Fast Private Adaptive Query Answering for Large Data Domains

Miguel Fuentes*
University of Massachusetts Amherst

Brett Mullins*
University of Massachusetts Amherst

Yingtai Xiao
TikTok

Daniel Kifer
Penn State University

Cameron Musco
University of Massachusetts Amherst

Daniel Sheldon
University of Massachusetts Amherst

Abstract

Privately releasing marginals of a tabular dataset is a foundational problem in differential privacy. However, state-of-the-art mechanisms suffer from a computational bottleneck when marginal estimates are reconstructed from noisy measurements. Recently, *residual queries* were introduced and shown to lead to highly efficient reconstruction in the batch query answering setting. We introduce new techniques to integrate residual queries into state-of-the-art adaptive mechanisms such as AIM. Our contributions include a novel conceptual framework for residual queries using multi-dimensional arrays, lazy updating strategies, and adaptive optimization of the per-round privacy budget allocation. Together these contributions reduce error, improve speed, and simplify residual query operations. We integrate these innovations into a new mechanism (AIM+GReM), which improves AIM by using fast residual-based reconstruction instead of a graphical model approach. Our mechanism is orders of magnitude faster than the original framework and demonstrates competitive error and greatly improved scalability.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Releasing marginal statistics under differential privacy (DP) is vital for sharing data insights while protecting privacy. Marginals (frequency distributions over attribute subsets) are crucial for descriptive statistics, downstream analyses, and various applications. We consider the problem of answering a workload $\mathcal{W}$ of marginal queries with a mechanism that satisfies differential privacy, where each $\gamma \in \mathcal{W}$ is a set of attributes for which we want to estimate a marginal μ_γ (an array of counts for each possible setting of those attributes). The most common strategy for this problem is the "select-measure-reconstruct" paradigm. Instead of naively measuring every query in $\mathcal{W}$ via a simple mechanism such as noise addition, this approach involves three steps: first, the mechanism intelligently *selects* a query or set of queries; second, it privately *measures* these selected queries; and third, it uses these measurements to *reconstruct* estimates for all queries in the original workload. We focus on *adaptive* or *data-dependent* approaches, which iteratively select and measure marginals. MWEM (Hardt and Rothblum, 2010) was an influential early mechanism of this type and has been followed by a family of successful mechanisms (McKenna et al., 2022; Fuentes et al., 2024; Liu et al., 2021a,b). Adaptive approaches typically give lower error than batch approaches such as the matrix mechanism family of algorithms (Li et al., 2015; McKenna et al., 2023; Xiao et al., 2023), especially with a very restrictive privacy budget, because they can save budget by only measuring poorly approximated marginals.

AIM (McKenna et al., 2022) is a state-of-the-art select-measure-reconstruct method for marginal queries and synthetic data. However, its performance is limited on large data domains by the computational tractabil-

ity of Private-PGM (Mckenna et al., 2019), its reconstruction method. Private-PGM solves a principled optimization problem to reconstruct a graphical model representation of the data distribution, but for some sets of measured marginals the representation becomes computationally intractable. AIM therefore employs a heuristic during selection to avoid measurements that would cause the model size to exceed a specified limit. Even with this safeguard, the running time of AIM is prohibitive in some scenarios.

Recent work has introduced a promising new reconstruction approach based on *residual queries* (Xiao et al., 2023; Mullins et al., 2024). Residuals are statistical queries that can be viewed as decompositions of marginals into tables with non-overlapping information. The GReM (Gaussian residuals-to-marginals) algorithm (Mullins et al., 2024) shows how to convert marginal measurements with Gaussian noise to equivalent residual measurements and then reconstruct workload marginals from the noisy residuals via a principled optimization problem. Unlike Private-PGM, this reconstruction is always efficient. Therefore, it is very promising for use in adaptive mechanisms.

Our work advances this line of research with several key contributions. First, we introduce a novel *in-axis formulation* that uses elementary array operations to define and convert between marginals and residuals. This approach clarifies these operations, simplifies the complexity analysis, and speeds marginal decomposition. Second, we contribute *lazy updating*, a procedure significantly speeding up GReM reconstruction in the context of an iterative mechanism. Third, we develop *conditional ResidualPlanner* to optimize noise allocation during measurement, named after the ResidualPlanner method which optimizes noise allocation in the batch setting (Xiao et al., 2023). Finally, we integrate these ideas into AIM to form AIM+GReM, an end-to-end adaptive mechanism that measures residuals and uses GReM for reconstruction. AIM+GReM is orders of magnitude faster than AIM and achieves favorable time-vs-utility tradeoffs compared to AIM and ResidualPlanner, the optimal batch method. Additionally, conditional ResidualPlanner provides an optimal solution for the problem of noise allocation under pre-existing measurements, a result of independent interest for any adaptive or multi-stage measurement strategy.

2 PRELIMINARIES

Let $\mathcal{D}$ be a discrete tabular dataset consisting of records $x^{(1)}, \ldots, x^{(N)}$. Each record $x = (x_1, \ldots, x_d)$ contains d categorical attributes. The ith attribute x_i belongs to the finite set $\mathcal{X}_i = \{1, \ldots, n_i\}$. The data universe is $\mathcal{X} = \prod_{i=1}^{d} \mathcal{X}_i$ and has size $n = \prod_i n_i$. We represent a dataset $\mathcal{D}$ as a multidimensional array $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with one dimension for each attribute, where $\mathcal{A}[i_1, \ldots, i_d] = \sum_{j=1}^{N} \prod_{k=1}^{d} \mathbb{I}[x_k^{(j)} = i_k]$. In prior work, the dataset $\mathcal{D}$ was represented as a vector $p \in \mathbb{R}^n$, called the *data vector*, obtained by flattening the array $\mathcal{A}$ into an $n \times 1$ vector (Hardt and Rothblum, 2010; Hardt et al., 2012; Li et al., 2015; McKenna et al., 2023; Xiao et al., 2023; Mullins et al., 2024).

Marginals. Marginals are a common type of statistical query that describe the frequency or count of combinations of values across a subset of attributes. For example, on a demographics dataset, the marginal over age group and education level counts the number of individuals for each combination of age group (e.g., 18-25, 26-35, 36-50, 50+) and level of education (e.g. no college, some college, obtained degree). For a subset of attributes $\gamma \subseteq [d]$, the marginal over γ, given by μ_γ, is an array with $|\gamma|$ dimensions, where $\mu_\gamma[i_1, \ldots, i_{|\gamma|}] = \sum_{j=1}^{N} \prod_{k \in \gamma} \mathbb{I}[x_k^{(j)} = i_k]$. Figure 1a shows an example marginal over age and education level, while Figure1b shows the marginal over age obtained by summing the values of education level from the two way marginal.

$$\mu_{\{\text{Age,Educ}\}} = \begin{bmatrix} 7 & 5 & 2 \\ 3 & 5 & 11 \\ 10 & 2 & 11 \\ 9 & 18 & 17 \end{bmatrix}$$

(a) Marginal over Age and Education.

$$\mu_{\{\text{Age}\}} = \begin{bmatrix} 14 \\ 19 \\ 23 \\ 44 \end{bmatrix} \qquad \zeta_{\{\text{Age}\}} = \begin{bmatrix} 5 \\ 9 \\ 30 \end{bmatrix}$$

(b) Marginal over Age. (c) Residual over Age.

Figure 1: Example Marginals and Residual.

Differential privacy. Differential privacy is a framework that ensures the privacy of individuals in a dataset by limiting the influence of any single record on the output of a computation. This approach is motivated by the need to balance data utility with privacy, enabling the safe analysis of sensitive information.

Definition 2.1. (Zero Concentrated Differential Privacy; Bun and Steinke (2016)) Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a randomized mechanism. For any datasets $\mathcal{D}, \mathcal{D}'$ that differ by at most one record, $\mathcal{M}$ satisfies ρ-zero concentrated differential privacy, denoted ρ-zCDP, if

$$D_\alpha \left(\mathcal{M}(\mathcal{D}) \| \mathcal{M}(\mathcal{D}') \right) \leq \alpha\rho \tag{1}$$

for all $\alpha \in (1, \infty)$, where D_α denotes the α-Renyi divergence.

This definition of differential privacy is convenient for our setting and can be converted to the more common

(ϵ, δ)-approximate differential privacy (Dwork et al., 2006). This conversion and other privacy properties are given in Appendix Section B.

3 RESIDUALS AND MARGINALS AS MULTI-DIMENSIONAL ARRAYS

In this section, we introduce residuals via a new and simplified framework. Traditionally, residuals are defined as transformations of marginals via query matrices constructed explicitly using Kronecker products and other algebraic techniques. Our approach departs from this vector-based approach, instead defining the same transformations via simple and efficient "in-axis" transformations of multi-dimensional arrays. Due to the complexity involved in the previous approach, here we present only the new view. In Appendix Section C, we give background on the previous approaches and demonstrate that our construction is equivalent to the methods in Xiao et al. (2023) and Mullins et al. (2024).

Residuals from marginals. The residual ζ_τ over attributes $\tau \subseteq [d]$ is an array with $|\tau|$ dimensions defined using Algorithm 1 as $\zeta_\tau = \texttt{Decomp}(\mu_\tau, \tau, \tau)$, where μ_τ is the τ marginal.[1] This applies a subtraction operator (details below) to each dimension of the τ marginal, leading to an array of size one smaller in each dimension. For example, $\zeta_{\{\text{Age}\}}$ in Figure 1 is obtained by subtracting the first entry of $\mu_{\{\text{Age}\}}$ from each of the last three entries. We can call $\texttt{Decomp}(\mu_\gamma, \gamma, \tau)$ to *decompose* μ_γ into component residuals ζ_τ for $\tau \subseteq \gamma$. The `Recon` procedure (Algorithm 2) then *reconstructs* a marginal from component residuals. A marginal and its component residuals express the same information in the sense that `Decomp` and `Recon` provide an invertible linear transformation between them:

Proposition 3.1. (Mullins et al., 2024) *The function* $T_\gamma(\mu_\gamma) = \{\texttt{Decomp}(\mu_\gamma, \gamma, \tau) \,|\, \tau \subseteq \gamma\}$ *is an invertible linear transformation between a marginal* μ_γ *and a set of residuals* $\zeta_\gamma = \{\zeta_\tau \mid \tau \subseteq \gamma\}$. *The inverse is given by* $T_\gamma^{-1}(\zeta_\gamma) = \sum_{\tau \subseteq \gamma} \texttt{Recon}(\zeta_\tau, \tau, \gamma)$.

Note that this bijection is between a single marginal, μ_γ, and the set of all its component residuals, $\{\zeta_\tau \mid \tau \subseteq \gamma\}$. A single marginal cannot be recovered from just one of its component residuals. A worked example of this process is provided in Appendix Section A along with related proofs.

Benefits of residuals While one marginal and its component residuals contain the same information, we can use residuals to efficiently perform inference tasks involving noisy marginals that are not tractable without residual decomposition. This process of inference from noisy marginals via residuals is covered in Section 4.1. Additionally, the running time of `Decomp` and `Recon` is nearly linear in the size n_γ of μ_γ, the larger of the input and output arrays, so it is tractable to use residuals as an intermediate representation:

Theorem 3.2. *Let* μ_γ *be a marginal over attributes* γ *and* ζ_τ *be a residual over attributes* τ *such that* $\tau \subseteq \gamma$. *Then* $\text{Decomp}(\mu_\gamma, \gamma, \tau)$ *and* $\text{Recon}(\zeta_\tau, \tau, \gamma)$ *take* $\mathcal{O}(|\gamma| n_\gamma)$ *time, where* $n_\gamma = \prod_{i \in \gamma} n_i$.

3.1 Transformations via In-Axis Operations

The primary advantage of our framework is its structural approach to the transformations used to define and compute with residuals. Instead of defining them via a complicated algebraic construction, we define them as a sequence of simple operations applied to the axes of a multi-dimensional array. This decomposition of a complex transformation into "in-axis" steps simplifies the conceptual model and its implementation. Specifically, each "in-axis" operation is a linear transformation applied to all 1D vectors along a specified axis of the array. For example, `sub(z, axis = i)` applies a differencing operation to every vector along axis i. The operations underpinning our `Decomp` and `Recon` procedures are detailed in Table 1.

Table 1: Along-dimension operations. (Note that `smear` is always applied to a singleton axis, making `v` a scalar.)

Operation	Implementation
`_sum(v)`	`sum(v)`
`_sub(v)`	`v[1:]-v[0]`
`_center(v)`	`u = [0, v]` `return u-mean(u)`
`_smear(v, k)`	`v/k * ones(k)`

These operations yield the following behavior when applied to the full array: `sum` marginalizes the specified axis to size one; `sub` reduces the axis size by one via differencing against the first slice to capture variations; `center` restores the axis dimension lost by `sub` by inserting an initial slice of zeros then subtracting the mean across the specified axis, and `smear` expands a unit axis to length `k` through scaling by `1/k` and broadcasting (so the sum across the axis is unchanged). These operations can be implemented efficiently with multi-dimensional array libraries, using built-in operations much faster than mapping individual transformations across all vectors along an axis.

[1] It is also true that $\zeta_\tau = \texttt{Decomp}(\mu_\gamma, \gamma, \tau)$ for any $\gamma \supseteq \tau$.

Algorithm 1 `Decomp` (μ, γ, τ)

Input: Marginal μ, marginal attributes γ, residual attributes $\tau \subseteq \gamma$
Output: Residual ζ for attributes τ

$\zeta \leftarrow \mu$
for $i \in \gamma \setminus \tau$ **do**
  $\zeta \leftarrow \texttt{sum}(\zeta, \texttt{axis} = i)$
for $j \in \tau$ **do**
  $\zeta \leftarrow \texttt{sub}(\zeta, \texttt{axis} = j)$
return ζ

Algorithm 2 `Recon` (ζ, τ, γ)

Input: Residual ζ, residual attributes τ, marginal attributes γ, axis lengths $\{n_i\}_{i \in \gamma \setminus \tau}$
Output: Marginal component $\mu_{\gamma,\tau}$ for attributes γ

$\mu_{\gamma,\tau} \leftarrow \zeta$
for $j \in \tau$ **do**
  $\mu_{\gamma,\tau} \leftarrow \texttt{center}(\mu_{\gamma,\tau}, \texttt{axis} = j)$
for $i \in \gamma \setminus \tau$ **do**
  $\mu_{\gamma,\tau} \leftarrow \texttt{smear}(\mu_{\gamma,\tau}, \texttt{axis} = i, \texttt{k} = n_i)$
return $\mu_{\gamma,\gamma}$

Figure 2: The decomposition (left, Algorithm 1) and reconstruction (right, Algorithm 2) procedures.

