an arbitrary covariance matrix with dimension $n \times n$. Fix any $\varepsilon \in (0, 1]$ and $\delta \in (0, 1/2]$, the mechanism $\mathcal{M}_\sigma(.)$ is (ε, δ)-DP for*

$$\sigma^2 \geq \frac{2\|\mathsf{diag}(\boldsymbol{C}^{-1})\|_\infty \log(2/\delta)}{\varepsilon^2}$$

where $\|\mathsf{diag}(\boldsymbol{C}^{-1})\|_\infty$ denotes the largest magnitude of the diagonal entries of $\boldsymbol{C}^{-1}$.

Theorem 3.8 is broad in scope yet challenging to apply. It requires the precision matrix (inverse of the covariance matrix), which is hard to estimate unless specifically designed. Fortunately, our proposed noise model has a clearly defined inverse matrix, simplifying analysis and making the theorem more practical. Our dataset includes $n = 2^k$ points, using multivariate Gaussian noise with a covariance of $\sigma^2\boldsymbol{C}_k$, as detailed in Definition 3.1. The key findings are presented in the following theorem.

Theorem 3.9. *Let $X \in \mathcal{X}^n$ be any dataset and let $\mathcal{M}_\sigma(X) = X + \mathsf{Noise}$ be our privacy mechanism, where $\mathsf{Noise} \sim \mathbb{N}(\mathbf{0}, \sigma^2\boldsymbol{C}_k)$ with $\boldsymbol{C}_k$ defined in Definition 3.1. Fix any $\varepsilon \in (0, 1]$ and $\delta \in (0, 1/2]$, our mechanism $\mathcal{M}_\sigma(.)$ is (ε, δ)-DP for*

$$\sigma^2 \geq \left(\frac{2}{\varepsilon^2} + \frac{2\log_2(n)}{3\varepsilon^2}\right)\log\frac{2}{\delta} = \Theta\left(\frac{\log n \log\frac{2}{\delta}}{\varepsilon^2}\right).$$

Theorem 3.9 is immediate by applying Theorem 3.8 and Corollary B.3.

3.4 Utility Analysis

We now evaluate the utility of our correlated input perturbation mechanism, $\mathcal{M}_\sigma(\boldsymbol{x}) = \boldsymbol{x} + \mathbb{N}(\mathbf{0}, \sigma^2\boldsymbol{C}_k)$, for a dataset $\boldsymbol{x} \in \mathcal{X}^n$ where $n = 2^k$. Given a workload matrix $\boldsymbol{W}$, our mechanism operates by applying this matrix to the perturbed dataset, which we represent as $\mathcal{W}_\sigma(\boldsymbol{x}) := \boldsymbol{W}\mathcal{M}_\sigma(\boldsymbol{x})$. This design satisfies all the desired properties described in Section 1. First, as an additive mechanism, $\mathcal{W}_\sigma$ enjoys unbiasedness as the privacy noise $\boldsymbol{e}$ has zero mean as guaranteed by design. Second, $\mathcal{W}_\sigma$ maintains consistency as $\mathcal{M}_\sigma(\boldsymbol{x})$ could be viewed as a potential legitimate input. Furthermore, the additive construction of $\mathcal{W}_\sigma$, coupled with the public knowledge of the noise distribution as a correlated Gaussian makes $\mathcal{W}_\sigma$ statistically transparent.

We outline various error metrics to quantify the discrepancy between $\mathcal{W}_\sigma(\boldsymbol{x})$ and $\boldsymbol{W}\boldsymbol{x}$. Since $\mathcal{W}_\sigma(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x} = \boldsymbol{W} \cdot \mathsf{Noise}$ where $\mathsf{Noise} \sim \mathbb{N}(\mathbf{0}, \sigma^2\boldsymbol{C}_k)$, we observe that the difference is a random vector that does not depend on the dataset $\boldsymbol{x}$. Some reasonable error metrics are as follows:

Definition 3.10 (Expected total squared error)**.** The expected total squared error is defined as

$$\begin{aligned}\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma) &:= \sup_{\boldsymbol{x}\in\mathcal{X}^n} \mathbb{E}\left[\|\mathcal{W}_\sigma(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_2^2\right] \\ &= \mathbb{E}_{\boldsymbol{s}\sim\mathbb{N}(\mathbf{0},\sigma^2\boldsymbol{C}_k)}\left[\|\boldsymbol{W}\boldsymbol{s}\|_2^2\right].\end{aligned}$$

Definition 3.11 (Worst-case expected error)**.** The worst-case expected error is defined as

$$\begin{aligned}\mathsf{err}^\infty_{\boldsymbol{W}}(\mathcal{W}_\sigma) &:= \sup_{\boldsymbol{x}\in\mathcal{X}^n} \left\|\mathbb{E}\left[|\mathcal{W}_\sigma(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}|\right]\right\|_\infty \\ &= \left\|\mathbb{E}_{\boldsymbol{s}\sim\mathbb{N}(\mathbf{0},\sigma^2\boldsymbol{C}_k)}[|\boldsymbol{W}\boldsymbol{s}|]\right\|_\infty.\end{aligned}$$

where $|v|$ applies to each component of v.

Definition 3.12 (Expected worst-case error)**.** The expected worst-case error is defined as

$$\begin{aligned}\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) &:= \sup_{\boldsymbol{x}\in\mathcal{X}^n} \mathbb{E}\left[\|\mathcal{W}_\sigma(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_\infty\right] \\ &= \mathbb{E}_{\boldsymbol{s}\sim\mathbb{N}(\mathbf{0},\sigma^2\boldsymbol{C}_k)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty].\end{aligned}$$

The difference between $\mathsf{err}^\infty_{\boldsymbol{W}}$ and $\mathsf{err}_{\boldsymbol{W},\infty}$ arises solely from the order in which $\mathbb{E}$ and the ℓ_∞ norm are taken. We have the following relationship between these errors.

Proposition 3.13. *For any given $\sigma > 0$ and query matrix W of size $m \times n$, we have:*

$$\sqrt{\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)/m} \leq \mathsf{err}^\infty_{\boldsymbol{W}}(\mathcal{W}_\sigma) \leq \mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma). \quad (3)$$

The focus of our analysis is on two scenarios. The first is that $\boldsymbol{W}$ represents all consecutive range queries. In this context, $\boldsymbol{W}$ is a binary matrix of dimensions $(\binom{n}{2}+n)\times n$, with each row comprising a sequence of consecutive ones. The second is when $\boldsymbol{W}$ represents all 'nodal' queries, indicating that $\boldsymbol{W}$ is a matrix of dimensions $(2n-1)\times n$, designed to query the values associated with every node in our binary tree. We also recall the notations introduced in Section 3.1, where $\{X_I\}_{I\in\{0,1\}^k}$ represents the Gaussian noise vector with a covariance matrix $\sigma^2\boldsymbol{C}_k$.

3.4.1 Continuous range queries

When $\boldsymbol{W}$ encompasses all continuous range queries, the related noise $\mathcal{W}_\sigma(\boldsymbol{x})-\boldsymbol{W}\boldsymbol{x}$ forms a vector of length $\binom{n}{2}+n$. Each element in this vector represents a consecutive sum $\sum_{L=I}^{I+j}X_L$. It is evident that each element is a univariate Gaussian with zero mean. The essential technical lemma below demonstrates that the maximum variance increases logarithmically with the number of data points.

Lemma 3.14. *Let $\{X_I\}_{I\in\{0,1\}^k}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_k)$ with $\boldsymbol{C}_k$ defined in Definition 3.1. Then the maximum variance among all the consecutive sums satisfies:*

$$\max_{I\in\{0,1\}^k,j\le 2^k-I}\mathsf{Var}[\sum_{L=I}^{I+j}X_L]/\sigma^2=\Theta(k)=\Theta(\log_2(n)).$$

With Lemma 3.14, we are ready to state the utilities of our privacy mechanism under different metrics.

Theorem 3.15. *Fix $\sigma>0$ and let query matrix $\boldsymbol{W}$ be all the continuous range queries, we have:*

$$\begin{aligned}\mathsf{err}^{\infty}_{\boldsymbol{W}}(\mathcal{W}_\sigma)&=\Theta\left(\sigma\sqrt{\log_2(n)}\right),\\ \mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)&=O\left(\sigma^2n^2\log_2(n)\right),\\ \mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)&=O(\sigma\log_2(n)).\end{aligned}$$

Choosing σ to accord to the privacy budget, the next corollary gives our mechanism's utility guarantee.

Corollary 3.16. *Let $X\in\mathcal{X}^n$ be any dataset and let $\mathcal{M}_\sigma(X)=X+\sigma\mathbb{N}(\boldsymbol{0},\boldsymbol{C}_K)$ be our privacy mechanism, where σ is chosen such that the mechanism satisfies (ε,δ)-DP. Let $\boldsymbol{W}$ be all the continuous range queries, we have:*

$$\begin{aligned}\mathsf{err}^{\infty}_{\boldsymbol{W}}(\mathcal{W}_\sigma)&=\Theta\left(\log(n)\sqrt{\log(2/\delta)}\varepsilon^{-1}\right),\\ \mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)&=O\left(n^2\log^2(n)\log(2/\delta)\varepsilon^{-2}\right),\\ \mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)&=O\left(\log^{1.5}(n)\sqrt{\log(2/\delta)}\varepsilon^{-1}\right).\end{aligned}$$

Theorem 3.15 differs from the scenario where independent Gaussian noise $\mathbb{N}(0,\sigma^2)$ is added to each leaf node. In the latter case, the three errors, $\mathsf{err}_{\boldsymbol{W},2}$, $\mathsf{err}^{\infty}_{\boldsymbol{W}}$, and $\mathsf{err}_{\boldsymbol{W},\infty}$, are $\Theta(n^3\sigma^2)$, $\Theta(\sqrt{n}\sigma)$, and $\Omega(\sqrt{n}\sigma)$ respectively. In contrast, our mechanism results in errors of $\Theta(n^2\log(n)\sigma^2)$, $O(\sqrt{\log(n)}\sigma)$, and $O(\log(n)\sigma)$, indicating lower error magnitudes for all evaluated metrics.

Comparison of bounds in Corollary 3.16: The bounds attained by our mechanism for $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ match previous work [26, 42] which can be viewed as instantiations of the matrix mechanism [31]. To our best knowledge, this is the first work that obtains comparable bound to specialized output perturbation mechanisms for range queries. We also show that empirically we match the performance of these methods in Section 5. Though these works do not explicitly analyze for $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$, [29] showed that the Binary Tree mechanism [13, 6] with Gaussian noise obtains the same $O(\log^{3/2}n)$ bound.