4 ALGORITHMIC IMPROVEMENTS FOR ITERATIVE MECHANISMS

Residuals are very promising for use in adaptive mechanisms. The GReM (Gaussian residuals-to-marginals) framework (Mullins et al., 2024) provides techniques for measuring noisy residuals and using them to reconstruct marginals. This section describes GReM and introduces algorithmic improvements to increase the speed and accuracy of measuring and reconstructing in an adaptive mechanism.

4.1 GReM

GReM is a method for reconstructing a set of marginals $\mathcal{W}$ from a collection of noisy marginal or residual measurements.

The measurement model assumes that noisy residuals are obtained by first adding isotropic Gaussian noise to a marginal and then applying the `Decomp` transformation. This procedure yields a noisy residual z_τ distributed as $z_\tau = \zeta_\tau + \mathcal{N}(0, \sigma_\tau^2 V_\tau)$, where ζ_τ is the true residual and V_τ is a covariance matrix derived from the `Decomp` transformation (Xiao et al., 2023; Mullins et al., 2024).[2]

The reconstruction is performed by the GReM-MLE algorithm, which is detailed in Algorithm 3. The procedure has two main stages. First, it computes a consolidated estimate for every residual τ in the downward closure of the target workload, $\mathcal{W}^\downarrow = \{\tau \subseteq \gamma \mid \gamma \in \mathcal{W}\}$. This is done by calculating the inverse-variance weighted average of all available measurements for that residual. If no measurements for τ exist, its estimate is zero. Second, the algorithm synthesizes each target marginal $\hat{\mu}_\gamma$ for $\gamma \in \mathcal{W}$ by summing the reconstructed components derived from the residual estimates $\{z_\tau \mid \tau \subseteq \gamma\}$.

[2]We interpret multidimensional arrays such as z_τ, ζ_τ as vectors in expressions for Gaussian distributions.

Algorithm 3 GReM-MLE Reconstruction (Mullins et al., 2024)

Input: Workload $\mathcal{W}$, noisy residuals $z_{\tau,i} = \zeta_\tau + \mathcal{N}(0, \sigma_{\tau,i}^2 V_\tau)$ for $\tau \in \mathcal{W}^\downarrow$, $i = 1, \dots k_\tau$
Output: Marginal estimates $\hat{\mu}_\gamma$ for each $\gamma \in \mathcal{W}$

$$z_\tau = \begin{cases} \frac{\sum_{i=1}^{k_i} z_{\tau,i} \sigma_{\tau,i}^{-2}}{\sum_{i=1}^{k_i} \sigma_{\tau,i}^{-2}} & k_\tau > 0 \\ 0 & k_\tau = 0 \end{cases} \text{ for } \tau \in \mathcal{W}^\downarrow$$

$\hat{\mu}_\gamma = \sum_{\tau \subseteq \gamma} \texttt{Recon}(z_\tau, \tau, \gamma)$ for $\gamma \in \mathcal{W}$

As formalized in Theorem 4.1, this two-stage procedure is guaranteed to produce estimates that are both consistent (i.e., there exists a data array $\hat{\mathcal{A}}$ that realizes all estimated marginals $\hat{\mu}_\gamma$ and residuals z_τ) and optimal in that they maximize the joint log-likelihood of the initial measurements.

Theorem 4.1. (Mullins et al., 2024) *GReM-MLE computes marginals $\hat{\mu}_\gamma$ and residuals z_τ that are consistent with a data array $\hat{\mathcal{A}}$ and, subject to this constraint, minimize the weighted squared error $\sum_{\tau \in \mathcal{W}^\downarrow} \sum_{i=1}^{k_\tau} \frac{1}{2\sigma_{\tau,i}^2} \|z_\tau - z_{\tau,i}\|_{V_\tau}^2$, where $\|u\|_{V_\tau}^2 = u^\top V_\tau^{-1} u$.*

4.2 Lazy Updating

When GReM-MLE is used within an iterative algorithm (e.g., one that acquires new measurements sequentially), its computations can be significantly accelerated. After each new measurement, a full reconstruction of all target marginals is inefficient and unnecessary.

Specifically, when a new measurement for a residual α is obtained, only the corresponding consolidated estimate z_α is affected by the inverse-variance weighted average update. All other residual estimates $\{z_\tau\}_{\tau \neq \alpha}$ remain unchanged. Due to the linear nature of the

reconstruction step, the impact of this single update on the marginals is localized and can be computed incrementally, as formalized below.

Proposition 4.2. *Suppose an algorithm maintains residual estimates $\{z_\tau\}_{\tau \in \mathcal{W}^\downarrow}$ and marginal estimates $\hat{\mu}_\gamma = \sum_{\tau \subseteq \gamma} \texttt{Recon}(z_\tau, \tau, \gamma)$ for all $\gamma \in \mathcal{W}$. If a single residual estimate z_α is updated to a new value z'_α, the new marginal estimates $\hat{\mu}'_\gamma$ are:*

$$\hat{\mu}'_\gamma = \begin{cases} \hat{\mu}_\gamma & \alpha \not\subseteq \gamma \\ \hat{\mu}_\gamma + \texttt{Recon}(z'_\alpha - z_\alpha, \alpha, \gamma) & \alpha \subseteq \gamma. \end{cases}$$

Proof. If $\alpha \not\subseteq \gamma$, the set of residuals summed to form $\hat{\mu}_\gamma$ is unchanged, so $\hat{\mu}'_\gamma = \hat{\mu}_\gamma$. Otherwise, if $\alpha \subseteq \gamma$:

$$\begin{aligned} \hat{\mu}'_\gamma &= \sum_{\tau \subseteq \gamma, \tau \neq \alpha} \texttt{Recon}(z_\tau, \tau, \gamma) + \texttt{Recon}(z'_\alpha, \alpha, \gamma) \\ &= (\hat{\mu}_\gamma - \texttt{Recon}(z_\alpha, \alpha, \gamma)) + \texttt{Recon}(z'_\alpha, \alpha, \gamma) \\ &= \hat{\mu}_\gamma + \texttt{Recon}(z'_\alpha - z_\alpha, \alpha, \gamma). \end{aligned}$$

The final step holds due to the linearity of the Recon operator with respect to its first argument. □

This result demonstrates a key principle for efficient implementation: by storing the full set of residual estimates, an update to a single z_α only requires recomputing the marginals $\{\hat{\mu}_\gamma\}_{\gamma \supseteq \alpha}$ that depend on it. This lazy updating strategy avoids the cost of a full reconstruction at each step.

4.3 Conditional ResidualPlanner

This section addresses the problem of optimally allocating a privacy budget when measuring the residuals associated with a target marginal γ. A baseline approach, which we term the "iid" strategy, involves adding i.i.d. Gaussian noise to the marginal μ_γ and then deriving the noisy residuals from this single measurement (Mullins et al., 2024). A natural question is whether the fixed noise allocation implied by this strategy is optimal, or if a more targeted allocation could achieve lower error for the same privacy cost.

The potential for improvement is especially apparent in iterative or adaptive settings. In these scenarios, some residuals $\tau \subseteq \gamma$ may have already been estimated in previous rounds. Intuitively, it is advantageous to allocate less of the current privacy budget to residuals that are already well-estimated, and more budget to those that are poorly estimated or unmeasured. To formalize this, we adapt machinery from ResidualPlanner (Xiao et al., 2023), which established the privacy cost and expected error for residual measurements.

Proposition 4.3. (Xiao et al., 2023) *Suppose noisy residuals $z_\tau = \zeta_\tau + \mathcal{N}(0, \sigma^2_\tau V_\tau)$ are released for each $\tau \subseteq \gamma$. This satisfies ρ-zCDP for $\rho = \sum_{\tau \subseteq \gamma} \frac{p_\tau}{2\sigma^2_\tau}$, where $p_\tau = \prod_{i \in \tau} \frac{n_i - 1}{n_i}$.*

Proposition 4.4. (Xiao et al., 2023) *Suppose noisy residuals $z_\tau = \zeta_\tau + \mathcal{N}(0, \sigma^2_\tau V_\tau)$ are measured for each $\tau \subseteq \gamma$. The expected squared error $\mathbb{E}[\|\hat{\mu}_\gamma - \mu_\gamma\|_2^2]$ of the reconstruction $\hat{\mu}_\gamma = \sum_{\tau \subseteq \gamma} \texttt{Recon}(z_\tau, \tau, \gamma)$ is $\sum_{\tau \subseteq \gamma} v_\tau \sigma^2_\tau$, where $v_\tau = \left(\prod_{i \in \tau} \frac{n_i - 1}{n_i}\right)\left(\prod_{j \in \gamma \setminus \tau} \frac{1}{n_j^2}\right)$.*

Our goal is to determine the optimal noise variances $\{\sigma^2_\tau\}_{\tau \subseteq \gamma}$ for a new round of measurements. We aim to minimize the final reconstruction error by combining our new measurements with prior information, subject to a privacy budget ρ for the current round only. Let $\tilde{\sigma}^2_\tau$ be the variance of the existing estimate for residual τ from previous rounds (if no prior estimate exists, $\tilde{\sigma}^2_\tau = \infty$). A new measurement with variance σ^2_τ yields a combined estimate with an updated variance $\bar{\sigma}^2_\tau$, calculated by summing the precisions: $\bar{\sigma}^{-2}_\tau = \tilde{\sigma}^{-2}_\tau + \sigma^{-2}_\tau$. This leads to the Conditional ResidualPlanner (CRP) convex optimization problem:

$$\begin{aligned} \underset{\{\sigma^2_\tau > 0\}_{\tau \subseteq \gamma}}{\text{argmin}} \quad & \sum_{\tau \subseteq \gamma} v_\tau \bar{\sigma}^2_\tau = \sum_{\tau \subseteq \gamma} \frac{v_\tau}{\frac{1}{\sigma^2_\tau} + \frac{1}{\tilde{\sigma}^2_\tau}} \\ \text{s.t.} \quad & \sum_{\tau \subseteq \gamma} \frac{p_\tau}{\sigma^2_\tau} \leq 2\rho. \end{aligned} \tag{2}$$

The problem is convex when re-parameterized in terms of precisions ($x_\tau = 1/\sigma^2_\tau$) and can be solved efficiently as a second-order cone program (Boyd, 2004).

Solving this optimization problem enables significant computational speedups. If the optimal variance σ^2_τ for a residual is very large, we can omit the measurement entirely, avoiding all the corresponding updates. This occurs frequently for low-order residuals that are already well-estimated from prior rounds like the total query. Furthermore, the optimization problem is solved with an efficient routine that first computes a closed-form analytical solution to an unconstrained version of the problem. In the common case that this solution is valid, a numerical solver is avoided completely. If not, the analytical result provides an excellent warm-start for an iterative solver (CVXPY with CLARABEL (Diamond et al., 2014; Goulart and Chen, 2024)), accelerating convergence. The full details of the cutoff for omitting measurements and the optimization strategy are in Appendix D.2.

5 AIM+GReM

To scale to high-dimensional datasets, we propose AIM+GReM: a mechanism that combines components from AIM+PGM[3] that adapt the selection and mea-

[3]For clarity, we refer to the original AIM mechanism as AIM+PGM to highlight the reconstruction method.

surement steps to the workload, privacy budget, and data domain with GReM-MLE (Mullins et al., 2024) for efficient reconstruction. AIM+GReM is shown in Alg. 4. In Appendix F.1, we detail components of AIM+PGM otherwise not described in this section.

The following components are shared between AIM+GReM and AIM+PGM. Line 3 initializes the mechanism by measuring all 1-way marginals in $\mathcal{W}^{\downarrow}$ using the Gaussian mechanism with noise scale σ_0^2, which spends $d\rho_{init}$ of the privacy budget (see Appendix F). Line 8 privately selects the next marginal to measure using the exponential mechanism with a score function that estimates the potential improvement in L_1 error for the final workload answers (McKenna et al., 2022). Line 16 updates the privacy parameters for the next round using budget annealing, a heuristic that dynamically adjusts the budget based on whether the previous measurement yielded a significant improvement in accuracy (see Appendix F.1). In the remainder of this section, we describe the components on which AIM+GReM and AIM+PGM differ and consider how these differences affect utility, scalability, and privacy.

5.1 Differences with AIM+PGM

Despite sharing a similar structure, AIM+GReM and AIM+PGM differ on aspects of the select, measure, and reconstruct steps.

Tractable selection. Since Private-PGM reconstruction can become intractable if the overall set of measured marginals causes the graphical model to become too complex, AIM+PGM filters out candidate marginals from $\mathcal{W}^{\downarrow}$ that increase the model size beyond a certain threshold. In contrast, AIM+GReM does not filter the candidate set $\mathcal{W}^{\downarrow}$, since the complexity of reconstruction with GReM-MLE does not depend on which specific marginals are measured but, rather, on the size of the marginals in $\mathcal{W}$ to be reconstructed.

Measurement noise optimization. When measuring a marginal, AIM+PGM uses the Gaussian Mechanism with isotropic noise. GReM was designed to take either marginals measured with isotropic noise or residuals measured with noise $\mathcal{N}(0, \sigma_\tau^2 V_\tau)$. This allows us to optimize the measurement noise scaled on the residual level in AIM+GreM by solving a CRP problem in each iteration (Line 9).

Reconstruction method. In lines 5 and 12-14 of Alg. 4, AIM+GReM uses GReM-MLE to reconstruct answers to marginals in $\mathcal{W}^{\downarrow}$ from prior measurements, while AIM+PGM uses Private-PGM.

5.2 Utility, Scalability, and Privacy

Utility and scalability. For an adaptive query answering mechanism, Private-PGM reconstruction presents a utility-scalability tradeoff. For a fixed set of measurements, when tractable, Private-PGM has lower error than GReM-MLE; however, GReM-MLE is often tractable when Private-PGM is not (Mullins et al., 2024). Since AIM+PGM filters candidates for tractability, it only uses a subset of the marginals in $\mathcal{W}^{\downarrow}$, limiting its capacity to model the data and the utility of its answers. Because AIM+GReM does not filter candidates, it can better model the data and, in some cases, produce lower-error answers than AIM+PGM.

Privacy. The privacy analysis for AIM+GReM follows as a corollary from the privacy analysis of AIM+PGM: Theorem 1 of McKenna et al. (2022). We provide a proof of Theorem 5.1 in Appendix F.2.

Theorem 5.1. *For $\rho > 0$,* AIM+GReM *satisfies ρ-zCDP.*

6 EXPERIMENTS

In this section, we empirically evaluate the performance of our proposed methods. The code, datasets, and instructions needed to reproduce these results are available at `https://github.com/Miguel-Fuentes/tens_daderp`.

Computing marginals and residuals. We compare the runtime of our in-axis operations against a baseline using traditional vector-based transformations via Kronecker matrix-vector multiplication (Plateau, 1985; McKenna et al., 2023; Xiao et al., 2023). We test two settings: (1) fixing attribute size $n_i = 16$ and varying marginal degree $|\gamma|$ from 2 to 7; and (2) fixing $|\gamma| = 3$ and varying n_i from 2 to 256. For each setting, we benchmark decomposition ($\zeta_\gamma = \texttt{Decomp}\,(\mu_\gamma, \gamma, \gamma)$) and reconstruction ($\texttt{Recon}\,(\zeta_\gamma, \gamma, \gamma)$).

As shown in Figure 3, our in-axis decomposition is faster across all settings, achieving average speedups of 3.14x for fixed attribute size and 3.75x for a fixed marginal degree. For reconstruction, our method is slightly slower than the baseline for low-degree marginals and small attribute sizes (max difference 5×10^{-5}s) but substantially faster for high-degree marginals and large attribute sizes (up to 9.75s faster).

Algorithmic improvements. We conduct an ablation study to isolate the impact of our algorithmic improvements, comparing our proposed `lazy+CRP` against two baselines: `lazy+IID` (lazy updates with isotropic noise) and `full+IID` (full reconstruction with isotropic noise). To ensure a fair comparison, we use a fixed, predetermined sequence of 3-way marginal measurements

Algorithm 4 AIM+GReM

Input: Workload $\mathcal{W}$, Budget ρ, Data Marginals μ_γ for $\gamma \in \mathcal{W}^\downarrow$
Output: Noisy marginals $\hat{\mu}_\gamma$ for $\gamma \in \mathcal{W}$

Initialize $z_\tau = \mathbf{0}$, $\lambda_\tau = 0$ for $\tau \in \mathcal{W}^\downarrow$ ▷ Initialize residual estimates and precisions
Set $\sigma_0^2 = \frac{|\mathcal{W}^\downarrow|}{0.9\rho}$, $\rho_{\text{init}} = \frac{1}{2\sigma_0^2}$
$z_\emptyset, \lambda_\emptyset, \ldots, z_{\{d\}}, \lambda_{\{d\}} \leftarrow$ measure 1-way marginals with budget ρ_{init} each and convert to residuals
Set $\rho_{\text{used}} = d\rho_{\text{init}}$; $t = d$; $\sigma_{t+1}^2 = \sigma_0^2$; $\epsilon_{t+1} = \sqrt{0.4\rho/|\mathcal{W}^\downarrow|}$
Set $\hat{\mu}_\gamma = \sum_{\tau \subseteq \gamma} \texttt{Recon}\,(z_\tau, \tau, \gamma)$ for each $\gamma \in \mathcal{W}$ ▷ Initialize marginal estimates
while $\rho_{\text{used}} < \rho$ **do**
    $t = t + 1$
    Select $\gamma_t \in \mathcal{W}^\downarrow$ using exponential mechanism with budget ϵ_t and weighted expected improvement score (McKenna et al., 2022) ▷ Select
    Set $\sigma_{\tau,t}^2$ for $\tau \subseteq \gamma_t$ by solving CRP with budget $\frac{1}{2\sigma_t^2}$ ▷ Optimize
    for $\tau \subseteq \gamma_t$ **do**
        $z_{\tau,t} = \texttt{MeasureResidual}(\mu_\tau, \sigma_{\tau,t}^2)$ ▷ Measure
        $z'_\tau = \frac{\lambda_\tau z_\tau + \sigma_{\tau,t}^{-2} z_{\tau,t}}{\lambda_\tau + \sigma_{\tau,t}^{-2}}$
        $\hat{\mu}_\gamma = \hat{\mu}_\gamma + \texttt{Recon}\,(z'_\tau - z_\tau, \tau, \gamma)$ for each $\gamma \in \mathcal{W}$ such that $\gamma \supseteq \tau$ ▷ Lazy reconstruct
        $z_\tau = z'_\tau$, $\lambda_\tau = \lambda_\tau + \sigma_{\tau,t}^{-2}$
    Update $\rho_{\text{used}} = \rho_{\text{used}} + \frac{1}{2\sigma_t^2} + \frac{\epsilon_t^2}{8}$
    Set σ_{t+1}^2, ϵ_{t+1} using budget annealing

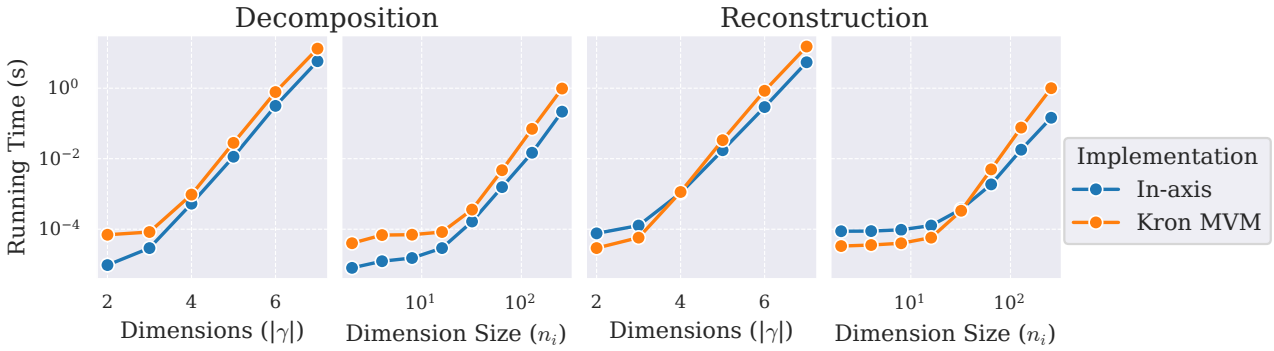

Figure 3: Running time in seconds of the `Decomp` and `Recon` operations with our in-axis implementation versus the baseline Kronecker-based approach.

and a 50/50 privacy budget split (1-way, then 3-way marginals). We evaluate runtime and final average L_1 error on the Adult (Kohavi et al., 1996), ACS (Ruggles et al., 2025), and Loans (Hay et al., 2016) datasets (details in Appendix G.1).

Table 2 shows two distinct sources of performance gain. First, lazy updating is significantly faster than full reconstruction (7.2x–23.2x speedup). Second, when using lazy updates, our CRP strategy is substantially faster than IID measurement (2.0x–12.9x speedup). This is because CRP strategically allocates a negligible privacy budget to many low-order residuals, which allows us to omit those measurements and their costly updates entirely. Taken together, `lazy+CRP` achieves overall speedups of 18.5x (Adult), 45.2x (ACS), and 92.5x (Loans) over the `full+IID` baseline. On the utility front, CRP measurement also provides modest but consistent error improvements (1.03x–1.13x) over IID noise.

Scalable query answering. We compare the utility-scalability tradeoff of AIM+GReM against AIM+PGM (McKenna et al., 2022) and ResidualPlanner (Xiao et al., 2023) on answering all 3-way marginals. We use a 50MB model for AIM+PGM, which provided the best utility-runtime trade-off in preliminary experiments (Appendix G.2). To test scalability, we use the original datasets without the binning performed in the original AIM paper. We evaluate the mean L_1 error vs. total runtime across a range of privacy budgets ($\delta = 10^{-9}$). We use a larger budget range for the Loans dataset to

Table 2: Running time (s) and workload error of different strategies. Results are averaged over five trials with $\epsilon = 1$ and $\delta = 10^{-9}$.

	Running time (s)			Workload error	
Dataset	`lazy+CRP`	`lazy+IID`	`full+IID`	`CRP`	`IID`
Adult	8.10	16.58	149.64	9.532	10.166
ACS	311.60	607.97	14094.69	0.053	0.060
Loans	37.87	489.46	3506.93	622.507	641.795

demonstrate that ResidualPlanner and AIM+GReM do eventually approach or improve on the error of AIM+PGM as ϵ goes to infinity.

Figure 4 visualizes the three-way tradeoff between privacy, utility, and runtime. To make this complex relationship easier to interpret, we present the time and error of the AIM variants relative to the ResidualPlanner baseline, with the privacy budget shown as a color gradient. For completeness, traditional 2D plots with absolute units are provided in Appendix Section G.2. The figure shows that AIM+GReM improves the scalability-utility tradeoff. It is orders of magnitude faster than AIM+PGM (up to 697x on Adult) while achieving competitive error. This speedup is especially impactful for ACS where the running time at high epsilon (epsilon = 10) is reduced to about 5 hours with AIM+GReM from over a week with AIM+PGM. Compared to ResidualPlanner, AIM+GReM often has lower error (especially at low ϵ) and is at worst only slightly higher (1.15x–1.37x). In contrast, AIM+PGM's error is significantly higher at high ϵ on the larger-domain datasets, up to 14.0x that of ResidualPlanner.

7 RELATED WORK

Answering linear queries under differential privacy (DP) is a foundational problem, with early work like Barak et al. (2007) appearing soon after DP's introduction. Much of this work used strategies based on query decomposition, including hierarchical methods (Hay et al., 2010; Qardaji et al., 2013), wavelet-based techniques (Xiao et al., 2011), and the low-rank mechanism (Yuan et al., 2012). The matrix mechanism (Li et al., 2015) formalized this decomposition approach, a framing adopted by subsequent work like (Xiao et al., 2023; Edmonds et al., 2020; Zhang et al., 2016). Another line of research focuses on data-dependent mechanisms, an area with significant development since early works like the Multiplicative Weights Exponential Mechanism (MWEM) (Hardt and Rothblum, 2010). We focus on the Adaptive Iterative Mechanism (AIM) as it performed best in recent benchmarks (Chen et al., 2025). Other notable contributions in this area include (Liu et al., 2021a; Aydore et al., 2021; Zhang et al., 2021; Cai et al., 2021).

We also considered other reconstruction methods for comparison with GReM-MLE in this work. These included Approx-PGM (McKenna et al., 2021), which attempts to speed up Private-PGM by relaxing consistency requirements. Another was GReM-LNN (Mullins et al., 2024), which tries to improve GReM's accuracy by imposing a local non-negativity constraint. However, preliminary experiments showed that GReM-LNN substantially increased GReM's runtime while not matching Private-PGM's accuracy. Approx-PGM did not meaningfully improve PGM's runtime in our setting and also reduced accuracy. Therefore, we did not include these methods in our experiments. In our experiments, we also considered GEM (Liu et al., 2021a), but were unable to scale the implementation to our workloads, encountering GPU memory limitations.

8 DISCUSSION

This research enhances differentially private adaptive query answering by improving the use of residual queries. We introduce an in-axis formulation for faster and simpler marginal/residual computations, a "lazy updating" strategy for GReM-based reconstruction that accelerates iterative updates by minimizing redundant calculations, and the conditional ResidualPlanner (CRP) for optimized, measurement-aware privacy budget allocation, reducing error and improving speed. These advancements are integrated into AIM+GReM, a novel mechanism significantly improving scalability and efficiency while maintaining competitive accuracy. AIM+GReM is orders of magnitude faster than AIM+PGM, offering a vital alternative for large data domains where AIM+PGM struggles. While AIM+PGM may yield lower error in low-budget scenarios, AIM+GReM provides a favorable utility-scalability trade-off, generally outperforming ResidualPlanner at low privacy budgets and avoiding AIM+PGM's high error on larger datasets at high budgets.

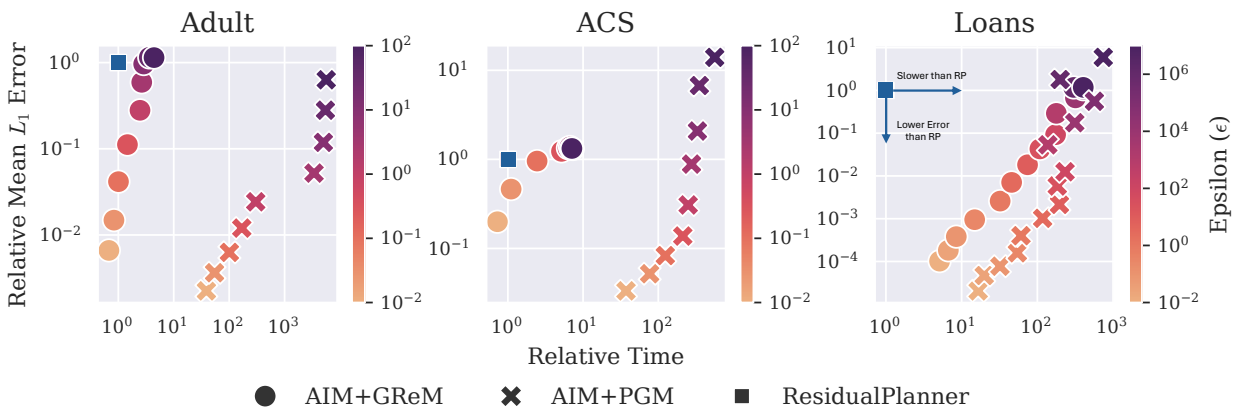

Figure 4: Mean L_1 error of AIM variants by running time on all 3-way marginals, relative to ResidualPlanner. Results are averaged across five trials. Privacy budgets ϵ are indicated in the vertical scale, with $\delta = 10^{-9}$. AIM+PGM uses a 50MB model. For 2D plots with absolute units see Appendix Section G.2

ACKNOWLEDGMENTS

This work utilized resources from Unity, a collaborative, multi-institutional high-performance computing cluster managed by UMass Amherst Research Computing and Data. This material is based upon work supported by the National Science Foundation under Grant Nos. 2046235, 2210979 and 2317232.

References

Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. 2021. Differentially Private Query Release Through Adaptive Projection. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 457–467. arXiv:2103.06641 doi:10.48550/arxiv.2103.06641

Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. 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 (SIGMOD/PODS07)*. ACM, 273–282. doi:10.1145/1265530.1265569

Stephen P. Boyd. 2004. *Convex optimization*. Cambridge University Press, Cambridge. Title from publisher's bibliographic system (viewed on 25 May 2016).

Mark Bun and Thomas Steinke. 2016. *Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds*. Springer Berlin Heidelberg, 635–658. doi:10.1007/978-3-662-53641-4_24

Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. 2021. Data synthesis via differentially private markov random fields. *Proceedings of the VLDB Endowment* 14, 11 (July 2021), 2190–2202. doi:10.14778/3476249.3476272

Clément L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 15676–15688. arXiv:2004.00010 doi:10.29012/jpc.784

Mark Cesar and Ryan Rogers. 2021. Bounding, Concentrating, and Truncating: Unifying Privacy Loss Composition for Data Analytics. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory (Proceedings of Machine Learning Research, Vol. 132)*, Vitaly Feldman, Katrina Ligett, and Sivan Sabato (Eds.). PMLR, 421–457. arXiv:2004.07223 doi:10.48550/arxiv.2004.07223

Kai Chen, Xiaochen Li, Chen Gong, Ryan McKenna, and Tianhao Wang. 2025. Benchmarking Differentially Private Tabular Data Synthesis. arXiv:2504.14061 [cs.CR] doi:10.48550/arxiv.2504.14061

Steven Diamond, Eric Chu, and Stephen Boyd. 2014. CVXPY: A Python-Embedded Modeling Language for Convex Optimization, version 0.2. http://cvxpy.org/.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. *Calibrating Noise to Sensitivity*

in Private Data Analysis. Springer Berlin Heidelberg, 265–284. doi:10.1007/11681878_14

Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. 2020. 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 '20)*. ACM, 425–438. doi:10.1145/3357713.3384297

Miguel Fuentes, Brett C. Mullins, Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2024. Joint Selection: Adaptively Incorporating Public Information for Private Synthetic Data. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 238)*, Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (Eds.). PMLR, 2404–2412. arXiv:2403.07797 doi:10.48550/arxiv.2403.07797

Paul J. Goulart and Yuwen Chen. 2024. Clarabel: An interior-point solver for conic programs with quadratic objectives. arXiv:2405.12762 [math.OC] doi:10.48550/arxiv.2405.12762

Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. arXiv:1012.4763 doi:10.48550/arxiv.1012.4763

Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 61–70. doi:10.1109/focs.2010.85

Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. 2016. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*. 139–154.

Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* 3, 1–2 (Sept. 2010), 1021–1032. doi:10.14778/1920841.1920970

Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *Kdd*, Vol. 96. 202–207.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (Aug. 2009), 455–500. doi:10.1137/07070111x

Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal* 24, 6 (Aug. 2015), 757–781. doi:10.1007/s00778-015-0398-x

Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Steven Wu. 2021b. Leveraging Public Data for Practical Private Query Release. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6968–6977. arXiv:2102.08598 doi:10.48550/arxiv.2102.08598

Terrance Liu, Giuseppe Vietri, and Steven Z. Wu. 2021a. Iterative Methods for Private Synthetic Data: Unifying Framework and New Methods. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 690–702. arXiv:2106.07153 doi:10.48550/arxiv.2106.07153

Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2023. Optimizing Error of High-Dimensional Statistical Queries Under Differential Privacy. *Journal of Privacy and Confidentiality* 13, 1 (Aug. 2023). doi:10.29012/jpc.791

Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. AIM: an adaptive and iterative mechanism for differentially private synthetic data. *Proceedings of the VLDB Endowment* 15, 11 (July 2022), 2599–2612. doi:10.14778/3551793.3551817

Ryan McKenna, Siddhant Pradhan, Daniel R Sheldon, and Gerome Miklau. 2021. Relaxed Marginal Consistency for Differentially Private Query Answering. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 20696–20707. arXiv:2109.06153 doi:10.48550/arxiv.2109.06153

Ryan Mckenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4435–4444. arXiv:1901.09136 doi:10.48550/arxiv.1901.09136

Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103. doi:10.1109/focs.2007.66

Brett Mullins, Miguel Fuentes, Yingtai Xiao, Daniel Kifer, Cameron Musco, and Daniel Sheldon. 2024. Efficient and Private Marginal Reconstruction with

Local Non-Negativity. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 141356–141389. arXiv:2410.01091 doi:10.48550/arxiv.2410.01091

Brigitte Plateau. 1985. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems - SIGMETRICS '85 (SIGMETRICS '85)*. ACM Press, 147–154. doi:10.1145/317795.317819

Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (Sept. 2013), 1954–1965. doi:10.14778/2556549.2556576

Steven Ruggles, Sarah Flood, Matthew Sobek, Daniel Backman, Grace Cooper, Julia A. Rivera Drew, Stephanie Richards, Renae Rodgers, Jonathan Schroeder, and Kari Williams. 2025. IPUMS USA: Version 16.0. doi:10.18128/D010.V16.0

Justin Whitehouse, Aaditya Ramdas, Ryan Rogers, and Steven Wu. 2023. Fully-adaptive composition in differential privacy. In *International conference on machine learning*. PMLR, 36990–37007.

Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential Privacy via Wavelet Transforms. *IEEE Transactions on Knowledge and Data Engineering* 23, 8 (Aug. 2011), 1200–1214. doi:10.1109/tkde.2010.247

Yingtai Xiao, Guanlin He, Danfeng Zhang, and Daniel Kifer. 2023. An Optimal and Scalable Matrix Mechanism for Noisy Marginals under Convex Loss Functions. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 20495–20539. arXiv:2305.08175 doi:10.48550/arxiv.2305.08175

Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. 2012. Low-rank mechanism: optimizing batch queries under differential privacy. *Proceedings of the VLDB Endowment* 5, 11 (July 2012), 1352–1363. doi:10.14778/2350229.2350252

Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. PrivTree: A Differentially Private Algorithm for Hierarchical Decompositions. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD/PODS'16)*. ACM, 155–170. doi:10.1145/2882903.2882928

Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. PrivSyn: Differentially Private Data Synthesis. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 929–946. https://www.usenix.org/conference/usenixsecurity21/presentation/zhang-zhikun

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary

A Marginals and Residuals Example

Let us continue the example from Section 2. For a demographics dataset, if age group (Age) takes on values in {18-24, 25-34, 35-50, 50+} and education level (Educ) takes on values {no college, some college, obtained degree}, then the marginal $\mu_{\{\text{Age,Educ}\}}$ is a 4×3 array. Suppose we observe $\mu_{\{\text{Age,Educ}\}}$:

$$\mu_{\{\text{Age,Educ}\}} = \begin{bmatrix} 7 & 5 & 2 \\ 3 & 5 & 11 \\ 10 & 2 & 11 \\ 9 & 18 & 17 \end{bmatrix}$$

Let us compute the residuals ζ_τ for $\tau \subseteq \{\text{Age}, \text{Educ}\}$ using the Decomp function.

$$\zeta_\emptyset = \text{Decomp}(\mu_{\{\text{Age,Educ}\}}, \{\text{Age}, \text{Educ}\}, \emptyset) = \begin{bmatrix} 100 \end{bmatrix}$$

$$\zeta_{\{\text{Age}\}} = \text{Decomp}(\mu_{\{\text{Age,Educ}\}}, \{\text{Age}, \text{Educ}\}, \{\text{Age}\}) = \begin{bmatrix} 5 \\ 9 \\ 30 \end{bmatrix}$$

$$\zeta_{\{\text{Educ}\}} = \text{Decomp}(\mu_{\{\text{Age,Educ}\}}, \{\text{Age}, \text{Educ}\}, \{\text{Educ}\}) = \begin{bmatrix} 1 & 12 \end{bmatrix}$$

$$\zeta_{\{\text{Age,Educ}\}} = \text{Decomp}(\mu_{\{\text{Age,Educ}\}}, \{\text{Age}, \text{Educ}\}, \{\text{Age}, \text{Educ}\}) = \begin{bmatrix} 4 & 13 \\ -6 & 6 \\ 11 & 13 \end{bmatrix}$$

To reconstruct the marginal $\mu_{\{\text{Age,Educ}\}}$, we can map each of the residuals ζ_τ above to a marginal component $\mu_{\{\text{Age,Educ}\},\tau}$ for $\tau \subseteq \{\text{Age}, \text{Educ}\}$ using the Recon function. Recall that a marginal is the sum of its components: $\mu_{\{\text{Age,Educ}\}} = \sum_{\tau \subseteq \{\text{Age,Educ}\}} \mu_{\{\text{Age,Educ}\},\tau}$.

$$\begin{aligned} \mu_{\{\text{Age,Educ}\},\emptyset} &= \text{Recon}(\zeta_\emptyset, \emptyset, \{\text{Age}, \text{Educ}\}) \\ &= \frac{1}{12} \begin{bmatrix} 100 & 100 & 100 \\ 100 & 100 & 100 \\ 100 & 100 & 100 \\ 100 & 100 & 100 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \mu_{\{\text{Age,Educ}\},\{\text{Age}\}} &= \text{Recon}(\zeta_{\{\text{Age}\}}, \{\text{Age}\}, \{\text{Age}, \text{Educ}\}) \\ &= \frac{1}{12} \begin{bmatrix} -44 & -44 & -44 \\ -24 & -24 & -24 \\ -8 & -8 & -8 \\ 76 & 76 & 76 \end{bmatrix} \end{aligned}$$

$$\mu_{\{\text{Age,Educ}\},\{\text{Educ}\}} = \text{Recon}(\zeta_{\{\text{Educ}\}}, \{\text{Educ}\}, \{\text{Age, Educ}\})$$
$$= \frac{1}{12}\begin{bmatrix} -13 & -10 & 23 \\ -13 & -10 & 23 \\ -13 & -10 & 23 \\ -13 & -10 & 23 \end{bmatrix}$$

$$\mu_{\{\text{Age,Educ}\},\{\text{Age,Educ}\}} = \text{Recon}(\zeta_{\{\text{Age,Educ}\}}, \{\text{Age, Educ}\}, \{\text{Age, Educ}\})$$
$$= \frac{1}{12}\begin{bmatrix} 41 & 14 & -55 \\ -27 & -6 & 32 \\ 41 & -58 & 17 \\ -55 & 50 & 5 \end{bmatrix}$$

B Differential Privacy

Differential privacy is a mathematical criterion of privacy that bounds the effect of any individual in the dataset on the output of a mechanism by adding noise to the computation. We say that two datasets $\mathcal{D}, \mathcal{D}'$ are neighboring if they differ by at most one record. This is referred to as the add/remove or unbounded neighborhood relation.

Definition B.1. (Differential Privacy; Dwork et al. (2006); Bun and Steinke (2016)) Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a randomized mechanism. For any neighboring datasets $\mathcal{D}, \mathcal{D}'$ that differ by at most one record, denoted $\mathcal{D} \sim \mathcal{D}'$, and all measurable subsets $S \subseteq \mathcal{Y}$:

- if $\Pr(\mathcal{M}(\mathcal{D}) \in S) \leq \exp(\epsilon) \cdot \Pr(\mathcal{M}(\mathcal{D}') \in S) + \delta$, then $\mathcal{M}$ satisfies (ϵ, δ)-approximate differential privacy, denoted (ϵ, δ)-DP;
- if $D_\gamma(\mathcal{M}(\mathcal{D})||\mathcal{M}(\mathcal{D}')) \leq \rho\gamma$ for all $\gamma \in (1, \infty)$ where D_γ is the γ-Renyi divergence between distributions $\mathcal{M}(\mathcal{D}), \mathcal{M}(\mathcal{D}')$, then $\mathcal{M}$ satisfies ρ-zCDP.

While (ϵ, δ)-DP is a more common notion, it is often more convenient to work with zCDP. There exists a conversion from zCDP to (ϵ, δ)-DP.

Proposition B.2 (zCDP to DP Conversion; Canonne et al. (2020))**.** *If mechanism $\mathcal{M}$ satisfies ρ-zCDP, then $\mathcal{M}$ satisfies (ϵ, δ)-DP for any $\epsilon > 0$ and $\delta = \min_{\alpha>1} \frac{\exp((\alpha-1)(\alpha\rho-\epsilon))}{\alpha-1}\left(1 - \frac{1}{\alpha}\right)^\alpha$.*

Next, we introduce two building block mechanisms. An important quantity in analyzing the privacy of a mechanism is sensitivity. The L_k sensitivity of a function $f : \mathcal{X} \to \mathbb{R}$ is given by $\Delta_k(f) = \max_{\mathcal{D}\sim\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_k$. If f is clear from the context, we write Δ_k. For a marginal μ_γ with any $\gamma \subseteq [d]$, $\Delta_k(\mu_\gamma) = 1$.

Proposition B.3 (zCDP of exponential mechanism; McSherry and Talwar (2007); Cesar and Rogers (2021))**.** *Let $\epsilon > 0$ and $Score_r : \mathcal{X} \to \mathbb{R}$ be a quality score of candidate $r \in \mathcal{R}$ for dataset $\mathcal{D}$. Then the exponential mechanism outputs a candidate $r \in \mathcal{R}$ according to the following distribution: $\Pr(\mathcal{M}(\mathcal{D}) = r) \propto \exp\left(\frac{\epsilon}{2\Delta_1(f)} Score_r(\mathcal{D})\right)$. The exponential mechanism satisfies $\frac{\epsilon^2}{8}$-zCDP.*

Proposition B.4 (zCDP of Gaussian mechanism; Xiao et al. (2023))**.** *Let f be a linear function of dataset $\mathcal{D}$, represented by matrix M. Given dataset $\mathcal{D}$, the Gaussian mechanism adds correlated Gaussian noise to $f(\mathcal{D})$ with covariance Σ, i.e., $\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + \mathcal{N}(0, \Sigma)$. Then the Gaussian Mechanism satisfies $\frac{\eta}{2}$-zCDP, where the privacy cost η is the largest diagonal element of the matrix $M^\top \Sigma^{-1} M$.*

Adaptive composition and postprocessing are two important properties of differential privacy that allow us to construct complex mechanisms from the above building blocks.

Proposition B.5 (zCDP Properties; Bun and Steinke (2016); Whitehouse et al. (2023))**.** *zCDP satisfies two properties of differential privacy:*

1. *(Fully Adaptive Composition) Let $\mathcal{M}_1 : \mathcal{X} \to \mathcal{Y}_1$ satisfy ρ_1-zCDP and $\mathcal{M}_2 : \mathcal{X} \times \mathcal{Y}_1 \to \mathcal{Y}_2$ satisfy ρ_2-zCDP. Then $\mathcal{M} = \mathcal{M}_2(\mathcal{D}, \mathcal{M}_1(\mathcal{D}))$ satisfies $(\rho_1 + \rho_2)$-zCDP.*

2. *(Postprocessing) Let $\mathcal{M}_1 : \mathcal{X} \to \mathcal{Y}$ satisfy ρ-zCDP and $f : \mathcal{Y} \to \mathcal{Z}$ be a randomized algorithm. Then $\mathcal{M} : \mathcal{X} \to \mathcal{Z} = f \circ \mathcal{M}_1$ satisfies ρ-zCDP.*

C Residual Properties

In this section we provide background on residuals and marginals and demonstrate that our multi-dimensional array perspective in Section 3 is equivalent to the original definitions. Residuals and marginals are usually defined using Kronecker product matrices. We will first review a well known correspondence between Kronecker product matrix-vector multiplications and n-mode products of tensors. We will then see that the n-mode products that arise with marginals and residuals correspond to the simple and interpretable multi-dimensional array operations from Section 3.

C.1 Background on Kronecker product matrices and tensor n-mode products

We first establish the necessary background on Kronecker product matrices and tensor n-mode products.

Kronecker products. If a matrix A can be written as

$$A = A_1 \otimes A_2 \otimes \cdots \otimes A_d = \bigotimes_{i=1}^{d} A_i$$

we will call it a Kronecker-product matrix with factors $A_1, \ldots, A_d$. In this paper we will only consider Kronecker-product matrices with d factors where d is the number of attributes of the dataset.

We will use the following fact: if $A = \bigotimes_{i=1}^{d} A_i$ and $B = \bigotimes_{i=1}^{d} B_i$ and the sizes of each A_i and B_i are compatible, then $AB = \bigotimes_{i=1}^{d} (A_i B_i)$.