In terms of tightness, our bound for $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ is optimal among all (ε,δ)-DP mechanisms. To see this, note that the workload matrix for continual counting (a lower triangular matrix of size $n\times n$ with 1 for entries on and below the diagonal and 0 elsewhere) forms a subset of the workload matrix for continuous queries. [27] (Thm. 4) show $\Omega(\log^2 n)$ lower bound on error for any (ε,δ)-DP mechanism for the continual counting which matches our upper bound (after scaling for the number of queries). For $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$, [17] (Thm. 3) show $\Omega(\log^2 n)$ lower bound on the *squared-infinity* error for any (ε,δ)-DP mechanism, leading to an $\Omega(\log n)$ lower bound for $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$. Consequently, our upper bound for $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ in Corollary 3.16 is only off by a factor of $\log^{0.5}n$ (due to Gaussian concentration) compared to the known lower bound. An intriguing open question is whether this gap can be narrowed from either side.

4 A brief review of related literature

This section provides an abridged review of existing DP mechanisms for linear queries, with a focus on how they can be used for the Census application with its considerations. Due to space constraints, we defer an extended literature review to Appendix A.

As a quintessential *output perturbation* method, *Gaussian mechanisms* add to the query output centered Gaussian noise with variance tailored to its sensitivity [10, 14, 12, 23, 36], which for linear queries is the largest norm of the columns of workload matrix $\boldsymbol{W}$. Gaussian mechanisms are unbiased and statistically transparent, but are not flexible in utility control because the accuracy of the entire output query is dictated by $\boldsymbol{W}$. They are not consistent in general unless

e is guaranteed to be in the column space of $\boldsymbol{W}$, as discussed in Remark 2.4.

The Census Bureau's *TopDown algorithm* [3, 1] is a massive endeavor to navigate through the complex requirements discussed in Section 1. It consists of two phases: 1) the "measurement" phase injects additive discrete Gaussian noise [5] to confidential queries, similar to the Gaussian mechanisms; 2) the "estimation" phase uses optimization-based post-processing to achieve consistency (internal and external) and complex utility control such as invariants/consistency, external constraints, and marginal accuracy targets. Post-processing does not damage privacy protection – which is one of the greatest traits of DP. Unfortunately, the post-processing phase of TopDown algorithm destroyed both unbiasedness [7, 19] and statistical transparency [20, 24] (noise distribution became intractable and regression on data output revealed unwanted bias). Researchers tend to agree that a major contributor to the bias is the nonnegativity constraint that TopDown's post-processing enforces; see e.g. [2].

In general consistency could be obtained by projection into a designated subspace [31] or other optimization-based transformations. Special care should be taken to ensure unbiasedness (e.g. [4, 26]), which could be lost if inequality constraints are present. Moreover, it would be ideal that an analytical description is available, e.g. [26], to maintain statistical transparency. In addition, though theoretically efficient algorithms are known, certain post-processing approaches are not practical in real-world use cases of very large data sets, e.g. projecting onto convex sets. For example, the post-processing phase of TopDown algorithm is a substantially non-trivial effort in terms of computational costs; see section 7.6 of [3].

To improve utility, a lot of work has been developed to carefully design how perturbations are added. Starting from the work in [31], the broadly defined family of *matrix mechanism* is a workload-dependent mechanism. In general, one finds a privileged factorization, $\boldsymbol{W} = RA$, and infuses Gaussian noise to the intermediate result Ah, with h being the histogram vector. Choice in the factorization allows for the attainment of near optimal utility. While this can be done for any general workload matrix, for the special case of one dimensional range query, there has been a number of excellent work such as Binary Tree [13, 6], Hierarchical [26], Wavelet [42], EigenDesign [30], HDMM [34] and explicit factorization [17] mechanisms that can be considered as carefully choosing the matrix factorization. Consistency of each of these mechanisms can be individually checked. We will review the details of these algorithms in Appendix A. We also show empirical results in the next section.

We would like to remark that our mechanism can also be written as a matrix factorization $M(x) = R(Ax+z)$ with $R = \sqrt{\boldsymbol{C}_k}$, $A = R^{-1}W$.In addition to showing asymptotically tight ℓ_2 error and near-optimal ℓ_∞ error (upto $\sqrt{\log n}$ factor), our mechanism also ensures that all queries share the *same* variances.

5 Experiments

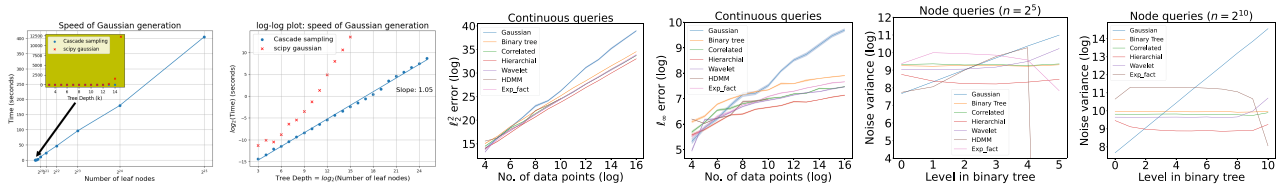
Efficiency of Cascade Sampling. The computational costs in terms of clock time for both scales, original and logarithmic are displayed in Figure 2(a). Cascade Sampling demonstrates remarkable efficiency, requiring less than a second for up to $n = 2^{17}$ $(131,072)$. Even for a over 33.5 million-dimensional Gaussian, our algorithm completes the task in under 9 minutes. Right part of Figure 2(a) features a scatterplot of log(Time) against log(Data Points), with a fitted straight line that has a slope of 1.05, confirming our assertion of linear cost. Additional details are provided in Appendix E.[1]

Utility comparison with existing algorithms. We evaluate the correlated input perturbation mechanism against the Gaussian (adding noise to $\boldsymbol{x}$ and querying from the privatized data), Binary Tree [13, 6], Hierarchical [26], Wavelet [42], HDMM [34] and explicit factorization [17]. We do not compare with an output Gaussian mechanism (adding noise to $\boldsymbol{W}\boldsymbol{x}$) as this performs substantially worse due to high sensitivity. Experiments comparing runtime are in Appendix E.4.

Experimental setup: We compare mechanisms on randomly generated data, where each element of $\boldsymbol{x} \in \mathbb{Z}^n$ is sampled uniformly at random from 1 to 1000. We consider the case where a person contributes a single count to an element in $\boldsymbol{x}$ and hence for neighboring $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$. We consider n in range $\{2^4, \cdots, 2^{15}\}$, except for HDMM which we restrict to 2^{10} due to computational overhead. We fix privacy parameters $\varepsilon = 0.1$ and $\delta = 10^{-9}$ for all our experiments and report $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ (Definition 3.10 and Definition 3.12) for different choices of workload matrices $\boldsymbol{W}$. The error values on graphs are plotted with ± 0.25 standard deviation across 10 independent runs for all experiments. Experiments for node and random queries and additional implementation details are included in Appendix E.

Continuous range queries: A continuous range query is specified by start and end index in $\{1, \ldots, n\}$ and the answer is sum of all elements within this range. We consider the workload matrix of all possible $\binom{n}{2} + n$

[1]Python code available at: `https://github.com/prathameshtd/DPRQ-correlated-noise`



(a) (b) (c) (d) (e)

Figure 2: (a): Comparison of Gaussian Generation Speeds (Left: original scale; Right:log-log scale). (b) & (c): $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ for continuous queries in log-log scale. (d) & (e): Variance of queries for levels of binary tree in log-log scale.

queries. As computing output of all these queries becomes computationally infeasible for very large n, we randomly sample the queries to obtain a Monte Carlo estimate of the errors, with details explained in Appendix E.5. We sample 5000 queries for each value of n and report estimated $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ in Figures 2(b) and 2(c).

Both $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ follow the same trend. Four mechanisms, ours, Wavelet, HDMM and Hierarchical, appear to be on the first tier (i.e., with smallest error) for both ℓ_2 and ℓ_∞. For small n, Gaussian mechanism performs as well as these methods but suffers variance scaling linearly in n for larger values. Our mechanism, owing to its design, exhibits significantly reduced error compared to the standard Gaussian approach, its primary input perturbation competitor. Our method achieves comparable error empirically with the state of the art methods, yet it carries additional nice statistical properties such as unbiasedness, consistency, and transparency.

Utility control: The ability to control noise variance across different 'levels' of summation was one of the motivations of our mechanism, which we demonstrate here. We consider a binary tree over the dataset vector $\boldsymbol{x}$, and the queries (which are nodes of the tree) correspond to intervals summing up the leaves of the sub-tree rooted at the node. We then consider the variance of noise across queries corresponding to different levels (and the root would correspond to the highest level summing up all the elements of $\boldsymbol{x}$). We perform this for $n = 2^5$ and $n = 2^{10}$ for 500 independent runs and the results are shown in Figures 2(d) and 2(e).

As the additive correlated noise in our mechanism is designed to have the same variance across levels, our mechanism and the Binary Tree mechanism, which adds i.i.d. Gaussian noises to each element of binary tree to answer queries have the same variance across all such queries. Comparatively, Gaussian mechanism has variance linearly increasing across levels. Wavelet has smaller consistent variance for smaller levels (corresponding to short ranges) but increases sharply for long ranges, whereas the opposite effect is observed for explicit factorization and HDMM. Hierarchical has the smallest variance in magnitude and also a typical trend, variance is larger for smaller and larger levels and less for intermediate, which helps illustrate the effect of post-processing and explains improved utility.

6 Discussion and Future Work

The proposed input perturbation mechanism leverages error correlation structures which, once combined with the query workload matrix, allows for fine-grained control over error magnitude across queries at different geographic levels, a feature that is highly desirable in applications such as the dissemination of confidential official statistical data products with complex structures. Our proposal permits straightforward generalizations to general (unbalanced) binary trees, as well as higher-dimensional query types, in particular 2-D range queries and contingency tables that recognize, for example, the geographic contiguity of the data points. Both generalizations need to modify the noise allocation scheme in the cascade sampling algorithm. Appendices D.2 and D.3 respectively discuss how to instantiate these generalizations. We believe this work leads to a generic framework of differential privacy mechanism design. In particular, beyond equality of error variance at every hierarchical level considered here, the Cascading Sampling Algorithm lends itself to a straightforward generalization where detailed variance control is required, i.e. when different error variances are needed at different levels. Furthermore, when the data or query workload structure is well understood, additional benefits can result from catering privacy perturbations to respect these known structures. These investigations are left to future work.

Acknowledgements

Guanayang Wang acknowledges support by the National Science Foundation through grant DMS-2210849. Prathamesh Dharangutte and Jie Gao are supported by NSF IIS-2229876, DMS-2220271, DMS-2311064, CCF-2208663, CCF-2118953.

References

[1] J. Abowd. Noisy measurements are important; the design of Census products is much more important. *Harvard Data Science Review*, 6(2), Apr. 2024.