Tensors and n-mode products. We now introduce tensors and n-mode products. A comprehensive background can be found in Kolda and Bader (2009). Let $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor or multi-dimensional array. We will use these two terms interchangeably. A mode of a tensor is one of the dimensions. In an array language like NumPy, this is called an axis.

The mode-k fibers of a tensor are the vectors along the kth mode. For each fixed tuple $i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d$ of indices for other dimensions, the values $u_{i_1 \cdots i_{k-1}\ j\ i_{k+1} \cdots i_d}$ for j ranging from 1 to n_k, taken as a column vector, form one mode-k fiber. There are $\prod_{i \neq k} n_i$ mode-k fibers in total.

An n-mode product is a linear transformation of the tensor along one of its modes, where n indicates the mode (e.g., a 1-mode product, 2-mode product, etc.). We will say "n-mode product" for the generic operation but "k-mode product" for the operation applied to a specific mode k, because we have already used n for another purpose in this paper.

The *k-mode product* $\mathcal{U} \times_k A$ with matrix $A \in \mathbb{R}^{m \times n_k}$ applies the linear transformation A to every mode-k fiber of $\mathcal{U}$. The result is a tensor $\mathcal{V} \in \mathbb{R}^{n_1 \times \cdots n_{k-1} \times m \times n_{k+1} \times \cdots n_d}$ with entries

$$v_{\ldots\, i \,\ldots} = \sum_{j=1}^{n_j} a_{ij} \cdot u_{\ldots\, j \,\ldots}$$

where each ellipsis suppresses shared indices between the LHS and RHS and the indices i and j appear in the kth position. So, $v_{\ldots\, i \,\ldots} = v_{i_1 \cdots i_{k-1}\ i\ i_{k+1} \cdots i_d}$ and $u_{\ldots\, j \,\ldots} = u_{i_1 \cdots i_{k-1}\ j\ i_{k+1} \cdots i_d}$ for fixed $i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d$.

A sequence of n-mode products is written as

$$\mathcal{U} \times_{i=1}^{d} A_i \ := \mathcal{U} \times_1 A_1 \times_2 \cdots \times_d A_d.$$

Sequences of n-mode products can be performed in any order and give the same result.

For a tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the vectorization operation $\text{vec}(\mathcal{U})$ reshapes $\mathcal{U}$ into a vector u of size $\prod_{i=1}^{d} n_i$. We will also define a tensorization operation $\text{ten}(u)$ that reshapes u back into a tensor of size $n_1 \times \cdots \times n_d$, with the sizes $n_1, \ldots, n_d$ implicit from context. This gives, for example, that $\mathcal{U} = \text{ten}(\text{vec}(\mathcal{U}))$.

Kronecker product matrix-vector multiplication as sequence of n-mode products. Let $A = A_1 \otimes \cdots \otimes A_d$ be a Kronecker-product matrix with $A_i \in \mathbb{R}^{m_i \times n_i}$ and let $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor of compatible size. The matrix-vector multiplication $A\,\mathrm{vec}(\mathcal{U})$ is equivalent to a sequence of n-mode products Kolda and Bader (2009):

$$A\,\mathrm{vec}(\mathcal{U}) = \mathrm{vec}(\mathcal{U} \times_{i=1}^{d} A_i).$$

Another way to write this is:

$$Au = \mathrm{vec}(\mathrm{ten}(u) \times_{i=1}^{d} A_i).$$

for any vector $u \in \mathbb{R}^n$ where $n = \prod_{i=1}^{d} n_i$.

Either way, we see that the multiplication of a Kronecker-product matrix and a vector can be conceptualized as applying a sequence of n-mode products to the tensor corresponding to the vector.

C.2 Marginals and Residuals

In Xiao et al. (2023) and Mullins et al. (2024), marginals and residuals are defined as Kronecker-product structured queries on the data vector $\mathbf{p} = \mathrm{vec}(\mathcal{A})$, i.e., the vectorized data array, which has length $n = \prod_{i=1}^{d} n_i$. A (vector-valued) query is a linear transformation $Q \in \mathbb{R}^{m \times n}$ that operates on $\mathbf{p}$ and gives query answers $Q\mathbf{p} \in \mathbb{R}^m$.

Marginal queries and the `sum` operation. A marginal query M_γ over attribute subset $\gamma \subseteq [d]$ is defined as:

$$M_\gamma = \bigotimes_{k=1}^{d} \begin{cases} \mathbb{I}_{n_k} & k \in \gamma \\ 1_{n_k}^\top & k \notin \gamma \end{cases},$$

where $\mathbb{I}_\ell$ is the $\ell \times \ell$ identity matrix and 1_ℓ is a column vector of ℓ ones. The marginal $\mu_\gamma = M_\gamma \mathbf{p}$ (shaped as a vector) is the vector of answers to the query M_γ. If we interpret $M_\gamma \mathbf{p}$ through the lens of n-mode products, we see it is equivalent to applying a sequence of transformations to the data array $\mathcal{A}$:

1. For each $k \in \gamma$, apply the identity transformation to dimension k,
2. For each $k \notin \gamma$, sum over dimension k.

This is exactly the definition of a marginal.

The operator $1_{n_k}^\top$, which sums over dimension k of the data array, is the first of the four elementary operations used for marginal and residual operations, and corresponds to the `_sum` operation in Table 1.

An important technical comment is that the sequence of n-mode products to compute a marginal from the data array does *not* reduce the number of dimensions of the multi-dimensional array. The sum operation $1_{n_k}^\top$ maps each mode-k fiber to a scalar, but this dimension is preserved as a dimension of size 1, or *singleton dimension*, in the resulting array. This allows us to follow the convention that every array—the data array $\mathcal{A}$ as well as each marginal μ_γ and, later, each residual array—has one dimension per attribute occuring in order from 1 to d, with singleton dimensions for marginalized attributes.[4]

Residual queries and the `sub` operation. A residual query R_τ over attribute subset $\tau \subseteq [d]$ is defined as:

$$R_\tau = \bigotimes_{k=1}^{d} \begin{cases} S_{(n_k)} & k \in \tau \\ 1_{n_k}^\top & k \notin \tau \end{cases}$$

where $S_{(\ell)} \in \mathbb{R}^{\ell-1 \times \ell}$ is a *basis matrix* defined below. The residual $\zeta_\tau = R_\tau \mathbf{p}$ (shaped as a vector) is the vector of answers to the residual query R_τ. There are different choices for which basis matrix to use. We use the *subtraction matrix* $S_{(\ell)}$ (Xiao et al., 2023), defined as

$$S_{(\ell)} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & 0 & 0 & \cdots & 1 & 0 \\ -1 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1_{\ell-1} & \mathbb{I}_{\ell-1} \end{bmatrix} \in \mathbb{R}^{\ell-1 \times \ell}.$$

[4] In Section 3 we defined μ_γ as a $|\gamma|$-dimensional array, but we will now follow the convention that it is a d-dimensional array with singleton dimensions for attributes not in γ.

A different basis was used in Mullins et al. (2024). Our definition of the subtraction matrix has the opposite sign of the one in Xiao et al. (2023).[5] With these definitions, we can see that the residual query operation $R_\tau \mathbf{p}$ is also equivalent to applying a sequence of transformations to the data array $\mathcal{A}$:

1. For each $k \in \tau$, apply the subtraction operator $S_{(n_k)}$ to dimension k,
2. For each $k \notin \tau$, sum over dimension k.

We can also see from its definition that $S_{(\ell)}$ has a simple interpretation as a linear transformation. For any $v \in \mathbb{R}^\ell$,

$$S_{(\ell)} v = v_{2:\ell} - v_1.$$

This is implemented by the `_sub(v)` operation from Table 1.

C.3 Converting between marginals and residuals: `Decomp` and `Recon`

Residual and marginal queries are closely related. Both sum over unwanted dimensions of the data array, and marginal queries preserve the remaining dimensions by applying the identity operator, while residual queries apply the subtraction operator.

Marginals to residuals with `Decomp`. From this it is clear we can compute the τ-residual from the γ-marginal for any $\tau \subseteq \gamma$ by summing over dimensions $k \in \gamma \setminus \tau$ and applying the subtraction operator to dimensions $k \in \tau$. Mathematically, for $\tau \subseteq \gamma$ we have that $R_\tau = T_{\tau,\gamma} M_\gamma$ where[6]

$$T_{\tau,\gamma} := \bigotimes_{k=1}^{d} \begin{cases} S_{(n_k)} & k \in \tau \\ 1_{n_k}^\top & k \in \gamma \setminus \tau \\ 1 & k \notin \gamma \end{cases}. \tag{3}$$

The operator $T_{\tau,\gamma}$ is therefore a linear transformation that converts a γ-marginal to a τ-residual, since $\zeta_\tau = R_\tau \mathbf{p} = T_{\tau,\gamma} M_\gamma \mathbf{p} = T_{\tau,\gamma} \mu_\gamma$.

The transformation $T_{\tau,\gamma}\mu_\gamma$ is exactly the $\texttt{Decomp}\,(\mu_\gamma, \gamma, \tau)$ operation in Algorithm 1. Using the n-mode product perspective, we see from Equation (3) that $T_{\tau,\gamma}\mu_\gamma$ is equivalent to a sequence of operations on the marginal μ_γ (shaped as a d-dimensional array):

1. For each $k \in \tau$, apply the subtraction operator $S_{(n_k)}$ to dimension k,
2. For each $k \in \gamma \setminus \tau$, sum over dimension k,
3. For each $k \notin \gamma$, preserve the singleton dimension k by applying the scalar identity operator.

This is exactly the definition of $\texttt{Decomp}\,(\mu_\gamma, \gamma, \tau)$.

Residuals to marginals with `Recon`. The `Recon` operation is the pseudoinverse of `Decomp`:

$$T_{\tau,\gamma}^+ = \bigotimes_{k=1}^{d} \begin{cases} S_{(n_k)}^+ & k \in \tau \\ \frac{1}{n_k} 1_{n_k} & k \in \gamma \setminus \tau \\ 1 & k \notin \gamma \end{cases} \tag{4}$$

We used the rule that $(A_1 \otimes \cdots \otimes A_d)^+ = A_1^+ \otimes \cdots \otimes A_d^+$ and simple calculations for the pseudoinverses of $1_{n_k}^\top \in \mathbb{R}^{1 \times n_k}$ and the scalar 1. To map Equation (4) to the `Recon` operation in Algorithm 2, it remains to

[5] This makes the corresponding `_sub` operation and its pseudoinverse `_center` slightly more interpretable.

[6] We can verify this is using Kronecker algebra:

$$T_{\tau,\gamma} M_\gamma \quad = \quad \bigotimes_{k=1}^{d} \begin{cases} S_{(n_k)} \mathbb{I}_{n_k} & k \in \tau \\ 1_{n_k}^\top \mathbb{I}_{n_k} & k \in \gamma \setminus \tau \\ 1 \cdot 1_{n_k}^\top & k \notin \gamma \end{cases} \quad = \quad \bigotimes_{k=1}^{d} \begin{cases} S_{(n_k)} & k \in \tau \\ 1_{n_k}^\top & k \notin \gamma \end{cases} \quad = \quad R_\tau$$

characterize $S^+_{(n_k)}$ and $\frac{1}{n_k}1_{n_k}$, which are the pseudoinverses of the subtraction and sum operations, as interpretable linear operators.

Center operator as pseudoinverse of subtraction operator. The pseudoinverse $S^+_{(\ell)} \in \mathbb{R}^{\ell \times \ell-1}$ of the subtraction matrix is (Xiao et al., 2023):

$$S^+_{(\ell)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} - \frac{1}{\ell} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 0^\top_{\ell-1} \\ \mathbb{I}_{\ell-1} \end{bmatrix} - \frac{1}{\ell} 1_\ell 1^\top_{\ell-1}$$

Its operation on a vector $v \in \mathbb{R}^{\ell-1}$ is:

$$S^+_{(\ell)} v = \begin{bmatrix} 0 \\ v \end{bmatrix} - 1_\ell \mu$$

where $\mu = \frac{1}{\ell}\sum_{i=1}^{\ell-1} v_i$, which is also the mean of the zero-padded vector $[0, v]^\top$. Thus, this operation can be implemented in two steps as described in the `_center` operation in Table 1: first pad v with an initial zero entry, then subtract off the mean.

Smear operator as pseudoinverse of sum operator. The pseudoinverse of the sum operator $1^\top_{n_k} \in \mathbb{R}^{1 \times n_k}$ is the "smear" operator $\frac{1}{n_k}1_{n_k} \in \mathbb{R}^{n_k \times 1}$. It operates on a scalar v as

$$v \mapsto \frac{1}{n_k} \begin{bmatrix} v \\ \vdots \\ v \end{bmatrix}$$

In words, it "smears" the value v evenly across the n_k entries of the resulting vector. This is implemented by the `_smear` operation in Table 1.

Put together: $T^+_{\tau,\gamma}$ as the `Recon` *operation.* We can now interpret the operation $T^+_{\tau,\gamma}\zeta_\tau$ as the following sequence of operations on a residual ζ_τ (shaped as a d-dimensional array):

1. For each $k \in \tau$, apply the center operator $S^+_{(n_k)}$ to dimension k,
2. For each $k \in \gamma \setminus \tau$, apply the smear operator $\frac{1}{n_k}1_{n_k}$ to dimension k,
3. For each $k \notin \gamma$, preserve the singleton dimension k by applying the scalar identity operator.

This is exactly the definition of $\texttt{Recon}\,(\zeta_\tau, \tau, \gamma)$ in Algorithm 2.

C.4 The invertible transformation T_γ

Prior work (Xiao et al., 2023; Mullins et al., 2024) established the invertible linear transformation between the marginal μ_γ and the residuals $[\zeta_\tau]_{\tau \subseteq \gamma}$ stated in Proposition 3.1. We briefly elaborate using the notation defined in this section. Define

$$T_\gamma = \left[\begin{array}{c} T_{\emptyset,\gamma} \\ \hline \vdots \\ \hline T_{\gamma,\gamma} \end{array}\right] \in \mathbb{R}^{n_\gamma \times n_\gamma}$$

to be the vertical concatenation of the decomposition operators $T_{\tau,\gamma}$ for all $\tau \subseteq \gamma$, where we recall that $n_\gamma = \prod_{k \in \gamma} n_k$. This matrix is invertible with inverse (e.g., see Theorem 2 of (Mullins et al., 2024)):

$$T^{-1}_\gamma = \left[\begin{array}{c|c|c} & & \\ T^+_{\emptyset,\gamma} & \cdots & T^+_{\gamma,\gamma} \\ & & \end{array}\right].$$

The operator T_γ applied to a marginal μ_γ gives the τ-residuals for all $\tau \subseteq \gamma$:

$$T_\gamma \mu_\gamma = \begin{bmatrix} T_{\emptyset,\gamma}\mu_\gamma \\ \vdots \\ T_{\gamma,\gamma}\mu_\gamma \end{bmatrix} = [T_{\tau,\gamma}\mu_\gamma]_{\tau\subseteq\gamma} = [\texttt{Decomp}\,(\mu_\gamma, \gamma, \tau)]_{\tau\subseteq\gamma} = [\zeta_\tau]_{\tau\subseteq\gamma}.$$

The operator T_γ^{-1} applied to the residuals $[\zeta_\tau]_{\tau\subseteq\gamma}$ gives the γ-marginal:

$$T_\gamma^{-1} \begin{bmatrix} \zeta_\emptyset \\ \vdots \\ \zeta_\gamma \end{bmatrix} = \sum_{\tau\subseteq\gamma} T^+_{\tau,\gamma}\zeta_\tau = \sum_{\tau\subseteq\gamma} \texttt{Recon}\,(\zeta_\tau, \tau, \gamma) = \mu_\gamma.$$

C.5 Residual Measurement

With the equivalence between the decomposition function $\texttt{Decomp}\,(\cdot, \gamma, \tau)$ and the residual transformation matrix $T_{\tau,\gamma}$ established, we can implement residual measurement as detailed in Algorithm 5.

Algorithm 5 $\texttt{MeasureResidual}\ (\mu_\tau, \sigma^2_\tau)$ (Xiao et al., 2023)

Input: Marginal μ_τ, variance σ^2_τ
Output: Noisy residual $z_\tau = \zeta_\tau + \mathcal{N}(0, \sigma^2_\tau V_\tau)$
1: $y_\tau = \mu_\tau + N(0, \sigma^2_\tau)$
2: Return $\texttt{Decomp}\,(y_\tau, \tau, \tau)$

D Conditional ResidualPlanner

This section provides a more detailed exposition of the Conditional ResidualPlanner (CRP) problem. We start by showing how our Proposition 4.3 (regarding privacy guarantees) and Proposition 4.4 (regarding error formulation) derive from results presented in Xiao et al. (2023). Next, we detail the specifics of our algorithm for solving the CRP problem.

D.1 CRP Parameters: Privacy and Error

To establish the connection between our framework and the results in Xiao et al. (2023), we first refer to Section C. The correspondence between in-axis operations and Kronecker products demonstrates that our residual measurement algorithm, Algorithm 5, is equivalent to the residual measuring mechanism $\mathcal{M}_A\left(\mathbf{x}; \sigma^2_A\right)$ described in Xiao et al. (2023). With this equivalence established, we can leverage the theorems from Xiao et al. (2023):

Privacy Cost Derivation (Proposition 4.3): Theorem 4.5 in Xiao et al. (2023) states that the "privacy cost" (as defined therein) of employing a mechanism equivalent to our Algorithm 5 to measure a specific residual, denoted τ, with noise variance σ^2_τ, is $\frac{p_\tau}{\sigma^2_\tau}$. Consequently, under this framework, our Algorithm 5 when applied to residual τ with noise variance σ^2_τ satisfies ρ-zero Concentrated Differential Privacy (ρ-zCDP) with:

$$\rho = \frac{p_\tau}{2\sigma^2_\tau} \tag{5}$$

This forms the basis for our Proposition 4.3.

Error Formulation Derivation (Proposition 4.4): Similarly, Theorem 4.7 in Xiao et al. (2023) addresses the variance of marginal estimates. Let $\hat{\mu}_\gamma$ be an estimate of a marginal γ, constructed as $\hat{\mu}_\gamma = \sum_{\tau\subseteq\gamma} \texttt{Recon}\,(\hat{z}_\tau, \tau, \gamma)$.

Theorem 4.7 states that the variance of each individual entry in the vector $\hat{\mu}_\gamma$ is given by $\sum_{\tau\subseteq\gamma} \sigma^2_\tau v_\tau$.

Let n_γ be the dimensionality of the marginal γ (i.e., the number of entries in the vector $\hat{\mu}_\gamma$). The noise introduced during the estimation process (such that $\hat{\mu}_\gamma = \mu_\gamma + \text{noise}$) is zero-mean, so the expected squared L2 error between the estimate $\hat{\mu}_\gamma$ and the true marginal μ_γ is:

$$E\left[\|\hat{\mu}_\gamma - \mu_\gamma\|_2^2\right] = \sum_{j=1}^{n_\gamma} \text{Var}\left((\hat{\mu}_\gamma)_j\right)$$

Given that Theorem 4.7 asserts that $\text{Var}\left((\hat{\mu}_\gamma)_j\right) = \sum_{\tau \subseteq \gamma} \sigma_\tau^2 v_\tau$ for each entry j, the expected squared L2 error becomes:

$$E\left[\|\hat{\mu}_\gamma - \mu_\gamma\|_2^2\right] = n_\gamma \sum_{\tau \subseteq \gamma} \sigma_\tau^2 v_\tau \tag{6}$$

This is the result presented in Proposition 4.4. Since n_γ is a constant positive term for a given marginal γ, it does not affect the optimal noise allocation in the CRP problem and can be ignored for optimization purposes.

D.2 Solving CRP

Recall the conditional ResidualPlanner (CRP) problem is defined as:

$$\begin{aligned} \underset{\sigma_\tau^2}{\text{argmin}} \quad & \sum_{\tau \subseteq \gamma} \frac{v_\tau}{\frac{1}{\sigma_\tau^2} + \frac{1}{\tilde{\sigma}_\tau^2}} \\ \text{s.t.} \quad & \sum_{\tau \subseteq \gamma} \frac{p_\tau}{\sigma_\tau^2} \leq C \\ & \sigma_\tau^2 > 0 \quad \forall \tau \end{aligned} \tag{7}$$

Here, σ_τ^2 is the noise scale to be optimized for measuring the residual τ in the current round, $\tilde{\sigma}_\tau^2$ is the noise scale of our current estimate of τ, $C = 2\rho$ is the privacy cost, with ρ being the zCDP value allocated to this measurement, and v_τ, p_τ are problem parameters defined in Proposition 4.3 and Proposition 4.4.

Stabilizing transformation. The original formulation Equation (7) presents two potential issues for numerical optimization. Firstly, the privacy cost value C can be very small, potentially leading to instability. Secondly, the objective function's dependence on $\frac{1}{\sigma_\tau^2}$ can create high curvature, also hindering optimization. To address these, we introduce the transformed variables $x_\tau = \frac{1}{C\sigma_\tau^2}$ and $a_\tau = \frac{1}{C\tilde{\sigma}_\tau^2}$. The original constraint $\sigma_\tau^2 > 0$ implies $x_\tau = \frac{1}{C\sigma_\tau^2} > 0$. However, in this setting, infinite variance ($\sigma_\tau^2 \to \infty$ and $x_\tau = 0$) corresponds to omitting the measurement of τ, which is a valid outcome. Thus, the constraint on the transformed variable x_τ becomes $x_\tau \geq 0$. Substituting these into the original problem leads to the transformed CRP problem:

$$\begin{aligned} \underset{x_\tau}{\text{argmin}} \quad & \frac{1}{C} \sum_{\tau \subseteq \gamma} \frac{v_\tau}{x_\tau + a_\tau} \\ \text{s.t.} \quad & \sum_{\tau \subseteq \gamma} x_\tau p_\tau \leq 1 \\ & x_\tau \geq 0 \quad \forall \tau \end{aligned} \tag{8}$$

Notice that the leading constant $1/C$ in the objective function can be dropped without altering the minimizer x_τ.

Solving relaxed problem. Solving the transformed CRP problem Equation (8) directly, which includes the non-negativity constraints $x_\tau \geq 0$ and an inequality budget constraint, can be challenging. Our strategy is to first address a simplified version. We define the relaxed CRP problem by omitting the non-negativity constraints $x_\tau \geq 0$ from Equation (8) and by treating the budget constraint as an equality:

$$\begin{aligned} \underset{x_\tau}{\text{argmin}} \quad & \sum_{\tau \subseteq \gamma} \frac{v_\tau}{x_\tau + a_\tau} \\ \text{s.t.} \quad & \sum_{\tau \subseteq \gamma} x_\tau p_\tau = 1 \end{aligned} \tag{9}$$

The budget constraint is formulated as an equality here because the objective function $\sum_{\tau \subseteq \gamma} \frac{v_\tau}{x_\tau + a_\tau}$ (where $v_\tau > 0$ and $a_\tau > 0$) is monotonically decreasing with respect to each x_τ. To minimize this sum, the values of x_τ should be made as large as the constraints permit. Therefore, an optimal solution to Equation (8) must exhaust the budget, satisfying $\sum_{\tau \subseteq \gamma} x_\tau p_\tau = 1$. This assumption simplifies the derivation of a closed-form solution for the relaxed problem.

Theorem D.1. *Consider the relaxed CRP problem defined in Equation* (9)*, which has the equality budget constraint* $\sum_{\tau \subseteq \gamma} p_\tau x_\tau = 1$ *and no non-negativity constraints on* x_τ*. The solution to this relaxed problem, denoted*

$x_\tau^{relaxed}$, is given by:

$$x_\tau^{relaxed} = \frac{1+Q}{S}\sqrt{\frac{v_\tau}{p_\tau}} - a_\tau \tag{10}$$

where $S = \sum_{\tau\subseteq\gamma}\sqrt{p_\tau v_\tau}$ *and* $Q = \sum_{\tau\subseteq\gamma} p_\tau a_\tau$.

Proof. We use the method of Lagrange multipliers to find the solution x_τ^{relaxed} for the relaxed problem Equation (9), which features the equality constraint $\sum_{\tau\subseteq\gamma} p_\tau x_\tau - 1 = 0$. The Lagrangian L is:

$$L(x,\lambda) = \sum_{\tau\subseteq\gamma}\frac{v_\tau}{x_\tau + a_\tau} + \lambda\left(\sum_{\tau\subseteq\gamma} p_\tau x_\tau - 1\right) \tag{11}$$

Here, λ is the Lagrange multiplier. By construction, $v_\tau > 0$, $p_\tau > 0$, and $a_\tau > 0$. The objective function terms require $x_\tau + a_\tau \neq 0$; given $a_\tau > 0$, we anticipate $x_\tau + a_\tau > 0$.

To find x_τ^{relaxed}, we take the partial derivative of L with respect to each x_τ and set it to zero (the first-order condition for optimality):

$$\frac{\partial L}{\partial x_\tau} = -\frac{v_\tau}{(x_\tau + a_\tau)^2} + \lambda p_\tau = 0 \tag{12}$$

Rearranging this equation yields:

$$\lambda p_\tau = \frac{v_\tau}{(x_\tau + a_\tau)^2}$$
$$(x_\tau + a_\tau)^2 = \frac{v_\tau}{\lambda p_\tau}$$

Assuming $x_\tau + a_\tau > 0$, we take the positive square root:

$$x_\tau + a_\tau = \sqrt{\frac{v_\tau}{\lambda p_\tau}}$$

This step implies $\lambda p_\tau > 0$. Since $p_\tau > 0$, we must have $\lambda > 0$. Solving for x_τ (which we denote x_τ^{relaxed} as it is the solution to the relaxed problem):

$$x_\tau^{\text{relaxed}} = \sqrt{\frac{v_\tau}{\lambda p_\tau}} - a_\tau \tag{13}$$

Next, we apply the equality budget constraint $\sum_{\tau\subseteq\gamma} p_\tau x_\tau^{\text{relaxed}} = 1$. Substituting the expression for x_τ^{relaxed} from Equation (13) into this constraint gives:

$$\sum_{\tau\subseteq\gamma} p_\tau\left(\sqrt{\frac{v_\tau}{\lambda p_\tau}} - a_\tau\right) = 1 \tag{14}$$

Simplifying the sum:

$$\sum_{\tau\subseteq\gamma}\left(\frac{1}{\sqrt{\lambda}}\sqrt{v_\tau p_\tau} - p_\tau a_\tau\right) = 1 \tag{15}$$

$$\frac{1}{\sqrt{\lambda}}\sum_{\tau\subseteq\gamma}\sqrt{v_\tau p_\tau} - \sum_{\tau\subseteq\gamma} p_\tau a_\tau = 1 \tag{16}$$

Let $S = \sum_{\tau\subseteq\gamma}\sqrt{p_\tau v_\tau}$ and $Q = \sum_{\tau\subseteq\gamma} p_\tau a_\tau$, as defined in Theorem D.1. Substituting S and Q into the equation yields:

$$\frac{S}{\sqrt{\lambda}} - Q = 1 \tag{17}$$

We now solve for $\sqrt{\lambda}$, noting that $S > 0$ and $1 + Q > 0$ by construction:

$$\frac{S}{\sqrt{\lambda}} = 1 + Q \tag{18}$$

$$\sqrt{\lambda} = \frac{S}{1+Q} \tag{19}$$

Finally, we substitute this expression for $\sqrt{\lambda}$ back into the equation for x_τ^{relaxed} from Equation (13):

$$\begin{aligned} x_\tau^{\text{relaxed}} &= \frac{1}{\sqrt{\lambda}}\sqrt{\frac{v_\tau}{p_\tau}} - a_\tau \\ &= \left(\frac{1+Q}{S}\right)\sqrt{\frac{v_\tau}{p_\tau}} - a_\tau \end{aligned} \tag{20}$$

This expression provides the solution x_τ^{relaxed} for the relaxed problem Equation (9), consistent with Equation (10). □

Iterative solver. Let x_τ^{relaxed} be the solution to the relaxed problem as given by Equation (10). We check if this solution satisfies the non-negativity constraints $x_\tau^{\text{relaxed}} \geq 0$ for all τ in Equation (8). If all $x_\tau^{\text{relaxed}} \geq 0$, then x_τ^{relaxed} is feasible and thus optimal for Equation (8). In this case, we set the final solution $x_\tau^* = x_\tau^{\text{relaxed}}$.

However, if any $x_\tau^{\text{relaxed}} < 0$, this analytical solution is infeasible for Equation (8). The true optimum of Equation (8) must then lie on the boundary where one or more $x_\tau = 0$. In this scenario, we employ an iterative numerical solver. We use the Clarabel solver (Goulart and Chen, 2024), an interior-point solver for convex optimization, which we found to be fast and reliable for this problem that is available via CVXPY (Diamond et al., 2014). We provide the (potentially infeasible) x_τ^{relaxed} values as a warm start to Clarabel to potentially speed up convergence, using a numerical tolerance of 10^{-8}. We denote the solution to the full CRP problem obtained from this process as x_τ^*.

Post-processing. In some cases, the solution x_τ^* to the full CRP problem includes measurements of residuals τ with very high noise. Although theoretically optimal, this is impractical. We therefore omit such measurements. If a measurement's share of the privacy cost, $p_\tau x_\tau^*$, is below a threshold η, we do not perform that measurement. For the remaining residuals, the optimal measurement noise is recovered as $\sigma_\tau^2 = \frac{1}{Cx_\tau^*}$. In experiments, we use $\eta = 10^{-3}$.