[2] J. Abowd, R. Ashmead, R. Cumings-Menon, S. Garfinkel, D. Kifer, P. Leclerc, W. Sexton, A. Simpson, C. Task, and P. Zhuravlev. An uncertainty principle is a price of privacy-preserving microdata. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11883–11895. Curran Associates, Inc., 2021.

[3] J. M. Abowd, R. Ashmead, R. Cumings-Menon, S. Garfinkel, M. Heineck, C. Heiss, R. Johns, D. Kifer, P. Leclerc, A. Machanavajjhala, B. Moran, W. Sexton, M. Spence, and P. Zhuravlev. The Census TopDown algorithm. *Harvard Data Science Review*, 2022.

[4] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282, 2007.

[5] C. L. Canonne, G. Kamath, and T. Steinke. The discrete Gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

[6] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):1–24, Nov 2011.

[7] D. L. Cork, C. F. Citro, and N. J. Kirkendall. *2020 Census Data Products: Data Needs and Privacy Considerations: Proceedings of a Workshop*. The National Academies Press, Washington, DC, 2020.

[8] R. Cumings-Menon, J. M. Abowd, R. Ashmead, D. Kifer, P. Leclerc, J. Ocker, M. Ratcliffe, and P. Zhuravlev. Geographic spines in the 2020 census disclosure avoidance system. *Journal of Privacy and Confidentiality*, 14(3), 2024.

[9] P. Dharangutte, J. Gao, R. Gong, and F.-Y. Yu. Integer subspace differential privacy. *Proceedings of the The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)*, pages 7349–7357, 2023. arXiv:2212.00936.

[10] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database*

Systems, PODS '03, page 202–210, New York, NY, USA, 2003. Association for Computing Machinery.

[11] K. Dvijotham, H. B. McMahan, K. Pillutla, T. Steinke, and A. Thakurta. Efficient and near-optimal noise generation for streaming differential privacy. https://arxiv.org/abs/2404.16706, 2024.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[13] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 715–724. ACM, 2010.

[14] C. Dwork and K. Nissim. Privacy-Preserving datamining on vertically partitioned databases. In *Advances in Cryptology – CRYPTO 2004*, pages 528–544. Springer Berlin Heidelberg, 2004.

[15] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] A. Edmonds, A. Nikolov, and J. 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*, STOC 2020, pages 425–438, New York, NY, USA, June 2020. Association for Computing Machinery.

[17] H. Fichtenberger, M. Henzinger, and J. Upadhyay. Constant matters: Fine-grained error bound on differentially private continual observation. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10072–10092. PMLR, 2023.

[18] F. Fioretto and P. Van Hentenryck. Differential privacy of hierarchical Census data: An optimization approach. In *International Conference on Principles and Practice of Constraint Programming*, pages 639–655. Springer, 2019.

[19] J. Gao, R. Gong, and F.-Y. Yu. Subspace differential privacy. In *Proceedings of the The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, pages 3986–3995, 2022. arXiv:2108.11527.

[20] R. Gong. Transparent Privacy Is Principled Privacy. *Harvard Data Science Review*, (Special Issue 2), jun 24 2022. https://hdsr.mitpress.mit.edu/pub/ld4smnnf.

[21] A. Gut. *An Intermediate Course in Probability*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[22] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. *Adv. Neural Inf. Process. Syst.*, 2:2339–2347, 2012.

[23] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714, 2010.

[24] M. Hawes and M. Spence. Using the 2020 census redistricting data (P.L. 94-171) noisy measurement file. https://www2.census.gov/about/training-workshops/2023/2023-06-15-nmf-presentation.pdf, 2023.

[25] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, D. Zhang, and G. Bissias. Exploring privacy-accuracy tradeoffs using DPComp. In F. Özcan, G. Koutrika, and S. Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 2101–2104. ACM, 2016.

[26] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings VLDB Endowment*, 3(1-2):1021–1032, Sept. 2010.

[27] M. Henzinger, J. Upadhyay, and S. Upadhyay. Almost tight error bounds on differentially private continual counting. In N. Bansal and V. Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 5003–5039. SIAM, 2023.

[28] J. Honaker. Efficient use of differentially private binary trees. *Theory and Practice of Differential Privacy (TPDP 2015), London, UK*, 2:26–27, 2015.

[29] P. Jain, S. Raskhodnikova, S. Sivakumar, and A. D. Smith. The price of differential privacy under continual observation. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023,*

Honolulu, Hawaii, USA, volume 202 of *Proceedings of Machine Learning Research*, pages 14654–14678. PMLR, 2023.

[30] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.*, 5(6):514–525, 2012.

[31] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[32] N. Linial and A. Shraibman. Lower bounds in communication complexity based on factorization norms. *Random Struct. Algorithms*, 34(3):368–394, May 2009.

[33] L. McKenna. Disclosure avoidance techniques used for the 1970 through 2010 decennial censuses of population and housing. Technical report, U. S. Census Bureau, November 2018.

[34] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Journal of Privacy and Confidentiality*, 13(1), Aug. 2023.

[35] S. Muthukrishnan and A. Nikolov. Optimal private halfspace counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1285–1292, 2012.

[36] A. Nikolov, K. Talwar, and L. Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360. ACM, 2013.

[37] A. Nikolov, K. Talwar, and L. Zhang. The geometry of differential privacy: The small database and approximate cases. *SIAM Journal on Computing*, 45(2):575–616, 2016.

[38] W. H. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *Proc. VLDB Endow.*, 6(14):1954–1965, 2013.

[39] J. Seeman, A. Slavkovic, and M. Reimherr. A formal privacy framework for partially private data. *arXiv preprint arXiv:2204.01102*, 2022.

[40] U.S. Census Bureau. Technical documentation – operational quality metrics, last updated 10.06.22. Technical report, U.S. Census Bureau, 2022.

[41] U.S. Census Bureau. 2020 census production disclosure avoidance system detailed summary metrics. Technical report, U.S. Census Bureau, 2023.

[42] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1200–1214, 2011.

[43] Y. Xiao, Z. Ding, Y. Wang, D. Zhang, and D. Kifer. Optimizing fitness-for-use of differentially private linear queries. *Proceedings VLDB Endowment*, 14(10):1730–1742, June 2021.

[44] Y. Xiao, Z. Ding, Y. Wang, D. Zhang, and D. Kifer. Optimizing fitness-for-use of differentially private linear queries. *Proc. VLDB Endow.*, 14(10):1730–1742, 2021.

[45] Y. Xiao, G. He, D. Zhang, and D. Kifer. An optimal and scalable matrix mechanism for noisy marginals under convex loss functions. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 20495–20539. Curran Associates, Inc., 2023.

A An extended review of related literature

Linear query is a central topic that has been studied extensively in the differential privacy literature. To supplement the brief review provided in Section 4, we discuss in greater detail existing work on privacy mechanisms for linear queries in relation to the properties of unbiasedness, consistency, statistical transparency, and flexible utility control.

A simple mechanism is input perturbation in which data entry either perturbed (in a discrete setting) or infused with an independent noise (often from a Laplace or Gaussian distribution) with suitable magnitude. The linear query can be answered by simply reporting the query answer on the perturbed data. As Section 4 alludes to, such mechanisms supports unbiasedness (if the element-wise perturbation has zero mean), logical consistency, statistical transparency, and are often amenable to distributed implementation. However, it does not support flexible utility specifications. In particular, when a query range contains a large number of data elements, the query utility (or error) grows in the order of $O(\sqrt{n})$ where n is the the number of elements in the range. The error blow-up for aggregated data substantially hinders the use of input perturbation in practice.

In contrast, mechanisms that use *output* perturbation consider adding noise to the query output. One classic example is the Gaussian mechanism [10, 14, 12]. The variance of independent Gaussian noise added to each query needs to be tailored to the query *sensitivity*, i.e., the maximum number of queries one element is involved, or the largest norm of the columns of workload matrix $\boldsymbol{W}$. Gaussian mechanisms satisfy unbiasedness and statistical transparency, but does not support flexible utility control. The noise magnitude could be high when the sensitivity of $\boldsymbol{W}$ is high. For consistency, if the perturbation noise $\boldsymbol{e}$ is in the column space of $\boldsymbol{W}$, then consistency holds – we can write the data output as $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{W}(\boldsymbol{x} + \boldsymbol{e}')$ with $\boldsymbol{W}\boldsymbol{e}' = \boldsymbol{e}$.

A notable class of Gaussian mechanisms provide improved utility to linear queries by leveraging their geometric structure [23, 36]. Specifically, consider the workload matrix W as a linear map that maps the input data histogram to a d-dimensional output vector, with $d \ll n$. [23] studied the image of the unit ℓ_1 ball under the linear map $\boldsymbol{W}$ (which is a convex polytope K) and use an instantiation of the exponential mechanism (via a randomly chosen point in K) for ε-differential privacy. In a later work, [36] considered the setting when the number of queries d is much larger than n, and (ε, δ)-differential privacy in general. They use a Gaussian mechanism where the parameters are guided by the minimum enclosing ellipsoid of a square submatrix of $\boldsymbol{W}$. These mechanisms have polylogarithmic approximation (in terms of error) to the lower bound for such queries, which are derived by using hereditary discrepancy of $\boldsymbol{W}$. As special cases of Gaussian mechanisms, these mechanisms also fall in the category of output perturbation, and enjoy similar properties as does the classic Gaussian mechanism in terms of bias, consistency, transparency, and utility control.

The generic family of matrix mechanism [31, 34], also called the factorization mechanism [37], is workload-dependent (i.e. the workload matrix $\boldsymbol{W}$ plays a role in its design) and is situated between input and output perturbation. The workload matrix $\boldsymbol{W}$ is factored into two matrices R and A, with R called the reconstruction matrix and A the strategy matrix. Gaussian errors are added to the intermediate result Ah, where h is the histogram vector. The final query result is taken as the multiplication of R and the noisy intermediate result. This way the sensitivity can be controlled by the maximum column norm of A. Depending on the utility objective (e.g., ℓ_∞ or ℓ_2 of the final error), one can choose the factorization $\boldsymbol{W} = RA$ and final query error is proportional to the multiplication of the max row norm of R [2] and maximum ℓ_2 column norm of A. By choosing an appropriate factorization $\boldsymbol{W} = RA$ (e.g., by using convex optimization [32] for final ℓ_∞ error), one can get near optimal utility [36, 16]. The matrix mechanism is unbiased, has statistical transparency and consistency if the noise is in the column space of A. Utility control depends on the specific choice of the matrix factorization. For efficient implementation, [34] give an efficient implementation of the matrix mechanism by considering different representations of the workload matrix thereby reducing the space for the optimization problem.

A number of prior work considered range query problems and developed hierarchical methods, which can be considered as special cases of the matrix mechanism [26, 6, 38, 42, 28]. We use [6] as an example to illustrate the main idea. [6] considered a specific type of range query that is derived from continuous counting in a data stream. Specifically, consider a set of binary data points $x_i \in \{0, 1\}$ arriving sequentially, the goal is to report the count at any time t, i.e., $X_t = \sum_{i=1}^{t} x_i$, for all t, with differential privacy guarantee for individual data points. They

[2] To optimize for the final ℓ_∞ (or ℓ_2) error one would minimize the max ℓ_2 row norm (or Frobenius norm) of R and maximum ℓ_2 column norm of A [31, 37].

developed a mechanism that renders the error utility of bound $O((1/\varepsilon)\log^{3/2} t)$ for answering the t-th counting query with ε-differential privacy. The main idea is to maintain a binary tree on the data stream $x_1, \cdots, x_t$ and add an independent Laplace noise for each internal node in the binary tree. The final query output will be derived by the summation of the (noisy) count of selected vertices in the binary tree. The mechanism can be applied to a general 1D range query problem of reporting $\sum_{k=i}^{j} x_k$, for any 1D range $[i, j]$, with $i \leq j$, with the same error bound. A number of other work use similar ideas: [4] uses subsets of Fourier basis, [42] uses wavelet transforms, and [35] considers general range queries with constant VC-dimension (e.g., half-space queries). [17] focuses on analysis of constant factors in the error bound using matrix factorization for the continual counting scenario and also provides concrete matrix choices for R and A. There are additional work that focuses on space efficiency of implementation in the streaming setting [11] or efficiently solving for the matrix mechanism for a wide variety of convex objective functions [45].

Subspace DP mechanisms [19, 9] impose external consistency on the data product without post-processing. External consistency is expressed through a set of invariants, exact queries from the confidential database, and may be understood as a special utility requirement of zero privacy noise. Subspace DP mechanisms allow utility control over all invariant queries, unbiasedness, and statistical transparency, but do not generally guarantee internal consistency.

We make a brief comment on the practicality of implementation. Among the mechanisms discussed here and in Section 4, some are mainly of theoretical interest [22, 36, 35]. The matrix mechanism has a practical instantiation [33], as does [26] and the subspace DP mechanisms of [19, 9]. In addition, the TopDown algorithm has been extensively instantiated, though it is difficult to independently replicate due to its formidable scale.

Finally, we remark that the local differential privacy model (LDP) refers to the scenario when each individual generates local noises and shares the perturbed value, without any need of a single trusted server. Although our noises are applied locally for each data value and the perturbed value is shared publicly, however, the generation of the local noises would require secured communication with either a trusted server (to run the cascade sampling algorithm) or with other peers/individuals (to exchange random seeds that they both use to help to install the negative correlation in their generated noises).

B Technical proofs

We first prove Theorem 3.3. Given that σ merely functions as a multiplicative constant within our privacy mechanism, our attention will now shift towards understanding some fundamental characteristics of $\boldsymbol{C}_k$, as listed below:

Proposition B.1. *For a fixed positive integer k, the matrix $\boldsymbol{C}_k$ exhibits the following properties:*

1. *The diagonal entries of $\boldsymbol{C}_k$ are all 1.*
2. *Every off-diagonal element of $\boldsymbol{C}_k$ is negative.*
3. *Summing over every column j for any row i, we have $\sum_j \boldsymbol{C}_k(i,j) = 2^{-k}$.*
4. *$\boldsymbol{C}_k$ is a positive semi-definite (PSD) matrix.*

Proof of Proporition B.1. The initial two properties can be readily validated based on the matrix's definition. For the third property, it is evidently correct when $k = 1$. Assuming its correctness for $k = l$, when $k = l + 1$, we can deduce from (1) that

$$\begin{aligned}\sum_j \boldsymbol{C}_{l+1}(i,j) &= \sum_j \boldsymbol{C}_l(i,j) + (-2^{-2l-1}) \times 2^l \\ &= 2^{-l} - 2^{-l-1} = 2^{-l-1} = 2^{-(l+1)},\end{aligned}$$

as desired. The final property can be directly applying the Gershgorin circle theorem based on the first three properties. □

Proof of Theorem 3.3. Assuming, without a loss of generality, that $\sigma = 1$, we begin our proof by considering the case where $k = 1$. This essentially requires us to demonstrate that $X_\varnothing := X_0 + X_1$ has a unit variance. We have:

$$\mathsf{Var}[X_\varnothing] = (1,1) \cdot \boldsymbol{C}_1 \cdot (1,1)^\top = 1 + 1 - 0.5 - 0.5 = 1,$$

as desired. Now, let's assume the aforementioned result holds true for $k = l$. For $k = l + 1$, the formula (1), combined with our inductive assumption, confirms that every node in both the left and right subtrees maintains the same distribution $\mathbb{N}(0, 1)$. Consequently, our task narrows down to computing the variance for the root node $X_\varnothing$, which is given by:

$$\begin{aligned}
\mathsf{Var}[X_\varnothing] &= (1, 1, \ldots, 1) \cdot \boldsymbol{C}_{l+1} \cdot (1, 1, \ldots, 1)^\top \\
&= \sum_{i,j} \boldsymbol{C}_{l+1}(i, j) \\
&= \sum_i \sum_j \boldsymbol{C}_{l+1}(i, j) \\
&= \sum_i 2^{-(l+1)} = 1 \qquad \text{by the Property 3 in Proposition B.1.}
\end{aligned}$$

This concludes our proof. □

Proof of Proposition 3.5. Without loss of generality, we assume $\sigma = 1$. We will prove by induction. When $k = 1$, it suffices to show the $\mathsf{Var}(X_0) = \mathsf{Var}(X_1) = 1$ and $\mathsf{Cov}(X_0, X_1) = -0.5$. For the former, observe that

$$\begin{aligned}
\mathsf{Var}(X_0) &= \mathsf{Var}\left(\frac{1}{2}X_\varnothing + \frac{\sqrt{3}}{2}Y_\varnothing\right) \\
&= \left(\frac{1}{2}\right)^2 \mathsf{Var}(X_\varnothing) + \left(\frac{\sqrt{3}}{2}\right)^2 \mathsf{Var}(Y_\varnothing) = \frac{1}{4} + \frac{3}{4} = 1.
\end{aligned}$$

Similarly $\mathsf{Var}(Y_0) = 1$. For the covariance, we have

$$\begin{aligned}
\mathsf{Cov}(X_0, X_1) &= \mathsf{Cov}\left(\frac{1}{2}X_\varnothing + \frac{\sqrt{3}}{2}Y_\varnothing, \frac{1}{2}X_\varnothing - \frac{\sqrt{3}}{2}Y_\varnothing\right) \\
&= \left(\frac{1}{2}\right)^2 \mathsf{Var}(X_\varnothing) - \left(\frac{\sqrt{3}}{2}\right)^2 \mathsf{Var}(Y_\varnothing) = \frac{1}{4} - \frac{3}{4} = -0.5.
\end{aligned}$$

This concludes our proof for the $k = 1$ case. Assuming the result holds for $k = l$, we note that for $k = l + 1$, the nodes in the left subtree (labelled $X_{0\star}$) and the nodes in the right subtree (labelled $X_{1\star}$) each constitute a complete binary tree of depth l. Furthermore, the recursive nature of Algorithm 1 indicates that the following three sets of random variables are identically distributed:

$$\{X_{0\star}\}_{\star \in \bigcup_{0 \le s \le l} \{0,1\}^s} \overset{d}{=} \{X_{1\star}\}_{\star \in \bigcup_{0 \le s \le l} \{0,1\}^s} \overset{d}{=} \{\tilde{X}_\star\}_{\star \in \bigcup_{0 \le s \le l} \{0,1\}^s}. \tag{4}$$

In this context, $\{X_{0\star}\}$ and $\{X_{1\star}\}$ represent the left and right subtrees generated by Algorithm 1 when $k = l+1$, while $\{\tilde{X}_\star\}$ denotes an independent execution of Algorithm 1 with $k = l$. Employing equation (4) and the inductive hypothesis for $k = l$ on the set $\{\tilde{X}_\star\}_{\star \in \bigcup_{0 \le s \le l} \{0,1\}^s}$, we know that the covariance matrices restricted to the left and right subtrees are both equivalent to $\boldsymbol{C}_l$. Hence, the two $2^l \times 2^l$ diagonal blocks of $\boldsymbol{C}_{l+1}$ are identical to $\boldsymbol{C}_l$.

To confirm that the entire covariance matrix is equal to $\boldsymbol{C}_{l+1}$ as specified in (1), we must demonstrate that any leaf in the left subtree is correlated with any leaf in the right subtree by -2^{-2l-1}. Take any pair of indices $I, J \in \{0, 1\}^l$, and denote $\tilde{I} = (0, I_1, \ldots, I_{l-1}) \in \{0, 1\}^l$ and $\tilde{J} = (1, J_1, \ldots, J_{l-1}) \in \{0, 1\}^l$. Algorithm 1 informs us that the noise at leaf X_{0I} is expressed as:

$$X_{0I} = \frac{1}{2}X_{\tilde{I}} + \frac{(-1)^{I_k}\sqrt{3}}{2}Y_{\tilde{I}},$$

and likewise for leaf X_{1J}:

$$X_{1J} = \frac{1}{2}X_{\tilde{J}} + \frac{(-1)^{J_k}\sqrt{3}}{2}Y_{\tilde{J}}.$$

Consequently:

$$\mathsf{Cov}(X_{0I}, X_{1J}) = \left(\frac{1}{2}\right)^2 \mathsf{Cov}(X_{\tilde{I}}, X_{\tilde{J}}), \tag{5}$$

since $Y_{\tilde{I}}$ is by construction independent of $X_{\tilde{J}}$ and $Y_{\tilde{I}}$, and $Y_{\tilde{J}}$ are independent of $X_{\tilde{I}}$. Given that $X_{\tilde{I}}$ and $X_{\tilde{J}}$ are leaf nodes on opposing sides of a binary tree of depth l, their covariance is -2^{-2k+1} by the inductive hypothesis. Inserting this into equation (5), we obtain:

$$\mathsf{Cov}(X_{0I}, X_{1J}) = 2^{-2} \times -2^{-2k+1} = -2^{-2k-1}, \tag{6}$$

which completes the proof. □

Proof of Theorem 3.8. To prove (ε, δ)-DP, we prove that with probability at least $1-\delta$ the likelihood ratio between $\mathcal{M}_\sigma(X)$ and $\mathcal{M}_\sigma(X')$ is upper bounded by $\exp(\epsilon)$, where X and X' are any two datasets that differ by at most one at only one entry. For now, we assume $X = X' + e_i$, where $e_i \in \mathcal{X}^n$ takes value 1 at the i-th coordinate, and 0 elsewhere. Let $s \in \mathbb{R}^n$ be any real vector, the logarithm of the likelihood ratio between $\mathcal{M}_\sigma(X)$ and $\mathcal{M}_\sigma(X')$ at point $X+s$ equals

$$\begin{aligned}
\log\left(\frac{f_{\mathcal{M}_\sigma(X)}(X+s)}{f_{\mathcal{M}_\sigma(X')}(X+s)}\right) &= \log\left(\frac{\exp\left(-\frac{1}{2\sigma^2}s^\top \boldsymbol{C}^{-1}s\right)}{\exp\left(-\frac{1}{2\sigma^2}(s+e_i)^\top \boldsymbol{C}^{-1}(s+e_i)\right)}\right) \\
&= -\frac{1}{2\sigma^2}\left(s^\top \boldsymbol{C}^{-1}s - (s+e_i)^\top \boldsymbol{C}^{-1}(s+e_i)\right) \\
&= \frac{1}{2\sigma^2}(\boldsymbol{C}^{-1})(i,i) + \frac{1}{\sigma^2}s^\top \boldsymbol{C}^{-1}e_i,
\end{aligned}$$