E In-axis Running Time

We now analyze the running time of the in-axis algorithms. Theorem 3.2 asserts that both $\texttt{Decomp}\,(\mu_\gamma, \gamma, \tau)$ and $\texttt{Recon}\,(\zeta_\tau, \tau, \gamma)$ take $\mathcal{O}(|\gamma| n_\gamma)$ time where $n_\gamma = \prod_{k\in\gamma} n_k$ is the size of μ_γ.

Each algorithm is a sequence of $|\gamma|$ n-mode products on intermediate arrays of either increasing size (`Recon`) or decreasing size (`Decomp`). The size of each intermediate array is always bounded by n_γ, the size of the marginal. We claim that each n-mode product takes $\mathcal{O}(n_\gamma)$ time, which will immediately yield the result. Write one of these n-mode products as $\mathcal{U} = \mathcal{V} \times_k A_k$ where $\mathcal{V}$ is the starting array and $\mathcal{U}$ is the resulting array: we claim that computing the n-mode product takes $\mathcal{O}(n_\mathcal{U} + n_\mathcal{V})$ time, where $n_\mathcal{U}$ and $n_\mathcal{V}$ are the sizes of $\mathcal{U}$ and $\mathcal{V}$, respectively. In words, each entry of $\mathcal{U}$ and $\mathcal{V}$ is "touched" a constant number of times. This follows because the computation can be divided into the operations $u = \texttt{op}(v)$ for each mode-k fiber v of $\mathcal{V}$, where `op` is one of the operations from Table 1, and it is straightforward to verify that each of those operations takes $\mathcal{O}(n_u + n_v)$ time, where n_u and n_v are the sizes u and v, respectively. Finally because $n_\mathcal{U}$ and $n_\mathcal{V}$ are both bounded by n_γ, the time for each n-mode product is $\mathcal{O}(n_\gamma)$.

It is also possible to establish these running times is by relation to the implementation using Kronecker matrices and the "shuffle algorithm" from Mullins et al. (2024) and Xiao et al. (2023). The shuffle algorithm multiplies a Kronecker-product matrix $A_1 \otimes \cdots \otimes A_d$ by a vector u via a sequence of operations that starts with the tensor $\text{ten}(u)$ and then repeatedly reshapes the tensor into a matrix and multiplies by one factor A_k. Each reshape-and-multiply operation implements one n-mode product and has the same time complexity as multiplying each k-mode fiber by A_k. The overall time complexity depends on the efficiency of the matrix-vector multiplications with the factors A_k. For the factors $A_k \in \mathbb{R}^{a\times b}$ that arise when converting between marginals and residuals, $A_k v$ can always be computed in $\mathcal{O}(a+b)$ time. With this fact, the shuffle implementation was shown in Mullins et al. (2024) and Xiao et al. (2023) to take $\mathcal{O}(|\gamma| n_\gamma)$ overall. Because our implementations perform the same sequences of n-mode products, they also take $\mathcal{O}(|\gamma| n_\gamma)$ time.

F AIM+GReM

In this section, we describe components of AIM+GReM and AIM+PGM not fully described in the main text. Let us first describe the initialization step of AIM+GReM, shown in Algorithm 6, which measures residuals equivalent to measuring 1-way marginals with variance σ_0^2. For each attribute i, we measure both the residual $z_{\{i\}}$ and the total query $z'_\emptyset$. Since we measure the total query multiple times, we update the total query $z_\emptyset$ and its precision $\lambda_\emptyset$ via inverse-variance weighting each round.

Algorithm 6 Initialization

Input: Marginals $\mu_0, \mu_{\{1\}}, \ldots, \mu_{\{d\}}$, variance σ_0^2
Output: Residuals $z_0, z_{\{1\}}, \ldots, z_{\{d\}}$, precisions $\lambda_0, \lambda_{\{1\}}, \ldots, \lambda_{\{d\}}$

1: **for** $i = 1, \ldots, d$ **do**
2: $\quad z_{\{i\}} = \texttt{MeasureResidual}(\mu_{\{i\}}, \sigma_0^2)$, $\lambda_{\{i\}} = \sigma_0^{-2}$ ▷ Measure 1-way residual
3: $\quad z'_\emptyset = \texttt{MeasureResidual}(\mu_\emptyset, n_i\sigma_0^2)$ ▷ Measure total query
4: $\quad z_\emptyset = \frac{\lambda_\emptyset z_\emptyset + n_i^{-1}\sigma_0^{-2} z'_\emptyset}{\lambda_0 + n_i^{-1}\sigma_0^{-2}}$, $\lambda_\emptyset = \lambda_\emptyset + n_i^{-1}\sigma_0^{-2}$ ▷ Update total query

This differs from AIM+PGM in two ways: AIM+PGM measures all 1-way marginals without decomposing them into residuals and uses a different noise scale. In line 1 of Alg. 4, AIM+GReM initializes the noise scale at $\sigma_0^2 = \frac{|\mathcal{W}^\downarrow|}{0.9\rho}$. In contrast, AIM+PGM initializes the noise scale at $\sigma_0^2 = \frac{8d}{0.9\rho}$, where d is the number of attributes in the data domain. The number of possible measurements AIM+GReM makes scales with the size of the workload $\mathcal{W}^\downarrow$, while AIM+PGM scales with the number of attributes in the data domain.

F.1 Shared AIM Components

AIM+GReM and AIM+PGM share two main algorithmic components: the selection step and the budget annealing step. We describe these components in detail below.

Selection step. The selection step of AIM+PGM uses the exponential mechanism to select a marginal to measure each round. For $\gamma \in \mathcal{W}^\downarrow$ at iteration t, the score function is given by

$$\text{Score}_\gamma = w_\gamma \left(\|\mu_\gamma - \hat{\mu}_\gamma\| - \sqrt{2/\pi} \cdot \sigma_t \cdot n_\gamma \right)$$

where $w_\gamma = \sum_{\pi \in \mathcal{W}} |\gamma \cap \pi|$. This can be interpreted as the improvement in L_1 error we can expect in the reconstructed answers by measuring a given marginal, weighted to prioritize lower-order marginals whose attributes are contained in higher-order marginals in $\mathcal{W}^\downarrow$.

Budget annealing. The annealing step, given in Alg. 7, updates the selection privacy parameter ϵ_{t+1} and the measurement noise scale σ_{t+1}^2 and for the next round. This approach checks if the change between the current and prior reconstructed answer for the selected marginal is greater or less than the expected improvement. If the difference is less, the mechanism doubles the privacy budget spent on both selection and measurement for the next round (lines 1-6). This allows the mechanism to adapt both the selection step and measurement step to the privacy budget. If the mechanism is close to exhausting the privacy budget and cannot run for two additional rounds, the mechanism spends the remaining budget on the next round (lines 7-9).

F.2 Privacy Analysis

Theorem 5.1. *For $\rho > 0$,* AIM+GReM *satisfies ρ-zCDP.*

Proof. The initialization step of AIM+GReM makes d 1-way marginal measurements with noise scale $\sigma_0^2 = \frac{|\mathcal{W}^\downarrow|}{0.9\rho}$, spending budget $\rho_{init} = \frac{1}{2\sigma_0^2} = \frac{0.9\rho}{2|\mathcal{W}^\downarrow|}$ by Proposition B.4. By Proposition B.5(1), the initialization step satisfies $d\rho_{init}$-zCDP. Observe that $\rho_{init} < \rho$, since $d < |\mathcal{W}^\downarrow|$. This shows that the privacy budget is not overspent during initialization. Let $t > d$ denote the round of the mechanism. Each round t of AIM+GReM calls the exponential mechanism with privacy parameter ϵ_t and the Gaussian mechanism with noise scales output by solving a CRP problem with budget $\frac{1}{2\sigma_t^2}$. By Propositions B.3, 4.3, B.5(1), round t satisfies ρ_t-zCDP, where

Algorithm 7 Budget Annealing

Output: Selection budget ϵ_{t+1}, measurement noise scale σ^2_{t+1}

if $\|\mu_\gamma - \hat{\mu}_\gamma\| \leq \sqrt{2/\pi} \cdot \sigma_t \cdot n_\gamma$ **then do** ▷ Checks the annealing condition
  $\epsilon_{t+1} = 2\epsilon_t$ ▷ Doubles the selection budget
  $\sigma^2_{t+1} = \sigma^2_t/4$ ▷ Doubles the measurement budget
else do
  $\epsilon_{t+1} = \epsilon_t$
  $\sigma^2_{t+1} = \sigma^2_t$
if $(\rho - \rho_{used}) \leq 2(1/2\sigma^2_{t+1} + \epsilon^2_{t+1}/8)$ **then do** ▷ Checks if at least two rounds can be run
  $\epsilon_{t+1} = \sqrt{0.8 \cdot (\rho - \rho_{used})}$
  $\sigma^2_{t+1} = 1/1.8(\rho - \rho_{used})$

$\rho_t = \frac{\epsilon_t^2}{8} + \frac{1}{2\sigma_t^2}$. The cumulative privacy budget spent at round t is $\rho_{used} = \rho_{init} + \sum_{i=d+1}^{t} \rho_i$. It remains to show that ρ_{used} never exceeds ρ. The annealing step (Alg. 7) checks the condition $(\rho - \rho_{used}) \leq 2(\rho_{t+1})$. If the condition is not met, then $(\rho - \rho_{used}) > 2(\rho_{t+1})$, so round $t+1$ can run without overspending the privacy budget. If the condition is met, then the mechanism spends the remaining privacy budget on round $t+1$ so that $\rho_{t+1} = \frac{\left(\sqrt{0.8(\rho-\rho_{used})}\right)^2}{8} + \frac{1}{\left(\frac{2}{1.8(\rho-\rho_{used})}\right)} = \frac{\rho-\rho_{used}}{10} + \frac{9(\rho-\rho_{used})}{10} = \rho - \rho_{used}$. □

G Experiments

G.1 Experimental Details

Compute environment. All experiments were run on an internal CPU compute cluster. Experiments were allocated 20GB of memory and two cores with a limit of at most 14 days running time.

Datasets. Datasets included in the evaluation are described in Table 3. Note that we process the datasets by dropping any attributes with fewer than two unique values, dropping any records with missing values, and subset the datasets such that no attribute has more than 100 unique values. From these processed datasets, we infer data domains.

Dataset	Records	Attributes	Average Attribute Size	Total Domain Size
Adult	48,842	15	16.4	$9.44 * 10^{15}$
ACS	378,817	66	4.22	$3.82 * 10^{18}$
Loans	42,535	43	50.01	$1.54 * 10^{63}$

Table 3: Summary of datasets used in the experiments.

G.2 Additional Experiments

In this section, we provide additional experimental results. In Figure 5, we report results from Figure 3 comparing our in-axis approach with the baseline Shuffle algorithm for the `Decomp` and `Recon` operations. The remaining plots compare AIM+GReM to AIM+PGM and ResidualPlanner. First, we display the results of Figure 4 in absolute terms. In Figure 6, we report we report the mean L_1 error of the three methods by the privacy budget ϵ. In Figure 7, we report we report the mean running time in seconds by the privacy budget ϵ. In Figure 8, we report the mean running time in seconds by mean L_1 error. Next, we display the mean L_2 and max L_1 error of the three methods on the task of reconstructing all 3-way marginals by running time, stratified by privacy budget, in Figures 9 and 10, respectively. These results are consistent with the findings in Figure 4, presented in the main text. Figure 11 shows results for AIM+PGM with varying model size from 1MB to 100MB by running time. We observe that the reported results are robust to the model size.

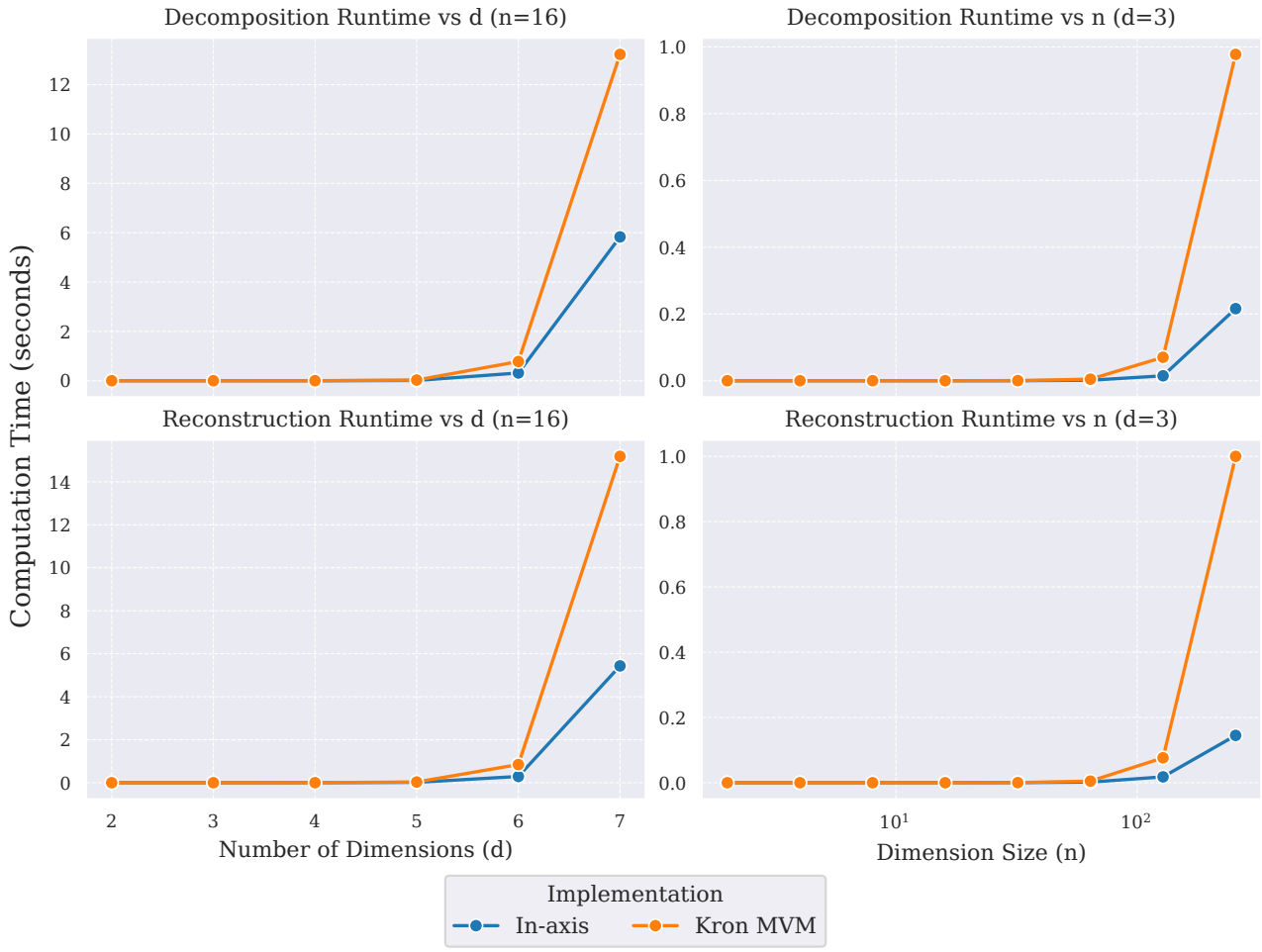

Figure 5: Running time in seconds of the `Decomp` and `Recon` operations with our in-axis implementation versus the baseline Kronecker-based approach.

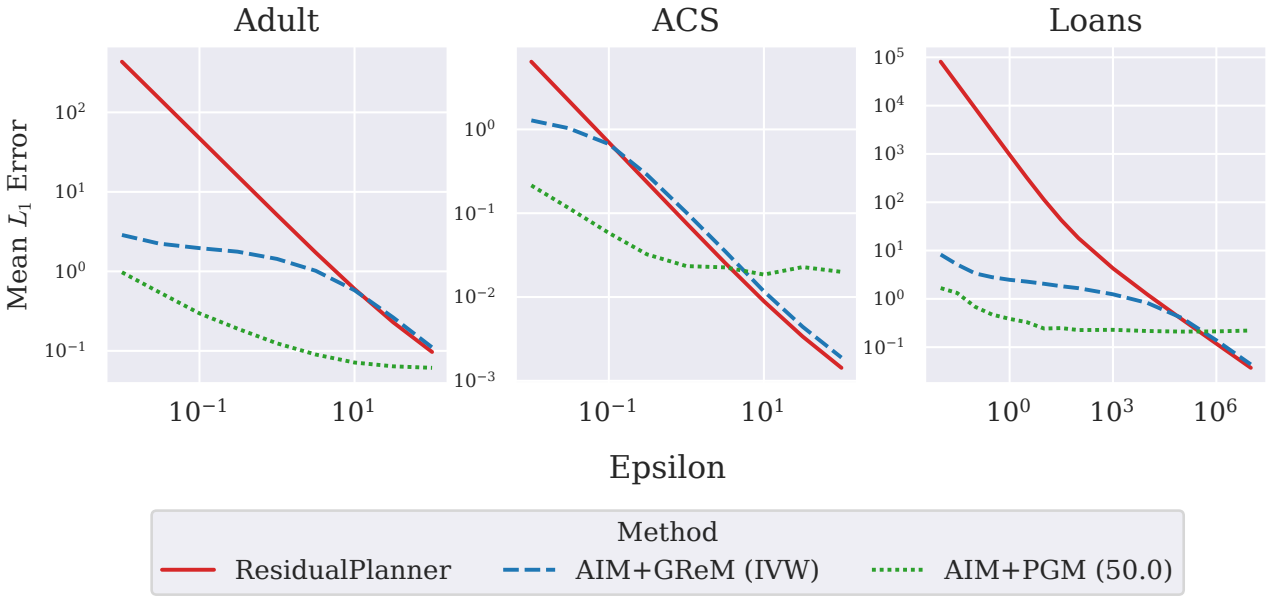

Figure 6: Mean L_1 error of AIM variants by ϵ on all 3-way marginals. Results are averaged across five trials. Privacy budgets ϵ indicated on the horizontal axis and $\delta = 10^{-9}$. AIM+PGM results use model size 50MB.

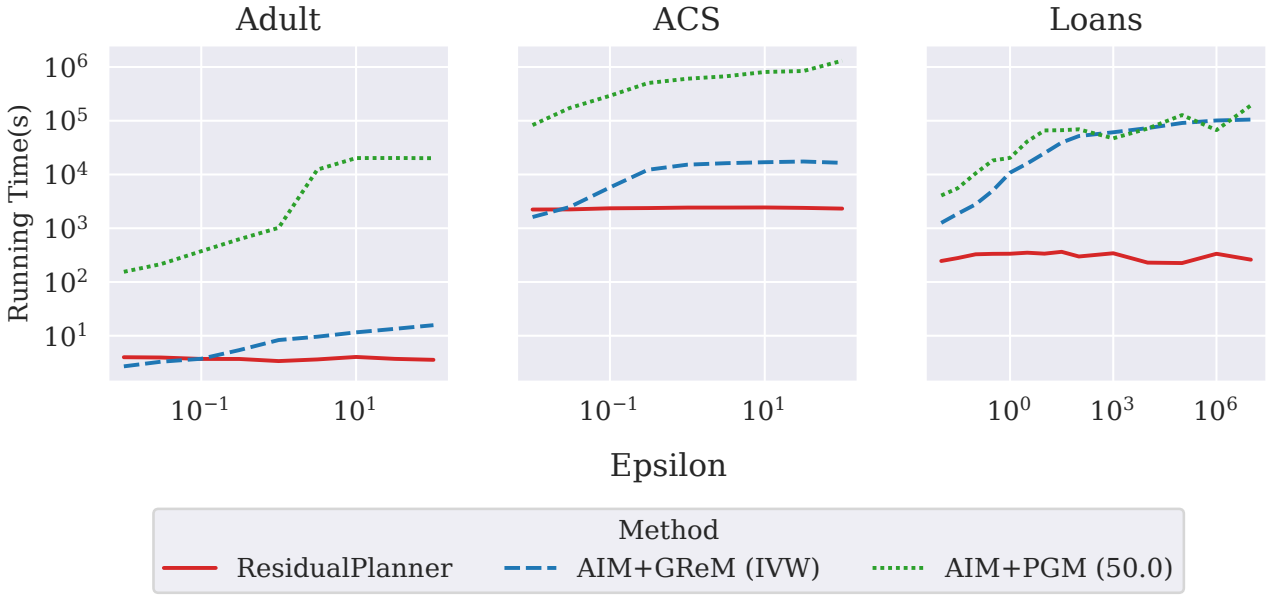

Figure 7: Running time of AIM variants in seconds by ϵ on all 3-way marginals. Results are averaged across five trials. Privacy budgets ϵ indicated on the horizontal axis and $\delta = 10^{-9}$. AIM+PGM results use model size 50MB.

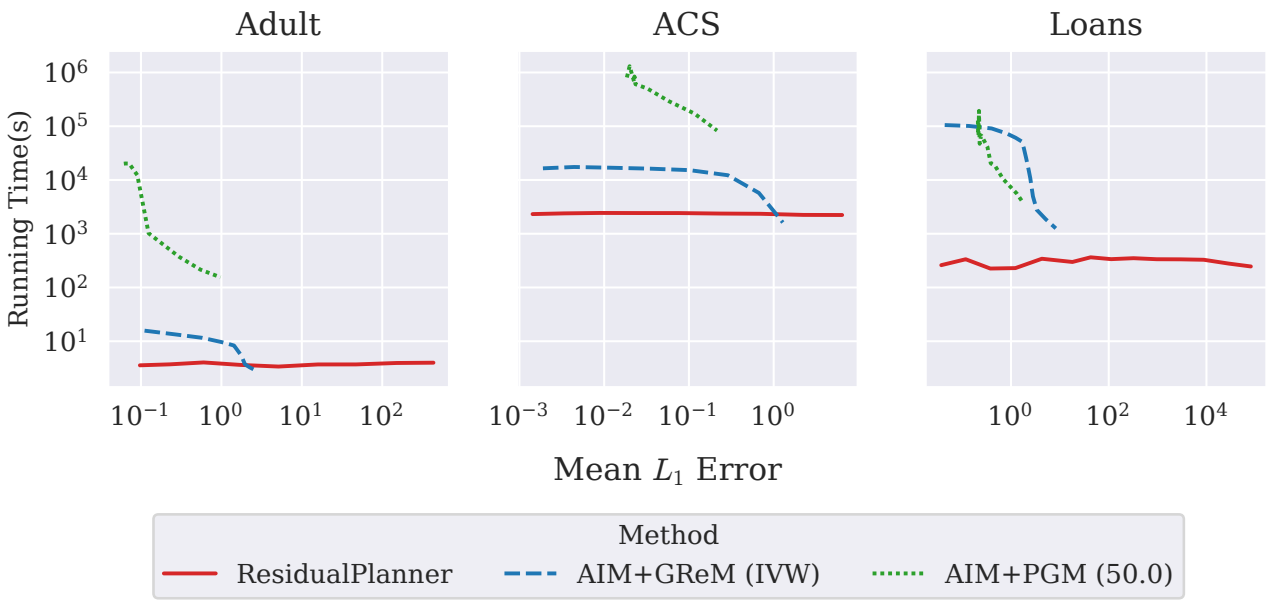

Figure 8: Running time of AIM variants in seconds by mean L_1 error on all 3-way marginals. Results are averaged across five trials. AIM+PGM results use model size 50MB.

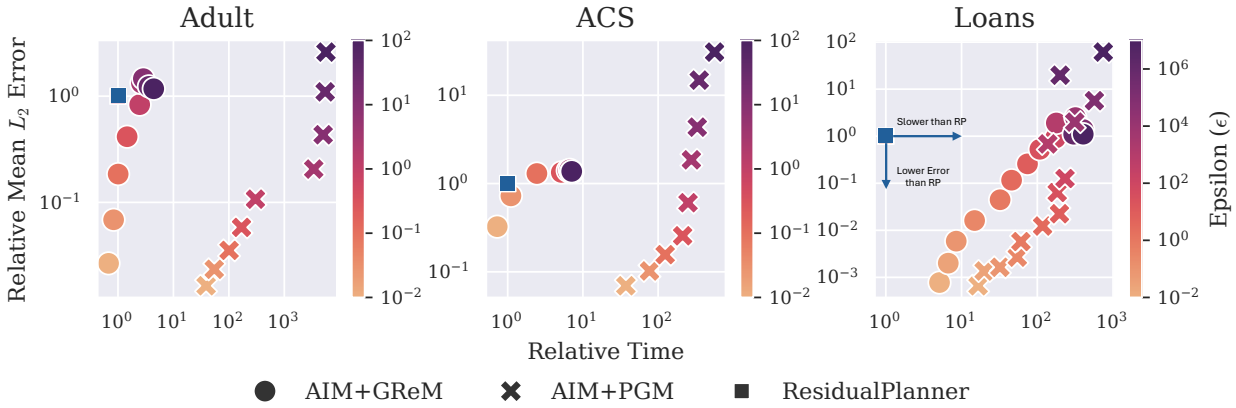

Figure 9: Mean L_2 error of AIM variants by running time on all 3-way marginals. Quantities are given relative to ResidualPlanner (i.e. AIM error / ResidualPlanner error). Results are averaged across five trials. Privacy budgets ϵ indicated in the vertical scale and $\delta = 10^{-9}$. AIM+PGM results use model size 50MB.

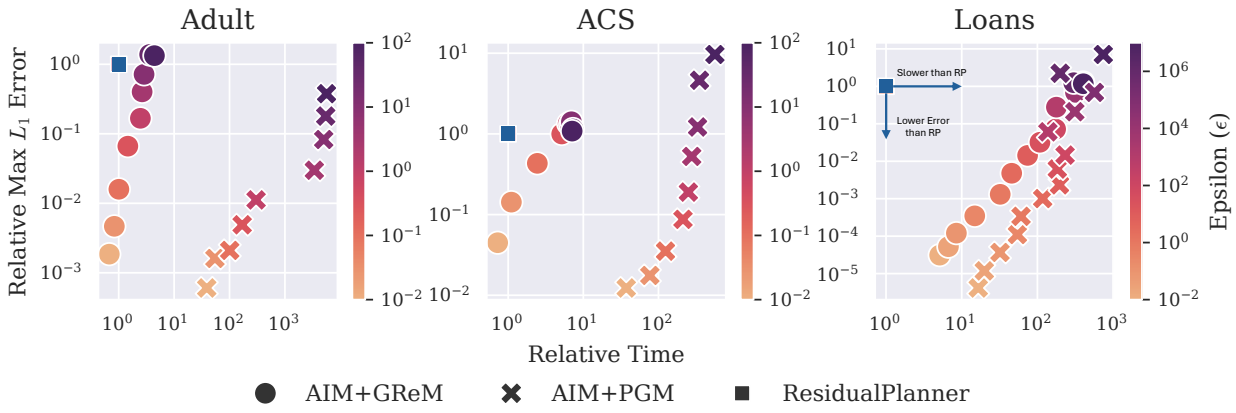

Figure 10: Max L_1 error of AIM variants by running time on all 3-way marginals. Quantities are given relative to ResidualPlanner (i.e. AIM error / ResidualPlanner error). Results are averaged across five trials. Privacy budgets ϵ indicated in the vertical scale and $\delta = 10^{-9}$. AIM+PGM results use model size 50MB.

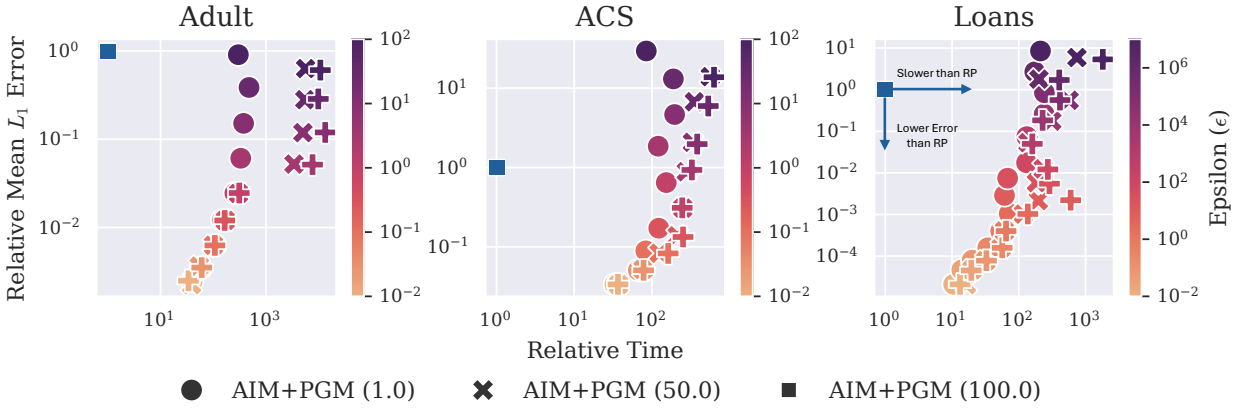

Figure 11: Mean L_1 error of AIM+PGM with varying model size by running time on all 3-way marginals. Quantities are given relative to ResidualPlanner (i.e. AIM error / ResidualPlanner error). Results are averaged across five trials. Privacy budgets ϵ indicated in the vertical scale and $\delta = 10^{-9}$.