In our noise mechanism, $s \sim \mathbb{N}(\mathbf{0}, \sigma^2\boldsymbol{C})$. Therefore the above log-likelihood ratio is a one-dimensional Gaussian with mean

$$\frac{1}{2\sigma^2}(\boldsymbol{C}^{-1})(i,i),$$

and variance

$$\frac{1}{\sigma^2}e_i^\top \boldsymbol{C}^{-1}\boldsymbol{C}\boldsymbol{C}^{-1}e_i = \frac{1}{\sigma^2}\boldsymbol{C}^{-1}(i,i).$$

Let $Y \sim \mathbb{N}(0,1)$ be a standard normal random variable, set $w = \boldsymbol{C}_K^{-1}(i,i)/\sigma^2$ for notational simplicity. It is then clear that $\sqrt{w}Y + w/2$ has the same distribution as $\log\left(\frac{f_{\mathcal{M}_\sigma(X)}(X+s)}{f_{\mathcal{M}_\sigma(X')}(X+s)}\right)$. Since $\sqrt{w}Y + w/2 \le \epsilon$ is equivalent to $Y \le \epsilon/\sqrt{w} - \sqrt{w}/2$, it suffices to find conditions on w such that $\mathbb{P}[|Y| \ge \epsilon/\sqrt{w} - \sqrt{w}/2] \le \delta$.

Applying the standard subgaussian tail bound on Y, we have

$$\mathbb{P}[Y > x] \le 2\exp(-x^2/2)$$

for every $x > 0$. Therefore it suffices to find w such that $\epsilon/\sqrt{w} - \sqrt{w}/2 > 0$, and

$$\left(\frac{\epsilon}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)^2 \ge \log\left(\frac{2}{\delta}\right).$$

Now we set $0 < w \le \epsilon^2/(2\log(2/\delta))$, and check every w in this range satisfies the above two conditions. Firstly,

$$w \le \epsilon^2/(2\log(2/\delta)) \le \epsilon^2/2\log(4) \le 2\epsilon,$$

therefore $\epsilon/\sqrt{w} - \sqrt{w}/2 > 0$. Secondly,

$$\frac{\epsilon^2}{w} \ge 2\log\left(\frac{2}{\delta}\right) \ge \log(2/\delta) + \log(4) > \log(2/\delta) + \epsilon.$$

Therefore

$$\left(\frac{\epsilon}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)^2 = \frac{\epsilon^2}{w} - \epsilon + \left(\frac{\sqrt{w}}{2}\right)^2 \geq \frac{\epsilon^2}{w} - \epsilon \geq \log(2/\delta),$$

as desired. This immediately translates to the noise bound

$$\sigma^2 \geq \frac{2(\boldsymbol{C}^{-1})(i,i)\log(2/\delta)}{\epsilon^2}. \tag{7}$$

The above calculation still goes through when $X = X' - e_i$. Since the preceding argument must be valid for any i, taking the maximum of i from Equation (7) shows our mechanism $\mathcal{M}_\sigma(.)$ is (ε, δ)-DP for

$$\sigma^2 \geq \frac{2\|\mathsf{diag}(\boldsymbol{C}^{-1})\|_\infty \log(2/\delta)}{\epsilon^2}$$

□

Proof of Theorem 3.9. Based on Corollary B.3, it becomes evident that the norm $\|\mathsf{diag}(\boldsymbol{C}_k^{-1})\|_\infty$ equals $1+k/3 = 1+\log_2(n)/3$. Applying this into Theorem 3.8 leads to our result. □

Lemma B.2. *For any positive integer j, $\boldsymbol{C}_j^{-1}$ satisfies $\boldsymbol{C}_1^{-1} := \left(\begin{smallmatrix} 4/3 & 2/3 \\ 2/3 & 4/3 \end{smallmatrix}\right)$ and*

$$\boldsymbol{C}_j^{-1} = \begin{pmatrix} \boldsymbol{C}_{j-1}^{-1} + \frac{1}{3}J_{j-1} & \frac{2}{3}J_{j-1} \\ \frac{2}{3}J_{j-1} & \boldsymbol{C}_{j-1}^{-1} + \frac{1}{3}J_{j-1}, \end{pmatrix},$$

where J_{j-1} is the all one matrix of size $2^{j-1} \times 2^{j-1}$.

Proof of Lemma B.2. We prove a slightly stronger result. We show for every $k \geq 1$, the followings all hold:

1. $J_k \boldsymbol{C}_k = \boldsymbol{C}_k J_k = 2^{-k} J_k$
2. $J_k J_k = 2^k J_k$
3. $J_k \boldsymbol{C}_k^{-1} = 2^k J_k$
4. We have

$$\boldsymbol{C}_1^{-1} = \begin{pmatrix} 4/3 & 2/3 \\ 2/3 & 4/3 \end{pmatrix}$$

and

$$\boldsymbol{C}_k^{-1} = \begin{pmatrix} \boldsymbol{C}_{k-1}^{-1} + \frac{1}{3}J_{k-1} & \frac{2}{3}J_{k-1} \\ \frac{2}{3}J_{k-1} & \boldsymbol{C}_{k-1}^{-1} + \frac{1}{3}J_{k-1} \end{pmatrix}.$$

We prove this by induction. All the four claims can be directly checked when $k = 1$. Assuming they are all satisfied when $k \leq K$, for $k = K+1$, facts 1 and 2 can still be directly checked. To show fact 4, it suffices to show

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} := \begin{pmatrix} \boldsymbol{C}_K & -2^{-2K-1}J_K \\ -2^{-2K-1}J_K & \boldsymbol{C}_K \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{C}_K^{-1} + \frac{1}{3}J_K, & \frac{2}{3}J_K \\ \frac{2}{3}J_K, & \boldsymbol{C}_K^{-1} + \frac{1}{3}J_K \end{pmatrix} = I_{2^{K+1}}$$

We directly check this by multiplying the two 2×2 block matrices on the left hand side. The diagonal blocks are

$$\begin{aligned} A = D &= I + \frac{1}{3}\boldsymbol{C}_K J_K - \frac{2}{3} \times 2^{-2K-1} J_K J_K \\ &= I + \frac{2^{-K}}{3} J_K - \frac{2^{-K}}{3} J_K = I \end{aligned}$$

where the second to last equality follows from facts 1 and 2 of our induction hypothesis.

The off-diagonal blocks are:

$$\begin{aligned} B = D &= \frac{2}{3}\boldsymbol{C}_K J_K - 2^{-2K-1} J_K \boldsymbol{C}_K^{-1} - \frac{2^{-2K-1}}{3} J_K J_K \\ &= \frac{4 \times 2^{-K-1}}{3} J_K - 2^{-K-1} J_k - \frac{2^{-K-1}}{3} J_k = 0 \end{aligned}$$

This concludes our induction for fact 4. Fact 3 is then immediate using fact 4 together with facts $1, 2$ from the induction hypothesis. □

Corollary B.3 below can be derived directly from Lemma B.2 by induction.

Corollary B.3. *For any positive integer k, the diagonal entries of $\boldsymbol{C}_j^{-1}$ all equal to $1 + j/3$.*

Proof of Proposition 3.13. The first inequality directly follows from the fact that

$$\|\boldsymbol{v}\|_2^2 = \sum_{i=1}^{m} v_i^2 \le m\|\boldsymbol{v}\|_\infty^2$$

for any m-dimensional vector $\boldsymbol{v}$.

The second inequality directly follows from the fact that

$$\|\mathbb{E}[V]\|_\infty \le \mathbb{E}[\|V\|_\infty]$$

for any random vector V. □

Proof of Lemma 3.14, upper bound. To prove the upper bound, our main idea is to argue that any fixed summation of the form $\sum_{L=I}^{I+k} X_L$ can be rewritten as a summation $\sum_{l=1}^{T} Y_l$, which satisfies the following properties:

- Each Y_l has variance σ^2
- Each pair of Y_i, Y_j has a non-positive correlation
- The term of summations $T = O(K)$.

Assuming all the three properties hold, putting these altogether shows

$$\mathsf{Var}[\sum_{L=I}^{I+k} X_L] = \mathsf{Var}[\sum_{l=1}^{T} Y_l] \le \sum_{l=1}^{T} \mathsf{Var}[Y_l] = \sigma^2 T \le CK\sigma^2.$$

The way we construct $\sum_{l=1}^{T} \mathsf{Var}[Y_l]$ is by defining a "merge" operation. Given a continuous summation $\sum_{L=I}^{I+k} X_L$, we will merge the summations using the equation $X_\star = X_{\star 0} + X_{\star 1}$ for any $\star \in \{0,1\}^k$ for $0 \le k \le K-1$ as much as possible. For example, summing over all leaves in the left subtree, i.e, $\sum_{L=000\ldots0}^{0111\ldots1} X_L$ can be merged to X_0. We can merge these summations until no further merging can happen, then we have $\sum_{L=I}^{I+k} X_L = \sum_{l=1}^{T} Y_l$, where each Y_l is the random variable associated with a (not-necessarily leaf) node in the original tree. It is proven in Theorem 3.3 that each Y_l has variance σ^2, which confirms Property 1 above. The rest two properties are justified by the two lemmas below. □

Lemma B.4. *Let S_1, S_2 be two non-empty subsets of $\{0,1\}^K$ and $S_1 \cap S_2 = \varnothing$. Let $Z_1 := \sum_{i\in S_1} X_i$ and $Z_2 = \sum_{j\in S_2} X_j$, then $\mathsf{Cov}(Z_1, Z_2) \le 0$. Therefore $\mathsf{Cov}(Y_l, Y_{l'}) \le 0$ for any $l \neq l'$.*

Proof of Lemme B.4. We observe $\mathsf{Cov}(Z_1, Z_2) = \sum_{i\in S_1, j\in S_2} \mathsf{Cov}(X_i, X_j)$. Since the covariance matrix $\sigma^2 \boldsymbol{C}_K$ in (3.1) has only positive entries on the diagonal, but the above summation does not include the diagonal, we conclude $\mathsf{Cov}(Z_1, Z_2) \le 0$. □

Lemma B.5. *Any continuous summation $\sum_{L=I}^{I+k} X_L$ can be merged as a summation $\sum_{l=1}^{T} Y_l$ with $T \le 2K-2$ when $K \ge 2$.*

Proof of Lemma B.5. We first prove any continuous summation of the form $\sum_{L=000,\dots,0}^{T} X_L$ can be merged into at most K summations. The case where $K = 1$ is straightforward. Suppose $K = m-1$ is proven, when $K = m$, we pick T such that the summation $\sum_{L=000\dots0}^{T} X_L$ has the largest number of summing terms after merging. We may assume without loss of generality that $L \ge 100, \dots, 0$, as otherwise the summation only happens on the left sub-tree which reduces to our inductive hypothesis. Now we can already split the first left-tree in the summation, and write $\sum_{L=000\dots0}^{T} X_L = X_0 + \sum_{L=100\dots0}^{T} X_L$. The second term has length at most $m-1$ by inductive hypothesis, as the summation happens at the leftmost node of the right subtree. Therefore the total length is at most m, as claimed. Moreover, the symmetries of the complete binary tree shows any summation of the form $\sum_{L=T'}^{111,\dots,1} X_L$ can also be merged into at most K summations.

Now we prove Lemma B.5. The $K = 2$ case is simple. We assume the claim holds when $K \le m-1$. When $K = m$, we pick the consecutive summation which has the largest number of terms after merging. Again, we may assume the summation includes both nodes on the left and right subtree, as otherwise it reduces to the $m-1$ case. Now we write $\sum_{L=I}^{I+k} X_L = W_1 + W_2$, where W_1 is the summation on the left part ending at the rightmost node of the left subtree, and W_2 on the right part starting at the leftmost node at the right subtree. By our previous induction, both W_1 and W_2 can be merged into a summation of at most $m-1$ terms. Therefore the whole summation has at most $2m-2$ terms. □

These results together give us the upper bound. Now we turn to the lower bound, we will inductively construct a sequence of continuous range summations for each K, and argue the variance scales as $\Theta(K)$.

We first assume K is an odd number, then we consider the consecutive sum by picking the first $M_K := 2^{K-1} + 2^{K-3} + 2^{K-5} + \dots + 1$ entries. Intuitively, we are picking all the left subtree, and the left subtree of the right subtree, and the left subtree of the right subtree of the right subtree, and so on.

Proof of Lemma 3.14, lower bound. For any odd number K, Let

$$V_K := \mathsf{Var}[\sum_{k=000\dots0}^{M_K} X_k],$$

where $M_K := 2^{K-1} + 2^{K-3} + 2^{K-5} + \dots + 1$. Then we claim $V_K - V_{K-2} \ge \frac{2}{3}\sigma^2$. Given the claim, we have $V_K/\sigma^2 = \Omega(K)$ for odd K. Since the maximum variance is a non-decreasing function of K, we know

$$\max_{I\in\{0,1\}^K, k\le 2^K - I} \frac{\mathsf{Var}[\sum_{L=I}^{I+k} X_L]}{\sigma^2} = \Omega(K).$$

To show the claim, we first merge the summation of the these M_K entries, and write

$$\sum_{k=000\dots0}^{M_K} X_k = \sum_{l=1}^{(K+1)/2} Y_l,$$

where Y_1 is the left subtree, Y_2 is the next summation of 2^{K-3} entries, and so on. Next we define $W_K := \sum_{l=2}^{(K+1)/2} Y_l$. By the recursive structure of our covariance matrix, W_K has variance V_{K-2}. Therefore

$$V_n = \mathsf{Var}\left[\sum_{k=000,\ldots,0}^{M_K} X_k\right] = \mathsf{Var}[Y_1 + W_K] = \sigma^2 + V_{K-2} + 2\mathsf{Cov}[Y_1, W_K].$$

The covariance term can be calculated through (3.1), which is $-2^{-2(K-1)-1}\sigma^2|Y_1||W_K|$. Therefore we can further bound V_K as

$$\begin{aligned}
V_K &= \mathsf{Var}\left[\sum_{k=000\ldots0}^{M_K} X_k\right] = \mathsf{Var}[Y_1 + W_K] = \sigma^2 + V_{K-2} + 2\mathsf{Cov}[Y_1, W_K] \\
&= \sigma^2 + V_{K-2} - \sigma^2 2^{-2(K-1)} 2^{K-1} M_{K-2} \\
&= \sigma^2 + V_{K-2} - \sigma^2 2^{-(K-1)}(2^{K-3} + 2^{K-5} + \ldots + 1) \\
&\geq \sigma^2 + V_{K-2} - \sigma^2 \frac{4}{3} 2^{-(K-1)} 2^{K-3} = V_{K-2} + \frac{2}{3}\sigma^2,
\end{aligned}$$

as desired. The last inequality is because

$$\begin{aligned}
2^{K-3} + 2^{K-5} + \ldots + 1 &= 2^{K-3}(1 + 1/4 + \ldots + 2^{-(K-3)}) \\
&\leq 2^{K-3}(1 + 1/4 + \ldots + 2^{-(K-3)} + \ldots) \\
&= \frac{4}{3} 2^{K-3}.
\end{aligned}$$

□

Proof of Theorem 3.15. Recall the classical fact that $\mathbb{E}[|X|] = \sigma\sqrt{2/\pi}$ for $X \sim \mathbb{N}(0, \sigma^2)$, we immediately have:

$$\begin{aligned}
\mathsf{err}^{\infty}_{\boldsymbol{W}}(\mathcal{W}_\sigma) &= \left\|\mathbb{E}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[|\boldsymbol{W}\boldsymbol{s}|]\right\|_\infty \\
&= \sqrt{\frac{2}{\pi}} \max_{I\in\{0,1\}^K, k\leq 2^K - I} \sqrt{\mathsf{Var}[\sum_{L=I}^{I+k} X_L]} \\
&= \Theta(\sigma\sqrt{K}) \\
&= \Theta\left(\sigma\sqrt{\log_2(n)}\right),
\end{aligned}$$

which proves the first claim. The second claim follows immediately from the first claim and Proposition 3.13.

For the last claim, recall the Gaussian tail bound $\mathbb{P}(|X| > t) \leq 2\exp(-t^2/2\sigma^2)$ given $X \sim \mathbb{N}(0, \sigma^2)$. We can bound the tail probability of the worst-case error as:

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty > t] &= \mathbb{P}[|\sum_{L=I}^{I+k} X_L| > t \text{ for some } \ I \in \{0,1\}^K, k] \\
&\leq \sum_{I,k} \mathbb{P}[|\sum_{L=I}^{I+k} X_L| > t] \\
&\leq 2\sum_{I,k} \exp\left(\frac{-t^2}{2\mathsf{Var}(\sum_{L=I}^{I+k} X_L)}\right) \\
&\leq 2\sum_{I,k} \exp\left(\frac{-t^2}{2\max \mathsf{Var}(\sum_{L=I}^{I+k} X_L)}\right) \\
&\leq 2\binom{n}{2} \exp\left(\frac{-t^2}{2c\sigma^2 K}\right) \\
&\leq n^2 \exp\left(\frac{-t^2}{2c\sigma^2 K}\right).
\end{aligned}$$

Here the first inequalty follows from the union bound, and the summation is over all the continuous range queries (and therefore $\binom{n}{2}$ terms). The second inequality uses the Gaussian tail bound. The last few inequalities use Lemma 3.14.

Now set $t_m = m\sqrt{4\log(n)c\sigma^2 K}$ for every positive integer m, we know

$$\begin{aligned}\mathbb{P}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty > t_m] &\le n^2 \exp\left(\frac{-t_m^2}{2c\sigma^2 K}\right)\\ &= n^2 \exp\left(\frac{-4m^2\log(n)c\sigma^2 K}{2c\sigma^2 K}\right)\\ &= n^2\exp\left(-2m^2\log(n)\right)\\ &= \frac{1}{n^{2m^2-2}}.\end{aligned}$$

Therefore we can bound

$$\begin{aligned}\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) &= \mathbb{E}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty]\\ &= \int_0^\infty \mathbb{P}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty > t]\mathrm{d}t\\ &\le t_1 + \sum_{m=1}^\infty \int_{t_m}^{t_{m+1}} \mathbb{P}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty > t]\mathrm{d}t\\ &\le \sqrt{4\log(n)c\sigma^2K} + \sqrt{4\log(n)c\sigma^2K}\sum_{m=1}^\infty \mathbb{P}_{\boldsymbol{s}\sim\mathbb{N}(\boldsymbol{0},\sigma^2\boldsymbol{C}_K)}[\|\boldsymbol{W}\boldsymbol{s}\|_\infty > t_m]\\ &= \sqrt{4\log(n)c\sigma^2K} + \sqrt{4\log(n)c\sigma^2K}\sum_{m=1}^\infty \frac{1}{n^{2m^2-2}}\\ &= O(\sqrt{4\log(n)c\sigma^2K}) = O(\sigma\log(n)) \qquad \text{as} \quad K = \log_2(n).\end{aligned}$$

□

Proof of Corollary 3.16. Corollary 3.16 is immediate when plugging the value of σ in Theorem 3.9 into Theorem 3.15. □

C Extra results on nodal queries

C.1 Theoretical results

When $\boldsymbol{W}$ represents the queries for all the nodes in the binary tree. Our results are summarized below:

Theorem C.1. *Fix $\sigma > 0$ and let query matrix $\boldsymbol{W}$ be all the nodal queries, we have:*

- $\mathsf{err}^\infty_{\boldsymbol{W}}(\mathcal{W}_\sigma) = \sqrt{2/\pi}\sigma$,
- $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma) = (2n-1)\sigma^2$,
- $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) = O(\sigma\sqrt{\log_2(n)})$.

Proof of Theorem C.1. The first two properties are follows from direct calculation, and are therefore omitted here.

The proof for the worst-case expected error is very similar to the calculation in Theorem 3.15. Recall that all the leaf nodes are labelled by X_I where $I \in \{0,1\}^K$ which has covariance matrix $\sigma^2\boldsymbol{C}_K$. The internal nodes are labelled by X_J where $J \in \{0,1\}^k$ for some $k \in \{0,1,\ldots,K-1\}$. Therefore $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) = \mathbb{E}[\max_{k,J}|X_J|, J \in \{0,1\}^k]$. We can again use the standard Gaussian concentration:

$$\begin{aligned}\mathbb{P}[\max_{k,J}|X_J| > t] &= \mathbb{P}[|X_J| > t \text{ for some } \ J \in \{0,1\}^k] \\ &\leq \sum_{J,k} \mathbb{P}[|X_J| > t] \\ &= (2n-1)\exp\left(\frac{-t^2}{2\sigma^2}\right) \qquad \text{total } 2n-1 \text{ summation terms.}\end{aligned}$$

Now set $t_m = m\sqrt{2\sigma^2\log(n)}$, we know

$$\mathbb{P}[\max_{k,J}|X_J| > t_m] \leq \frac{(2n-1)}{n^{m^2}}.$$

Therefore

$$\begin{aligned}\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) &= \int_0^\infty \mathbb{P}[\max_{k,J}|X_J| > t]\mathrm{d}t \\ &= \sqrt{2\sigma^2\log(n)} + \sqrt{2\sigma^2\log(n)}\sum_{m=1}^{\infty}\frac{(2n-1)}{n^{m^2}} \\ &= O(\sigma\sqrt{\log(n)}),\end{aligned}$$

as announced. □

Corollary C.2. *Let $X \in \mathcal{X}^n$ be any dataset and let $\mathcal{M}_\sigma(X) = X + \sigma\mathbb{N}(\mathbf{0}, \boldsymbol{C}_K)$ be our privacy mechanism, where σ is chosen such that the mechanism satisfies (ε,δ)-DP. Let $\boldsymbol{W}$ be all the continuous range queries, we have:*

- $\mathsf{err}^{\infty}_{\boldsymbol{W}}(\mathcal{W}_\sigma) = \Theta\left(\sqrt{\log n\log(2/\delta)}\varepsilon^{-1}\right)$,
- $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma) = \Theta\left(n^2\log(n)\log(2/\delta)\varepsilon^{-2}\right)$,
- $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma) = O\left(\log(n)\sqrt{\log(2/\delta)}\varepsilon^{-1}\right)$.

D Generalizations

D.1 Cascade Sampling for Streaming data

Algorithm 2: Cascade Sampling Algorithm for Streaming Data

Input: Depth of the current binary tree $k \geq 1$, current root noise value X_0, variance σ^2 determined by the privacy budget.
Output: Noise values $\{X_I\}$ for all nodes $I \in \cup_{0\leq i\leq k-1}\{1\}\times\{0,1\}^i$; new root noise X'.
Sample $Y_\star \sim \mathbb{N}(0,\sigma^2)$ independently;
Set $X_1 := -\frac{1}{2}X_0 + \frac{\sqrt{3}}{2}Y_\star, \quad X' := X_0 + X_1$;
for *each node* $\star \in \{1\}\times\{0,1\}^i$ *at depth* $1 \leq i \leq k-1$ **do**
  Sample $Y_\star \sim \mathbb{N}(0,\sigma^2)$ independently;
  Define the noise values for the children of $\star$: ;
  $X_{\star 0} := \frac{1}{2}X_\star + \frac{\sqrt{3}}{2}Y_\star, \quad X_{\star 1} := \frac{1}{2}X_\star - \frac{\sqrt{3}}{2}Y_\star$;
end

D.2 General binary trees

Our noise mechanism is designed under the assumption that our data points can be viewed as leaves of a perfect binary tree. Nevertheless, thank to the top-down nature of our cascade sampling algorithm (Algorithm 1), it

can be readily adapted to work with a general binary tree. We will still use binary bits to represent data points. They are still leaves of a binary tree of maximum depth k, but not necessarily have the same depth (i.e., same length in the bit string). In our binary tree, the root node is still indexed by $\varnothing$. For any non-leaf node identified by the bit string $\star$, its children are indexed as $\star 0$ and $\star 1$ if it has two children, or solely as $\star 0$ if it has only one child.

Our adapted algorithm closely mirrors Algorithm 1, with the primary distinction being the evaluation of the number of children at each step. Starting from the root, the process proceeds downward. Given the assigned value to a node $\star$, the algorithm first determines the number of $\star$'s children. If $\star$ has two children, it generates correlated noises in the same way as Algorithm 1 (or as described in formula 2). If $\star$ has only one child, the same noise is assigned to $\star 0$. If $\star$ is itself a leaf, the algorithm simply moves on to the next node. Detailed description is in Algorithm 3 below.

Algorithm 3: Noise Allocation Mechanism for a General Binary Tree

Input: Binary tree with maximum depth k, variance σ^2 determined by the privacy budget.
Output: Noise values $\{X_I\}$ for all nodes $I \in \cup_{0\le l\le k}\{0,1\}^l$ in a binary tree.
for *each node* $\star \in \{0,1\}^i$ *at depth* $0 \le i \le k-1$ **do**
  if $\star = \varnothing$ **then**
    Assign $X_\varnothing \sim \mathbb{N}(0,\sigma^2)$
    Sample $Y_\star \sim \mathbb{N}(0,\sigma^2)$ independently
    Define the noise values for the children nodes of $\star$:
    if $\star$ *has two children* **then**
      $X_{\star 0} := \frac{1}{2}X_\star + \frac{\sqrt{3}}{2}Y_\star$
    end
    $X_{\star 1} := \frac{1}{2}X_\star - \frac{\sqrt{3}}{2}Y_\star$
    else if $\star$ *has one child* **then**
      $X_{\star 0} = X_\star$
    end
  end
end

Algorithm 3 continues to ensure that the marginal distributions for each node adhere to $\mathbb{N}(0,\sigma^2)$. Moreover, the noise level for any given node is equivalent to the sum of the noise levels of its children, thereby maintaining consistency. The theoretical analysis for this scenario closely parallels that of the perfect binary tree case discussed in Section 3. Extending this approach to non-binary trees is also possible, though more complicated. This further exploration will be reserved for our future studies.

D.3 Two-dimensional range queries and contingency tables

Remember that our approach, as outlined in Algorithm 1, is appropriate for data arranged in one-dimensional arrays that have hierarchical structures. We are now expanding this method to apply to two-dimensional contingency tables. An example of this is that each data point represents the population of a village, positioned in two dimensions for longitude and latitude. The higher-level hierarchies might represent the populations of larger entities like cities, provinces, or countries.

Putting it formally, we consider data points labeled as $\{X_{I,J}\}_{I\in\{0,1\}^{k_1},J\in\{0,1\}^{k_2}}$. It is convenient to think of these data points as elements of a matrix with 2^{k_1} rows and 2^{k_2} columns. Each row and column corresponds to the leaf nodes of a perfect binary tree of heights k_1 and k_2, respectively. Each pair of (internal) nodes (I_1, J_1) from the two binary trees represents a submatrix. For instance, the pair $(\varnothing,\varnothing)$ corresponds to the entire data matrix. We aim to develop a noise mechanism where the summation of noises applied to each submatrix, as described by $(I_1,J_1)_{I_1\in\{0,1\}^{i_1},I_2\in\{0,1\}^{i_2},i_1\le k_1,i_2\le k_2}$, maintain the same variance.

Remember that for the one-dimensional scenario, the essence of our design hinges on the observation that if

$X, Y \sim \mathbb{N}(0,1)$, then the variables X_0, X_1 given by

$$X_0 = \frac{1}{2}X + \frac{\sqrt{3}}{2}Y, \quad X_1 = \frac{1}{2}X - \frac{\sqrt{3}}{2}Y.$$

also follow a $\mathbb{N}(0,1)$ distribution. The above construction can be generalized to higher dimensions. For example, let X, Y, Z, W be i.i.d. $\mathbb{N}(0,1)$ random variables, and define:

$$X_0 = \frac{1}{2}X + \frac{\sqrt{3}}{2}Y, \quad X_1 = \frac{1}{2}X - \frac{\sqrt{3}}{2}Y$$
$$\tilde{X}_0 = \frac{1}{2}Z + \frac{\sqrt{3}}{2}W, \quad \tilde{X}_1 = \frac{1}{2}Z - \frac{\sqrt{3}}{2}W.$$

Then we can construct

$$(X_{00}, X_{01}) = \frac{1}{2}(X_0, X_1) + \frac{\sqrt{3}}{2}\left(\tilde{X}_0, \tilde{X}_1\right)$$
$$(X_{10}, X_{11}) = \frac{1}{2}(X_0, X_1) - \frac{\sqrt{3}}{2}\left(\tilde{X}_0, \tilde{X}_1\right),$$

or equivalently

$$X_{00} = \frac{1}{4}X + \frac{\sqrt{3}}{4}Y + \frac{\sqrt{3}}{4}Z + \frac{3}{4}W$$
$$X_{01} = \frac{1}{4}X - \frac{\sqrt{3}}{4}Y + \frac{\sqrt{3}}{4}Z - \frac{3}{4}W$$
$$X_{10} = \frac{1}{4}X + \frac{\sqrt{3}}{4}Y - \frac{\sqrt{3}}{4}Z - \frac{3}{4}W$$
$$X_{11} = \frac{1}{4}X - \frac{\sqrt{3}}{4}Y - \frac{\sqrt{3}}{4}Z + \frac{3}{4}W.$$

We can verify that the matrix

$$\begin{pmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{pmatrix}$$

satisfies the following:

- Each entry is a standard normal
- The summations of each row, each column are all standard normals
- The summation of all entries are standard normals.
- $X_{00} + X_{10} = X_0, X_{01} + X_{11} = X_1$.

Building on this concept, we will now describe how to extend our method (Algorithm 1) for creating private noise mechanisms suitable for 2-dimensional range queries. Initially, we'll utilize Algorithm 1 to sample the summation of all 2^{k_2} columns, meaning the noise values indexed as $(\varnothing, J)$. Following that, we will implement the aforementioned strategy to divide each of these column totals into smaller groups of $2, 4, 8, \ldots, 2^{k_1}$ values. The detailed algorithm is described below.

It is not hard to verify from the construction that every $X_{I,J}$ has the same marginal distribution, and satisfies $X_{I,J} = X_{I0,J} + X_{I1,J}$ and $X_{I,J} = X_{I,J0} + X_{I,J1}$. The computational cost of Algorithm 4 is also linear with the number of data points. More detailed analysis of the appropriate level for the privacy budget, along with an analysis of utility, will be deferred to our future research.

Algorithm 4: Noise Allocation Mechanism for two-dimensional range queries

Input: Depths K_1, K_2 of the binary tree indexing rows and columns, variance σ^2 determined by the privacy budget.

Output: Noise values $\{X_{I,J}\}$ for all node pairs $(I,J) \in \cup_{0\le k_1\le K_1}\{0,1\}^{k_1} \times \cup_{0\le k_2\le K_2}\{0,1\}^{k_2}$

Fix $I = \varnothing$, implement Algorithm 1 to assign $X_{\varnothing J}$ for every $J \in \cup_{0\le k_2\le K_2}\{0,1\}^{k_2}$;

for *each node* $\star \in \{0,1\}^{k_1}$ *for some* $0 \le k_1 \le K-1$ **do**

Implement Algorithm 1 to assign $Y_{\star J}$ for every $J \in \cup_{0\le k_2\le K_2}\{0,1\}^{k_2}$;

Define the noise values for the children nodes of $\star$:;

$X_{\star 0,J} := \frac{1}{2}X_{\star,J} + \frac{\sqrt{3}}{2}Y_{\star,J}$ for every J;

$X_{\star 1,J} := \frac{1}{2}X_{\star,J} - \frac{\sqrt{3}}{2}Y_{\star,J}$ for every J;

end

E Additional experiments

In this section, we demonstrate numerical experiments for when the workload matrix consists of node and random queries and also runtime comparisons.

E.1 Efficiency of Cascade Sampling

To justify the practical efficiency of the Cascade Sampling algorithm described in Section 3.2, we compare it against `Scipy`'s built-in function for generating multivariate Gaussian distributions. On a personal laptop, we use both methods to produce Gaussians with a covariance matrix C_k for $k \in \{3,4,\dots,25\}$, which corresponds to dimensionalities ranging from $2^3 = 8$ to $2^{25} \approx 33.5$ million. In comparison to our sampling algorithm, `Scipy`'s `multivariate_normal` function consistently takes longer time. Its efficiency diminishes at higher dimensions. For example, it takes over 28 minutes to sample a $16,384$ (2^{14})-dimensional Gaussian, and more than 3 hours for a $32,768$ (2^{15})-dimensional Gaussian, making it impractical for larger dimensions. Samplers from other libraries like `Numpy` were also tested but could not execute for dimensions exceeding 1024.

E.2 Node queries

Consider a binary tree such that elements of $\boldsymbol{x}$ form the leaves of the tree. The queries are the nodes (including leaves) of this tree, which correspond to an interval summing up the leaves of tree rooted at the particular node. The error values are shown in Figure 3.

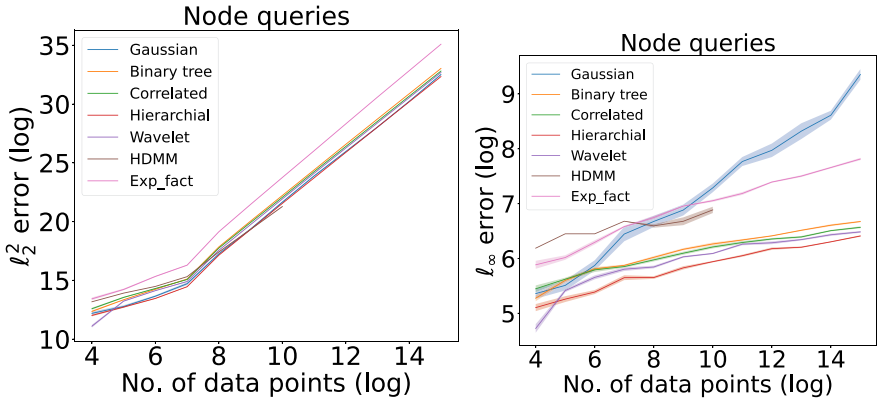


Figure 3: $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ for node queries.

The gap in utility across mechanisms for $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ is small with Hierarchial, Wavelet and ours providing smallest error values, then closely followed by remaining mechanisms. But that is not the case for $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$, where Gaussian, HDMM and explicit factorization perform significantly worse. The remaining mechanisms perform very similarly (except for Binary Tree which has slightly larger error) as n increases. The gap in

correlated input perturbation and Binary tree mechanism is easier to see which comes from the smaller constant for noise that is required for guaranteeing privacy ($2 + \frac{2}{3}\log n$ vs $\log n$).

E.3 Random queries

Here, each query asks for sum of random elements from $\boldsymbol{x}$. Note that unlike first two settings, the query is not contiguous over elements in $\boldsymbol{x}$. We sample the queries as follows - for each query we sample a number $k \in \{n/4, \cdots, n\}$ (to simulate dense queries). We then sample k indices uniformly at random from $[n]$ and these indices form the query (these indices are set to 1 for the particular row in $\boldsymbol{W}$). We fix number of queries to be 2500. The results are shown in Figure 4

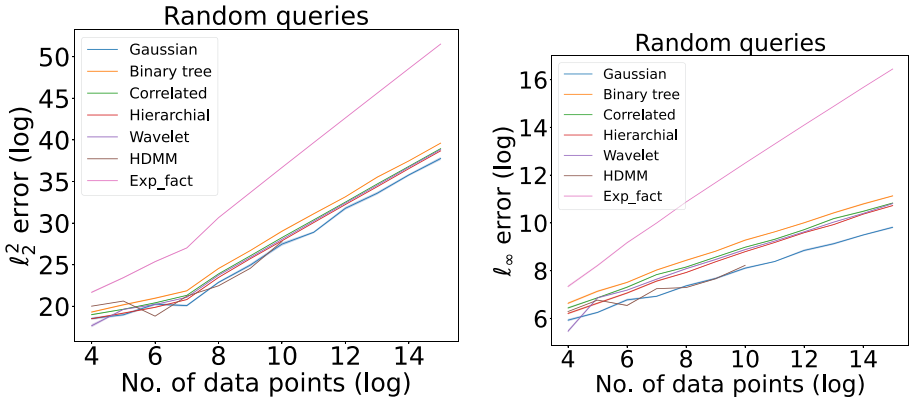


Figure 4: $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ for random queries.

For this setting of workload matrix, HDMM and Gaussian mechanism perform best utility wise for both $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$. Wavelet, ours and Hierarchical perform almost similarly w.r.t noise magnitude whereas explicit factorization mechanism has the largest noise. Explicit factorization uses a very special decomposition tailored to continual counting hence it has worse performance and Hierarchical, ours and Binary tree mechanisms are more suited for sum over contiguous arrays by design, and as a result for random queries they perform slightly worse. Gaussian mechanism has the least error majorly due to the fact that the noise scale is smallest due to smaller sensitivity. To see this, note that the queries sum over $O(n)$ elements in $\boldsymbol{x}$. For Gaussian, this corresponds to sum of $O(n)$ i.i.d Gaussians, whereas for Hierarchical and Correlated mechanism, worst case we still sum over $O(n)$ elements but with higher variance (by a factor of $O(\log n)$), resulting in larger noise added to the query answers.

E.4 Runtime comparison

We compare runtime for different mechanisms in seconds for answering the queries provided by the given workload matrix. For runtime, Gaussian and Wavelet are typically the fastest across different settings of workload matrix and n whereas our mechanism is competitive with these and faster than Hierarchical. HDMM typically takes the longnest which is expected as it involves expensive optimization to achieve the matrix decomposition. The time required for explicit factorization mechanism majorly comes from answering queries, as their decomposition is fast but for each query, the answer has to be constructed via difference of two other queries.

E.5 Implementation Details

We first describe the implementation for all mechanisms used in evaluations. All experiments were performed on a Macbook Pro with M1 processor and 16GB of RAM. Code is implemented in python language.

Binary Tree: Here the mechanism builds a binary tree with elements being the leaves of the tree, and appropriately scaled Gaussian noise (for preserving privacy) is added to the the counts of nodes. We use the segment tree package for building and answering queries.

Hierarchical and Wavelet: Hierarchical mechanisn follows the same methodology as the Binary tree mechanism, followed by post-processing to find consistent sums for the internal nodes. We use the implementation from dpcomp-core package [25] for Hierarchical and Wavelet.

HDMM: We use the implementation from dpcompcore library, using the default parameter for number of restarts.

Explicit factorization [17] provides implicit matrix factorization for continual counting, i.e., counting of ranges $[1, i]$, for all i. Notice that this is a special case of range queries $[i, j]$ for all . One can easily retrieve the query answer $[i, j]$ by calculating the privatized query answer for $[1, j]$ minus that of $[1, i-1]$. Privacy is preserved (post-processing) although variance for query of $[i, j]$ can possibly increase (at most twice).

Gaussian and Correlated mechanism: Implementing these mechanisms is straight forward by adding appropriately scaled noise for privacy and then using the private histograms to answer queries.

Monte Carlo sampling for estimating errors of continuous queries: Now we discuss some implementation details of the continuous range queries in Section 5. After implementing the Cascade Sampling algorithm for the correlated perturbation $\mathcal{M}_\sigma(\boldsymbol{x}) = \boldsymbol{x} + \mathbb{N}(\mathbf{0}, \sigma^2 \boldsymbol{C}_k)$, calculating the error introduced by *all* the $\Theta(n^2)$ continuous range queries requires a cost of $O(n^3)$ to sum up all the noises in the corresponding entries, which is prohibitive when n is large.

Our primary goal is to evaluate the errors $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$ in comparison to current algorithms, so we employ the Monte Carlo method for estimating these errors. It is clear from the mean and standard deviation plot presented in Figure 2 that our estimation is sufficiently accurate.

We now outline the implementation process for $n \geq 2^8$. In each iteration, we uniformly select one range from all $\binom{n}{2} + n$ queries. This step is repeated m times. Using the m sampled errors $E_1, \ldots, E_m$, we estimate $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$ and $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$. For instance, $(\binom{n}{2} + n) \times \sum_{i=1}^{m}(E_i^2)/m$ serves as an unbiased estimator for $\mathsf{err}_{\boldsymbol{W},2}(\mathcal{W}_\sigma)$, becoming more accurate as m increases. Additionally, $\max_i E_i$ consistently estimates $\mathsf{err}_{\boldsymbol{W},\infty}(\mathcal{W}_\sigma)$.

In further details, to uniformly select a continuous range, our sampling method involves: 1. Flip a coin with head probability $2/(n+1)$. 2. If heads, then random choose one element in $\{1, \ldots, n\}$. 3. If tails, uniformly choose two distinct elements and use $\min\{n_1, n_2\}$ to $\max\{n_1, n_2\}$ as our range.

A final remark is that one could compute all $\binom{n}{2} + n$ errors without Monte Carlo methods using faster than $O(n^3)$ algorithms by properly storing intermediate results. However, these are not implemented in our current work.

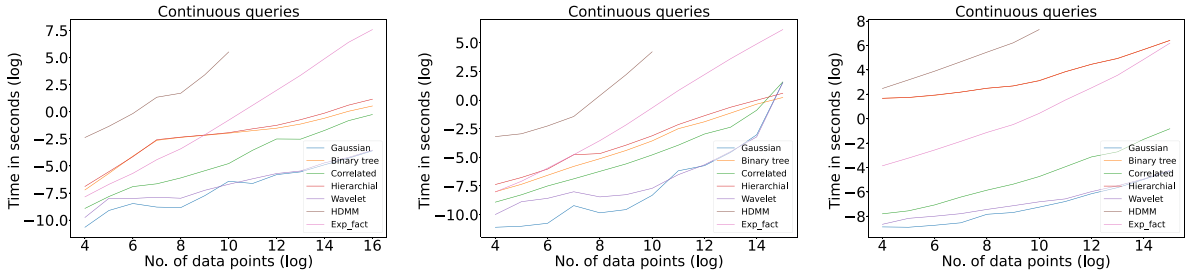


Figure 5: Runtime comparison for different configurations of the workload matrix.